aws

Entwicklerhandbuch

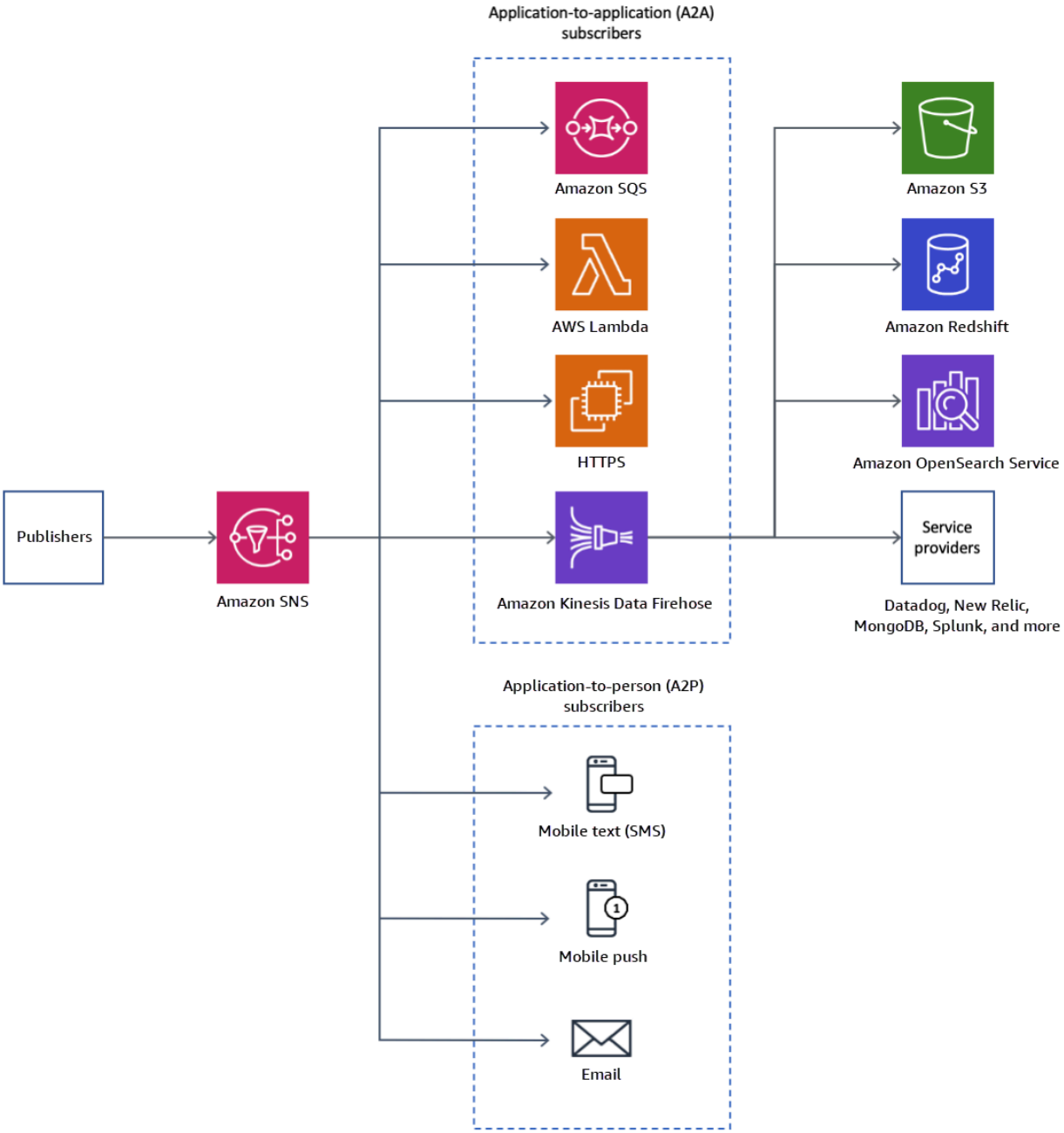
Amazon Simple Notification Service

Amazon Simple Notification Service: Entwicklerhandbuch

Table of Contents

Was ist Amazon SNS?

Amazon Simple Notification Service (Amazon SNS) ist ein verwalteter Service, der die Nachrichtenzustellung von Publishern an Abonnenten (auch bekannt alsProduzentenundKonsumenten) umfasst. Herausgeber kommunizieren asynchron mit Abonnenten, indem sie eine Nachricht erstellen und an ein Thema senden, bei dem es sich um einen logischen Zugriffspunkt und Kommunikationskanal handelt. Kunden können das SNS-Thema abonnieren und veröffentlichte Nachrichten über einen unterstützten Endpunkttyp wie Amazon Data Firehose, Amazon SQS, HTTP, E-Mail AWS Lambda, mobile Push-Benachrichtigungen und mobile Textnachrichten (SMS) empfangen.



Themen

- Funktionen und Funktionen
- Zugehörige Services
- Auf Amazon SNS zugreifen

- Preismodell für Amazon SNS
- Amazon SNS-Szenarien
- Amazon SNS mit einem AWS SDK verwenden

Funktionen und Funktionen

Amazon SNS bietet die folgenden grundlegenden Funktionen und Möglichkeiten:

- Eine Nachricht pplication-to-application

  Ein pplication-to-application Messaging unterstützt Abonnenten wie Amazon Data Firehose-Lieferstreams, Lambda-Funktionen, Amazon SQS SQS-Warteschlangen, HTTP/S-Endpunkte und Event Fork-Pipelines. AWS Weitere Informationen finden Sie unter Anwendung-zu-Anwendung-Messaging (A2A).
- A-Benachrichtigungen pplication-to-person

  pplication-to-person A-Benachrichtigungen enthalten Benutzerbenachrichtigungen für Abonnenten wie mobile Anwendungen, Handynummern und E-Mail-Adressen. Weitere Informationen finden Sie unter Application-to-Person-Messaging (A2P).
- Standard- und FIFO-Themen

  Verwenden Sie ein FIFO-Thema, um eine strikte Nachrichtenreihenfolge sicherzustellen, Nachrichtengruppen zu definieren und Nachrichtenduplizierungen zu verhindern. Sie können sowohl FIFO-, als auch Standard-Warteschlangen verwenden, um ein FIFO-Thema zu abonnieren. Weitere Informationen finden Sie unter Reihenfolge und Deduplizierung von Nachrichten (FIFO-Themen).

  Verwenden Sie ein Standardthema, wenn der Nachrichtenzustellungsauftrag und mögliche Nachrichtenduplizierung nicht kritisch sind. Alle unterstützten Bereitstellungsprotokolle können ein Standardthema abonnieren.
- Speicherdauer der Nachrichten

  Amazon SNS verwendet eine Reihe von Strategien, die zusammenarbeiten, um die Haltbarkeit von Nachrichten zu gewährleisten:

  - Veröffentlichte Nachrichten werden auf mehreren geografisch getrennten Servern und Rechenzentren gespeichert.

  - Wenn ein abonnierter Endpunkt nicht verfügbar ist, führt Amazon SNS eineRichtlinie zur Wiederholung von Zustellungenaus.
  - Um Nachrichten beizubehalten, die nicht zugestellt werden, bevor die Richtlinie für die Wiederholung der Zustellung endet, können Sie eineQueue unzustellbare Nachrichteneinrichten.
- Archivierung, Wiederholung und Analyse von Nachrichten

  Sie können Nachrichten auf verschiedene Arten mit Amazon SNS archivieren, z. B. indem Sie Firehose-Lieferstreams für SNS-Themen abonnieren, wodurch Sie Benachrichtigungen an Analyseendpunkte wie Amazon Simple Storage Service (Amazon S3) -Buckets, Amazon Redshift Redshift-Tabellen und mehr senden können. Darüber hinaus unterstützen Amazon SNS FIFO-Themen die Archivierung und Wiederholung von Nachrichten als integriertes Nachrichtenarchiv ohne Code, mit dem Themenbesitzer Nachrichten innerhalb ihres Themas speichern (oder archivieren) können. Themen-Suscriber können archivierten Nachrichten zurück an einen abonnierten Endpunkt abrufen (oder wiederholen). Weitere Informationen finden Sie unter Archivierung und Wiederholung von Nachrichten für FIFO-Themen.
- Nachrichtenattribute

  Nachrichtenattribute ermöglichen Ihnen die Bereitstellung strukturierter Metadatenelemente zu einer Nachricht. the section called “Nachrichtenattribute”.
- Nachrichtenfilterung

  Standardmäßig erhalten Abonnementen von Themen jede Nachricht, die zum Thema veröffentlicht wird. Um nur eine Teilmenge der Nachrichten zu erhalten, müssen Abonnenten dem Themen-Abonnement eine Filterrichtlinie zuweisen. Ein Abonnent kann auch den Geltungsbereich der Filterrichtlinie definieren, um eine nutzlastbasierte oder attributbasierte Filterung zu aktivieren. Der Standardwert für den Geltungsbereich der Filterrichtlinie ist `MessageAttributes`. Wenn die Attribute für eingehende Nachrichten mit den Filterrichtlinienattributen übereinstimmen, wird die Nachricht an den abonnierten Endpunkt übermittelt. Andernfalls wird die Nachricht herausgefiltert. Wenn der Geltungsbereich der Filterrichtlinie `MessageBody` ist, werden die Attribute der Filterrichtlinie mit der Nutzlast abgeglichen. Weitere Informationen finden Sie unter Nachrichtenfilterung.
- Nachrichten-Sicherheit

  Serverseitige Verschlüsselung schützt den Inhalt von Nachrichten, die in Amazon SNS SNS-Themen gespeichert sind, mithilfe von Verschlüsselungsschlüsseln, die von bereitgestellt werden. AWS KMS Weitere Informationen finden Sie unter the section called “Verschlüsselung im Ruhezustand”.

Sie können auch eine private Verbindung zwischen Amazon SNS und Ihrer Virtual Private Cloud (VPC) herstellen. Für weitere Informationen siehe:the section called “Richtlinie für den Datenverkehr zwischen Netzwerken”

Zugehörige Services

Sie können mit Amazon SNS folgenden Services verwenden:

- Amazon SQS bietet eine sichere, dauerhafte und verfügbare gehostete Queue, die es Ihnen ermöglicht, verteilte Softwaresysteme und -komponenten zu integrieren und zu entkoppeln. Amazon SQS ist auf folgende Weise mit Amazon SNS verbunden:
  - Amazon SNS stellt-Queues für unzustellbare Nachrichtenunterstützt von Amazon SQS für unzustellbare Nachrichten.
  - Sie können eine Amazon SQS-Warteschlange für ein Amazon-SNS-Thema abonnieren.
  - Sie können eine Amazon-SQS-FIFO-Warteschlange oder eine Standard-Warteschlange für ein Amazon-SNS-FIFO-Thema abonnieren. Nur Amazon-SQS-FIFO-Warteschlangen garantieren, dass Nachrichten in der richtigen Reihenfolge und ohne Duplikate empfangen werden.
- AWS Lambda ermöglicht es Ihnen, Anwendungen zu erstellen, die schnell auf neue Informationen reagieren. Lambda führt Ihren Code über eine hochverfügbaren Computing-Infrastruktur aus. Weitere Informationen finden Sie im AWS Lambda -Entwicklerhandbuch. Sie könnenSo abonnieren Sie eine Lambda Funktion für ein SNS-Themaaus.
- AWS Identity and Access Management (IAM) hilft Ihnen dabei, den Zugriff Ihrer Benutzer auf AWS Ressourcen sicher zu kontrollieren. Verwenden Sie IAM, um zu steuern, wer Ihre -Amazon SNS-Themen (Authentifizierung) verwenden kann, welche Themen verwendet werden können und auf welche Weise (Autorisierung). Weitere Informationen finden Sie unter Verwenden von identitätsbasierten Richtlinien mit Amazon SNS.
- AWS CloudFormationermöglicht es Ihnen, Ihre AWS Ressourcen zu modellieren und einzurichten. Erstellen Sie eine Vorlage, die die gewünschten AWS Ressourcen beschreibt, einschließlich Amazon SNS SNS-Themen und Abonnements. AWS CloudFormation kümmert sich um die Bereitstellung und Konfiguration dieser Ressourcen für Sie. Weitere Informationen finden Sie im AWS CloudFormation -Benutzerhandbuch.

Auf Amazon SNS zugreifen

Sie können SNS-Themen und -Abonnements mithilfe der Amazon SNS-Konsole, Befehlszeilentools oder AWS SDKs konfigurieren und verwalten.

- DieAmazon-SNS-Konsolebietet eine komfortable Benutzeroberfläche zum Erstellen von Themen und Abonnements, zum Senden und Empfangen von Nachrichten sowie zur Überwachung von Ereignissen und Protokollen.
- Mit AWS Command Line Interface (AWS CLI) erhalten Sie direkten Zugriff auf die Amazon SNS SNS-API für erweiterte Konfiguration und Automatisierung. Weitere Informationen finden Sie unter Verwenden von Lambda mit Amazon SNS AWS CLI.
- AWS stellt SDKs in verschiedenen Sprachen bereit. Weitere Informationen finden Sie unter SDKs und Tools.

Preismodell für Amazon SNS

Amazon SNS hat keine Vorabkosten. Sie zahlen basierend auf der Anzahl der Nachrichten, die Sie veröffentlichen, der Anzahl der Benachrichtigungen, die Sie bereitstellen, und allen zusätzlichen API-Aufrufen für die Verwaltung von Themen und Abonnements. Die Preise für die Lieferung variieren je nach Endpunkttyp. Mit dem kostenlosen Amazon-SNS-Kontingent können Sie kostenlos loslegen.

Weitere Informationen finden Sie unter Amazon SNS-Preise.

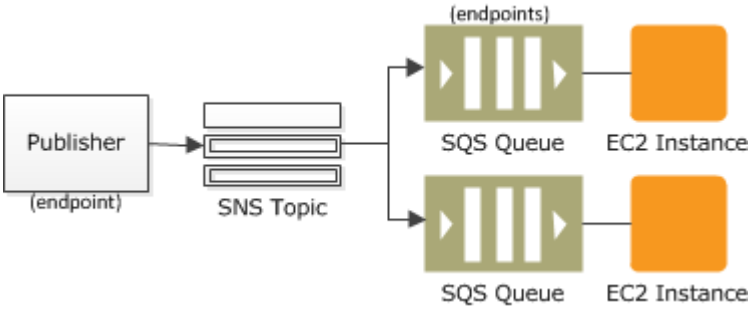
Amazon SNS-Szenarien

Anwendungsintegration

Beim Fanout-Szenario wird eine zu einem SNS-Thema veröffentlichte Nachricht repliziert und an mehrere Endpunkte wie Firehose-Zustellungsstreams, Amazon SQS SQS-Warteschlangen, HTTP (S) -Endpunkte und Lambda-Funktionen weitergeleitet. Auf diese Weise kann eine parallele, asynchrone Verarbeitung erfolgen.

Sie können z. B. eine Anwendung entwickeln, die immer dann eine Nachricht an ein SNS-Thema sendet, wenn eine Bestellung für ein Produkt eingeht. Auf diese Weise erhalten alle SQS-Queues, die dieses SNS-Thema abonniert haben, identische Benachrichtigungen für die neue Bestellung. Eine Amazon Elastic Compute Cloud (Amazon EC2)-Server-Instance, die einer der SQS-Queues zugewiesen ist, kann die Verarbeitung oder Ausführung der Bestellung ausführen. Außerdem

können Sie eine andere Amazon EC2-Server-Instance an ein Data Warehouse anhängen, um alle eingegangenen Bestellungen zu analysieren.



Eine weitere Möglichkeit zur Verwendung von des Verteilens liegt in der Replikation von Daten, die an Ihre Produktionsumgebung gesendet wurden, um sie mit Ihrer Testumgebung zu teilen. Um beim vorherigen Beispiel zu bleiben könnten Sie demselben SNS-Thema noch eine weitere SQS-Queue für neue eingehende Bestellungen hinzufügen. Wenn Sie diese neue SQS-Queue nun Ihrer Testumgebung anfügen, können Sie Ihre Anwendung mithilfe von Daten aus der Produktionsumgebung weiter verbessern und testen.

> **Important**
>
> Stellen Sie sicher, dass Sie Datenschutz und Sicherheit berücksichtigen, bevor Sie Produktionsdaten an Ihre Testumgebung senden.

Weitere Informationen finden Sie in den folgenden Ressourcen:

- Fanout zu Firehose-Bereitstellungsdatenströmen
- Aufrufen von Lambda-Funktionen
- Verteilung an Amazon-SQS-Warteschlangen
- Verteilung zu HTTP(S)-Endpunkten
- Event-Driven Computing mit Amazon SNS und AWS Rechen-, Speicher-, Datenbank- und Netzwerkservices

Warnungen für die Anwendung

Bei Anwendungs- und Systemwarnungen handelt es sich um Benachrichtigungen, die anhand vordefinierter Schwellenwerte ausgelöst und per SMS und/oder E-Mail an angegebene Benutzer gesendet werden. Amazon SNS kann diese Benachrichtigungen per SMS und E-Mail an bestimmte

Benutzer senden. Sie können beispielsweise sofort benachrichtigt werden, wenn ein Ereignis eintritt, z. B. eine bestimmte Änderung an Ihrer Amazon EC2 Auto Scaling Scaling-Gruppe, eine neue Datei, die in einen Amazon S3 S3-Bucket hochgeladen wurde, oder wenn ein metrischer Schwellenwert in Amazon überschritten wurde. CloudWatch Weitere Informationen finden Sie unter Amazon SNS SNS-Benachrichtigungen einrichten im CloudWatch Amazon-Benutzerhandbuch.

Benutzerbenachrichtigungen

Amazon SNS kann Push-E-Mails und Textnachrichten (SMS-Nachrichten) an Einzelpersonen oder Gruppen senden. Beispielsweise könnten Sie E-Commerce-Auftragsbestätigungen als Benutzerbenachrichtigungen senden. Weitere Informationen finden Sie unter Senden von Amazon-SNS-Nachrichten in Text-Messaging (SMS).

Mobile Push-Benachrichtigungen

Mobile Push-Benachrichtigungen ermöglichen es Ihnen, Nachrichten direkt an mobile Apps zu senden. Sie können Amazon SNS beispielsweise verwenden, um Aktualisierungsbenachrichtigungen an eine App zu senden. Die Benachrichtigung kann einen Link zum Herunterladen und Installieren der Aktualisierung enthalten. Weitere Informationen zur Verwendung von Amazon SNS zum Senden von Push-Benachrichtigungen finden Sie unter.Mobile Push-Benachrichtigungenaus.

Amazon SNS mit einem AWS SDK verwenden

AWS Software Development Kits (SDKs) sind für viele gängige Programmiersprachen verfügbar. Jedes SDK bietet eine API, Codebeispiele und Dokumentation, die es Entwicklern erleichtern, Anwendungen in ihrer bevorzugten Sprache zu erstellen.

SDK-Dokumentation	Codebeispiele
AWS SDK for C++	AWS SDK for C++ Code-Beispiele
AWS CLI	AWS CLI Code-Beispiele
AWS SDK for Go	AWS SDK for Go Code-Beispiele
AWS SDK for Java	AWS SDK for Java Code-Beispiele
AWS SDK for JavaScript	AWS SDK for JavaScript Code-Beispiele

SDK-Dokumentation	Codebeispiele
AWS SDK for Kotlin	AWS SDK for Kotlin Code-Beispiele
AWS SDK for .NET	AWS SDK for .NET Code-Beispiele
AWS SDK for PHP	AWS SDK for PHP Code-Beispiele
AWS Tools for PowerShell	Tools für PowerShell Codebeispiele
AWS SDK for Python (Boto3)	AWS SDK for Python (Boto3) Code-Beispiele
AWS SDK for Ruby	AWS SDK for Ruby Code-Beispiele
AWS SDK for Rust	AWS SDK for Rust Code-Beispiele
AWS SDK für SAP ABAP	AWS SDK für SAP ABAP Code-Beispiele
AWS SDK for Swift	AWS SDK for Swift Code-Beispiele

Weitere Beispiele für Amazon SNS finden Sie unter Codebeispiele für Amazon SNS mit AWS SDKs.

> **Beispiel Verfügbarkeit**
>
> Sie können nicht finden, was Sie brauchen? Fordern Sie ein Codebeispiel an, indem Sie unten den Link Provide feedback (Feedback geben) auswählen.

Amazon SNS Ereignisquellen und -ziele

Amazon SNS kann ereignisgesteuerte Benachrichtigungen von vielen AWS-Quellen und Benachrichtigungen sowohl an Application-to-Application-Ziele (A2A) als auch Anwendung-an-Person (A2P) verteilen. Dieser Abschnitt listet unterstützte Ereignisquellen und -ziele auf und enthält Links zu weiterführenden Informationen.

Themen

Amazon-SNS-Ereignisquellen

Diese Seite listet die AWS-Services, die Ereignisse in Amazon SNS-Themen veröffentlichen können, gruppiert nach ihrenAWSProduktkategorien auf.

> **Note**
>
> Amazon SNS führte FIFO-Themen im Oktober 2020 ein. Derzeit unterstützen die meisten AWS-Services nur das Senden von Ereignissen an Standardthemen.

Analyseservices

AWS-Service	Vorteile der Verwendung mit Amazon SNS
Amazon Athena – Ermöglicht Ihnen die Analyse von Daten in Amazon S3 mit Standard-SQL.	Erhalten Sie Benachrichtigungen, wenn Kontrolllimits überschritten werden. Weitere Informationen finden Sie unter Festlegen von Limits zur Kontrolle der Datennutzung im Benutzerhandbuch zu Amazon Athena.
AWS Data Pipeline – Hilft bei der Automatis ierung der Bewegung und Umwandlung von Daten.	Erhalten Sie Benachrichtigungen über den Status von Pipeline-Komponenten. Weitere Informationen finden Sie unter SnsAlarm im AWS Data Pipeline-Entwicklerhandbuch.

AWS-Service	Vorteile der Verwendung mit Amazon SNS
Amazon Redshift – Verwaltet alle Arbeiten zur Einrichtung, zum Betrieb und zum Skalieren eines Data Warehouse.	Erhalten Sie Benachrichtigungen über Amazon-Redshift-Ereignisse. Weitere Informationen finden Sie unter Amazon-Redshift-Ereignisbenachrichtigungen im Amazon-Redshift-Verwaltungshandbuch.

Services für die Anwendungsintegration

AWS-Service	Vorteile der Verwendung mit Amazon SNS
Amazon EventBridge – Stellt einen Stream von Echtzeitdaten aus Ihren eigenen Anwendungen, software-as-a-service (SaaS)-Anwendungen und -AWSServices bereit und leitet diese Daten an Ziele weiter, einschließlich Amazon SNS . EventBridge wurde früher als CloudWatch Ereignisse bezeichnet.	Erhalten Sie Benachrichtigungen über EventBridge Ereignisse. Weitere Informationen finden Sie unter Amazon EventBridge-Ziele im Amazon- EventBridge Benutzerhandbuch.
AWS Step Functions – Ermöglicht das Kombinieren von AWS Lambda-Funktionen und anderen AWS-Services zum Erstellen von geschäftskritischen Anwendungen.	Erhalten Sie eine Benachrichtigung über Step Functions-Ereignisse. Weitere Informationen finden Sie unter Aufrufen von Amazon SNS mit Step Functions im AWS Step Functions Entwicklerhandbuch.

Fakturierungs- und Kostenmanagementservices

AWS-Service	Vorteile der Verwendung mit Amazon SNS
AWS Billing and Cost Management – Bietet Funktionen, mit denen Sie Ihre Kosten überwachen und Ihre Rechnung bezahlen können.	Erhalten Sie Budgetbenachrichtigungen, Preisänderungsbenachrichtigungen und Anomaliewarnungen. Weitere Informationen finden Sie auf den folgenden Seiten im AWS Billing-Benutzerhandbuch:

AWS-Service	Vorteile der Verwendung mit Amazon SNS
	• Erstellen eines Amazon-SNS-Themas für Budget-Benachrichtigungen • Einrichten von Benachrichtigungen • Erkennen von ungewöhnlichen Ausgaben mit AWS Cost Anomaly Detection

Services für Geschäftsanwendungen

AWS-Service	Vorteile der Verwendung mit Amazon SNS
Amazon Chime – Ermöglicht Ihnen, geschäftliche Anrufe innerhalb und außerhalb Ihrer Organisation zu tätigen, zu chatten und sich zu treffen.	Erhalten Sie wichtige Benachrichtigungen über Meetingveranstaltungen. Weitere Informationen finden Sie unter Amazon-Chime-SDK-Ereignisbenachrichtigungen im Amazon-Chime-Entwicklerhandbuch.

Datenverarbeitungsservices

AWS-Service	Vorteil der Verwendung mit Amazon SNS
Amazon EC2 Auto Scaling – Damit erhalten Sie die richtige Anzahl von Amazon-Elastic-Compute-Cloud-Instances (Amazon EC2), die für die Handhabung Ihrer Anwendungslast verfügbar sind.	Erhalten Sie Benachrichtigungen, wenn Auto Scaling Amazon-EC2-Instances in Ihrer Auto-Scaling-Gruppe startet oder beendet. Weitere Informationen finden Sie unter Abrufen von Amazon-SNS-Benachrichtigungen über Skalierungen einer Auto-Scaling-Gruppe im Amazon-EC2-Auto-Scaling-Benutzerhandbuch.
EC2 Image Builder – Hilft bei der Automatisierung der Erstellung, Verwaltung und Bereitstellung benutzerdefinierter, sicherer und up-to-	Erhalten Sie Benachrichtigungen, wenn Builds abgeschlossen sind. Weitere Informationen finden Sie unter Tracking der neuesten

AWS-Service	Vorteil der Verwendung mit Amazon SNS
date Server-Images, die vorinstalliert und mit Software und Einstellungen vorkonfiguriert sind, um bestimmte IT-Standards zu erfüllen.	Serverbilder in EC2-Image-Builder-Pipelines auf dem AWS Compute-Blog.
AWS Elastic Beanstalk – Übernimmt automatisch die Kapazitätsbereitstellung, Lastverteilung, Skalierung und Überwachung des Anwendungsstatus.	Erhalten Sie Benachrichtigungen zu wichtigen Ereignissen, die sich auf Ihre Anwendung auswirken. Weitere Informationen finden Sie unter Elastic-Beanstalk-Umgebungsbenachrichtigungen mit Amazon SNS im AWS Elastic Beanstalk Entwicklerhandbuch.
AWS Lambda – Ermöglicht die Ausführung von Code ohne Bereitstellung oder Verwaltung von Servern.	Erhalten Sie Funktionsausgabedaten, indem Sie ein SNS-Thema als Lambda-Warteschlange für unzustellbare Nachrichten oder als Lambda-Ziel festlegen. Weitere Informationen finden Sie unter Asynchroner Aufruf im AWS Lambda Entwicklerhandbuch.
Amazon Lightsail – Hilft Entwicklern beim Einstieg mit AWS, um Websites oder Webanwendungen zu erstellen.	Erhalten Sie Benachrichtigungen, wenn eine Metrik für eine Ihrer Instances, Datenbanken oder Load Balancer einen angegebenen Schwellenwert überschreitet. Weitere Informationen finden Sie unter Hinzufügen von Benachrichtigungskontakten in Amazon Lightsail im Amazon-Lightsail-Entwicklerhandbuch.

Containerservices

AWS-Service	Vorteil der Verwendung mit Amazon SNS
Amazon EKS Distro – Ermöglicht Ihnen die Erstellung von zuverlässigen und sicheren Clustern, wo immer Ihre Anwendungen bereitgestellt werden.	Verfolgen Sie Updates und Sicherheits-Patches für Cluster, die mit Amazon EKS Distro erstellt wurden. Weitere Informationen finden Sie unter Einführung von Amazon EKS Distro –

AWS-Service	Vorteil der Verwendung mit Amazon SNS
	einer Open-Source-Kubernetes-Verteilung, verwendet von Amazon EKS.
Amazon Elastic Container Service (Amazon ECS) – Ermöglicht das Ausführen, Anhalten und Verwalten von Containern in einem Cluster.	Erhalten Sie Benachrichtigungen, wenn ein neues Amazon-ECS-optimiertes AMI verfügbar ist. Weitere Informationen finden Sie unter Abonnieren von Amazon-ECS-optimierten AMI-Update-Benachrichtigungen im Amazon-Elastic-Container-Service-Entwicklerhandbuch.

Services zur Kundenbindung

AWS-Service	Vorteil der Verwendung mit Amazon SNS
Amazon Connect – Sie können ein Omnichannel-Cloud-Kontaktcenter einrichten, um mit Ihren Kunden zu interagieren.	Erhalten Sie Warnungen und Validierungen. Weitere Informationen finden Sie unter Leistungsfähigkeit von AWS mit Amazon Connect im Amazon-Connect-Administratorhandbuch.
Amazon Pinpoint – Hilft Ihnen, Ihre Kunden zu engagieren, indem Sie ihnen E-Mails, SMS und Sprachnachrichten sowie Push-Benachrichtigungen senden.	Konfigurieren Sie bidirektionale SMS-Nachrichten, sodass Sie Nachrichten von Kunden empfangen können. Weitere Informationen finden Sie unter Verwenden von bidirektionalem SMS Messaging in Amazon Pinpoint im Amazon-Pinpoint-Benutzerhandbuch.
Amazon Simple Email Service (Amazon SES) – Bietet eine kostengünstige Möglichkeit, E-Mails mit Ihrer eigenen E-Mail-Adresse und Domänen zu senden und zu empfangen.	Empfangen von Benachrichtigungen zu Unzustellbarkeiten, Beschwerden und Zustellungen. Weitere Informationen finden Sie unter Konfigurieren von Amazon SNS-Benachrichtigungen für Amazon SES im Amazon-Simple-Email-Service-Entwicklerhandbuch.

Datenbankservices

AWS-Service	Vorteil der Verwendung mit Amazon SNS
AWS Database Migration Service – Migriert Daten aus lokalen Datenbanken in die AWS-Cloud.	Empfang von Benachrichtigungen, wenn AWS DMS-Ereignisse auftreten, z. B. wenn eine Replikationsinstanz erstellt oder gelöscht wird. Weitere Informationen finden Sie unter Arbeiten mit Ereignissen und Benachrichtigungen in AWS Database Migration Service im AWS Database Migration Service Benutzerhandbuch.
Amazon DynamoDB – Bietet eine schnelle und planbare Leistung mit nahtloser Skalierbarkeit in diesem vollständig verwalteten NoSQL-Datenbank-Service.	Erhalten Sie Benachrichtigungen, wenn Wartungsereignisse auftreten. Weitere Informationen finden Sie unter Anpassen der DAX-Cluster-Einstellungen im Amazon-DynamoDB-Entwicklerhandbuch.
Amazon ElastiCache – Bietet einen leistungsstarken, anpassbaren und kosteneffektiven In-Memory-Cache, ohne die mit der Bereitstellung und Verwaltung einer verteilten Cache-Umgebung verbundene Komplexität.	Erhalten Sie Benachrichtigungen, wenn wichtige Ereignisse auftreten. Weitere Informationen finden Sie unter Ereignisbenachrichtigungen und Amazon SNS im Benutzerhandbuch für Amazon ElastiCache für Memcached.
Amazon Neptune – Sie können Anwendungen erstellen und ausführen, die mit hochgradig verbundenen Datensätzen funktionieren.	Erhalten Sie Benachrichtigungen, wenn ein Neptune-Ereignis eintritt. Weitere Informationen finden Sie unter Verwenden von Neptune-Ereignisbenachrichtigungen im Neptune-Benutzerhandbuch.
Amazon Redshift – Verwaltet alle Arbeiten zur Einrichtung, zum Betrieb und zum Skalieren eines Data Warehouse.	Erhalten Sie Benachrichtigungen über Amazon-Redshift-Ereignisse. Weitere Informationen finden Sie unter Amazon-Redshift-Ereignisbenachrichtigungen im Amazon-Redshift-Verwaltungshandbuch.

AWS-Service	Vorteil der Verwendung mit Amazon SNS
Amazon Relational Database Service – Vereinfacht das Einrichten, Betreiben und Skalieren einer relationalen Datenbank in der AWS-Cloud.	Erhalten Sie Benachrichtigungen über Amazon-RDS-Ereignisse. Weitere Informationen finden Sie unter Verwenden von Amazon-RDS-Ereignisbenachrichtigungen im Amazon-RDS-Benutzerhandbuch.

Services für Entwickler-Tools

AWS-Service	Vorteil der Verwendung mit Amazon SNS
AWS CodeBuild – Kompiliert den Quellcode , führt Komponententests aus und erzeugt Artefakte, die bereitgestellt werden können.	Erhalten Sie Benachrichtigungen, wenn Builds erfolgreich sind, fehlschlagen oder von einer Build-Phase in eine andere verschoben werden. Weitere Informationen finden Sie unter Beispiel für Build-Benachrichtigungen für CodeBuild im AWS CodeBuild -Benutzerhandbuch.
AWS CodeCommit – Bietet Versionskontrolle für die private Speicherung und Verwaltung von Assets in der Cloud.	Erhalten Sie Benachrichtigungen über CodeCommit Repository-Ereignisse. Weitere Informationen finden Sie unter Beispiel: Erstellung eines AWS CodeCommit-Auslösers für ein Amazon-SNS-Thema im AWS CodeCommit Benutzerhandbuch.
AWS CodeDeploy – Automatisiert die Anwendungsbereitstellungen auf Amazon-EC2-Instances, lokalen Instances, serverless Lambda-Funktionen oder Amazon-ECS-Services.	Erhalten Sie Benachrichtigungen für CodeDeploy Bereitstellungen oder Instance-Ereignisse. Weitere Informationen finden Sie unter Erstellen eines Auslösers für ein CodeDeploy Ereignis im AWS CodeDeploy -Benutzerhandbuch.
Amazon CodeGuru – sammelt Laufzeitleistungsdaten aus Ihren Live-Anwendungen	Erhalten Sie Benachrichtigungen, wenn Anomalien auftreten. Weitere Informationen

AWS-Service	Vorteil der Verwendung mit Amazon SNS
und bietet Empfehlungen, mit denen Sie die Leistung Ihrer Anwendung optimieren können.	finden Sie unter Arbeiten mit Anomalien und Empfehlungsberichten im Amazon- CodeGuru Benutzerhandbuch.
AWS CodePipeline – Automatisiert die Schritte, die erforderlich sind, um Softwareänderungen kontinuierlich freizugeben.	Erhalten Sie Benachrichtigungen über Genehmigungsaktionen. Weitere Informationen finden Sie unter Verwalten von Genehmigungsaktionen in CodePipeline im AWS CodePipeline -Benutzerhandbuch.
AWS CodeStar – Erstellen, Verwalten und Arbeiten mit Softwareentwicklungsprojekten auf AWS.	Empfangen von Benachrichtigungen zu Ereignissen, die in den von Ihnen verwendeten Ressourcen auftreten. Weitere Informationen finden Sie unter Konfigurieren von Amazon-SNS-Themen für Benachrichtigungen im Developer-Tools-Console-Benutzerhandbuch.

Front-End-Web- und Mobildienste

AWS-Service	Vorteil der Verwendung mit Amazon SNS
Amazon Pinpoint – Hilft Ihnen, Ihre Kunden zu engagieren, indem Sie ihnen E-Mails, SMS und Sprachnachrichten sowie Push-Benachrichtigungen senden.	Konfigurieren Sie bidirektionale SMS-Nachrichten, sodass Sie Nachrichten von Kunden empfangen können. Weitere Informationen finden Sie unter Verwenden von bidirektionalem SMS-Messaging in Amazon Pinpoint im Amazon-Pinpoint-Benutzerhandbuch.

Services für Spieleentwicklung

AWS-Service	Vorteile der Verwendung mit Amazon SNS
Amazon GameLift – Bietet Lösungen für das Hosten sitzungsbasierter Multiplayer-	Erhalten Sie Benachrichtigungen über Matchmaking und Warteschlangenereignisse.

AWS-Service	Vorteile der Verwendung mit Amazon SNS
Spielserver in der Cloud, einschließlich eines vollständig verwalteten Service für die Bereitstellung, den Betrieb und die Skalierung von Spielservern.	Weitere Informationen finden Sie auf den folgenden Seiten: • Matchmaking-Benachrichtigungen finden Sie unter Einrichten von FlexMatch Ereignisbenachrichtigungen im Amazon- GameLift FlexMatch Entwicklerhandbuch. • Informationen zu Warteschlangenbenachrichtigungen finden Sie unter Einrichten von Ereignisbenachrichtigungen für die Platzierung von Spielsitzungen im Amazon- GameLift Entwicklerhandbuch.

Internet-of-Things-Services

AWS-Service	Vorteil der Verwendung mit Amazon SNS
AWS IoT Core – Bietet die Cloud-Dienste, die Ihre IoT-Geräte mit anderen Geräten und AWS-Cloud-Services verbinden.	Erhalten Sie Benachrichtigungen von AWS IoT Core-Ereignissen. Weitere Informationen finden Sie unter Erstellen einer Amazon SNS-Regel im AWS IoT Entwicklerhandbuch.
AWS IoT Device Defender – Ermöglicht das Überprüfen der Konfiguration Ihrer Geräte und Überwachen der verbundenen Geräte, um anormales Verhalten zu ermitteln und Sicherheitsrisiken zu minimieren.	Erhalten Sie Alarme, wenn ein Gerät gegen ein Verhalten verstößt. Weitere Informationen finden Sie unter Funktionsweise von AWS IoT Device Defender-Detekt im AWS IoT Entwicklerhandbuch.
AWS IoT Events – Ermöglicht Ihnen die Überwachung Ihrer Geräte und Geräteflotten auf Fehler oder Änderungen im Betrieb sowie für das Auslösen von Aktionen, wenn solche Ereignisse auftreten.	Erhalten Sie Benachrichtigungen von AWS IoT Events-Ereignissen. Weitere Informationen finden Sie unter Amazon Simple Notification Service im AWS IoT Events Entwicklerhandbuch.

AWS-Service	Vorteil der Verwendung mit Amazon SNS
AWS IoT Greengrass – Erweitert AWS auf physische Geräte, so dass diese auf die dort erzeugten Daten lokal einwirken können, während die Cloud weiterhin für Verwaltun g, Analytik und dauerhafte Speicherung verwendet wird.	Erhalten Sie Benachrichtigungen von AWS IoT Greengrass-Ereignissen. Weitere Informationen finden Sie unter SNS-Konnektor im AWS IoT Greengrass Version 1 Entwicklerhandbuch.

Machine-Learning-Services

AWS-Service	Vorteile der Verwendung mit Amazon SNS
Amazon CodeGuru – sammelt Laufzeitl eistungsdaten aus Ihren Live-Anwendungen und bietet Empfehlungen, mit denen Sie die Leistung Ihrer Anwendung optimieren können.	Erhalten Sie Benachrichtigungen, wenn Anomalien auftreten. Weitere Informationen finden Sie unter Arbeiten mit Anomalien und Empfehlungsberichten im Amazon- CodeGuru Benutzerhandbuch.
Amazon DevOps Guru – Generiert betriebliche Einblicke mithilfe von Machine Learning, um die Leistung Ihrer betrieblichen Anwendungen zu verbessern.	Weiterleiten von Erkenntnissen und Bestätigu ngen. Weitere Informationen finden Sie unter Liefern von ML-gestützten betrieblichen Einblicken an Ihre Bereitschaftsteams über PagerDuty mit Amazon DevOps Guru im AWS Management & Governance Blog.
Amazon Lookout for Metrics – Findet Anomalien in Ihren Daten, bestimmt deren Ursachen und ermöglicht Ihnen, schnell Maßnahmen zu ergreifen.	Erhalten Sie Benachrichtigungen über Anomalien. Weitere Informationen finden Sie unter Verwenden von Amazon SNS mit Lookout for Metrics im Amazon Lookout for Metrics Entwicklerhandbuch.
Amazon Rekognition – Sie können Bild- und Videoanalysen zu Ihren Anwendungen hinzufügen	Erhalten Sie Benachrichtigungen über Anforderungsergebnisse. Weitere Informati onen finden Sie unter Referenz: Videoanal

AWS-Service	Vorteile der Verwendung mit Amazon SNS
	yse Ergebnisbenachrichtigung im Amazon-Re kognition-Entwicklerhandbuch.
Amazon SageMaker – Ermöglicht es Datenwiss enschaftlern und Entwicklern, Machine-L earning-Modelle zu erstellen und zu trainieren und sie dann direkt in einer produktionsbereiten gehosteten Umgebung bereitzustellen.	Erhalten Sie Benachrichtigungen, wenn ein Datenobjekt gekennzeichnet ist. Weitere Informationen finden Sie unter Erstellen eines Streaming-Kennzeichnungsauftrags im Amazon- SageMaker Entwicklerhandbuch.

Management- und Governance-Services

AWS-Service	Vorteile der Verwendung mit Amazon SNS
AWS Chatbot – Ermöglicht DevOps es und Softwareentwicklungsteams, Amazon Chime- und Slack-Chatrooms zu verwenden, um Betriebsereignisse in der AWS Cloud zu überwachen und darauf zu reagieren.	Senden von Benachrichtigungen an Chatrooms . Weitere Informationen finden Sie unter Einrichten von AWS Chatbot im AWS Chatbot Administrator-Handbuch.
AWS CloudFormation – Ermöglicht es Ihnen, AWS-Infrastrukturen vorhersagbar und wiederholt zu erstellen und bereitzustellen.	Erhalten Sie Benachrichtigungen, wenn Stacks erstellt und aktualisiert werden. Weitere Informationen finden Sie unter Einrichten von AWS CloudFormation Stack-Optionen im AWS CloudFormation-Benutzerhandbuch.
AWS CloudTrail – Bietet den Ereignisverlauf Ihrer AWS-Konto-Aktivität.	Erhalten Sie Benachrichtigungen, wenn neue Protokolldateien in Ihrem Amazon S3-Bucket CloudTrail veröffentlicht. Weitere Informationen finden Sie unter Konfigurieren von Amazon SNS-Benachrichtigungen für CloudTrail im AWS CloudTrail -Benutzerhandbuch.
Amazon CloudWatch – Überwacht Ihre AWS Ressourcen und die Anwendungen, auf denen Sie ausführen, AWS in Echtzeit.	Erhalten Sie Benachrichtigungen, wenn Aufträge den Status ändern. Weitere Informati onen finden Sie unter Verwenden von Amazon-

AWS-Service	Vorteile der Verwendung mit Amazon SNS
	CloudWatch Alarmen im Amazon- CloudWatch Benutzerhandbuch.
AWS Config – Ermöglicht eine detaillierte Ansicht der Konfiguration von AWS-Ressourcen in AWS-Konto.	Benachrichtigungen erhalten, wenn Ressourcen aktualisiert werden oder wenn AWS Config benutzerdefinierte oder verwaltete Regeln anhand Ihrer Ressourcen ausgewertet werden. Weitere Informationen finden Sie unter Benachrichtigungen, die AWS Config an ein SNS-Thema sendet und Beispiel für Benachrichtigungen bei Änderung eines Konfigurationselements im AWS Config Entwicklerhandbuch.
AWS Control Tower – Ermöglicht Ihnen die Einrichtung und Verwaltung einer sicheren und konformen Multi-Account–AWS-Umgebung.	Verwenden Sie Warnungen, um Drift innerhalb Ihrer Landing Zone zu verhindern und Compliance-Benachrichtigungen zu erhalten. Weitere Informationen finden Sie unter Nachverfolgen von Warnungen mit Amazon Simple Notification Service im AWS Control Tower Benutzerhandbuch.
AWS License Manager – Hilft Ihnen bei der zentralen Verwaltung Ihrer Softwarelizenzen von Softwareanbietern in AWS-Umgebungen und Ihren lokalen Umgebungen.	Erhalten Sie LicenseManager-Benachrichtigungen- und Warnungen. Weitere Informationen finden Sie unter Einstellungen in License Manager im License Manager-Benutzerhandbuch und Erstellen von ServiceNow Vorfällen für AWS License ManagerBenachrichtigungen im Management & Governance Blog. AWS
AWS Service Catalog – Damit können IT-Administratoren Portfolios genehmigter Produkte erstellen, verwalten und für Endbenutzer verteilen, die dann in einem personalisierten Portal Zugriff auf die jeweils benötigten Produkte haben.	Erhalten Sie Benachrichtigungen über Stack-Ereignisse. Weitere Informationen finden Sie unter AWS Service Catalog Benachrichtigungseinschränkungen im Servicekatalog-Administrator-Handbuch.

AWS-Service	Vorteile der Verwendung mit Amazon SNS
AWS Systems Manager – Ermöglicht das Anzeigen und Steuern Ihrer Infrastruktur auf AWS.	Erhalten Sie Benachrichtigungen über den Status von Befehlen. Weitere Informationen finden Sie unter Überwachen von Systems-Manager-Statusänderungen mit Amazon-SNS-Benachrichtigungen im AWS Systems Manager Benutzerhandbuch.

Medienservices

AWS-Service	Vorteil der Verwendung mit Amazon SNS
Amazon Elastic Transcoder – Damit können Sie Mediendateien, die in Amazon S3 gespeichert werden, in ein Mediendateienformat konvertieren, das für Wiedergabegeräte der Kunden geeignet ist.	Erhalten Sie Benachrichtigungen, wenn Aufträge den Status ändern. Weitere Informationen finden Sie unter Benachrichtigungen über den Status eines Auftrags im Amazon-Elastic-Transcoder-Entwicklerhandbuch.

Migrations- und Übertragungsservices

AWS-Service	Vorteil der Verwendung mit Amazon SNS
AWS Application Discovery Service – Hilft Ihnen bei der Planung Ihrer Migration in die AWS-Cloud durch Sammeln von Nutzungs- und Konfigurationsdaten über Ihre lokalen Server.	Erhalten Sie Benachrichtigungen von Ereignissen durch AWS CloudTrail. Weitere Informationen finden Sie unter Protokollieren von Application-Discovery-Service-API-Aufrufen mit AWS CloudTrail im Application-Discovery-Service-Benutzerhandbuch.
AWS Database Migration Service – Migriert Daten aus lokalen Datenbanken in die AWS-Cloud.	Empfang von Benachrichtigungen, wenn AWS DMS-Ereignisse auftreten, z. B. wenn eine Replikationsinstanz erstellt oder gelöscht wird. Weitere Informationen finden Sie unter Arbeiten mit Ereignissen und Benachric

AWS-Service	Vorteil der Verwendung mit Amazon SNS
	htigungen in AWS Database Migration Service im AWS Database Migration Service Benutzerhandbuch.
AWS Snowball – Verwendet physische Speichergeräte, um große Datenmengen schnell zwischen Amazon S3 und Ihrem On-Premises-Datenspeicherort zu übertragen faster-than-internet .	Erhalten Sie Benachrichtigungen für Snowball-Aufträge. Weitere Informationen finden Sie hier: • Snowball-Benachrichtigungen im AWS Snowball Benutzerhandbuch • Schritt 5: Wählen Sie Ihre Benachrichtigungseinstellungen im AWS Snowball-Edge-Entwicklerhandbuch • Schritt 5: Wählen Sie Ihre Benachrichtigungseinstellungen im AWS Snowcone-Benutzerhandbuch

Services für Netzwerke und zur Bereitstellung von Inhalten

AWS-Service	Vorteile der Verwendung mit Amazon SNS
Amazon API Gateway – Ermöglicht es Ihnen, Ihre eigenen REST- und - WebSocket APIs in jeder Größenordnung zu erstellen und bereitzustellen.	Empfangen von Nachrichten, die an einen API-Gateway-Endpunkt gepostet wurden. Weitere Informationen finden Sie unter Tutorial: Erstellen einer API Gateway REST API mit AWS Integration im API-Gateway-Entwicklerhandbuch.
Amazon CloudFront – Beschleunigt die Verteilung Ihrer statischen und dynamischen Webinhalte wie HTML-, CSS-, PHP-, Bild- und Mediendateien.	Erhalten Sie Benachrichtigungen, wenn Alarme auf der Grundlage bestimmter CloudFront Metriken auftreten. Weitere Informationen finden Sie unter Festlegen von Alarmen für den

AWS-Service	Vorteile der Verwendung mit Amazon SNS
	Empfang von Benachrichtigungen im Amazon-CloudFront Entwicklerhandbuch.
AWS Direct Connect – Verknüpft Ihr internes Netzwerk über ein standardmäßiges Glasfaser kabel mit einem AWS Direct Connect-Standort.	Erhalten Sie Benachrichtigungen, wenn Alarme, die den Status einer AWS Direct Connect-Verbindung überwachen, den Status ändern. Weitere Informationen finden Sie unter Erstellen von CloudWatch Alarmen zur Überwachung von AWS Direct Connect Verbindungen im AWS Direct Connect - Benutzerhandbuch.
Elastic Load Balancing – Verteilt Ihren eingehenden Datenverkehr automatisch auf mehrere Ziele, z. B. Amazon-EC2-Instances, Container und IP-Adressen oder mehr Availabil ity Zones.	Erhalten Sie Benachrichtigungen über Alarme, die Sie für Load-Balancer-Ereignisse erstellt haben. Weitere Informationen finden Sie unter Erstellen von CloudWatch Alarmen für Ihren Load Balancer im Benutzerhandbuch für Classic Load Balancer.
Amazon Route 53 – Bietet Domänenregistrieru ng, DNS-Routing und Zustandsprüfung.	Erhalten Sie Benachrichtigungen, wenn die Zustandsprüfung fehlerhaft ist. Weitere Informationen finden Sie unter So erhalten Sie eine Amazon-SNS-Benachrichtigung, wenn bei der Zustandsprüfung der Status fehlerhaf t ist (Konsole) im Amazon Route 53 Entwickle rhandbuch.
Amazon Virtual Private Cloud (Amazon VPC) – Ermöglicht Ihnen das Starten von AWS-Resso urcen in einem virtuellen Netzwerk, das Sie definiert haben.	Sie können eine Benachrichtigung erstellen, um Warnungen bei bestimmten Ereignissen zu erhalten, die auf Ihrem Schnittstellenendp unkt stattfinden. Weitere Informationen finden Sie unter Erstellen und verwalten einer Benachrichtigung für einen Endpunkt-Service im Amazon-VPC-Benutzerhandbuch.

Sicherheits-, Identitäts- und Compliance-Services

AWS-Service	Vorteil der Verwendung mit Amazon SNS
AWS Directory Service – Bietet mehrere Möglichkeiten zur Nutzung von Microsoft Active Directory (AD) mit anderen AWS-Services.	Erhalten Sie E-Mail- oder Text- (SMS-)Nachrichten, wenn sich der Status Ihres Verzeichnisses ändert. Weitere Informationen finden Sie unter Konfigurieren von Benachrichtigungen über den Verzeichnisstatus im AWS Directory Service Administration-Handbuch.
Amazon GuardDuty – Bietet eine kontinuierliche Sicherheitsüberwachung, um unerwartete und potenziell nicht autorisierte oder böswillige Aktivitäten in Ihrer -AWSUmgebung zu identifizieren.	Erhalten Sie Benachrichtigungen über neu veröffentlichte Suchtypen, Updates der vorhandenen Suchtypen und andere Änderungen in den Funktionalitäten. Weitere Informationen finden Sie unter Abonnieren von GuardDuty Ankündigungen im SNS-Thema im Amazon- GuardDuty Benutzerhandbuch.
Amazon Inspector – Testen Sie die Netzwerkzugänglichkeit Ihrer Amazon-EC2-Instances und den Sicherheitsstatus Ihrer Anwendungen, die auf diesen Instances ausgeführt werden.	Erhalten Sie Benachrichtigungen für Amazon Inspector Ereignisse. Weitere Informationen finden Sie unter Einrichten eines SNS-Themas für Amazon-Inspector-Benachrichtigungen im Amazon-Inspector-Benutzerhandbuch.
AWS Security Hub: Automatisiert AWS-Sicherheitsüberprüfungen und zentralisiert Sicherheitswarnungen.	Erhalten Sie Benachrichtigungen über AWS Security Hub-Ankündigungen, einschließlich Benachrichtigungen über AWS Security Hub-Steuerelemente oder Standards, die hinzugefügt, bearbeitet oder entfernt wurden. Weitere Informationen finden Sie unter: Abonnieren einer AWS Security Hub-Ankündigung mit Amazon SNS.

Serverless-Services

AWS-Service	Vorteil der Verwendung mit Amazon SNS
Amazon DynamoDB – Bietet eine schnelle und planbare Leistung mit nahtloser Skalierbarkeit in diesem vollständig verwalteten NoSQL-Dat enbank-Service.	Erhalten Sie Benachrichtigungen, wenn Wartungsereignisse auftreten. Weitere Informationen finden Sie unter Anpassen der DAX-Cluster-Einstellungen im Amazon-Dy namoDB-Entwicklerhandbuch.
Amazon EventBridge – Stellt einen Stream von Echtzeitdaten aus Ihren eigenen Anwendungen, software-as-a-service (SaaS)-Anwendungen und -AWSServices bereit und leitet diese Daten an Ziele weiter, einschließlich Amazon SNS . EventBridge wurde früher als CloudWatch Ereignisse bezeichnet.	Erhalten Sie Benachrichtigungen über EventBridge Ereignisse. Weitere Informationen finden Sie unter Amazon EventBridge-Ziele im Amazon- EventBridge Benutzerhandbuch.
AWS Lambda – Ermöglicht die Ausführung von Code ohne Bereitstellung oder Verwaltung von Servern.	Erhalten Sie Funktionsausgabedaten, indem Sie ein SNS-Thema als Lambda-Warteschlan ge für unzustellbare Nachrichten oder als Lambda-Ziel festlegen. Weitere Informationen finden Sie unter Asynchroner Aufruf im AWS Lambda Entwicklerhandbuch.

Speicherservices

AWS-Service	Vorteil der Verwendung mit Amazon SNS
AWS Backup – Hilft bei der Zentralisierung und Automatisierung der Sicherung von Daten über AWS-Services in der Cloud und vor Ort	Erhalten Sie Benachrichtigungen von AWS Backup-Ereignissen. Weitere Informationen finden Sie unter Verwenden von Amazon SNS zur Nachverfolgung von AWS Backup-Er eignissen im AWS Backup Entwicklerhandbuch .

AWS-Service	Vorteil der Verwendung mit Amazon SNS
Amazon Elastic File System – Stellt Dateispeicher für Ihre Amazon-EC2-Instances bereit.	Erhalten Sie Benachrichtigungen über Alarme, die Sie für Amazon-EFS-Ereignisse erstellt haben. Weitere Informationen finden Sie unter Automatisierte Überwachungstools im Amazon-Elastic-File-System-Benutzerhandbuch.
Amazon S3 Glacier – Stellt Speicher für selten verwendete Daten bereit.	Sie können eine Benachrichtigungskonfiguration für einen Tresor einrichten, so dass nach Abschluss eines Auftrags eine Nachricht an ein SNS-Thema gesendet wird. Weitere Informationen finden Sie unter Konfigurieren von Tresorbenachrichtigungen in Amazon S3 Glacier im Amazon-S3-Glacier-Entwicklerhandbuch.
Amazon Simple Storage Service (Amazon S3) – Bietet Objektspeicherung.	Erhalten Sie Benachrichtigungen, wenn Änderungen an einem Amazon-S3-Bucket auftreten oder in dem seltenen Fall, dass Objekte nicht in ihre Zielregion repliziert werden. Weitere Informationen finden Sie unter Walkthrough: Konfigurieren eines Buckets für Benachrichtigungen (SNS-Thema und SQS-Warteschlange) und Überwachung des Fortschritts mit Replikationsmetriken und Amazon-S3-Ereignisbenachrichtigungen im Benutzerhandbuch zur Amazon-Simple-Storage-Service-Konsole.

AWS-Service	Vorteil der Verwendung mit Amazon SNS
AWS Snowball – Verwendet physische Speichergeräte, um große Datenmengen schnell zwischen Amazon S3 und Ihrem On-Premises-Datenspeicherort zu übertragen faster-than-internet .	Erhalten Sie Benachrichtigungen für Snowball-Aufträge. Weitere Informationen finden Sie hier: • Snowball-Benachrichtigungen im AWS Snowball Benutzerhandbuch • Schritt 5: Wählen Sie Ihre Benachrichtigungseinstellungen im AWS Snowball-Edge-Entwicklerhandbuch • Schritt 5: Wählen Sie Ihre Benachrichtigungseinstellungen im AWS Snowcone-Benutzerhandbuch

Zusätzliche Ereignisquellen

Quelle	Vorteile der Verwendung mit Amazon SNS
Tägliche Updates für AWS-Funktionen	Erhalten Sie über ein Amazon-SNS-Thema zeitnah detaillierte Informationen zu Veröffentlichungen und Updates für AWS. Zu diesen Versionen gehören AWS-Regionen, AWS-Services, Amazon-VPC-Endpunkte, die in AWS Service Quotas AWS-Services integriert sind, Amazon EC2-Instance-Typen, Amazon-Instance SageMaker -Typen, Amazon-Nimble-Studio-Instance-Typen, Amazon-RDS-Datenbank-Engine-Versionen und Amazon-MSK-Apache-Kafka-Versionen. Service Quotas Weitere Informationen finden Sie im AWS-News-Blog unter Abonnieren täglicher Updates für AWS-Funktionen über Amazon SNS.

Quelle	Vorteile der Verwendung mit Amazon SNS
AWSIP-Adressbereiche	Erhalten Sie Benachrichtigungen über Änderungen an AWS-IP-Bereichen über ein Amazon-SNS-Thema. Weitere Informationen finden Sie unter Benachrichtigungen von AWS-IP-Adressbereichen in der Allgemeine Amazon Web Services-Referenz und Abonnieren von Änderungen der öffentlichen AWS-IP-Adressen im AWS-News-Blog.

Weitere Informationen zum ereignisgesteuerten Computing finden Sie in den folgenden Quellen:

- Was ist eine ereignisgesteuerte Architektur?
- Ereignisgesteuertes Computing mit Amazon SNS und AWS Computing-, Speicher-, Datenbanken- und Netzwerk-Services auf dem AWS Compute Blog
- Anreicherung ereignisgesteuerter Architekturen mit AWS Event Fork Pipelines auf dem AWS Compute Blog

Amazon SNS Eventziele

Auf dieser Seite werden alle Ziele aufgeführt, die Informationen zu Ereignissen erhalten können, gruppiert nach application-to-application (A2A)-Nachrichten und application-to-person (A2P)-Benachrichtigungen.

> **Note**
>
> Amazon SNS eingeführtFIFO-Themenim Oktober 2020. Derzeit sind die meistenAWS-Dienste unterstützen nur das Empfangen von Ereignissen aus SNS-Standardthemen. Amazon SQS unterstützt den Empfang von Ereignissen aus SNS-Standard- und FIFO-Themen.

A2A-Ziele

Eventziel	Vorteile der Verwendung mit Amazon SNS
Amazon Data Firehose	Bereitstellen von Ereignissen für Delivery Streams zu Archivierungs- und Analysezwecken. Über Bereitstellungsdatenströme können Sie Ereignisse an AWS Ziele wie Amazon Simple Storage Service (Amazon S3), Amazon Redshift und Amazon OpenSearch Service (OpenSearch Service) oder an Ziele von Drittanbietern wie Datadog, New Relic, MongoDB und Splunk liefern. Weitere Informationen finden Sie unter Fanout zu Firehose-Bereitstellungsdatenströmen.
AWS Lambda	Ereignisse an Funktionen liefern zum Auslösen der Ausführung benutzerdefinierter Geschäftslogik. Weitere Informationen finden Sie unter Aufrufen von Lambda-Funktionen.
Amazon SQS	Bereitstellen von Ereignissen in Queues für Anwendungsintegrationszwecke. Weitere Informationen finden Sie unter Verteilung an Amazon-SQS-Warteschlangen.
AWS Event Fork Pipelines	Bereitstellen von Ereignissen für Ereignissicherung und -speicherung, Ereignissuche und -analysen oder Ereigniswiedergabepipelines. Weitere Informationen finden Sie unter Fanout zu AWS Event Fork Pipelines.
HTTP/S	Liefern von Ereignissen an externe Webhooks. Weitere Informationen finden Sie unter Verteilung zu HTTP(S)-Endpunkten.

A2P-Ziele

Eventziel	Vorteile der Verwendung mit Amazon SNS
SMS	Bereitstellen von Ereignissen an Mobiltelefone als Textnachrichten. Weitere Informationen finden Sie unter Text-Messaging (SMS).
Email	Übermitteln Sie Ereignisse als E-Mail-Nachrichten an Posteingänge. Weitere Informationen finden Sie unter E-Mail-Benachrichtigungen.
Plattformendpunkt	Bereitstellen von Ereignissen auf Mobiltelefonen als native Push-Benachrichtigungen. Weitere Informationen finden Sie unter Mobile Push-Benachrichtigungen.
AWS Chatbot	Liefern Sie Ereignisse an Amazon Chime Chatrooms oder Slack-Kanäle. Weitere Informationen finden Sie auf den folgenden Seiten imAWS Chatbot-Administratorhandbuch: • Einrichten von AWS Chatbot mit Amazon Chime • Einrichten AWS Chatbot mit Slack • Verwenden von AWS Chatbot mit anderen AWS-Services
PagerDuty	Bereitstellen von operativen Einblicken für Bereitschaftsteams. Weitere Informationen finden Sie unter Liefern von ML-gestützten betrieblichen Erkenntnissen für Ihre Bereitschaftsteams über PagerDuty mit Amazon DevOps

Eventziel	Vorteile der Verwendung mit Amazon SNS
	Guru im AWS Management & Governance Blog.

> **Note**
>
> Sie können sowohl nativeAWSEreignisse und benutzerdefinierte Ereignisse, um Apps zu chatten:
>
> - NativeAWS-EreignisseSie können AWS Chatbotnative Ereignisse AWSüber Amazon SNS -Themen an Amazon Chime und Slack senden. Die unterstützten nativen AWS Ereignisse umfassen Ereignisse von AWS Billing and Cost Management, AWS Health, CloudWatch, AWS CloudFormationAmazon und mehr. Weitere Informationen finden Sie unterAWS Chatbotmit anderen -Services verwendenimAWS Chatbot-Administratorhandbuch.
> - Benutzerdefinierte Ereignisse— Sie können Ihre benutzerdefinierten Ereignisse auch über Amazon SNS-Themen an Amazon Chime, Slack und Microsoft Teams senden. Dazu veröffentlichen Sie benutzerdefinierte Ereignisse in einem SNS-Thema, das die Ereignisse an eine abonnierte Lambda-Funktion übermittelt. Die Lambda-Funktion verwendet dann den Webhook der Chat-App, um die Ereignisse an Empfänger zu übermitteln. Weitere Informationen finden Sie unterWie verwende ich Webhooks, um Amazon SNS Nachrichten in Amazon Chime, Slack oder Microsoft Teams zu veröffentlichen?

Einrichten des Zugriffs für Amazon SNS

Bevor Sie Amazon SNS zum ersten Mal verwenden können, müssen Sie die folgenden Schritte abschließen.

Themen

Schritt 1: Einen AWS-Konto und einen IAM-Benutzer erstellen

Um auf einen AWS Dienst zugreifen zu können, müssen Sie zunächst einen AWS-Kontoerstellen. Sie können Ihren verwenden AWS-Konto , um Ihre Aktivitäts- und Nutzungsberichte einzusehen und Authentifizierung und Zugriff zu verwalten.

Melden Sie sich an für eine AWS-Konto

Wenn Sie noch keine haben AWS-Konto, führen Sie die folgenden Schritte aus, um eine zu erstellen.

Um sich für eine anzumelden AWS-Konto

1. Öffnen Sie https://portal.aws.amazon.com/billing/signup.
2. Folgen Sie den Online-Anweisungen.

   Bei der Anmeldung müssen Sie auch einen Telefonanruf entgegennehmen und einen Verifizierungscode über die Telefontasten eingeben.

   Wenn Sie sich für einen anmelden AWS-Konto, wird ein AWS-Konto Root-Benutzer erstellt. Der Root-Benutzer hat Zugriff auf alle AWS-Services und Ressourcen des Kontos. Aus Sicherheitsgründen sollten Sie einem Benutzer Administratorzugriff zuweisen und nur den Root-Benutzer verwenden, um Aufgaben auszuführen, für die Root-Benutzerzugriff erforderlich ist.

AWS sendet Ihnen nach Abschluss des Anmeldevorgangs eine Bestätigungs-E-Mail. Sie können jederzeit Ihre aktuelle Kontoaktivität anzeigen und Ihr Konto verwalten. Rufen Sie dazu https://aws.amazon.com/ auf und klicken Sie auf Mein Konto.

Erstellen Sie einen Benutzer mit Administratorzugriff

Nachdem Sie sich für einen angemeldet haben AWS-Konto, sichern Sie Ihren AWS-Konto Root-Benutzer AWS IAM Identity Center, aktivieren und erstellen Sie einen Administratorbenutzer, sodass Sie den Root-Benutzer nicht für alltägliche Aufgaben verwenden.

Schützen Sie Ihren AWS-Konto Root-Benutzer

1. Melden Sie sich AWS Management Consoleals Kontoinhaber an, indem Sie Root-Benutzer auswählen und Ihre AWS-Konto E-Mail-Adresse eingeben. Geben Sie auf der nächsten Seite Ihr Passwort ein.

   Hilfe bei der Anmeldung mit dem Root-Benutzer finden Sie unter Anmelden als Root-Benutzer im AWS-Anmeldung Benutzerhandbuch zu.
2. Aktivieren Sie die Multi-Faktor-Authentifizierung (MFA) für den Root-Benutzer.

   Anweisungen finden Sie unter Aktivieren eines virtuellen MFA-Geräts für Ihren AWS-Konto Root-Benutzer (Konsole) im IAM-Benutzerhandbuch.

Erstellen Sie einen Benutzer mit Administratorzugriff

1. Aktivieren Sie das IAM Identity Center.

   Anweisungen finden Sie unter Aktivieren AWS IAM Identity Center im AWS IAM Identity Center Benutzerhandbuch.
2. Gewähren Sie einem Benutzer in IAM Identity Center Administratorzugriff.

   Ein Tutorial zur Verwendung von IAM-Identity-Center-Verzeichnis als Identitätsquelle finden Sie unter Benutzerzugriff mit der Standardeinstellung konfigurieren IAM-Identity-Center-Verzeichnis im AWS IAM Identity Center Benutzerhandbuch.

Melden Sie sich als Benutzer mit Administratorzugriff an

- Um sich mit Ihrem IAM-Identity-Center-Benutzer anzumelden, verwenden Sie die Anmelde-URL, die an Ihre E-Mail-Adresse gesendet wurde, als Sie den IAM-Identity-Center-Benutzer erstellt haben.

  Hilfe bei der Anmeldung mit einem IAM Identity Center-Benutzer finden Sie im AWS-Anmeldung Benutzerhandbuch unter Anmeldung beim AWS Zugriffsportal.

Weisen Sie weiteren Benutzern Zugriff zu

1. Erstellen Sie in IAM Identity Center einen Berechtigungssatz, der der bewährten Methode zur Anwendung von Berechtigungen mit den geringsten Rechten folgt.

   Anweisungen finden Sie im Benutzerhandbuch unter Einen Berechtigungssatz erstellen.AWS IAM Identity Center
2. Weisen Sie Benutzer einer Gruppe zu und weisen Sie der Gruppe dann Single Sign-On-Zugriff zu.

   Anweisungen finden Sie im AWS IAM Identity Center Benutzerhandbuch unter Gruppen hinzufügen.

Nächste Schritte

Da Sie nun auf die Arbeit mit Amazon SNS vorbereitet sind, sollten Sie die Ersten Schritte ausführen, indem Sie ein Thema erstellen, ein Abonnement für das Thema erstellen, eine Nachricht zum Thema veröffentlichen und das Abonnement und das Thema löschen.

Erste Schritte mit Amazon SNS

In diesem Abschnitt lernen Sie Amazon SNS besser kennen, indem Ihnen die Verwaltung von Themen, Abonnements und Nachrichten über die Amazon SNS-Konsole gezeigt wird.

Themen

Voraussetzungen

Bevor Sie beginnen, führen Sie die Schritte in Einrichten des Zugriffs für Amazon SNS aus.

Schritt 1: Erstellen von Themen

1. Melden Sie sich bei der Amazon-SNS-Konsole an.
2. Wählen Sie im linken Navigationsbereich Topics (Themen).
3. Wählen Sie auf der Seite Topics (Themen) Create New Topic (Neues Thema erstellen) aus.
4. Standardmäßig erstellt die Konsole ein FIFO-Thema. Klicken Sie auf Standard.
5. Geben Sie im Abschnitt Details einen Name nfür das Thema ein, z. B. `MyTopic`.
6. Scrollen Sie zum Ende des Formulars und wählen Sie Create topic (Erstellen eines Themas) aus.

   In der Konsole wird die Seite Details geöffnet.

Schritt 2: Erstellen von Themenabonnements für Endpunkte

1. Wählen Sie im linken Navigationsbereich Subscriptions (Abonnements).

2. Wählen Sie auf der Seite Subscriptions (Abonnements) die Option Create subscription (Abonnement erstellen)aus.
3. Klicken Sie auf der Create subscription (Abonnement erstellen) und wählen Sie die Option Topic ARN (ARN-Thema) aus, um eine Liste der Themen in Ihrem AWS-Konto zu sehen.
4. Wählen Sie das Mesh aus, das Sie im vorherigen Schritt erstellt haben.
5. Wählen Sie unter Protocol (Protokoll) die Option Email (E-Mail) aus.
6. Geben Sie unter Endpoint (Endpunkt) eine E-Mail-Adresse ein, um die Benachrichtigungen zu empfangen.
7. Wählen Sie Create subscription (Abonnement erstellen) aus.

   Die Konsole öffnet die Seite Details.
8. Überprüfen Sie Ihren E-Mail-Posteingang und wählen Sie Das Abonnement bestätigen in der E-Mail von AWS Benachrichtigungen. Die Absender-ID lautet normalerweise „no-reply@sns.amazonaws.com“.
9. Amazon SNS öffnet Ihren Webbrowser und zeigt eine Abonnementbestätigung mit Ihrer Abonnement-ID an.

Schritt 3: Veröffentlichen von Nachrichten zu Themen

1. Wählen Sie im linken Navigationsbereich Topics (Themen).
2. Wählen Sie auf der Seite Topics (Themen) das zuvor erstellte Thema und anschließend Publish message (Nachricht veröffentlichen) aus.

   Die Konsole öffnet die Seite Publish message to topic (Nachricht im Thema veröffentlichen).
3. (Optional) Geben Sie im Abschnitt Message details (Nachrichtendetails) den Subject (Betreff) ein, z. B.:

   ```
   Hello from Amazon SNS!
   ```
4. Im Message body (Nachrichtentext) wählen Sie die Option Identical payload for all delivery protocols (Identische Nutzlast für alle Lieferprotokolle) aus, und geben dann einen Nachrichtentext ein, z. B.:

   ```
   Publishing a message to an SNS topic.
   ```
5. Wählen Sie Publish message (Nachricht veröffentlichen) aus.

Die Nachricht wird im Thema veröffentlicht. Die Konsole öffnet die Seite Details.

6. Überprüfen Sie Ihren E-Mail-Posteingang, und stellen Sie sicher, dass Sie eine E-Mail von Amazon SNS mit der veröffentlichten Nachricht erhalten haben.

Schritt 4: Löschen von Abonnements und Themen

1. Wählen Sie im Navigationsbereich Subscriptions (Abonnements) aus.
2. Wählen Sie auf der Abonnement-Seite ein bestätigtes Abonnement aus und wählen dann Delete (Löschen).

> **Note**
>
> Eine ausstehende Bestätigung kann nicht gelöscht werden. Nach 48 Stunden löscht Amazon SNS es automatisch.

3. Wählen Sie im Dialogfeld Delete Subnet (Subnetz löschen) die Option Delete Subnet (Subnetz löschen) aus.

   Das Abonnement wird gelöscht.
4. Wählen Sie im Navigationsbereich Topics (Themen) aus.
5. Wählen Sie auf der Seite Topics (Themen) ein Thema und anschließend Edit (Bearbeiten) aus.

> **Important**
>
> Wenn Sie ein Thema löschen, löschen Sie auch alle Abonnements zu diesem Thema.

6. Geben Sie im Dialogfeld Thema löschen ***MyTopic*** ein `delete me` und wählen Sie dann Löschen aus.

   Das Thema wird gelöscht.

Nächste Schritte

Nachdem Sie nun ein Thema mit einem Abonnement erstellt und Nachrichten an das Thema gesendet haben, können Sie Folgendes ausprobieren:

- Erkunden Sie das AWS-Entwickler-Center.
- Informationen zum Schutz Ihrer Daten und zum Zugriff auf diese finden Sie im Abschnitt Sicherheit.
- Aktivieren der serverseitigen Verschlüsselung für ein Thema.
- Aktivieren Sie die serverseitige Verschlüsselung für ein Thema mit einem Abbonement für Amazon Simple Queue Service (Amazon SQS).
- Abonnieren Sie AWS Event Fork Pipelines für ein Thema.

Amazon SNS Konfiguration

Verwenden Sie die Amazon SNS Konsole, um Amazon SNS-Themen und -Abonnements zu erstellen und zu konfigurieren. Weitere Informationen zu Amazon SNS finden Sie hier: Was ist Amazon SNS?

Themen

Erstellen Sie ein Amazon SNS-Thema

Ein Amazon SNS-Thema ist ein logischer Zugriffspunkt, der als ein Kommunikationskanal fungiert. Mit einem Thema können Sie mehrere Endpunkte gruppieren (z. B. AWS Lambda Amazon SQS, HTTP/S oder eine E-Mail-Adresse).

Zum Senden der Nachrichten eines Nachrichtenproduktionssystems (etwa einer e-Commerce-Website), die mit mehreren anderen Services zusammenarbeiten, die dessen Nachrichten benötigen (beispielsweise Checkout- und Erfüllungssysteme) können Sie ein Thema für Ihr Produktionssystem erstellen.

Die erste und häufigste Amazon SNS-Aufgabe ist das Erstellen eines Themas. Auf dieser Seite wird gezeigt AWS Management Console, wie Sie die, und die verwenden können AWS SDK for Java, um ein Thema AWS SDK for .NET zu erstellen.

Während der Erstellung wählen Sie einen Thementyp (Standard oder FIFO) und benennen das Thema. Der Thementyp oder -name kann nach dem Erstellen eines Themas nicht geändert werden. Alle anderen Konfigurationsoptionen sind während der Themenerstellung optional und Sie können sie später bearbeiten.

> **Important**
>
> Fügen Sie keine persönlich identifizierbare Informationen (PII) oder andere vertrauliche oder sensible Informationen in Themennamen hinzu. Themennamen sind für andere Amazon Web

Services zugänglich, einschließlich CloudWatch Logs. Themennamen sind nicht für private oder sensible Daten gedacht.

Themen

Um ein Thema mit dem zu erstellen AWS Management Console

1. Melden Sie sich bei der Amazon-SNS-Konsole an.
2. Führen Sie eine der folgenden Aktionen aus:
   - Wenn noch nie zuvor Themen unter Ihrem AWS-Konto Konto erstellt wurden, lesen Sie die Beschreibung von Amazon SNS auf der Startseite.
   - Wenn AWS-Konto zuvor Themen unter Ihrem Konto erstellt wurden, wählen Sie im Navigationsbereich die Option Themen aus.
3. Klicken Sie auf der Seite Themen auf Create New Topic.
4. Führen Sie auf der Seite Create subscription (Abonnement erstellen) im Abschnitt Details die folgenden Schritte aus:
   a. FürTypWählen Sie einen Thementyp aus (Standard-oderFIFO).
   b. Geben Sie den Namen des neuen Themas ein. Für einFIFO-ThemaFügen SieFIFOam Ende des Namens hinzu.
   c. (Optional) Geben Sie einen Display name (Anzeigenamen) für Ihr Thema ein.

> **Important**
>
> Wenn Sie einen E-Mail-Endpunkt abonnieren, darf die kombinierte Zeichenanzahl für den Anzeigenamen des Amazon-SNS-Themas und die sendende E-Mail-Adresse (z. B. no-reply@sns.amazonaws.com) 320 UTF-8-Zeichen nicht überschreiten. Sie können ein Verschlüsselungstool eines Drittanbieters verwenden, um die Länge der Absenderadresse zu überprüfen, bevor Sie einen Anzeigenamen für Ihr Amazon-SNS-Thema konfigurieren.

d. (Optional) Für ein FIFO-Thema können SieDeduplizierung für inhaltsbasierte Nachrichten, um die Standardnachrichtendeduplizierung zu aktivieren. Weitere Informationen finden Sie unter Nachrichtendeduplizierung für FIFO-Themen.

5. (Optional) Erweitern Sie den Abschnitt Encryption (Verschlüsselung) und gehen Sie wie folgt vor. Weitere Informationen finden Sie unter Verschlüsselung im Ruhezustand.

   a. Wählen Sie Enable encryption (Verschlüsselung aktivieren) aus.

   b. Geben Sie den AWS KMS Schlüssel an. Weitere Informationen finden Sie unter Wichtige Begriffe.

      Für jeden KMS-Typ werden Description (Beschreibung), Account (Konto) und KMS ARN (KMS-ARN) angezeigt.

      > **Important**
      >
      > Wenn Sie nicht der Besitzer des KMS-Schlüssels sind oder wenn Sie sich mit einem Konto anmelden, das über keine `kms:ListAliases`- und `kms:DescribeKey`-Berechtigungen verfügt, können Sie auf der Amazon-SNS-Konsole keine Informationen über den KMS aufrufen.
      > Bitten Sie den Inhaber des KMS, Ihnen diese Berechtigungen zu erteilen. Beispiele und weitere Informationen zu AWS KMS -Berechtigungen finden Sie unter API-Berechtigungen: Referenztabelle für Aktionen und Ressourcen im AWS Key Management Service Benutzerhandbuch.

      - Der AWS verwaltete KMS für Amazon SNS (Standard) Alias/aws/sns ist standardmäßig ausgewählt.

        > **Note**
        >
        > Beachten Sie Folgendes:
        >
        > - Wenn Sie AWS Management Console zum ersten Mal den AWS verwalteten KMS für Amazon SNS für ein Thema angeben, AWS KMS wird der AWS verwaltete KMS für Amazon SNS erstellt.

- Alternativ können Sie, wenn Sie die `Publish` Aktion zum ersten Mal für ein Thema mit aktivierter SSE verwenden, das AWS KMS AWS verwaltete KMS für Amazon SNS erstellen.

- Um ein benutzerdefiniertes KMS von Ihrem AWS Konto aus zu verwenden, wählen Sie das KMS-Schlüsselfeld und dann das benutzerdefinierte KMS aus der Liste aus.

> **Note**
>
> Weitere Informationen zum Erstellen benutzerdefinierter KMSs finden Sie unter Erstellen von Schlüsseln im AWS Key Management Service -Entwicklerhandbuch.

- Um einen benutzerdefinierten KMS-ARN von Ihrem AWS Konto oder einem anderen AWS Konto zu verwenden, geben Sie ihn in das Feld KMS-Schlüssel ein.

6. (Optional) Standardmäßig kann nur der Eigentümer des Themas das Thema abonnieren oder Veröffentlichungen dazu vornehmen. Um zusätzliche Zugriffsberechtigungen zu konfigurieren, erweitern Sie den Abschnitt Access policy (Zugriffsrichtlinie). Weitere Informationen finden Sie unter Identity and Access Management in Amazon SNS und Beispiele für die Zugriffskontrolle in Amazon SNS.

> **Note**
>
> Wenn Sie ein Thema mit der Konsole erstellen, verwendet die Standardrichtlinie den Bedingungsschlüssel `aws:SourceOwner`. Dieser Schlüssel ist ähnlich wie `aws:SourceAccount`.

7. (Optional) Um zu konfigurieren, wie Amazon SNS fehlgeschlagene Nachrichtenzustellversuche wiederholt, erweitern Sie den Abschnitt Delivery retry policy (HTTP/S) (Richtlinie für die Zustellungswiederholung (HTTP/S)). Weitere Informationen finden Sie unter Wiederholungsversuche bei der Nachrichtenzustellung Amazon SNS.
8. (Optional) Um zu konfigurieren, wie Amazon SNS die Zustellung von Nachrichten protokolliert CloudWatch, erweitern Sie den Abschnitt Protokollierung des Lieferstatus. Weitere Informationen finden Sie unter Zustellungsstatus von Amazon SNS Nachrichtenübermittmittungen.
9. (Optional) Wenn Sie Metadaten-Tags zum Thema hinzufügen, erweitern Sie den Tag-Abschnitt, geben Sie einen Schlüssel und einen Wert ein (optional), und wählen Sie Tag hinzufügen aus. Weitere Informationen finden Sie unter Markierung von Amazon-SNS-Themen.

10. Wählen Sie Thema erstellen aus.

    Das Thema wird erstellt und die ***MyTopic***Seite wird angezeigt.

    Der Name, der ARN, (optional) der Anzeigename und die AWS Konto-ID des Themenbesitzers werden im Abschnitt Details angezeigt.
11. Kopieren Sie das Thema ARN in die Zwischenablage, zum Beispiel:

```
arn:aws:sns:us-east-2:123456789012:MyTopic
```

Um ein Thema mit einem AWS SDK zu erstellen

Um ein AWS SDK verwenden zu können, müssen Sie es mit Ihren Anmeldeinformationen konfigurieren. Weitere Informationen finden Sie unter Freigegebene Konfigurations- und Anmeldeinformationsdateien im AWS -Referenzhandbuch zu SDKs und Tools.

Die folgenden Codebeispiele zeigen, wie Sie es verwenden`CreateTopic`.

.NET

AWS SDK for .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

Erstellen Sie ein Thema mit einem bestimmten Namen.

```
using System;
using System.Threading.Tasks;
using Amazon.SimpleNotificationService;
using Amazon.SimpleNotificationService.Model;

/// <summary>
/// This example shows how to use Amazon Simple Notification Service
/// (Amazon SNS) to add a new Amazon SNS topic.
/// </summary>
public class CreateSNSTopic
```

```
    {
        public static async Task Main()
        {
            string topicName = "ExampleSNSTopic";

            IAmazonSimpleNotificationService client = new
 AmazonSimpleNotificationServiceClient();

            var topicArn = await CreateSNSTopicAsync(client, topicName);
            Console.WriteLine($"New topic ARN: {topicArn}");
        }

        /// <summary>
        /// Creates a new SNS topic using the supplied topic name.
        /// </summary>
        /// <param name="client">The initialized SNS client object used to
        /// create the new topic.</param>
        /// <param name="topicName">A string representing the topic name.</param>
        /// <returns>The Amazon Resource Name (ARN) of the created topic.</
returns>
        public static async Task<string>
 CreateSNSTopicAsync(IAmazonSimpleNotificationService client, string topicName)
        {
            var request = new CreateTopicRequest
            {
                Name = topicName,
            };

            var response = await client.CreateTopicAsync(request);

            return response.TopicArn;
        }
    }
```

Erstellen Sie ein neues Thema mit einem Namen und spezifischen FIFO- und Deduplizierungsattributen.

```
    /// <summary>
    /// Create a new topic with a name and specific FIFO and de-duplication
 attributes.
    /// </summary>
```

```
    /// <param name="topicName">The name for the topic.</param>
    /// <param name="useFifoTopic">True to use a FIFO topic.</param>
    /// <param name="useContentBasedDeduplication">True to use content-based de-
duplication.</param>
    /// <returns>The ARN of the new topic.</returns>
    public async Task<string> CreateTopicWithName(string topicName, bool
 useFifoTopic, bool useContentBasedDeduplication)
    {
        var createTopicRequest = new CreateTopicRequest()
        {
            Name = topicName,
        };

        if (useFifoTopic)
        {
            // Update the name if it is not correct for a FIFO topic.
            if (!topicName.EndsWith(".fifo"))
            {
                createTopicRequest.Name = topicName + ".fifo";
            }

            // Add the attributes from the method parameters.
            createTopicRequest.Attributes = new Dictionary<string, string>
            {
                { "FifoTopic", "true" }
            };
            if (useContentBasedDeduplication)
            {
                createTopicRequest.Attributes.Add("ContentBasedDeduplication",
 "true");
            }
        }

        var createResponse = await
 _amazonSNSClient.CreateTopicAsync(createTopicRequest);
        return createResponse.TopicArn;
    }
```

- Einzelheiten zur API finden Sie CreateTopicin der AWS SDK for .NET API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

```
//! Create an Amazon Simple Notification Service (Amazon SNS) topic.
/*!
  \param topicName: An Amazon SNS topic name.
  \param topicARNResult: String to return the Amazon Resource Name (ARN) for the
 topic.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SNS::createTopic(const Aws::String &topicName,
                              Aws::String &topicARNResult,
                              const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::SNS::Model::CreateTopicRequest request;
    request.SetName(topicName);

    const Aws::SNS::Model::CreateTopicOutcome outcome =
 snsClient.CreateTopic(request);

    if (outcome.IsSuccess()) {
        topicARNResult = outcome.GetResult().GetTopicArn();
        std::cout << "Successfully created an Amazon SNS topic " << topicName
                  << " with topic ARN '" << topicARNResult
                  << "'." << std::endl;

    }
    else {
        std::cerr << "Error creating topic " << topicName << ":" <<
                  outcome.GetError().GetMessage() << std::endl;
        topicARNResult.clear();
    }
```

```
    return outcome.IsSuccess();
}
```

- Einzelheiten zur API finden Sie CreateTopicin der AWS SDK for C++ API-Referenz.

CLI

AWS CLI

So erstellen Sie ein SNS-Thema

Das folgende `create-topic`-Beispiel erstellt ein SNS-Thema namens `my-topic`.

```
aws sns create-topic \
    --name my-topic
```

Ausgabe:

```
{
    "ResponseMetadata": {
        "RequestId": "1469e8d7-1642-564e-b85d-a19b4b341f83"
    },
    "TopicArn": "arn:aws:sns:us-west-2:123456789012:my-topic"
}
```

Weitere Informationen finden Sie unter Verwenden der AWS Befehlszeilenschnittstelle mit Amazon SQS und Amazon SNS im Benutzerhandbuch für die AWS Befehlszeilenschnittstelle.

- Einzelheiten zur API finden Sie unter CreateTopic AWS CLIBefehlsreferenz.

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

```
// SnsActions encapsulates the Amazon Simple Notification Service (Amazon SNS)
 actions
// used in the examples.
type SnsActions struct {
 SnsClient *sns.Client
}



// CreateTopic creates an Amazon SNS topic with the specified name. You can
 optionally
// specify that the topic is created as a FIFO topic and whether it uses content-
based
// deduplication instead of ID-based deduplication.
func (actor SnsActions) CreateTopic(topicName string, isFifoTopic bool,
 contentBasedDeduplication bool) (string, error) {
 var topicArn string
 topicAttributes := map[string]string{}
 if isFifoTopic {
  topicAttributes["FifoTopic"] = "true"
 }
 if contentBasedDeduplication {
  topicAttributes["ContentBasedDeduplication"] = "true"
 }
 topic, err := actor.SnsClient.CreateTopic(context.TODO(), &sns.CreateTopicInput{
  Name:       aws.String(topicName),
  Attributes: topicAttributes,
 })
 if err != nil {
  log.Printf("Couldn't create topic %v. Here's why: %v\n", topicName, err)
 } else {
  topicArn = *topic.TopicArn
 }

 return topicArn, err
}
```

- Einzelheiten zur API finden Sie CreateTopicin der AWS SDK for Go API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.CreateTopicRequest;
import software.amazon.awssdk.services.sns.model.CreateTopicResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class CreateTopic {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <topicName>

                Where:
                   topicName - The name of the topic to create (for example,
 mytopic).

                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }
```

```
        String topicName = args[0];
        System.out.println("Creating a topic with name: " + topicName);
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        String arnVal = createSNSTopic(snsClient, topicName);
        System.out.println("The topic ARN is" + arnVal);
        snsClient.close();
    }

    public static String createSNSTopic(SnsClient snsClient, String topicName) {
        CreateTopicResponse result;
        try {
            CreateTopicRequest request = CreateTopicRequest.builder()
                    .name(topicName)
                    .build();

            result = snsClient.createTopic(request);
            return result.topicArn();

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
}
```

- Einzelheiten zur API finden Sie CreateTopicin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie den Client in einem separaten Modul und exportieren Sie ihn.

```
import { SNSClient } from "@aws-sdk/client-sns";

// The AWS Region can be provided here using the `region` property. If you leave
 it blank
// the SDK will default to the region set in your AWS config.
export const snsClient = new SNSClient({});
```

Importieren Sie das SDK- und Client-Module und rufen Sie die API auf.

```
import { CreateTopicCommand } from "@aws-sdk/client-sns";
import { snsClient } from "../libs/snsClient.js";

/**
 * @param {string} topicName - The name of the topic to create.
 */
export const createTopic = async (topicName = "TOPIC_NAME") => {
  const response = await snsClient.send(
    new CreateTopicCommand({ Name: topicName }),
  );
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: '087b8ad2-4593-50c4-a496-d7e90b82cf3e',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   TopicArn: 'arn:aws:sns:us-east-1:xxxxxxxxxxxx:TOPIC_NAME'
  // }
  return response;
};
```

- Weitere Informationen finden Sie im AWS SDK for JavaScript -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie CreateTopicin der AWS SDK for JavaScript API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

```
suspend fun createSNSTopic(topicName: String): String {

    val request = CreateTopicRequest {
        name = topicName
    }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.createTopic(request)
        return result.topicArn.toString()
    }
}
```

- API-Details finden Sie CreateTopicin der API-Referenz zum AWS SDK für Kotlin.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;
```

```
/**
 * Create a Simple Notification Service topics in your AWS account at the
 requested region.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/
guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$topicname = 'myTopic';

try {
    $result = $SnSclient->createTopic([
        'Name' => $topicname,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- Weitere Informationen finden Sie im AWS SDK for PHP -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie CreateTopicin der AWS SDK for PHP API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

```
class SnsWrapper:
    """Encapsulates Amazon SNS topic and subscription functions."""

    def __init__(self, sns_resource):
        """
        :param sns_resource: A Boto3 Amazon SNS resource.
        """
        self.sns_resource = sns_resource


    def create_topic(self, name):
        """
        Creates a notification topic.

        :param name: The name of the topic to create.
        :return: The newly created topic.
        """
        try:
            topic = self.sns_resource.create_topic(Name=name)
            logger.info("Created topic %s with ARN %s.", name, topic.arn)
        except ClientError:
            logger.exception("Couldn't create topic %s.", name)
            raise
        else:
            return topic
```

- Einzelheiten zur API finden Sie CreateTopicin AWS SDK for Python (Boto3) API Reference.

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu. GitHub Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

```
# This class demonstrates how to create an Amazon Simple Notification Service
 (SNS) topic.
class SNSTopicCreator
  # Initializes an SNS client.
  #
  # Utilizes the default AWS configuration for region and credentials.
  def initialize
    @sns_client = Aws::SNS::Client.new
  end

  # Attempts to create an SNS topic with the specified name.
  #
  # @param topic_name [String] The name of the SNS topic to create.
  # @return [Boolean] true if the topic was successfully created, false
 otherwise.
  def create_topic(topic_name)
    @sns_client.create_topic(name: topic_name)
    puts "The topic '#{topic_name}' was successfully created."
    true
  rescue Aws::SNS::Errors::ServiceError => e
    # Handles SNS service errors gracefully.
    puts "Error while creating the topic named '#{topic_name}': #{e.message}"
    false
  end
end

# Example usage:
if $PROGRAM_NAME == __FILE__
  topic_name = "YourTopicName" # Replace with your topic name
  sns_topic_creator = SNSTopicCreator.new

  puts "Creating the topic '#{topic_name}'..."
  unless sns_topic_creator.create_topic(topic_name)
    puts "The topic was not created. Stopping program."
    exit 1
  end
end
```

- Weitere Informationen finden Sie im AWS SDK for Ruby -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie CreateTopicin der AWS SDK for Ruby API-Referenz.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

```
async fn make_topic(client: &Client, topic_name: &str) -> Result<(), Error> {
    let resp = client.create_topic().name(topic_name).send().await?;

    println!(
        "Created topic with ARN: {}",
        resp.topic_arn().unwrap_or_default()
    );

    Ok(())
}
```

- Einzelheiten zur API finden Sie CreateTopicin der API-Referenz zum AWS SDK für Rust.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

```
    TRY.
        oo_result = lo_sns->createtopic( iv_name = iv_topic_name ). " oo_result
 is returned for testing purposes. "
        MESSAGE 'SNS topic created' TYPE 'I'.
      CATCH /aws1/cx_snstopiclimitexcdex.
```

```
        MESSAGE 'Unable to create more topics. You have reached the maximum
number of topics allowed.' TYPE 'E'.
   ENDTRY.
```

- Einzelheiten zur API finden Sie CreateTopicin der API-Referenz zum AWS SDK für SAP ABAP.

Abonnieren eines Amazon-SNS-Themas

Zum Abrufen der für ein Thema veröffentlichten Nachrichten müssen Sie einen Endpunkt für das Thema abonnieren. Wenn ein Endpunkt ein Thema abonniert, beginnt der Endpunkt, für das zugehörige Thema veröffentlichte Nachrichten zu empfangen.

> **Note**
>
> HTTP(S)-Endpunkte, E-Mail-Adressen und AWS-Ressourcen in anderen AWS-Konten benötigen eine Bestätigung des Abonnements, bevor sie Nachrichten empfangen können.

So richten Sie ein Endpunkt-Abonnement für ein Amazon-SNS-Thema ein

1. Melden Sie sich bei der Amazon-SNS-Konsole an.
2. Wählen Sie im linken Navigationsbereich Subscriptions (Abonnements).
3. Wählen Sie auf der Seite Subscriptions (Abonnements) die Option Create subscription (Abonnement erstellen)aus.
4. Führen Sie auf der Seite Create subscription (Abonnement erstellen) im Abschnitt Details die folgenden Schritte aus:

   a. Wählen Sie für Topic ARN den Amazon Resource Name (ARN) eines Themas. Dieser Wert ist der AWS-ARN, der generiert wurde, als Sie das Amazon-SNS-Thema erstellt haben. Zum Beispiel `arn:aws:sns:us-east-2:123456789012:your_topic`.

   b. Für Protocol wählen Sie einen Endpunkttypen. Folgende Endpunkttypen stehen zur Verfügung:

      - HTTP/HTTPS
      - E-Mail/E-Mail-JSON

- Amazon Data Firehose
- Amazon SQS

> **Note**
>
> Um ein SNS FIFO-Thema zu abonnieren, wählen Sie diese Option.

- AWS Lambda
- Plattformanwendungs-Endpunkt
- SMS

c. Für Endpoint, geben Sie den Endpunktwert ein, wie z. B. eine E-Mail-Adresse oder die ARN einer Amazon-SQS-Warteschlange.

d. Nur Firehose-Endpunkte: Geben Sie für Abonnementrollen-ARN den ARN der IAM-Rolle an, die Sie zum Schreiben in Firehose-Bereitstellungsdatenströme erstellt haben. Weitere Informationen finden Sie unter Voraussetzungen für das Abonnieren von Firehose-Bereitstellungsdatenströmen für Amazon SNS-Themen.

e. (Optional) Für Firehose, Amazon SQS, HTTP/S-Endpunkte können Sie auch die Übermittlung unformatierter Nachrichten aktivieren. Weitere Informationen finden Sie unter Übermittlung unformatierter Nachrichten Amazon SNS.

f. (Optional) Um eine Filterrichtlinie zu konfigurieren, erweitern Sie den Abschnitt Abonnement-Filterrichtlinie. Weitere Informationen finden Sie unter Filterrichtlinien für Amazon-SNS-Abonnements.

g. (Optional) Zur Aktivierung der nutzlastbasierten Filterung konfigurieren Sie `Filter Policy Scope` auf `MessageBody`. Weitere Informationen finden Sie unter Filterrichtlinien für Amazon-SNS-Abonnements – Geltungsbereich.

h. (Optional) Um eine Warteschlange für unzustellbare Nachrichten für das Abonnement zu konfigurieren, erweitern Sie den Abschnitt Redrive-Richtlinie (Warteschlange für unzustellbare Nachrichten). Weitere Informationen finden Sie unter Amazon SNS Queues für unzustellbare Nachrichten (DLQs).

i. Wählen Sie Create subscription (Abonnement erstellen) aus.

   Die Konsole erstellt das Abonnement und öffnet die Seite Details des Abonnements.

Löschen eines Amazon SNS SNS-Themas und -Abonnements

Wenn ein Thema gelöscht wird, werden die zugehörigen Abonnements asynchron gelöscht. Kunden können zwar weiterhin auf diese Abonnements zugreifen, die Abonnements sind jedoch nicht mehr mit dem Thema verknüpft — auch wenn Sie das Thema mit demselben Namen neu erstellen.

Wenn ein Subscriber versucht, eine Nachricht zu dem gelöschten Thema zu veröffentlichen, erhält der Publisher eine Fehlermeldung, die darauf hinweist, dass das Thema nicht vorhanden ist. Ebenso führt jeder Versuch, das gelöschte Thema zu abonnieren, zu einer Fehlermeldung.

Ein Abonnement, das noch nicht bestätigt ist, kann nicht gelöscht werden. Amazon SNS löscht nicht bestätigte Abonnement nach 48 Stunden automatisch.

Themen

- Um ein Amazon SNS SNS-Thema oder -Abonnement zu löschen, verwenden Sie den AWS Management Console
- So löschen Sie ein Amazon SNS-Abonnement und Amazon SNS-Thema mithilfe der AWS SDK

Um ein Amazon SNS SNS-Thema oder -Abonnement zu löschen, verwenden Sie den AWS Management Console

Um ein Thema mit dem zu löschen AWS Management Console

1. Melden Sie sich bei der Amazon-SNS-Konsole an.
2. Wählen Sie im linken Navigationsbereich Topics (Themen).
3. Wählen Sie auf der Seite Topics (Themen) ein Thema und anschließend Edit (Bearbeiten) aus.
4. Geben Sie im Dialogfenster Delete (Löschen) den Text `delete me` ein und wählen Sie dann Delete (Löschen) aus.

   In der Konsole wird das Thema gelöscht.

Um ein Abonnement mit dem zu löschen AWS Management Console

1. Melden Sie sich bei der Amazon-SNS-Konsole an.
2. Wählen Sie im linken Navigationsbereich Subscriptions (Abonnements).

3. Wählen Sie auf der Seite Abonnements ein Abonnement mit dem Status Bestätigt aus und klicken Sie dann auf Löschen.
4. Wählen Sie im Dialogfeld Delete subscription die Option Delete aus.

   Die Konsole löscht das Abonnement.

So löschen Sie ein Amazon SNS-Abonnement und Amazon SNS-Thema mithilfe der AWS SDK

Um ein AWS SDK verwenden zu können, müssen Sie es mit Ihren Anmeldeinformationen konfigurieren. Weitere Informationen finden Sie unter Freigegebene Konfigurations- und Anmeldeinformationsdateien im AWS -Referenzhandbuch zu SDKs und Tools.

Die folgenden Codebeispiele zeigen, wie Sie es verwenden`DeleteTopic`.

.NET

AWS SDK for .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

Löschen Sie ein Thema mit seinem Themen-ARN.

```
/// <summary>
/// Delete a topic by its topic ARN.
/// </summary>
/// <param name="topicArn">The ARN of the topic.</param>
/// <returns>True if successful.</returns>
public async Task<bool> DeleteTopicByArn(string topicArn)
{
    var deleteResponse = await _amazonSNSClient.DeleteTopicAsync(
        new DeleteTopicRequest()
        {
            TopicArn = topicArn
        });
```

```
        return deleteResponse.HttpStatusCode == HttpStatusCode.OK;
    }
```

- Einzelheiten zur API finden Sie DeleteTopicin der AWS SDK for .NET API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

```
//! Delete an Amazon Simple Notification Service (Amazon SNS) topic.
/*!
  \param topicARN: The Amazon Resource Name (ARN) for an Amazon SNS topic.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SNS::deleteTopic(const Aws::String &topicARN,
                              const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::SNS::Model::DeleteTopicRequest request;
    request.SetTopicArn(topicARN);

    const Aws::SNS::Model::DeleteTopicOutcome outcome =
 snsClient.DeleteTopic(request);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully deleted the Amazon SNS topic " << topicARN <<
 std::endl;
    }
    else {
        std::cerr << "Error deleting topic " << topicARN << ":" <<
                  outcome.GetError().GetMessage() << std::endl;
    }
```

```
    return outcome.IsSuccess();
}
```

- Einzelheiten zur API finden Sie DeleteTopicin der AWS SDK for C++ API-Referenz.

CLI

AWS CLI

So löschen Sie das SNS-Thema

Das folgende `delete-topic`-Beispiel löscht die angegebene SNS-Thema.

```
aws sns delete-topic \
    --topic-arn "arn:aws:sns:us-west-2:123456789012:my-topic"
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

- Einzelheiten zur API finden Sie DeleteTopicin der AWS CLI Befehlsreferenz.

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

```
// SnsActions encapsulates the Amazon Simple Notification Service (Amazon SNS)
 actions
// used in the examples.
type SnsActions struct {
 SnsClient *sns.Client
}
```

```
// DeleteTopic delete an Amazon SNS topic.
func (actor SnsActions) DeleteTopic(topicArn string) error {
 _, err := actor.SnsClient.DeleteTopic(context.TODO(), &sns.DeleteTopicInput{
  TopicArn: aws.String(topicArn)})
 if err != nil {
  log.Printf("Couldn't delete topic %v. Here's why: %v\n", topicArn, err)
 }
 return err
}
```

- Einzelheiten zur API finden Sie DeleteTopicin der AWS SDK for Go API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.DeleteTopicRequest;
import software.amazon.awssdk.services.sns.model.DeleteTopicResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DeleteTopic {
    public static void main(String[] args) {
```

```
        final String usage = """

                Usage:     <topicArn>

                Where:
                   topicArn - The ARN of the topic to delete.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String topicArn = args[0];
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        System.out.println("Deleting a topic with name: " + topicArn);
        deleteSNSTopic(snsClient, topicArn);
        snsClient.close();
    }

    public static void deleteSNSTopic(SnsClient snsClient, String topicArn) {
        try {
            DeleteTopicRequest request = DeleteTopicRequest.builder()
                    .topicArn(topicArn)
                    .build();

            DeleteTopicResponse result = snsClient.deleteTopic(request);
            System.out.println("\n\nStatus was " +
 result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Einzelheiten zur API finden Sie DeleteTopicin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie den Client in einem separaten Modul und exportieren Sie ihn.

```
import { SNSClient } from "@aws-sdk/client-sns";

// The AWS Region can be provided here using the `region` property. If you leave
 it blank
// the SDK will default to the region set in your AWS config.
export const snsClient = new SNSClient({});
```

Importieren Sie das SDK- und Client-Module und rufen Sie die API auf.

```
import { DeleteTopicCommand } from "@aws-sdk/client-sns";
import { snsClient } from "../libs/snsClient.js";

/**
 * @param {string} topicArn - The ARN of the topic to delete.
 */
export const deleteTopic = async (topicArn = "TOPIC_ARN") => {
  const response = await snsClient.send(
    new DeleteTopicCommand({ TopicArn: topicArn }),
  );
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: 'a10e2886-5a8f-5114-af36-75bd39498332',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   }
```

```
  // }
};
```

- Weitere Informationen finden Sie im AWS SDK for JavaScript -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie DeleteTopicin der AWS SDK for JavaScript API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

```
suspend fun deleteSNSTopic(topicArnVal: String) {

    val request = DeleteTopicRequest {
        topicArn = topicArnVal
    }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        snsClient.deleteTopic(request)
        println("$topicArnVal was successfully deleted.")
    }
}
```

- API-Details finden Sie DeleteTopicin der API-Referenz zum AWS SDK für Kotlin.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Deletes an SNS topic and all its subscriptions.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/
guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$topic = 'arn:aws:sns:us-east-1:111122223333:MyTopic';

try {
    $result = $SnSclient->deleteTopic([
        'TopicArn' => $topic,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- Einzelheiten zur API finden Sie DeleteTopicin der AWS SDK for PHP API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

```
class SnsWrapper:
    """Encapsulates Amazon SNS topic and subscription functions."""

    def __init__(self, sns_resource):
        """
        :param sns_resource: A Boto3 Amazon SNS resource.
        """
        self.sns_resource = sns_resource


    @staticmethod
    def delete_topic(topic):
        """
        Deletes a topic. All subscriptions to the topic are also deleted.
        """
        try:
            topic.delete()
            logger.info("Deleted topic %s.", topic.arn)
        except ClientError:
            logger.exception("Couldn't delete topic %s.", topic.arn)
            raise
```

- Einzelheiten zur API finden Sie DeleteTopicin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

```
TRY.
    lo_sns->deletetopic( iv_topicarn = iv_topic_arn ).
    MESSAGE 'SNS topic deleted.' TYPE 'I'.
  CATCH /aws1/cx_snsnotfoundexception.
    MESSAGE 'Topic does not exist.' TYPE 'E'.
ENDTRY.
```

- Einzelheiten zur API finden Sie DeleteTopicin der API-Referenz zum AWS SDK für SAP ABAP.

Markierung von Amazon-SNS-Themen

Amazon SNS unterstützt die Markierung von Amazon-SNS-Themen. Dank Markierung können Sie die mit Ihren Themen verbundenen Kosten nachverfolgen und verwalten, die Sicherheit Ihrer AWS Identity and Access Management (IAM)-Richtlinien erhöhen und tausende Themen problemlos durchsuchen oder filtern. Mit Markierungen können Sie Ihre Amazon-SNS-Themen mithilfe von AWS Resource Groups verwalten. Weitere Informationen zu Resource Groups finden Sie im Benutzerhandbuch zu AWS Resource Groups.

Themen

- Markierungen für die Kostenzuordnungen
- Markierungen für die Zugriffssteuerung
- Markierungen für die Suche und Filterung von Ressourcen
- Konfigurieren von Tags für ein Amazon-SNS-Thema

Markierungen für die Kostenzuordnungen

Zur Organisation und Identifizierung Ihrer Amazon-SNS-Themen für die Kostenzuordnung können Sie Tags hinzufügen, die den Zweck eines Themas identifizieren. Das ist vor allem dann nützlich, wenn Sie viele Themen haben. Organisieren Sie Ihre AWS-Rechnung mit Kostenzuordnungs-Tags, damit Sie Ihre eigene Kostenstruktur wiedergeben können. Melden Sie sich an, um Ihre AWS-Kontorechnung mit Tag-Schlüsseln und Werten zu erhalten. Weitere Informationen finden Sie unter Einrichten eines monatlichen Kostenzuordnungsberichts im Benutzerhandbuch zu AWS Billing and Cost Management.

Beispielsweise können Sie wie folgt Tags hinzufügen, die die Kostenstelle und den Zweck Ihrer Amazon-SNS-Topics repräsentieren.

Ressource	Schlüssel	Wert
Thema 1	Kostenstelle	43289
	Anwendung	Auftragsverarbeitung
Thema 2	Kostenstelle	43289
	Anwendung	Zahlungsverarbeitung
Thema 3	Kostenstelle	76585
	Anwendung	Archivierung

Dieses Markierungsschema ermöglicht es Ihnen, zwei Themen zu gruppieren, die verwandte Aufgaben für dieselbe Kostenstelle ausführen, während Sie eine unabhängige Aktivität mit einem anderen Tag für die Kostenzuordnung markieren.

Markierungen für die Zugriffssteuerung

AWS Identity and Access Management unterstützt die Steuerung des Zugriffs auf Ressourcen basierend auf Tags. Geben Sie nach der Markierung Ihrer Ressourcen Informationen über die Ressourcen-Tags im Bedingungselement einer Identity and Access Management (IAM)-Richtlinie an, um den tagbasierten Zugriff zu verwalten. Weitere Informationen darüber, wie Sie Ihre Ressourcen mit der Amazon-SNS-Konsole oder dem AWS-SDK markieren, finden Sie unter Konfigurieren von Tags.

Sie können den Zugriff auf eine IAM-Identität einschränken. Sie können z. B. den Zugriff von `Publish` und `PublishBatch` auf alle Amazon-SNS-Themen einschränken, die ein Tag mit dem Schlüssel `environment`und dem Wert `production` enthalten, während gleichzeitig Zugriff auf alle anderen Amazon-SNS-Themen gewährt wird. Im folgenden Beispiel schränkt die Richtlinie die Möglichkeit ein, Nachrichten zu Themen zu veröffentlichen, die mit `production` markiert sind, während Nachrichten zu Themen, die mit `development`gekennzeichnet sind, veröffentlicht werden können. Weitere Informationen finden Sie unter Zugriffssteuerung mit Tags im IAM-Benutzerhandbuch.

> **Note**
>
> Das Einrichten der IAM-Berechtigung für `Publish` legt die Berechtigung für sowohl für `Publish` als auch für `PublishBatch` fest.

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Deny",
    "Action": [
 "sns:Publish"
    ],
    "Resource": "arn:aws:sns:*:*:*",
    "Condition": {
      "StringEquals": {
        "aws:ResourceTag/environment": "production"
      }
    }
  },
  {
    "Effect": "Allow",
    "Action": [
      "sns:Publish"
    ],
    "Resource": "arn:aws:sns:*:*:*",
    "Condition": {
      "StringEquals": {
        "aws:ResourceTag/environment": "development"
      }
    }
  }]
```

```
}
```

Markierungen für die Suche und Filterung von Ressourcen

Ein AWS-Konto kann zehntausende Amazon-SNS-Themen enthalten (siehe Details in Amazon-SNS-Kontingente). Durch das Markieren Ihrer Themen erleichtern Sie sich das Durchsuchen oder Filtern von Themen.

Möglicherweise haben Sie z. B. hunderte Themen, die mit Ihrer Produktionsumgebung verknüpft sind. Anstatt manuell nach diesen Themen suchen zu müssen, können Sie alle Themen mit einem bestimmten Tag abfragen:

```
import com.amazonaws.services.resourcegroups.AWSResourceGroups;
import com.amazonaws.services.resourcegroups.AWSResourceGroupsClientBuilder;
import com.amazonaws.services.resourcegroups.model.QueryType;
import com.amazonaws.services.resourcegroups.model.ResourceQuery;
import com.amazonaws.services.resourcegroups.model.SearchResourcesRequest;
import com.amazonaws.services.resourcegroups.model.SearchResourcesResult;

public class Example {
    public static void main(String[] args) {
        // Query Amazon SNS Topics with tag "keyA" as "valueA"
        final String QUERY = "{\"ResourceTypeFilters\":[\"AWS::SNS::Topic\"],
\"TagFilters\":[{\"Key\":\"keyA\", \"Values\":[\"valueA\"]}]}";

        // Initialize ResourceGroup client
        AWSResourceGroups awsResourceGroups = AWSResourceGroupsClientBuilder
            .standard()
            .build();

        // Query all resources with certain tags from ResourceGroups
        SearchResourcesResult result = awsResourceGroups.searchResources(
            new SearchResourcesRequest().withResourceQuery(
                new ResourceQuery()
                .withType(QueryType.TAG_FILTERS_1_0)
                .withQuery(QUERY)
            ));
        System.out.println("SNS Topics with certain tags are " +
 result.getResourceIdentifiers());
    }
}
```

Konfigurieren von Tags für ein Amazon-SNS-Thema

Auf dieser Seite wird gezeigt, wie Sie das AWS Management Console, ein AWS SDK und die AWS CLI verwenden können, um Tags für ein Amazon SNS SNS-Thema zu konfigurieren.

> **Important**
>
> Fügen Sie keine personenbezogenen Daten (Personally Identifiable Information, PII) oder andere vertrauliche Informationen in Tags hinzu. Tags sind für andere Amazon Web Services zugänglich, einschließlich der Abrechnung. Tags sind nicht für private oder vertrauliche Daten gedacht.

Themen

Auflisten, Hinzufügen und Entfernen von Tags für ein Amazon SNS SNS-Thema mithilfe der AWS Management Console

1. Melden Sie sich bei der Amazon-SNS-Konsole an.
2. Wählen Sie im Navigationsbereich Topics (Themen) aus.
3. Wählen Sie auf der Seite Topics (Themen) ein Thema und anschließend Edit (Bearbeiten) aus.
4. Erweitern Sie den Abschnitt Tags.

   Die dem Thema hinzugefügten Tags werden aufgelistet.
5. Ändern von Themen-Tags:

   - Um ein Tag hinzuzufügen, wählen Sie Add tag (Tag hinzufügen) aus und geben Sie einen Key (Schlüssel) und einen Value (Wert) (optional) ein.
   - Zum Entfernen eines Tags wählen Sie neben einem Schlüssel-Wert-Paar Remove tag (Tag entfernen) aus.
6. Wählen Sie Save Changes (Änderungen speichern) aus.

Hinzufügen von Tags zu einem Thema über ein AWS -SDK

Um ein AWS SDK zu verwenden, müssen Sie es mit Ihren Anmeldeinformationen konfigurieren. Weitere Informationen finden Sie unter Freigegebene Konfigurations- und Anmeldeinformationsdateien im AWS -Referenzhandbuch zu SDKs und Tools.

Die folgenden Codebeispiele zeigen, wie Sie es verwenden`TagResource`.

CLI

AWS CLI

So fügen Sie einem Thema ein Tag hinzu

Das folgende `tag-resource`-Beispiel fügt dem angegebenen Amazon-SNS-Thema ein Metadaten-Tag hinzu.

```
aws sns tag-resource \
    --resource-arn arn:aws:sns:us-west-2:123456789012:MyTopic \
    --tags Key=Team,Value=Alpha
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

- Einzelheiten zur API finden Sie TagResourcein der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SnsException;
import software.amazon.awssdk.services.sns.model.Tag;
import software.amazon.awssdk.services.sns.model.TagResourceRequest;
import java.util.ArrayList;
import java.util.List;
```

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class AddTags {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <topicArn>

                Where:
                   topicArn - The ARN of the topic to which tags are added.

                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String topicArn = args[0];
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        addTopicTags(snsClient, topicArn);
        snsClient.close();
    }

    public static void addTopicTags(SnsClient snsClient, String topicArn) {
        try {
            Tag tag = Tag.builder()
                    .key("Team")
                    .value("Development")
                    .build();

            Tag tag2 = Tag.builder()
                    .key("Environment")
```

```
                    .value("Gamma")
                    .build();

            List<Tag> tagList = new ArrayList<>();
            tagList.add(tag);
            tagList.add(tag2);

            TagResourceRequest tagResourceRequest = TagResourceRequest.builder()
                    .resourceArn(topicArn)
                    .tags(tagList)
                    .build();

            snsClient.tagResource(tagResourceRequest);
            System.out.println("Tags have been added to " + topicArn);

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Einzelheiten zur API finden Sie TagResourcein der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

```
suspend fun addTopicTags(topicArn: String) {

    val tag = Tag {
        key = "Team"
        value = "Development"
    }
```

```
    val tag2 = Tag {
        key = "Environment"
        value = "Gamma"
    }

    val tagList = mutableListOf<Tag>()
    tagList.add(tag)
    tagList.add(tag2)

    val request = TagResourceRequest {
        resourceArn = topicArn
        tags = tagList
    }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        snsClient.tagResource(request)
        println("Tags have been added to $topicArn")
    }
}
```

- Einzelheiten zur API finden Sie TagResourcein der API-Referenz zum AWS SDK für Kotlin.

Verwalten von Tags mit SNS-API-Aktionen von Amazon

Um Tags über die Amazon-SNS-API zu verwalten, verwenden Sie die folgenden API-Aktionen:

- ListTagsForResource
- TagResource
- UntagResource

API-Aktionen, die ABAC unterstützen

Im Folgenden finden Sie eine Liste von API-Aktionen, die Attribute-based Access Control (ABAC, attributbasierte Zugriffssteuerung) unterstützen. Weitere Informationen zu ABAC finden Sie unter Wofür ist ABAC? AWS im IAM-Benutzerhandbuch.

- AddPermission
- ConfirmSubscription
- DeleteTopic

- GetDataProtectionPolicy
- GetSubscriptionAttributes
- GetTopicAttributes
- ListSubscriptionsByTopic
- ListTagsForResource
- Publish
- PublishBatch
- PutDataProtectionPolicy
- RemovePermission
- SetSubscriptionAttributes
- SetTopicAttributes
- Subscribe
- TagResource
- Unsubscribe
- UntagResource

Reihenfolge und Deduplizierung von Nachrichten (FIFO-Themen)

Sie können Amazon-SNS-FIFO-Themen (First in, First out) und Amazon-SQS-FIFO-Warteschlangen zusammen verwenden, um eine strikte Nachrichtenreihenfolge und Nachrichtendeduplizierung bereitzustellen. Die FIFO-Funktionen jeder dieser Dienste arbeiten zur Integration verteilter Anwendungen, die Datenkonsistenz in nahezu Echtzeit erfordern, zusammen, um als vollständig verwalteter Service zu fungieren. Wenn Sie Amazon-SQS-Standardwarteschlangen für Amazon-SNS-FIFO-Themen abonnieren, erhalten Sie bestmögliche Bestellungen und mindestens eine Lieferung.

Themen

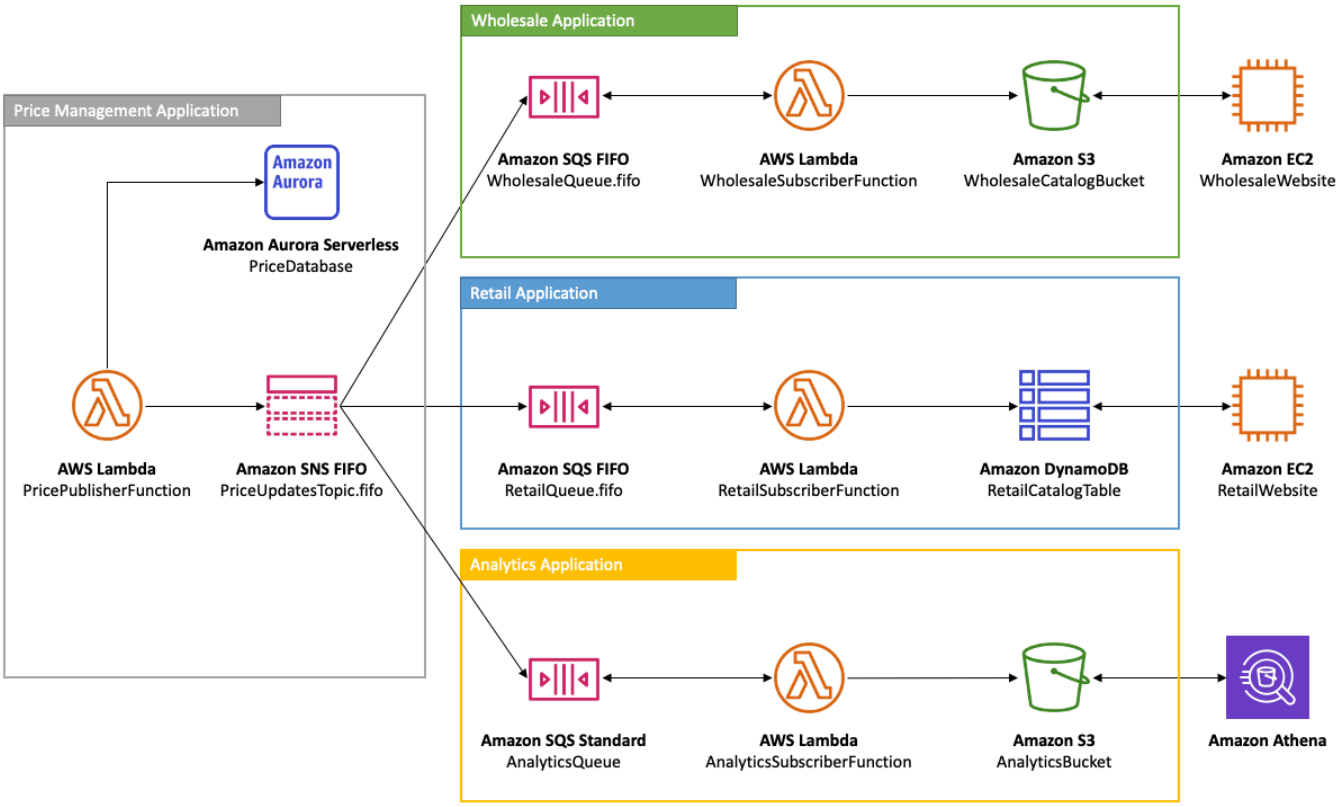
Anwendungsfall für FIFO-Themen

Im folgenden Beispiel wird eine E-Commerce-Plattform beschrieben, die von einem Hersteller von Autoteilen mithilfe von Amazon-SNS-FIFO-Themen und Amazon-SQS-Warteschlangen erstellt wurde. Die Plattform besteht aus drei Serverless-Anwendungen:

- Lagerverwalter verwenden eine Preisverwaltungsanwendung, um den Preis für jeden Artikel auf Lager festzulegen. Bei diesem Unternehmen können sich die Produktpreise aufgrund von Wechselkursschwankungen, Marktnachfrage und Verschiebungen in der Vertriebsstrategie ändern. Die Preismanagement-Anwendung verwendet eine AWS Lambda-Funktion, die

Preisaktualisierungen für ein Amazon SNS-FIFO-Thema veröffentlicht, wenn sich die Preise ändern.

- Eine Wholesale-Anwendung bietet das Backend für eine Website, wo Auto-Karosseriewerkstätten und Autohersteller die Autoteile des Unternehmens in großen Mengen kaufen können. Um Preisänderungsbenachrichtigungen zu erhalten, abonniert die Großhandels-Anwendung ihre Amazon-SQS-FIFO-Warteschlange für das Amazon-SNS-FIFO-Thema der Preisverwaltungsanwendung.
- Eine Einzelhandelsanwendung bietet das Backend für eine andere Website, auf der Autobesitzer und Auto-Tuning-Enthusiasten einzelne Autoteile für ihre Fahrzeuge kaufen können. Um Preisänderungsbenachrichtigungen zu erhalten, abonniert die Einzelhandelsanwendung ebenfalls auch ihre Amazon-SQS-FIFO-Warteschlange für das Amazon-SNS-FIFO-Thema der Preisverwaltungsanwendung.
- Eine Analyseanwendung, die Preisaktualisierungen zusammenfasst und in einem Amazon-S3-Bucket speichert, ermöglicht Amazon Athena, den Bucket für Business Intelligence (BI)-Zwecke abzufragen. Um Preisänderungsbenachrichtigungen zu erhalten, abonniert die Analyse-Anwendung ihre Amazon-SQS-Standard-Warteschlange für das Amazon-SNS-FIFO-Thema der Preisverwaltungsanwendung. Im Gegensatz zu den anderen Anwendungen erfordert die Analyse-Anwendung keine strikte Reihenfolge der Preisaktualisierungen.

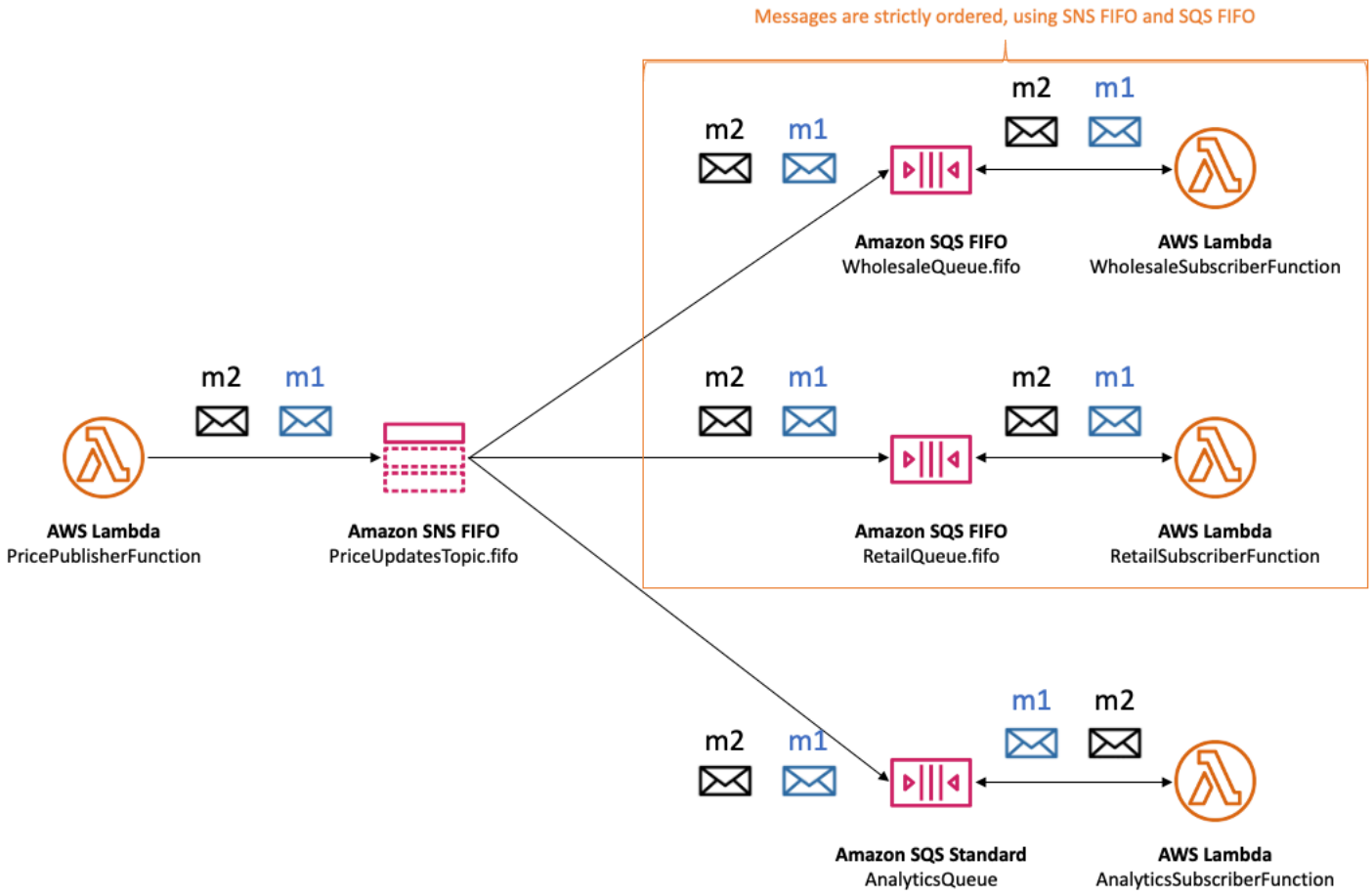


Damit die Groß- und Einzelhandelsanwendungen Preisaktualisierungen in der richtigen Reihenfolge erhalten, muss die Preisverwaltungsanwendung ein streng geordnetes Nachrichtenverteilungssystem verwenden. Die Verwendung von Amazon-SNS-FIFO-Themen und Amazon-SQS-FIFO-Warteschlangen ermöglicht die Verarbeitung von Nachrichten in der Reihenfolge und ohne Duplizierung. Weitere Informationen finden Sie unter Details zur Nachrichtenbestellung für FIFO-Themen. Code-Snippets, die diesen Anwendungsfall implementieren, finden Sie unterCodebeispiele für FIFO-Themen.

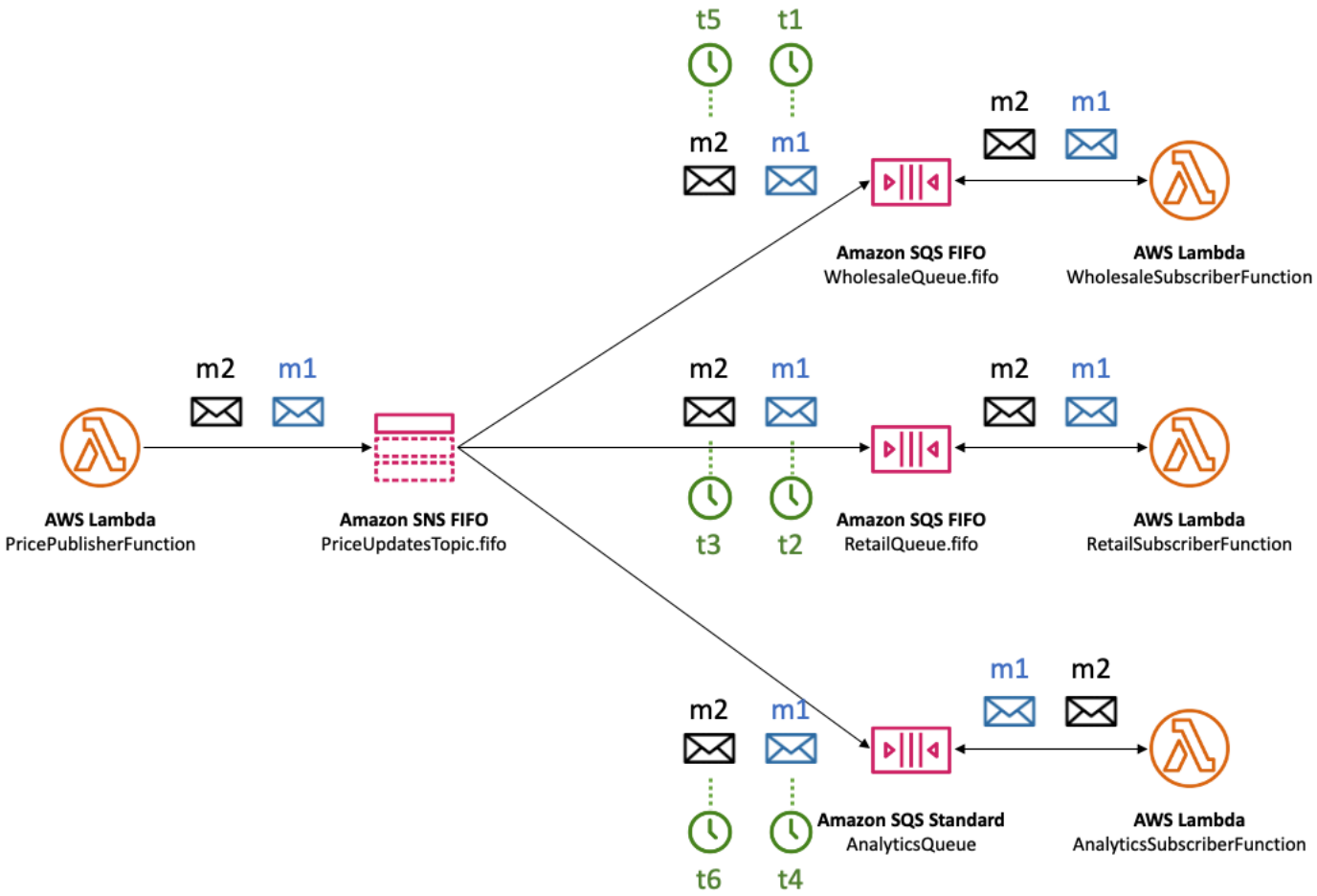
Details zur Nachrichtenbestellung für FIFO-Themen

In einem Amazon-SNS-FIFO-Thema werden Nachrichten an abonnierte Amazon-SQS-Warteschlangen in der genauen Reihenfolge übermittelt, in der die Nachrichten für das Thema veröffentlicht werden, und zwar nur einmal. Beim Abonnement einer Amazon-SQS-FIFO-Warteschlange erhält der Benutzer der Warteschlange die Nachrichten in der exakten Reihenfolge, in der die Nachrichten an die Warteschlange zugestellt werden, und ohne Duplikate. Wenn eine -Amazon-SQS-Standard-Warteschlange abonniert ist, kann es jedoch vorkommen, dass der Benutzer der Warteschlange Nachrichten nicht in der richtigen Reihenfolge erhält, und dazu mehr als einmal. Dies ermöglicht eine weitere Entkopplung der Abonnenten von den Herausgebern, was den Abonnenten mehr Flexibilität in Bezug auf die Nachrichtennutzung und die Kostenoptimierung bietet,

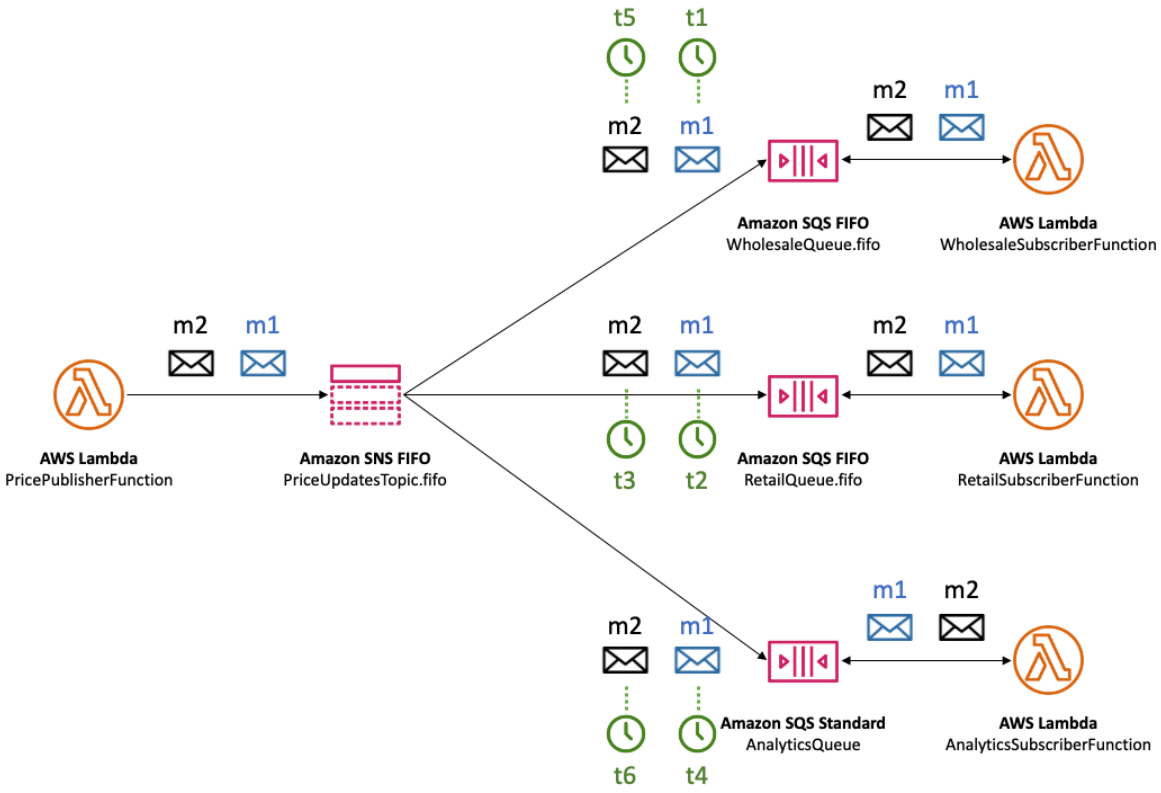
wie das folgende Diagramm zeigt, das auf der Grundlage von Anwendungsfall für FIFO-Themen erstellt wurde.



Beachten Sie, dass es keine implizite Bestellung der Abonnenten gibt. Das folgende Beispiel zeigt, dass die Nachricht m1 zuerst an den Großhandels-Abonnenten, dann an den Einzelhandel- und schließlich an den Analyse-Abonnenten geht. Nachricht m2 wird zuerst an den Einzelhandels, dann an den Großhandels- und schließlich an den Analyse-Abonnenten zugestellt. Obwohl die beiden Nachrichten an die Abonnenten in einer anderen Reihenfolge zugestellt werden, wird die Nachrichtenreihenfolge für jeden Amazon-SQS-FIFO-Abonnenten beibehalten. Jeder Teilnehmer wird isoliert von anderen Abonnenten wahrgenommen.

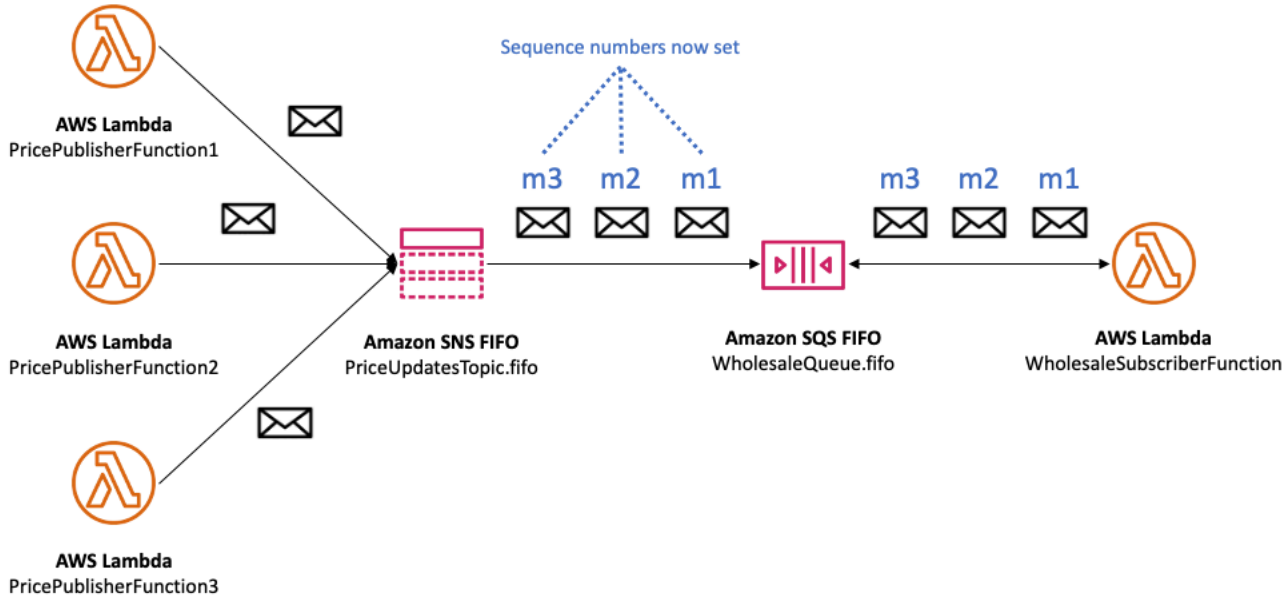


Wenn ein Amazon-SQS-Warteschlangenabonnent nicht erreichbar ist, ist er nicht mehr synchronisiert. Angenommen, der Besitzer der Wholesale-AnwendungsQueue fälschlicherweise ändert dieAmazon-SQS-Warteschlangenrichtlinieauf eine Weise, die verhindert, dass der Amazon-SNS-Serviceprinzipal Nachrichten an die Queue sendet. In diesem Fall schlagen Preisaktualisierungen an die Großhandels-Warteschlange fehl, während die Lieferungen an die Einzelhandels- und die Analyse-Warteschlangen erfolgreich sind, was dazu führt, dass die Abonnenten nicht synchron sind. Wenn der Besitzer der Warteschlange der Großhandelsanwendung die Warteschlangenrichtlinie ändert, setzt Amazon SNS das Zustellen der Nachrichten an die abonnierte Warteschlange fort. Alle Nachrichten, die für das Thema veröffentlicht wurden und auf die falsch konfigurierte Warteschlange abzielen, werden gelöscht, es sei denn, für das Abonnement ist eine Warteschlange für unzustellbare Nachrichten konfiguriert.



Sie können mehrere Anwendungen (oder mehrere Threads innerhalb derselben Anwendung) gleichzeitig Nachrichten in einem SNS-FIFO-Thema veröffentlichen. Wenn Sie dies tun, delegieren Sie die Nachrichtensequenzierung effektiv an den Amazon SNS-Service. Um die etablierte Sequenz von Nachrichten zu bestimmen, können Sie die Sequenznummer überprüfen.

Die Sequenznummer ist eine große, nicht fortlaufende Zahl, die Amazon SNS jeder Nachricht zuweist. Die Länge der Sequenznummer beträgt 128 Bit und nimmt für jede Nachrichtengruppe weiter zu. Die Sequenznummer wird an die abonnierten Amazon-SQS-Warteschlangen als Teil des Nachrichtentextes übergeben. Wenn Sie jedoch Rohnachrichtenzustellung aktivieren, enthält die Nachricht, die an die Amazon-SQS-Warteschlange übermittelt wird, weder die Sequenznummer noch andere Amazon-SNS-Nachrichtenmetadaten.



Amazon SNS FIFO-Themen definieren die Reihenfolge im Kontext einer Nachrichtengruppe. Weitere Informationen finden Sie unter Nachrichtengruppierung für FIFO-Themen.

Nachrichtengruppierung für FIFO-Themen

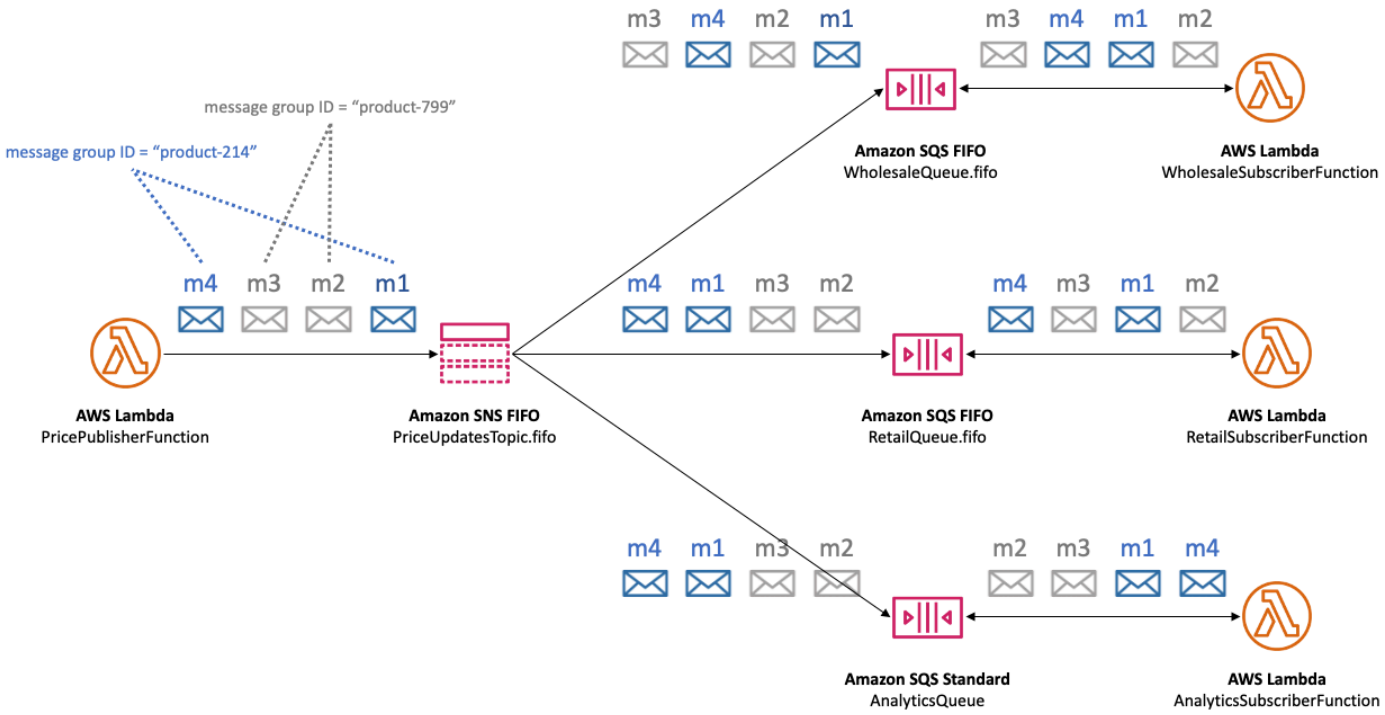
Nachrichten, die derselben Gruppe angehören, werden nacheinander in einer strengen Reihenfolge in Bezug auf die Gruppe verarbeitet.

Wenn Sie Nachrichten in einem Amazon SNS FIFO-Thema veröffentlichen, legen Sie die Nachrichtengruppen-ID fest. Die Gruppen-ID ist ein obligatorisches Token, das angibt, dass eine Nachricht zu einer bestimmten Nachrichtengruppe gehört. Das SNS-FIFO-Thema übergibt die Gruppen-ID an die abonnierten Amazon SQS FIFO-Queues. Es gibt keine Begrenzung für die Anzahl der Gruppen-IDs in SNS-FIFO-Themen oder SQS-FIFO-Queues. Die Nachrichtengruppen-ID wird nicht an Amazon-SQS-Standard-Warteschlangen weitergegeben.

Es gibt keine Affinität zwischen einer Nachrichtengruppe und einem Abonnement. Daher werden Nachrichten, die für eine Nachrichtengruppe veröffentlicht werden, an alle abonnierten Queues übermittelt, abhängig von den Filterrichtlinien, die mit Abonnements verknüpft sind. Weitere Informationen finden Sie unter Nachrichtenzustellung für FIFO-Themen und Nachrichtenfilterung für FIFO-Themen.

In Anwendungsbeispiel des Autoteile-Preis-Management gibt es für jedes Produkt, das auf der Plattform verkauft wird, eine dedizierte Nachrichtengruppe. Das gleiche Amazon-SNS-FIFO-Thema wird für die Verarbeitung aller Preisaktualisierungen verwendet. Die Reihenfolge der

Preisaktualisierungen bleibt im Kontext eines einzelnen Autoteile-Produkts erhalten, aber nicht über mehrere Produkte hinweg. In der folgenden Abbildung ist diese Funktionsweise dargestellt. Beachten Sie, dass für das Produkt mit der Nachrichtengruppen-ID produkt-214 die Nachricht m1 immer vor der Nachricht m4 verarbeitet wird. Diese Sequenz wird während des gesamten Workflows beibehalten, von Amazon-SNS-FIFO bis zu Amazon-SQS-FIFO. Ebenso wird für das Produkt, dessen Nachrichtengruppen-ID produkt-799 ist, die Nachricht m2 vor der Nachricht m3 verarbeitet, sofern die Workflows Amazon-SNS-FIFO und Amazon-SQS-FIFO verwenden. Bei Verwendung von Amazon-SQS-Standard-Warteschlangen ist die Nachrichtenreihenfolge jedoch nicht mehr garantiert, und Nachrichtengruppen existieren nicht. Die produkt-214 und produkt-799-Nachrichtengruppen sind unabhängig voneinander, so dass es keine Beziehung zwischen der Sequenzierung ihrer Nachrichten gibt.



Verteilen von Daten nach Nachrichtengruppen-IDs zur Verbesserung der Leistung

Um den Zustellungsdurchsatz zu optimieren, stellen Amazon-SNS-FIFO-Themen Nachrichten aus verschiedenen Nachrichtengruppen parallel zu, wobei die Nachrichtenreihenfolge innerhalb der einzelnen Nachrichtengruppen strikt eingehalten wird. Jede einzelne Nachrichtengruppe kann maximal 300 Nachrichten pro Sekunde übermitteln. Um einen hohen Durchsatz für ein einzelnes

Thema zu erreichen, sollten Sie daher eine große Anzahl unterschiedlicher Nachrichtengruppen-IDs verwenden. Durch die Verwendung einer Vielzahl von Nachrichtengruppen verteilen Amazon-SNS-FIFO-Themen Nachrichten automatisch auf eine größere Anzahl parallel Partitionen.

> **Note**
>
> Amazon-SNS-FIFO-Themen sind für die gleichmäßige Verteilung von Nachrichten auf Nachrichtengruppen-IDs optimiert, unabhängig von der Anzahl der Gruppen. AWS empfiehlt die Verwendung einer großen Anzahl unterschiedlicher Nachrichtengruppen-IDs, um die Leistung zu optimieren.

Wenn Sie in Ihrem Amazon-SNS-FIFO-Thema mit hohem Durchsatz veröffentlichen und mindestens eine Amazon-SQS-FIFO-Warteschlange abonniert ist, wird empfohlen, den hohen Durchsatz für Ihre Warteschlangen zu aktivieren. Weitere Informationen finden Sie unter Hoher Durchsatz für FIFO-Warteschlangen im Entwicklerhandbuch zu Amazon Simple Queue Service.

Nachrichtenzustellung für FIFO-Themen

Amazon-SNS-FIFO-Themen (First in, First out) unterstützen die Bereitstellung sowohl an Amazon-SQS-Standard- als auch an FIFO-Warteschlangen, um Kunden Flexibilität und Kontrolle bei der Integration verteilter Anwendungen zu bieten, die Datenkonsistenz nahezu in Echtzeit erfordern.

Für Workloads, die eine strikte Nachrichtenreihenfolge oder Deduplizierung beibehalten müssen, bietet die Kombination von Amazon-SNS-FIFO-Themen mit Amazon-SQS-FIFO-Warteschlangen, die als Zustellungsendpunkt abonniert werden, eine bessere Nachrichtenübermittlung zwischen Anwendungen, wenn die Reihenfolge der Vorgänge und Ereignisse wichtig ist, oder wenn Duplikate nicht toleriert werden können.

Für Workloads, die eine Bestellung nach bestem Aufwand und eine mindestens einmalige Lieferung tolerieren, bietet das Abonnieren von Amazon-SQS-Standard-Warteschlangen für Amazon-SNS-FIFO-Themen die Möglichkeit, Kosten zu senken und Warteschlangen für Workloads, die kein FIFO verwenden, gemeinsam zu nutzen.

> **Note**
>
> Um Nachrichten von Amazon SNS FIFO-Themen an AWS Lambda-Funktionen zu verteilen, sind zusätzliche Schritte erforderlich. Abonnieren Sie zunächst Amazon-SQS-FIFO-Warteschlangen für das Thema. Konfigurieren Sie dann die Queues, um die Funktionen

auszulösen. Weitere Informationen finden Sie unter SQS FIFO als Ereignisquelle im AWSCompute Blog.

SNS-FIFO-Themen können keine Nachrichten an vom Kunden verwaltete Endpunkte wie E-Mail-Adressen, mobile Apps, Telefonnummern für SMS oder HTTP(S)-Endpunkte übermitteln. Bei diese Endpunkttypen ist nicht garantiert, dass die strikte Nachrichtenreihenfolge beibehalten wird. Versuche, vom Kunden verwaltete Endpunkte für SNS-FIFO-Themen zu abonnieren, führen zu Fehlern.

SNS-FIFO-Themen unterstützen dieselben Nachrichtenfilterfunktionen wie Standardthemen. Weitere Informationen finden Sie unter Nachrichtenfilterung für FIFO-Themenund Vereinfachen Sie Ihr Pub/Sub-Messaging mit Amazon SNS NachrichtenfilterungVeröffentlichen im AWSCompute Blog.

Nachrichtenfilterung für FIFO-Themen

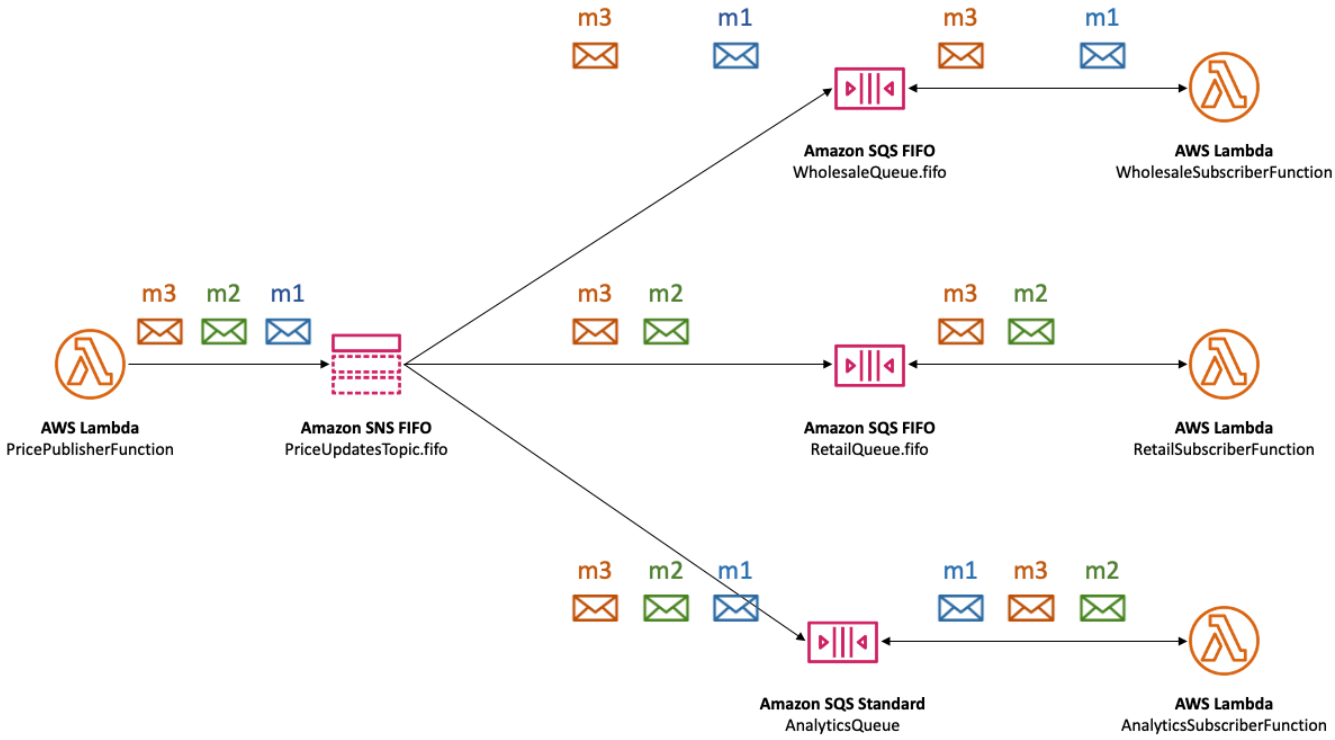
Amazon SNS FIFO-Themen unterstützen die Nachrichtenfilterung. Die Verwendung von Nachrichtenfiltern vereinfacht die Architektur, indem die Abonnenten von der Nachrichten-Routing-Logik von den Publisher-Systemen und die Abonnenten von der Nachrichtenfilterlogik befreit werden.

Wenn Sie eine Amazon-SQS-FIFO- oder Standard-Warteschlange für ein SNS-FIFO-Thema abonnieren, können Sie mithilfe der Nachrichtenfilterung angeben, dass der Abonnent anstelle aller Nachrichten eine Teilmenge empfängt. Jeder Abonnent kann seine eigene Filterrichtlinie als Abonnementattribute festlegen. Basierend auf dem Geltungsbereich der Filterrichtlinie wird diese mit den Attributen oder dem Nachrichtentext abgeglichen. Wenn die Filterrichtlinie übereinstimmt, übermittelt das Thema dem Abonnenten eine Kopie der Nachricht. Wenn keine Übereinstimmung vorhanden ist, liefert das Thema keine Kopie der Nachricht.

Im Beispiel für das Preismanagement von Autoteilen wird davon ausgegangen, dass die folgenden Amazon-SNS-Filterrichtlinien festgelegt sind und der Geltungsbereich der Filterrichtlinie `MessageBody` ist:

- Bei der Großhandelswarteschlange stimmt die Filterrichtlinie `{"business":["wholesale"]}` mit jeder Nachricht überein, die einen Schlüssel namens `business` und die Zeichenfolge `wholesale` im Wertesatz enthält. Im folgenden Diagramm ist einer der Schlüssel in der Nachricht m1 `business` mit dem Wert `wholesale`. Einer der Schlüssel in Nachricht m3 ist `business` mit dem Wert `["wholesale,retail"]`. Daher gilt:m1 und m3 entsprechen den Kriterien der Filterrichtlinie, und beide Nachrichten werden an die Großhandelsqueue übermittelt.

- Für die Einzelhandelswarteschlange stimmt die Filterrichtlinie `{"business":["retail"]}` mit jeder Nachricht überein, die einen Schlüssel namens `business` und die Zeichenfolge `retail` im Wertesatz enthält. Im Diagramm ist einer der Schlüssel in der Nachricht m2 `business` mit dem Wert `retail`. Einer der Schlüssel in Nachricht m3 ist `business` mit dem Wert `["wholesale,retail"]`. Daher gilt:m2 und m3 den Kriterien der Filterrichtlinie entsprechen, und beide Nachrichten werden an die Einzelhandelswarteschlange übermittelt.
- Für die Analyse-Warteschlange möchten wir, dass Amazon Athena alle Datensätze empfängt, so dass keine Filterrichtlinie angewendet wird.



SNS-FIFO-Themen unterstützen eine Vielzahl von übereinstimmenden Operatoren, einschließlich Attributzeichenfolgenwerte, numerische Attributwerte und Attributschlüssel. Weitere Informationen finden Sie unter Amazon SNS Nachrichtenfilterung.

SNS-FIFO-Themen liefern keine doppelten Nachrichten an abonnierte Endpunkte. Weitere Informationen finden Sie unter Nachrichtendeduplizierung für FIFO-Themen.

Nachrichtendeduplizierung für FIFO-Themen

Amazon SNS FIFO-Themen und Amazon SQS FIFO-Queues unterstützen die Nachrichtendeduplizierung, die eine exakte Nachrichtenübermittlung und -verarbeitung ermöglicht, solange die folgenden Bedingungen erfüllt sind:

- Die abonnierte Amazon-SQS-FIFO-Warteschlange ist vorhanden und verfügt über Berechtigungen, die es dem Amazon-SNS-Service-Prinzipal ermöglichen, Nachrichten an die Warteschlange zu senden.
- Der Nutzer der Amazon-SQS-FIFO-Warteschlange verarbeitet die Nachricht und löscht sie aus der Warteschlange, bevor das Zeitlimit für die Sichtbarkeit abläuft.
- Das Thema Amazon SNS Abonnement“ enthält keine Nachrichtenfilterung. Wenn Sie die Nachrichtenfilterung konfigurieren, unterstützen Amazon SNS FIFO-Themen die at-most-once Zustellung, da Nachrichten auf der Grundlage Ihrer Abonnementfilterrichtlinien herausgefiltert werden können.
- Es gibt keine Netzwerkunterbrechungen, die die Bestätigung der Nachrichtenzustellung verhindern.

> **Note**
>
> Die Nachrichtendeduplizierung gilt für ein gesamtes Amazon-SNS-FIFO-Thema, nicht für eine einzelne Nachrichtengruppe.

Wenn Sie eine Nachricht in einem Amazon-SNS-FIFO-Thema veröffentlichen, muss die Nachricht eine Deduplizierungskennung enthalten. Diese Kennung ist in der Nachricht enthalten, die das Amazon-SNS-FIFO-Thema an die abonnierten Amazon-SQS-FIFO-Warteschlangen übermittelt.

Wenn eine Nachricht mit einer bestimmten Deduplizierungs-ID erfolgreich in einem Amazon-SNS-FIFO-Thema veröffentlicht wurde, werden alle Nachrichten, die mit derselben Deduplizierungs-ID innerhalb des fünfminütigen Deduplizierungsintervalls veröffentlicht wurden, akzeptiert, aber nicht zugestellt. Das Amazon-SNS-FIFO-Thema verfolgt weiterhin die Nachrichtendeduplizierungs-ID, auch nachdem die Nachricht an abonnierte Endpunkte übermittelt wurde.

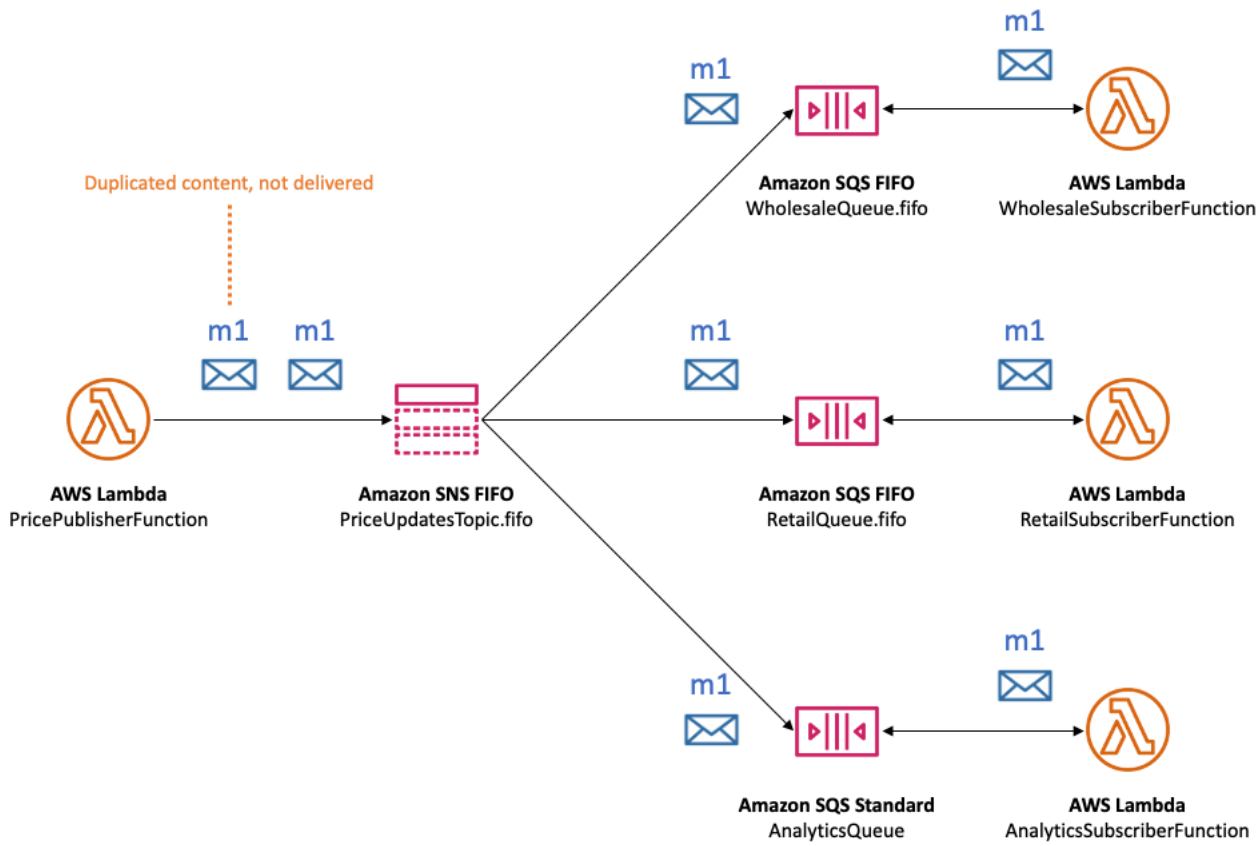
Wenn der Nachrichtentext für jede veröffentlichte Nachricht garantiert eindeutig ist, können Sie die inhaltsbasierte Deduplizierung für ein Amazon SNS FIFO-Thema und die abonnierten Amazon-SQS-FIFO-Warteschlangen aktivieren. Amazon SNS verwendet den Nachrichtentext, um einen

eindeutigen Hashwert zu generieren, der als Deduplizierungs-ID für jede Nachricht verwendet wird. Daher müssen Sie beim Senden jeder Nachricht keine Deduplizierungs-ID festlegen.

> **Note**
>
> Nachrichtenattribute sind nicht in der Hashberechnung enthalten.

Wenn die inhaltsbasierte Deduplizierung für ein Amazon SNS FIFO-Thema aktiviert ist und eine Nachricht mit einer Deduplizierungs-ID veröffentlicht wird, überschreibt die veröffentlichte Deduplizierungs-ID die generierte inhaltsbasierte Deduplizierungs-ID.

Im Beispiel des Autoteile-Preismanagement muss das Unternehmen für jede Preisaktualisierung eine universell eindeutige Deduplizierungs-ID festlegen. Dies liegt daran, dass der Nachrichtentext identisch sein kann, selbst wenn sich das Nachrichtenattribut für Groß- und Einzelhandel unterscheidet. Wenn das Unternehmen jedoch den Unternehmenstyp (Groß- oder Einzelhandel) neben der Produkt-ID und dem Produktpreis dem Nachrichtentext hinzugefügt hat, könnte es eine inhaltsbasierte Duplizierung im Amazon-SNS-FIFO-Thema und den abonnierten Amazon-SQS-FIFO-Warteschlangen ermöglichen.



Zusätzlich zur Reihenfolge und Deduplizierung von Nachrichten unterstützen Amazon SNS FIFO-Themen die serverseitige Nachrichtenverschlüsselung (SSE) mit AWS KMS Schlüsseln und den Nachrichtenschutz über VPC-Endpunkte mit. AWS PrivateLink Weitere Informationen finden Sie unter Nachrichtensicherheit für FIFO-Themen.

Nachrichtensicherheit für FIFO-Themen

Sie können festlegen, dass Amazon SNS und Amazon SQS Nachrichten verschlüsseln, die an FIFO-Themen und -Queues gesendet werden, indem Sie AWS Key Management Service (AWS KMS) Kundenmasterschlüssel (CMKs) auswählen. Sie können verschlüsselte FIFO-Themen und -Queues erstellen oder vorhandene FIFO-Themen und -Queues verschlüsseln. Amazon SNS und Amazon SQS verschlüsseln nur den Nachrichtentext. Sie verschlüsseln die Nachrichtenattribute, Ressourcenmetriken oder Ressourcenmetriken nicht.

> **Note**
>
> Durch das Hinzufügen von Verschlüsselung zu einem vorhandenen FIFO-Thema oder einer vorhandenen Queue werden keine nachgelagerten Nachrichten verschlüsselt. Durch das

Entfernen der Verschlüsselung aus einem Thema oder einer Queue werden nachgelagerte Nachrichten verschlüsselt.

SNS-FIFO-Themen entschlüsseln die Nachrichten unmittelbar vor der Zustellung an abonnierte Endpunkte. SQS FIFO-Queues entschlüsseln die Nachricht, kurz bevor sie an die Verbraucheranwendung zurückgesendet werden. Weitere Informationen finden Sie unter Datenverschlüsselungund Verschlüsseln von auf Amazon SNS veröffentlichten Nachrichten mit AWS KMSVeröffentlichen auf dem AWS Compute Blogsaus.

Darüber hinaus unterstützen SNS FIFO-Themen und SQS FIFO-Queues den Datenschutz bei Schnittstellen-VPC Endpunktepowered by AWS PrivateLink aus. Mithilfe von Schnittstellenendpunkten können Sie Nachrichten von Amazon Virtual Private Cloud (Amazon VPC)-Subnetzen an FIFO-Themen und -Queues senden, ohne das öffentliche Internet zu durchlaufen. Dieses Modell hält Ihr Messaging innerhalb der AWS-Infrastruktur und -Netzwerk, die die allgemeine Sicherheit Ihrer Anwendung erhöht. Wenn Sie AWS PrivateLink nutzen brauchen Sie kein Internet-Gateway, eine Network Address Translation (NAT) oder ein Virtual Private Network (VPN) einrichten. Weitere Informationen finden Sie unter Richtlinie für den Datenverkehr zwischen Netzwerken und Schutz von auf Amazon SNS veröffentlichten Nachrichten mit AWS PrivateLink im AWSSicherheitsblog.

SNS-FIFO-Themen unterstützen auch Queues für unzustellbare Nachrichten und Nachrichtenspeicher über Availability Zones hinweg. Weitere Informationen finden Sie unter Speicherdauer der Nachrichten für FIFO-Themen.

Speicherdauer der Nachrichten für FIFO-Themen

Amazon-SNS-FIFO-Themen und Amazon-SQS-Warteschlangen sind dauerhaft. Beide Ressourcentypen speichern Nachrichten redundant über mehrere Availability Zones hinweg und stellen Queues für Dead-Letter bereit, um Ausnahmefälle zu bearbeiten.

In Amazon SNS schlägt die Nachrichtenzustellung fehl, wenn aufgrund eines clientseitigen oder serverseitigen Fehlers nicht auf eine abonnierte Amazon SQS Queue zugreifen kann:

- Client-seitige Fehler treten auf, wenn das Amazon-SNS-FIFO-Thema über veraltete Abonnementmetadaten verfügt. Zwei häufige Fälle von clientseitigen Fehlern treten auf, wenn der Eigentümer der Amazon-SQS-FIFO-Warteschlange eine der folgenden Aktionen ausführt:
  - Es löscht die Queue.

- Ändert die Queuesrichtlinie so, dass der Amazon SNS Dienstprinzipal daran hindert, Nachrichten an ihn zu senden.

  Amazon SNS versucht nicht erneut, Nachrichten zuzustellen, die aufgrund von clientseitigen Fehlern fehlgeschlagen sind.

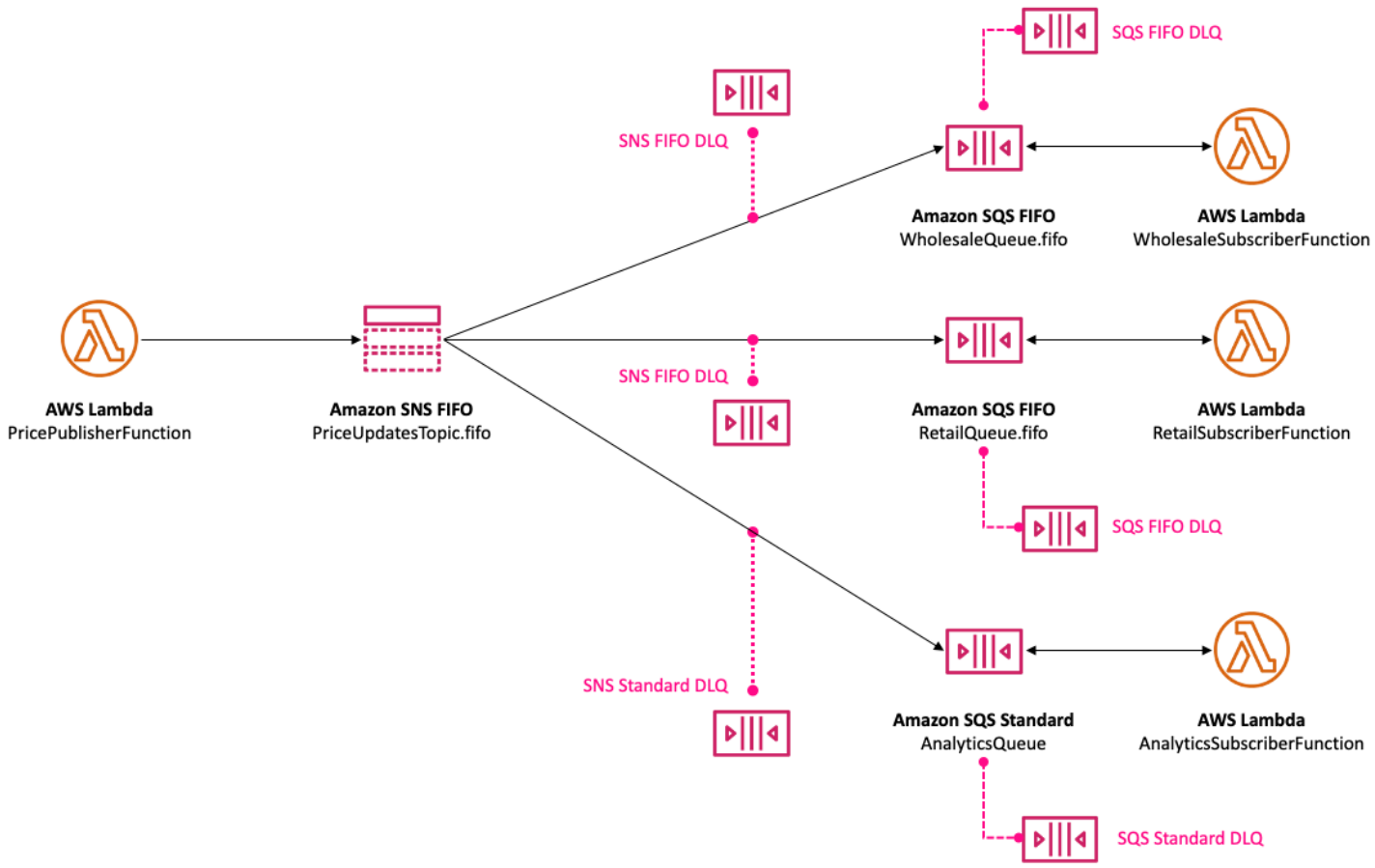
- In diesen Situationen können serverseitige Fehler auftreten:
  - Der Amazon SQS Dienst ist nicht verfügbar.
  - Amazon SQS verarbeitet keine gültige Anforderung des Amazon SNS-Service.

  Bei serverseitigen Fehlern versuchen die Amazon-SNS-FIFO-Themen, die fehlgeschlagenen Zustellungen über 23 Tage hinweg bis zu 100.015 mal zu wiederholen. Weitere Informationen finden Sie unter Wiederholungsversuche bei der Nachrichtenzustellung Amazon SNS.

Bei jeder Art von Fehler kann Amazon SNS Nachrichten an Amazon SQS Unzustellbare Nachrichten Queues abstellen, sodass Daten nicht verloren gehen.

In Amazon SQS schlägt die Nachrichtenverarbeitung fehl, wenn die Verbraucheranwendung die Nachricht nicht empfängt, verarbeitet und aus der Queue gelöscht wird. Wenn die maximale Anzahl von Empfangsanforderungen fehlschlägt, kann Amazon SQS Nachrichten an Queues mit Dead-Lettern abstellen, sodass Daten nicht verloren gehen.

Im Beispiel für das Preismanagement von Autoteilen kann das Unternehmen jedem Amazon-SNS-FIFO-Themenabonnement sowie jeder abonnierten Amazon-SQS-Warteschlange eine Amazon-SQS-Warteschlange für unzustellbare Nachrichten (DLQ) zuweisen. Dies schützt das Unternehmen vor jeglichen Preisaktualisierungsverlusten.



Bei der Warteschlange für unzustellbare Nachrichten (DLQ) für ein Amazon-SNS-Abonnement muss es sich um eine Amazon-SQS-Warteschlange handeln, die dem Typ der abonnierenden Warteschlange entspricht. Zum Beispiel muss ein Amazon-SNS-FIFO-Abonnement für eine Amazon-SQS-FIFO-Warteschlange eine Amazon-SQS-FIFO-Warteschlange für unzustellbare Nachrichten haben. Zum Beispiel muss ein Amazon-SNS-FIFO-Abonnement für eine Amazon-SQS-Standard-Warteschlange eine Amazon-SQS-Standard-Warteschlange für unzustellbare Nachrichten haben. Weitere Informationen finden Sie unter Amazon SNS Queues für unzustellbare Nachrichten (DLQs) und Entwickeln von dauerhaften serverless Apps mit DLQs für Amazon SNS, Amazon SQS, AWS Lambda Veröffentlichen im AWS Compute Blog.

Für eine längere Lebensdauer zur Unterstützung der Wiederherstellung nach Ausfällen nachgelagerten Anwendungen können Themenbesitzer auch FIFO-Themen verwenden, um Nachrichten bis zu 365 Tage lang zu archivieren. Themen-Subscriber können diese Nachrichten an einem abonnierten Endpunkt wiederholen, um Nachrichten wiederherzustellen, die auf einen Fehler in einer nachgelagerten Anwendung zurückzuführen sind, oder um den Status einer vorhandenen Anwendung zu replizieren. Weitere Informationen finden Sie unter Archivierung und Wiederholung von Nachrichten für FIFO-Themen.

Archivierung und Wiederholung von Nachrichten für FIFO-Themen

Themen

Was ist die Nachrichtenarchivierung und -wiederholung?

Die Nachrichtenarchivierung und -wiederholung von Amazon SNS ist ein integriertes Nachrichtenarchiv ohne Code, mit dem Themenbesitzer Nachrichten innerhalb ihres Themas speichern (oder archivieren) können. Themen-Suscriber können archivierte Nachrichten zurück an einen abonnierten Endpunkt abrufen (oder wiederholen), um Folgendes zu erreichen:

- Nachrichten wiederherstellen, die möglicherweise aufgrund eines Fehlers in einer nachgeschalteten Anwendung verloren gegangen sind.
- Den Status einer vorhandenen Anwendung auf eine neue Anwendung replizieren, indem Sie den neuen Endpunkt abonnieren und den gewünschten Zeitstempel für die Replikation auswählen.

Sie können die Nachrichtenarchivierung und -wiederholung mit der AWS-API, dem SDK, AWS CloudFormation und der AWS Management Console verwenden.

> **Note**
>
> Die Amazon SNS Nachrichtenarchivierung und -wiederholung ist nur für Application-to-Application (A2A) FIFO-Themen verfügbar.

Die Nachrichtenarchivierung und -wiederholung besteht aus zwei Hauptkomponenten:

1. Nachrichtenarchivierung – Der Themeneigentümer aktiviert die Archivierungs- und Wiederholungsfunktion für ein Thema und legt einen Aufbewahrungszeitraum für Nachrichten fest (bis zu 365 Tage). Der Themeneigentümer kann archivierte Nachrichten auch anhand von Amazon CloudWatch-Metriken überwachen. Weitere Informationen finden Sie unter Nachrichtenarchivierung für Besitzer von FIFO-Themen.

2. Nachrichtenwiedergabe – Der Themen-Suscriber initiiert die Wiederholung einer Reihe von Nachrichten vom Thema bis zum abonnierten Endpunkt. Weitere Informationen finden Sie unter Nachrichtenwiederholung für Subscriber des FIFO-Themas.

Nachrichtenarchivierung für Besitzer von FIFO-Themen

Mit der Nachrichtenarchivierung können Sie eine einzige Kopie aller zu Ihrem Thema veröffentlichten Nachrichten archivieren. Sie können veröffentlichte Nachrichten in Ihrem Thema speichern, indem Sie die Nachrichtenarchivierungsrichtlinie für das Thema aktivieren, wodurch die Nachrichtenarchivierung für alle Abonnements aktiviert wird, die mit diesem Thema verknüpft sind. Nachrichten können für mindestens einen Tag bis maximal 365 Tage archiviert werden.

Bei der Festlegung einer Archivrichtlinie fallen zusätzliche Gebühren an. Informationen zu den Preisen finden Sie unter Amazon SNS – Preise.

Themen

- Eine Nachrichtenarchivierungsrichtlinie mit AWS Management Console erstellen
- Eine Nachrichtenarchivierungsrichtlinie mit der API erstellen
- Eine Nachrichtenarchivierungsrichtlinie mit dem SDK erstellen
- Eine Nachrichtenarchivierungsrichtlinie mit AWS CloudFormation erstellen
- Zugriff auf ein verschlüsseltes Archiv gewähren
- Nachrichtenarchivierungsmetriken mit Amazon CloudWatch überwachen

Eine Nachrichtenarchivierungsrichtlinie mit AWS Management Console erstellen

Verwenden Sie diese Option zum Erstellen einer neuen Nachrichtenarchivrichtlinie mit AWS Management Console.

1. Melden Sie sich bei der Amazon-SNS-Konsole an.
2. Wählen Sie ein Thema aus oder erstellen Sie ein neues Thema. Weitere Informationen zum Erstellen von Themen finden Sie unter Erstellen eines Amazon SNS-Themas.

> **Note**
>
> Die Amazon SNS Nachrichtenarchivierung und -wiederholung ist nur für Application-to-Application (A2A) FIFO-Themen verfügbar.

3. Erweitern Sie auf der Seite Edit topic (Thema bearbeiten) den Abschnitt Archive policy (Archivrichtlinie) .
4. Aktivieren Sie die Funktion Archive policy (Archivrichtlinie) und geben Sie im Thema die Anzahl der Tage ein, für die Sie Nachrichten archivieren möchten.
5. Wählen Sie Änderungen speichern.

Anzeigen, Bearbeiten und Deaktivieren einer Themenrichtlinie für die Nachrichtenarchivierung

- Auf der Seite Topic details (Themendetails) zeigt die Retention policy (Aufbewahrungsrichtlinie) den Status der Archivrichtlinie an, einschließlich der Anzahl der Tage, für die sie festgelegt ist. Wählen Sie die Registerkarte Archive policy (Archivrichtlinie), um die folgenden Details zum Nachrichtenarchiv anzuzeigen:
  - Status — Der Archivierungs- und Wiederholungsstatus wird als active (aktiv) angezeigt, wenn eine Archivrichtlinie angewendet wird. Der Archivierungs- und Wiederholungsstatus wird als inactive (inaktiv) angezeigt, wenn die Archivrichtlinie auf ein leeres JSON-Objekt festgelegt ist.
  - Message retention period (Aufbewahrungszeitraum für Nachrichten) – Die angegebene Anzahl von Tagen für die Aufbewahrung von Nachrichten.
  - Archive start date (Startdatum des Archivs) – Das Datum, ab dem Suscriber Nachrichten wiederholen können.
  - JSON preview (JSON-Vorschau) – Die JSON-Vorschau der Archivrichtlinie.
- (Optional) Um eine Archivrichtlinie zu bearbeiten, wechseln Sie zur Seite mit der Themenzusammenfassung und wählen Sie Edit (Bearbeiten) aus.
- (Optional) Um eine Archivrichtlinie zu deaktivieren, wechseln Sie zur Seite mit der Themenzusammenfassung und wählen Sie Edit (Bearbeiten) aus. Deaktivieren Sie die Archive Policy (Archivrichtlinie) und wählen Sie Save changes (Änderungen speichern).
- (Optional) Um ein Thema mit einer Archivrichtlinie zu löschen, müssen Sie zuerst die Archivrichtlinie wie zuvor beschrieben deaktivieren.

> **Important**
>
> Um das versehentliche Löschen von Nachrichten zu vermeiden, ist es nicht möglich, ein Thema mit einer aktiven Nachrichtenarchivierungsrichtlinie zu löschen. Die Nachrichtenarchivierungsrichtlinie des Themas muss deaktiviert werden, bevor das Thema gelöscht werden kann. Wenn Sie eine Nachrichtenarchivierungsrichtlinie deaktivieren, löscht Amazon SNS alle archivierten Nachrichten. Beim Löschen eines Themas werden

Abonnements entfernt und alle Nachrichten, die gerade übertragen werden, können möglicherweise nicht zugestellt werden.

Eine Nachrichtenarchivierungsrichtlinie mit der API erstellen

Um mithilfe der API eine Nachrichtenarchivierungsrichtlinie zu erstellen, müssen Sie das Attribut `ArchivePolicy` zu Ihrem Thema hinzufügen. Sie können `ArchivePolicy` mithilfe der API-Aktionen `CreateTopic` und `SetTopicAttributes` festlegen. `ArchivePolicy` hat einen einzigen Wert, `MessageRetentionPeriod`, der die Anzahl der Tage darstellt, an denen Amazon SNS Nachrichten aufbewahrt. Um die Nachrichtenarchivierung für Ihr Thema zu aktivieren, legen Sie `MessageRetentionPeriod` auf einen ganzzahligen Wert größer als Null fest. Um beispielsweise Nachrichten 30 Tage lang in Ihrem Archiv aufzubewahren, legen Sie für `ArchivePolicy` Folgendes fest:

```
{
    "ArchivePolicy": {
        "MessageRetentionPeriod": "30"
    }
}
```

Um die Nachrichtenarchivierung für Ihr Thema zu deaktivieren und das Archiv zu löschen, deaktivieren Sie `ArchivePolicy` wie folgt:

```
{}
```

Eine Nachrichtenarchivierungsrichtlinie mit dem SDK erstellen

Um ein AWS-SDK zu verwenden, müssen Sie es mit Ihren Anmeldeinformationen konfigurieren. Weitere Informationen finden Sie unter Freigegebene `config`- und `credentials`-Dateien im Referenzhandbuch zu AWSSDKs und Tools.

Im folgenden Codebeispiel wird veranschaulicht, wie Sie `ArchivePolicy` für ein Amazon SNS-Thema festlegen, damit alle Nachrichten, die zum Thema veröffentlicht wurden, 30 Tage lang aufbewahrt werden.

```
// Specify the ARN of the Amazon SNS topic to set the ArchivePolicy for.
String topicArn =
    "arn:aws:sns:us-east-2:123456789012:MyArchiveTopic.fifo";
```

```
// Set the MessageRetentionPeriod to 30 days for the ArchivePolicy.
String archivePolicy =
    "{\"MessageRetentionPeriod\":\"30\"}";

// Set the ArchivePolicy for the Amazon SNS topic
SetTopicAttributesRequest request = new SetTopicAttributesRequest()
    .withTopicArn(topicArn)
    .withAttributeName("ArchivePolicy")
    .withAttributeValue(archivePolicy);
sns.setTopicAttributes(request);
```

Eine Nachrichtenarchivierungsrichtlinie mit AWS CloudFormation erstellen

Für Informationen zum Erstellen einer Archivrichtlinie mithilfe von AWS CloudFormation siehe `AWS::SNS::Topic` im AWS CloudFormation-Benutzerhandbuch.

Zugriff auf ein verschlüsseltes Archiv gewähren

Bevor ein Suscriber mit der Wiederholung von Nachrichten aus einem verschlüsselten Thema beginnen kann, müssen Sie die folgenden Schritte ausführen. Da frühere Nachrichten wiederholt werden, muss Amazon SNS `Decrypt`-Zugriff auf den KMS-Schlüssel erhalten, der zur Verschlüsselung der Nachrichten im Archiv verwendet wurde.

1. Wenn Sie Nachrichten mit einem KMS-Schlüssel verschlüsseln und innerhalb des Themas speichern, müssen Sie Amazon SNS die Möglichkeit geben, diese Nachrichten über die Schlüsselrichtlinie zu entschlüsseln. Weitere Informationen finden Sie unter Amazon SNS Entschlüsselungsberechtigungen gewähren.
2. Aktivieren Sie AWS KMS für Amazon SNS. Weitere Informationen finden Sie unter Konfigurieren von AWS KMS-Berechtigungen.

> **Important**
>
> Ändern Sie beim Hinzufügen der neuen Abschnitte zur KMS-Schlüsselrichtlinie keinen der vorhandenen Abschnitte. Wenn die Verschlüsselung für ein Thema aktiviert ist und der KMS-Schlüssel deaktiviert oder gelöscht wird oder die KMS-Schlüsselrichtlinie nicht korrekt für Amazon SNS konfiguriert ist, kann Amazon SNS keine Nachrichten für Ihre Subscriber wiederholen.

Amazon SNS Entschlüsselungsberechtigungen gewähren

Damit Amazon SNS auf verschlüsselte Nachrichten aus dem Archiv Ihres Themas zugreifen und sie auf abonnierten Endpunkten archivieren und wiederholen kann, müssen Sie das Amazon SNS-Serviceprinzip aktivieren, um diese Nachrichten zu entschlüsseln.

Im Folgenden finden Sie eine Beispielrichtlinie, die erforderlich ist, damit der Prinzipal des Amazon SNS gespeicherte Nachrichten während einer Wiederholung von historischen Nachrichten aus Ihrem Thema entschlüsseln kann.

```
{
    "Sid": "Allow SNS to decrypt archived messages",
    "Effect": "Allow",
    "Principal": {
        "Service": "sns.amazonaws.com"
    },
    "Action": [
        "kms:Decrypt",
        "kms:GenerateDataKey"
    ],
    "Resource": "*"
}
```

Nachrichtenarchivierungsmetriken mit Amazon CloudWatch überwachen

Sie können archivierte Nachrichten mit Amazon CloudWatch anhand der folgenden Metriken überwachen. Um über Anomalien in Ihren Workloads informiert zu werden und entsprechende Auswirkungen zu vermeiden, können Sie Amazon CloudWatch-Alarme für diese Metriken konfigurieren. Weitere Details finden Sie unter Protokollierung und Überwachung in Amazon SNS.

Kennzahl	Beschreibung
ApproximateNumberOfMessagesArchived	Stellt dem Eigentümer des Themas die Gesamtanzahl der im Themenarchiv archivierten Nachrichten mit einer Auflösung von 60 Minuten zur Verfügung.
ApproximateNumberOfBytesArchived	Stellt dem Eigentümer des Themas die Gesamtanzahl der archivierten Bytes für

Kennzahl	Beschreibung
	alle Nachrichten im Themenarchiv mit einer Auflösung von 60 Minuten zur Verfügung.
NumberOfMessagesArchiveProcessing	Nennt dem Eigentümer des Themas die Anzahl der Nachrichten, die während des Intervalls im Themenarchiv gespeichert wurden, mit einer Auflösung von 1 Minute.
NumberOfBytesArchiveProcessing	Nennt dem Eigentümer des Themas die Gesamtzahl der Bytes, die während des Intervalls im Themenarchiv gespeichert wurden, mit einer Auflösung von 1 Minute.

Die `GetTopicAttributes`-API verfügt über eine `BeginningArchiveTime`-Eigenschaft, die den ältesten Zeitstempel darstellt, zu dem ein Subscriber eine Wiederholung starten kann. Im Folgenden finden Sie eine Beispielantwort für diese API-Aktion:

```
{
 "ArchivePolicy": {
    "MessageRetentionPeriod": "<integer>"
  },
  "BeginningArchiveTime": "<timestamp>",
  ...
}
```

Nachrichtenwiederholung für Subscriber des FIFO-Themas

Mit Amazon SNS Replay können Themen-Subscriber archivierte Nachrichten aus dem Themendatenspeicher abrufen und sie erneut an einen abonnierten Endpunkt senden (oder wiederholen). Nachrichten können wiederholt werden, sobald das Abonnement erstellt wurde. Eine wiederholt Nachricht hat denselben Inhalt und denselben Inhalt, `MessageId` und `Timestamp`, wie die Originalkopie und enthält außerdem das Attribut `Replayed`, anhand dessen Sie erkennen können, dass es sich um eine wiederholte Nachricht handelt. Um nur ausgewählte Nachrichten zu wiederholen, können Sie Ihrem Abonnement eine Filterrichtlinie hinzufügen. Weitere Informationen zum Filtern von Nachrichten finden Sie unter Filtert wiederholte Nachrichten.

Themen

Eine Nachrichtenwiederholungsrichtlinie mit AWS Management Console erstellen

Verwenden Sie diese Option zum Erstellen einer neuen Wiederholungsrichtlinie mit AWS Management Console.

1. Melden Sie sich bei der Amazon-SNS-Konsole an.
2. Wählen Sie ein Themenabonnement aus oder erstellen Sie ein neues Thema. Weitere Informationen zum Erstellen von Abonnements finden Sie unter Abonnieren eines Amazon-SNS-Themas.
3. Um die Nachrichtenwiederholung zu starten, rufen Sie das Dropdown-Menü Replay (Wiederholung) auf und wählen Sie Start replay (Wiederholung starten).
4. Treffen Sie im Modal Replay timeframe (Zeitrahmen für Wiederholung) die folgenden Auswahlen:

    a. Choose replay start date and time (Startdatum und Uhrzeit der Wiederholung wählen) – Wählen Sie unter date das Datum (Format JJJJ/MM/TT) und bei time die Uhrzeit (24-Stunden-Format hh:mm:ss) aus, ab dem Sie mit der Wiederholung archivierter Nachrichten beginnen möchten. Die Startzeit muss nach Beginn der ungefähren Archivierungszeit liegen.

    b. (Optional) Choose replay end date and time (Enddatum und Uhrzeit der Wiederholung wählen) – Wählen Sie unter date das Datum (Format JJJJ/MM/TT) und bei time die Uhrzeit (24-Stunden-Format hh:mm:ss) aus, ab dem die Wiederholung archivierter Nachrichten beendet werden soll.

    c. Wählen Sie Start replay „Wiederholung starten“.
5. (Optional) Um die Wiedergabe einer Nachricht zu beenden, rufen Sie die Seite mit den Subscription details (Abonnementdetails) auf und wählen Sie im Dropdown-Menü Replay (Wiederholung) die Option Stop replay (Wiederholung beenden) aus.
6. (Optional) Informationen zur Überwachung der Nachrichtenwiederholungsmetriken innerhalb dieses Workflows mithilfe von CloudWatch finden Sie unter Nachrichtenwiederholungsmetriken mit Amazon CloudWatch überwachen.

Informationen zum Anzeigen und Bearbeiten einer Nachrichtenwiederholungsrichtlinie

Auf der Seite Subscription details (Abonnementdetails) können Sie die folgenden Aktionen ausführen:

- Um den Wiederholungsstatus der Nachricht anzuzeigen, werden im Feld Replay status (Wiederholungsstatus) die folgenden Werte angezeigt:
  - Completed (Abgeschlossen) – Die Wiederholung hat alle Nachrichten erfolgreich erneut zugestellt und stellt jetzt neu veröffentlichte Nachrichten zu.
  - In progress (In Bearbeitung) – Die Wiederholung wiederholt derzeit die ausgewählten Nachrichten.
  - Failed (Fehlgeschlagen) – Die Wiederholung konnte nicht abgeschlossen werden.
  - Pending (Ausstehend) – Der Standardstatus, während die Wiederholung gestartet wird.
- (Optional) Um eine Nachrichtenwiederholungsrichtlinie zu ändern, rufen Sie die Seite Subscription details (Abonnementdetails) auf und wählen Sie im Dropdown-Menü Replay (Wiederholung) die Option Start replay (Wiederholung starten) aus. Durch das Starten einer Wiederholung wird die bestehende Wiederholung ersetzt.

Dem Abonnement mithilfe der API eine Wiederholungsrichtlinie hinzufügen

Verwenden Sie das Attribut `ReplayPolicy`, um archivierte Nachrichten zu wiederholen. `ReplayPolicy` kann mit den API-Aktionen `Subscribe` und `SetSubscriptionAttributes` verwendet werden. Diese Richtlinie hat folgende Werte:

- `StartingPoint` (Obligatorisch) – Zeigt an, ab welchem Punkt die Wiederholung von Nachrichten gestartet werden soll.
- `EndingPoint`(Optional) – Zeigt an, wann die Nachrichtenwiederholung beendet werden soll. Wenn `EndingPoint` weggelassen wird, wird die Wiederholung fortgesetzt, bis die aktuelle Uhrzeit erreicht ist.
- `PointType` (Obligatorisch) – Legt den Typ der Start- und Endpunkte fest. Derzeit wird `Timestamp` als einziger Wert für `PointType` unterstützt.

Um beispielsweise nach dem Ausfall einer nachgeschalteten Anwendung eine Wiederherstellung durchzuführen und alle Nachrichten für einen Zeitraum von zwei Stunden am 1. Oktober 2023 erneut zu senden, verwenden Sie die API-Aktion `SetSubscriptionAttributes`, um wie folgt ein `ReplayPolicy` festzulegen:

```
{
  "PointType":"Timestamp",
  "StartingPoint":"2023-10-01T10:00:00.000Z",
  "EndingPoint":"2023-10-01T12:00:00.000Z"
}
```

Um alle Nachrichten, die am 1. Oktober 2023 an das Thema gesendet wurden, zu wiederholen und weiterhin alle neu veröffentlichten Nachrichten zu Ihrem Thema zu erhalten, verwenden Sie die API-Aktion `SetSubscriptionAttributes`, um wie folgt ein `ReplayPolicy` für Ihr Abonnement festzulegen:

```
{
  "PointType":"Timestamp",
  "StartingPoint":"2023-10-01T00:00:00.000Z"
}
```

Um zu überprüfen, ob eine Nachricht wiederholt wurde, wird jeder wiedergegebenen Nachricht das boolesche Attribut `Replayed` hinzugefügt.

Dem Abonnement mithilfe des SDK eine Wiederholungsrichtlinie hinzufügen

Um ein AWS-SDK zu verwenden, müssen Sie es mit Ihren Anmeldeinformationen konfigurieren. Weitere Informationen finden Sie unter Freigegebene `config`- und `credentials`-Dateien im Referenzhandbuch zu AWS-SDKs und Tools.

Das folgende Codebeispiel zeigt, wie Sie `ReplayPolicy` bei einem Abonnement festlegen, damit Nachrichten aus dem Archiv des Amazon SNS FIFO-Themas für ein 2-stündiges Zeitfenster am 1. Oktober 2023 erneut zugestellt werden.

```
// Specify the ARN of the Amazon SNS subscription to initiate the ReplayPolicy on.
String subscriptionArn =
    "arn:aws:sns:us-
east-2:123456789012:MyArchiveTopic.fifo:1d2a3e9d-7f2f-447c-88ae-03f1c68294da";

// Set the ReplayPolicy to replay messages from the topic's archive
// for a 2 hour time period on October 1st 2023 between 10am and 12pm UTC.
String replayPolicy =
    "{\"PointType\":\"Timestamp\",\"StartingPoint\":\"2023-10-01T10:00:00.000Z\",
\"EndingPoint\":\"2023-10-01T12:00:00.000Z\"}";
```

```
// Set the ArchivePolicy for the Amazon SNS topic
SetSubscriptionAttributesRequest request = new SetSubscriptionAttributesRequest()
    .withSubscriptionArn(subscriptionArn)
    .withAttributeName("ReplayPolicy")
    .withAttributeValue(replayPolicy);
sns.setSubscriptionAttributes(request);
```

Filtert wiederholte Nachrichten

Mit der Amazon SNS-Nachrichtenfilterung können Sie steuern, welche Nachrichten die Amazon SNS an Ihrem Abonnentenendpunkt wiederholt. Wenn sowohl Nachrichtenfilterung als auch Nachrichtenarchivierung aktiviert sind, ruft Amazon SNS die Nachricht zunächst aus dem Datenspeicher des Themas ab und ordnet sie dann dem `FilterPolicy` des Abonnements zu. Die Nachricht wird an den abonnierten Endpunkt übermittelt, wenn eine Übereinstimmung vorliegt, andernfalls wird die Nachricht herausgefiltert. Weitere Informationen finden Sie unter Filterrichtlinien für Amazon-SNS-Abonnements.

Nachrichtenwiederholungsmetriken mit Amazon CloudWatch überwachen

Sie können wiederholte Nachrichten mit Amazon CloudWatch anhand der folgenden Metriken überwachen. Um über Anomalien in Ihren Workloads informiert zu werden und entsprechende Auswirkungen zu vermeiden, können Sie Amazon CloudWatch-Alarme für diese Metriken konfigurieren. Weitere Details finden Sie unter Protokollierung und Überwachung in Amazon SNS.

Kennzahl	Beschreibung
NumberOfReplayedNotificationsDelivered	Nennt dem Suscriber die Gesamtanzahl der aus dem Themenarchiv wiederholten Nachricht en mit einer Auflösung von 1 Minute.
NumberOfReplayedNotificationsFailed	Nennt dem Suscriber die Gesamtanzahl der aus dem wiederholten Nachrichten mit einer Auflösung von 1 Minute, die nicht aus dem Themenarchiv zugestellt werden konnten.

Codebeispiele für FIFO-Themen

Sie können die folgenden Codebeispiele verwenden, um den Beispiel-Anwendungsfall für das Preismanagement von Autoteilen mit einem Amazon-SNS-FIFO-Thema und einer Amazon-SQS-FIFO- oder Standard-Warteschlange zu verwenden.

Themen

- Verwenden eines AWSSDKs
- AWS CloudFormation einsetzen

Verwenden eines AWSSDKs

Unter Verwendung eines AWS-SDK erstellen Sie ein Amazon-SNS-FIFO-Thema, indem Sie dessen `FifoTopic`-Attribut auf **`true`** setzen. Sie erstellen eine Amazon-SQS-FIFO-Warteschlange, indem Sie ihr `FifoQueue`-Attribut auf **`true`** setzen. Außerdem müssen Sie das Suffix **`.fifo`**zum Namen jeder FIFO-Ressource hinzufügen. Nachdem Sie ein FIFO-Thema oder eine Queue erstellt haben, können Sie es nicht in ein Standardthema oder eine Standardqueue konvertieren.

In den folgenden Codebeispielen werden diese FIFO- und Standard-Warteschlangenressourcen erstellt:

- Das Amazon-SNS-FIFO-Thema, das die Preisaktualisierungen verteilt
- Die Amazon-SQS-FIFO-Warteschlangen, die diese Updates für die Groß- und Einzelhandel-Anwendung bereitstellen
- Die Amazon-SQS-Standardwarteschlange für die Analyseanwendung, in der Datensätze gespeichert werden, die für Business Intelligence (BI) abgefragt werden können
- Die Amazon-SNS-FIFO-Abonnements, die beide Warteschlangen mit dem Thema verbinden

In diesem Beispiel werden Filterrichtlinien in den Abonnements eingestellt. Wenn Sie das Beispiel überprüfen, indem Sie eine Nachricht im -Thema veröffentlichen, stellen Sie sicher, dass Sie die Nachricht mit dem `business`-Attribut veröffentlichen. Geben Sie entweder `retail`oder `wholesale`als Attributwert an. Andernfalls wird die Nachricht herausgefiltert und nicht an die abonnierten Queues übermittelt. Weitere Informationen finden Sie unter Nachrichtenfilterung für FIFO-Themen.

Java

SDK für Java 2.x

> **Note**
>
> Auf GitHub finden Sie noch mehr. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS-Code-Beispiel- einrichten und ausführen.

Dieses Beispiel

- erstellt ein Amazon-SNS-FIFO-Thema, zwei Amazon SQS-FIFO-Warteschlangen und eine Standard-Warteschlange.
- abonniert die Warteschlangen für das Thema und veröffentlicht eine Nachricht zu dem Thema.

Der Test überprüft den Eingang der Nachricht in jeder Warteschlange. Das vollständige Beispiel zeigt auch das Hinzufügen von Zugriffsrichtlinien und löscht die Ressourcen am Ende.

```
public class PriceUpdateExample {
    public final static SnsClient snsClient = SnsClient.create();
    public final static SqsClient sqsClient = SqsClient.create();

    public static void main(String[] args) {

        final String usage = "\n" +
                "Usage: " +
                "    <topicName> <wholesaleQueueFifoName> <retailQueueFifoName>
 <analyticsQueueName>\n\n" +
                "Where:\n" +
                "   fifoTopicName - The name of the FIFO topic that you want to
 create. \n\n" +
                "   wholesaleQueueARN - The name of a SQS FIFO queue that will be
 created for the wholesale consumer. \n\n"
                +
                "   retailQueueARN - The name of a SQS FIFO queue that will
 created for the retail consumer. \n\n" +
                "   analyticsQueueARN - The name of a SQS standard queue that
 will be created for the analytics consumer. \n\n";
        if (args.length != 4) {
            System.out.println(usage);
```

```
            System.exit(1);
        }

        final String fifoTopicName = args[0];
        final String wholeSaleQueueName = args[1];
        final String retailQueueName = args[2];
        final String analyticsQueueName = args[3];

        // For convenience, the QueueData class holds metadata about a queue:
ARN, URL,
        // name and type.
        List<QueueData> queues = List.of(
                new QueueData(wholeSaleQueueName, QueueType.FIFO),
                new QueueData(retailQueueName, QueueType.FIFO),
                new QueueData(analyticsQueueName, QueueType.Standard));

        // Create queues.
        createQueues(queues);

        // Create a topic.
        String topicARN = createFIFOTopic(fifoTopicName);

        // Subscribe each queue to the topic.
        subscribeQueues(queues, topicARN);

        // Allow the newly created topic to send messages to the queues.
        addAccessPolicyToQueuesFINAL(queues, topicARN);

        // Publish a sample price update message with payload.
        publishPriceUpdate(topicARN, "{\"product\": 214, \"price\": 79.99}",
"Consumables");

        // Clean up resources.
        deleteSubscriptions(queues);
        deleteQueues(queues);
        deleteTopic(topicARN);
    }

    public static String createFIFOTopic(String topicName) {
        try {
            // Create a FIFO topic by using the SNS service client.
            Map<String, String> topicAttributes = Map.of(
                    "FifoTopic", "true",
                    "ContentBasedDeduplication", "false");
```

```
            CreateTopicRequest topicRequest = CreateTopicRequest.builder()
                    .name(topicName)
                    .attributes(topicAttributes)
                    .build();

            CreateTopicResponse response = snsClient.createTopic(topicRequest);
            String topicArn = response.topicArn();
            System.out.println("The topic ARN is" + topicArn);

            return topicArn;

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }

    public static void subscribeQueues(List<QueueData> queues, String topicARN) {
        queues.forEach(queue -> {
            SubscribeRequest subscribeRequest = SubscribeRequest.builder()
                    .topicArn(topicARN)
                    .endpoint(queue.queueARN)
                    .protocol("sqs")
                    .build();

            // Subscribe to the endpoint by using the SNS service client.
            // Only Amazon SQS queues can receive notifications from an Amazon
SNS FIFO
            // topic.
            SubscribeResponse subscribeResponse =
snsClient.subscribe(subscribeRequest);
            System.out.println("The queue [" + queue.queueARN + "] subscribed to
the topic [" + topicARN + "]");
            queue.subscriptionARN = subscribeResponse.subscriptionArn();
        });
    }

    public static void publishPriceUpdate(String topicArn, String payload, String
groupId) {

        try {
            // Create and publish a message that updates the wholesale price.
```

```
            String subject = "Price Update";
            String dedupId = UUID.randomUUID().toString();
            String attributeName = "business";
            String attributeValue = "wholesale";

            MessageAttributeValue msgAttValue = MessageAttributeValue.builder()
                    .dataType("String")
                    .stringValue(attributeValue)
                    .build();

            Map<String, MessageAttributeValue> attributes = new HashMap<>();
            attributes.put(attributeName, msgAttValue);
            PublishRequest pubRequest = PublishRequest.builder()
                    .topicArn(topicArn)
                    .subject(subject)
                    .message(payload)
                    .messageGroupId(groupId)
                    .messageDeduplicationId(dedupId)
                    .messageAttributes(attributes)
                    .build();

            final PublishResponse response = snsClient.publish(pubRequest);
            System.out.println(response.messageId());
            System.out.println(response.sequenceNumber());
            System.out.println("Message was published to " + topicArn);

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- API-Details finden Sie in den folgenden Themen der AWS SDK for Java 2.x-API-Referenz.
  - CreateTopic
  - Veröffentlichen
  - Abonnieren

Python

SDK für Python (Boto3)

> **Note**
>
> Auf GitHub finden Sie noch mehr. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS-Code-Beispiel- einrichten und ausführen.

Erstellen Sie ein Amazon-SNS-FIFO-Thema, abonnieren Sie eine Amazon-SQS-FIFO- und eine Standard-Warteschlange für das Thema und veröffentlichen Sie eine Nachricht zu dem Thema.

```
def usage_demo():
    """Shows how to subscribe queues to a FIFO topic."""
    print("-" * 88)
    print("Welcome to the `Subscribe queues to a FIFO topic` demo!")
    print("-" * 88)

    sns = boto3.resource("sns")
    sqs = boto3.resource("sqs")
    fifo_topic_wrapper = FifoTopicWrapper(sns)
    sns_wrapper = SnsWrapper(sns)

    prefix = "sqs-subscribe-demo-"
    queues = set()
    subscriptions = set()

    wholesale_queue = sqs.create_queue(
        QueueName=prefix + "wholesale.fifo",
        Attributes={
            "MaximumMessageSize": str(4096),
            "ReceiveMessageWaitTimeSeconds": str(10),
            "VisibilityTimeout": str(300),
            "FifoQueue": str(True),
            "ContentBasedDeduplication": str(True),
        },
    )
    queues.add(wholesale_queue)
    print(f"Created FIFO queue with URL: {wholesale_queue.url}.")
```

```
    retail_queue = sqs.create_queue(
        QueueName=prefix + "retail.fifo",
        Attributes={
            "MaximumMessageSize": str(4096),
            "ReceiveMessageWaitTimeSeconds": str(10),
            "VisibilityTimeout": str(300),
            "FifoQueue": str(True),
            "ContentBasedDeduplication": str(True),
        },
    )
    queues.add(retail_queue)
    print(f"Created FIFO queue with URL: {retail_queue.url}.")

    analytics_queue = sqs.create_queue(QueueName=prefix + "analytics",
Attributes={})
    queues.add(analytics_queue)
    print(f"Created standard queue with URL: {analytics_queue.url}.")

    topic = fifo_topic_wrapper.create_fifo_topic("price-updates-topic.fifo")
    print(f"Created FIFO topic: {topic.attributes['TopicArn']}.")

    for q in queues:
        fifo_topic_wrapper.add_access_policy(q, topic.attributes["TopicArn"])

    print(f"Added access policies for topic: {topic.attributes['TopicArn']}.")

    for q in queues:
        sub = fifo_topic_wrapper.subscribe_queue_to_topic(
            topic, q.attributes["QueueArn"]
        )
        subscriptions.add(sub)

    print(f"Subscribed queues to topic: {topic.attributes['TopicArn']}.")

    input("Press Enter to publish a message to the topic.")

    message_id = fifo_topic_wrapper.publish_price_update(
        topic, '{"product": 214, "price": 79.99}', "Consumables"
    )

    print(f"Published price update with message ID: {message_id}.")

    # Clean up the subscriptions, queues, and topic.
    input("Press Enter to clean up resources.")
```

```
    for s in subscriptions:
        sns_wrapper.delete_subscription(s)

    sns_wrapper.delete_topic(topic)

    for q in queues:
        fifo_topic_wrapper.delete_queue(q)

    print(f"Deleted subscriptions, queues, and topic.")

    print("Thanks for watching!")
    print("-" * 88)



class FifoTopicWrapper:
    """Encapsulates Amazon SNS FIFO topic and subscription functions."""

    def __init__(self, sns_resource):
        """
        :param sns_resource: A Boto3 Amazon SNS resource.
        """
        self.sns_resource = sns_resource

    def create_fifo_topic(self, topic_name):
        """
        Create a FIFO topic.
        Topic names must be made up of only uppercase and lowercase ASCII
 letters,
        numbers, underscores, and hyphens, and must be between 1 and 256
 characters long.
        For a FIFO topic, the name must end with the .fifo suffix.

        :param topic_name: The name for the topic.
        :return: The new topic.
        """
        try:
            topic = self.sns_resource.create_topic(
                Name=topic_name,
                Attributes={
                    "FifoTopic": str(True),
                    "ContentBasedDeduplication": str(False),
                },
            )
```

```
            logger.info("Created FIFO topic with name=%s.", topic_name)
            return topic
        except ClientError as error:
            logger.exception("Couldn't create topic with name=%s!", topic_name)
            raise error


    @staticmethod
    def add_access_policy(queue, topic_arn):
        """
        Add the necessary access policy to a queue, so
        it can receive messages from a topic.

        :param queue: The queue resource.
        :param topic_arn: The ARN of the topic.
        :return: None.
        """
        try:
            queue.set_attributes(
                Attributes={
                    "Policy": json.dumps(
                        {
                            "Version": "2012-10-17",
                            "Statement": [
                                {
                                    "Sid": "test-sid",
                                    "Effect": "Allow",
                                    "Principal": {"AWS": "*"},
                                    "Action": "SQS:SendMessage",
                                    "Resource": queue.attributes["QueueArn"],
                                    "Condition": {
                                        "ArnLike": {"aws:SourceArn": topic_arn}
                                    },
                                }
                            ],
                        }
                    )
                }
            )
            logger.info("Added trust policy to the queue.")
        except ClientError as error:
            logger.exception("Couldn't add trust policy to the queue!")
            raise error
```

```
    @staticmethod
    def subscribe_queue_to_topic(topic, queue_arn):
        """
        Subscribe a queue to a topic.

        :param topic: The topic resource.
        :param queue_arn: The ARN of the queue.
        :return: The subscription resource.
        """
        try:
            subscription = topic.subscribe(
                Protocol="sqs",
                Endpoint=queue_arn,
            )
            logger.info("The queue is subscribed to the topic.")
            return subscription
        except ClientError as error:
            logger.exception("Couldn't subscribe queue to topic!")
            raise error


    @staticmethod
    def publish_price_update(topic, payload, group_id):
        """
        Compose and publish a message that updates the wholesale price.

        :param topic: The topic to publish to.
        :param payload: The message to publish.
        :param group_id: The group ID for the message.
        :return: The ID of the message.
        """
        try:
            att_dict = {"business": {"DataType": "String", "StringValue":
"wholesale"}}
            dedup_id = uuid.uuid4()
            response = topic.publish(
                Subject="Price Update",
                Message=payload,
                MessageAttributes=att_dict,
                MessageGroupId=group_id,
                MessageDeduplicationId=str(dedup_id),
            )
            message_id = response["MessageId"]
```

```
            logger.info("Published message to topic %s.", topic.arn)
        except ClientError as error:
            logger.exception("Couldn't publish message to topic %s.", topic.arn)
            raise error
        return message_id


    @staticmethod
    def delete_queue(queue):
        """
        Removes an SQS queue. When run against an AWS account, it can take up to
        60 seconds before the queue is actually deleted.

        :param queue: The queue to delete.
        :return: None
        """
        try:
            queue.delete()
            logger.info("Deleted queue with URL=%s.", queue.url)
        except ClientError as error:
            logger.exception("Couldn't delete queue with URL=%s!", queue.url)
            raise error
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS-SDK für Python (Boto3).
  - CreateTopic
  - Veröffentlichen
  - Abonnieren

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Auf GitHub finden Sie noch mehr. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS-Code-Beispiel- einrichten und ausführen.

Erstellen Sie ein FIFO-Thema, abonnieren Sie eine Amazon-SQS-FIFO-Warteschlange für das Thema und veröffentlichen Sie eine Nachricht zu einem Amazon-SNS-Thema.

```
    " Creates a FIFO topic. "
    DATA lt_tpc_attributes TYPE /aws1/
cl_snstopicattrsmap_w=>tt_topicattributesmap.
    DATA ls_tpc_attributes TYPE /aws1/
cl_snstopicattrsmap_w=>ts_topicattributesmap_maprow.
    ls_tpc_attributes-key = 'FifoTopic'.
    ls_tpc_attributes-value = NEW /aws1/cl_snstopicattrsmap_w( iv_value =
 'true' ).
    INSERT ls_tpc_attributes INTO TABLE lt_tpc_attributes.

    TRY.
        DATA(lo_create_result) = lo_sns->createtopic(
               iv_name = iv_topic_name
               it_attributes = lt_tpc_attributes
        ).
        DATA(lv_topic_arn) = lo_create_result->get_topicarn( ).
        ov_topic_arn = lv_topic_arn.                                        "
 ov_topic_arn is returned for testing purposes. "
        MESSAGE 'FIFO topic created' TYPE 'I'.
      CATCH /aws1/cx_snstopiclimitexcdex.
        MESSAGE 'Unable to create more topics. You have reached the maximum
 number of topics allowed.' TYPE 'E'.
    ENDTRY.

    " Subscribes an endpoint to an Amazon Simple Notification Service (Amazon
 SNS) topic. "
    " Only Amazon Simple Queue Service (Amazon SQS) FIFO queues can be subscribed
 to an SNS FIFO topic. "
```

```
    TRY.
        DATA(lo_subscribe_result) = lo_sns->subscribe(
               iv_topicarn = lv_topic_arn
               iv_protocol = 'sqs'
               iv_endpoint = iv_queue_arn
           ).
        DATA(lv_subscription_arn) = lo_subscribe_result->get_subscriptionarn( ).
        ov_subscription_arn = lv_subscription_arn.                        "
 ov_subscription_arn is returned for testing purposes. "
        MESSAGE 'SQS queue was subscribed to SNS topic.' TYPE 'I'.
      CATCH /aws1/cx_snsnotfoundexception.
        MESSAGE 'Topic does not exist.' TYPE 'E'.
      CATCH /aws1/cx_snssubscriptionlmte00.
        MESSAGE 'Unable to create subscriptions. You have reached the maximum
 number of subscriptions allowed.' TYPE 'E'.
    ENDTRY.

    " Publish message to SNS topic. "
    TRY.
        DATA lt_msg_attributes TYPE /aws1/
cl_snsmessageattrvalue=>tt_messageattributemap.
        DATA ls_msg_attributes TYPE /aws1/
cl_snsmessageattrvalue=>ts_messageattributemap_maprow.
        ls_msg_attributes-key = 'Importance'.
        ls_msg_attributes-value = NEW /aws1/cl_snsmessageattrvalue( iv_datatype =
 'String' iv_stringvalue = 'High' ).
        INSERT ls_msg_attributes INTO TABLE lt_msg_attributes.

        DATA(lo_result) = lo_sns->publish(
             iv_topicarn = lv_topic_arn
             iv_message = 'The price of your mobile plan has been increased from
 $19 to $23'
             iv_subject = 'Changes to mobile plan'
             iv_messagegroupid = 'Update-2'
             iv_messagededuplicationid = 'Update-2.1'
             it_messageattributes = lt_msg_attributes
      ).
        ov_message_id = lo_result->get_messageid( ).                      "
 ov_message_id is returned for testing purposes. "
        MESSAGE 'Message was published to SNS topic.' TYPE 'I'.
      CATCH /aws1/cx_snsnotfoundexception.
        MESSAGE 'Topic does not exist.' TYPE 'E'.
    ENDTRY.
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz des AWS-SDK für SAP ABAP.
  - CreateTopic
  - Veröffentlichen
  - Abonnieren

Empfangen von Nachrichten von FIFO-Abonnements

Sie können jetzt Preisaktualisierungen in den drei abonnierten Anwendungen erhalten. Wie in the section called “Anwendungsfall für FIFO-Themen” gezeigt ist der Einstiegspunkt für jede Verbraucheranwendung die Amazon-SQS-Warteschlange, die ihre entsprechende AWS Lambda-Funktion automatisch abfragen kann. Wenn eine Amazon-SQS-Warteschlange eine Ereignisquelle für eine Lambda-Funktion ist, skaliert Lambda seine Pollerflotte nach Bedarf, um Nachrichten effizient zu verbrauchen.

Weitere Informationen finden Sie unter Verwenden von AWS Lambdamit Amazon SQS im AWS Lambda-Entwicklerhandbuch. Informationen zum Schreiben eigener Queue-Abfragen finden Sie unterEmpfehlungen für Amazon SQS Standard- und FIFO-Queues im Amazon Simple Queue Service-Entwicklerhandbuch undReceiveMessage in der Amazon Simple Queue Service-API-Referenz.

AWS CloudFormation einsetzen

Mit AWS CloudFormation können Sie eine Vorlagendatei verwenden, um eine Sammlung von AWS-Ressourcen als eine Einheit zu erstellen und zu konfigurieren. Dieser Abschnitt enthält eine Beispieltemplate, das Folgendes erstellt:

- Das Amazon-SNS-FIFO-Thema, das die Preisaktualisierungen verteilt
- Die Amazon-SQS-FIFO-Warteschlangen, die diese Updates für die Groß- und Einzelhandel-Anwendung bereitstellen
- Die Amazon-SQS-Standardwarteschlange für die Analyseanwendung, in der Datensätze gespeichert werden, die für Business Intelligence (BI) abgefragt werden können
- Die Amazon-SNS-FIFO-Abonnements, die beide Warteschlangen mit dem Thema verbinden
- Filterrichtlinien, die angeben, dass Abonnentenanwendungen nur die Preisaktualisierungen erhalten, die sie benötigen

> **Note**
>
> Wenn Sie dieses Codebeispiel testen, indem Sie eine Nachricht im Thema veröffentlichen, stellen Sie sicher, dass Sie die Nachricht mit dem `business`-Attribut veröffentlichen. Geben Sie entweder `retail`oder `wholesale`als Attributwert an. Andernfalls wird die Nachricht herausgefiltert und nicht an die abonnierten Queues übermittelt.

```
{
  "AWSTemplateFormatVersion": "2010-09-09",
  "Resources": {
    "PriceUpdatesTopic": {
      "Type": "AWS::SNS::Topic",
      "Properties": {
        "TopicName": "PriceUpdatesTopic.fifo",
        "FifoTopic": true,
        "ContentBasedDeduplication": false,
        "ArchivePolicy": {
        "MessageRetentionPeriod": "30"
        }
      }
    },
    "WholesaleQueue": {
      "Type": "AWS::SQS::Queue",
      "Properties": {
        "QueueName": "WholesaleQueue.fifo",
        "FifoQueue": true,
        "ContentBasedDeduplication": false
      }
    },
    "RetailQueue": {
      "Type": "AWS::SQS::Queue",
      "Properties": {
        "QueueName": "RetailQueue.fifo",
        "FifoQueue": true,
        "ContentBasedDeduplication": false
      }
    },
    "AnalyticsQueue": {
      "Type": "AWS::SQS::Queue",
      "Properties": {
        "QueueName": "AnalyticsQueue"
```

```
      }
    },
    "WholesaleSubscription": {
      "Type": "AWS::SNS::Subscription",
      "Properties": {
        "TopicArn": {
          "Ref": "PriceUpdatesTopic"
        },
        "Endpoint": {
          "Fn::GetAtt": [
            "WholesaleQueue",
            "Arn"
          ]
        },
        "Protocol": "sqs",
        "RawMessageDelivery": "false",
        "FilterPolicyScope": "MessageBody",
        "FilterPolicy": {
          "business": [
            "wholesale"
          ]
        }
      }
    },
    "RetailSubscription": {
      "Type": "AWS::SNS::Subscription",
      "Properties": {
        "TopicArn": {
          "Ref": "PriceUpdatesTopic"
        },
        "Endpoint": {
          "Fn::GetAtt": [
            "RetailQueue",
            "Arn"
          ]
        },
        "Protocol": "sqs",
        "RawMessageDelivery": "false",
        "FilterPolicyScope": "MessageBody",
        "FilterPolicy": {
          "business": [
            "retail"
          ]
        }
```

```
      }
    },
    "AnalyticsSubscription": {
      "Type": "AWS::SNS::Subscription",
      "Properties": {
        "TopicArn": {
          "Ref": "PriceUpdatesTopic"
        },
        "Endpoint": {
          "Fn::GetAtt": [
            "AnalyticsQueue",
            "Arn"
          ]
        },
        "Protocol": "sqs",
        "RawMessageDelivery": "false"
      }
    },
    "SalesQueuesPolicy": {
      "Type": "AWS::SQS::QueuePolicy",
      "Properties": {
        "PolicyDocument": {
          "Statement": [
            {
              "Effect": "Allow",
              "Principal": {
                "Service": "sns.amazonaws.com"
              },
              "Action": [
                "sqs:SendMessage"
              ],
              "Resource": "*",
              "Condition": {
                "ArnEquals": {
                  "aws:SourceArn": {
                    "Ref": "PriceUpdatesTopic"
                  }
                }
              }
            }
          ]
        },
        "Queues": [
          {
```

```
            "Ref": "WholesaleQueue"
          },
          {
            "Ref": "RetailQueue"
          },
          {
            "Ref": "AnalyticsQueue"
          }
        ]
      }
    }
  }
}
```

Weitere Informationen zur Bereitstellung von AWS-Ressourcen mithilfe einer AWS CloudFormation-Vorlage finden Sie unter Erste Schritte im AWS CloudFormation-Benutzerhandbuch.

Amazon-SNS-Nachrichten veröffentlichen

Nachdem Sie ein Amazon-SNS-Thema erstellt haben und ein Endpunkt dieses abonniert hat, können Sie Nachrichten zu einem Thema veröffentlichen. Wenn eine Nachricht veröffentlicht wird, versucht Amazon SNS, die Nachricht an die abonnierte Endpunkte auszuliefern.

Themen

- Um Nachrichten in Amazon-SNS-Themen mithilfe von AWS Management Console zu veröffentlichen
- So veröffentlichen Sie eine Nachricht für ein Thema mit AWS -SDK
- Veröffentlichen großer Nachrichten mit Amazon SNS und Amazon S3
- Amazon-SNS-Nachrichtenattribute
- Batch-Messaging in Amazon SNS

Um Nachrichten in Amazon-SNS-Themen mithilfe von AWS Management Console zu veröffentlichen

1. Melden Sie sich bei der Amazon-SNS-Konsole an.
2. Wählen Sie im linken Navigationsbereich Topics (Themen).
3. Klicken Sie auf der Seite Topics ein Thema aus und dann Publish to topic.

   Die Konsole öffnet die Nachricht im Thema veröffentlichenSeite
4. Gehen Sie im Abschnitt Basic details (Grundlegende Details) wie folgt vor:

   a. (Optional) Geben Sie den Betreff einer Nachricht ein.

   b. Für ein FIFO-Thema geben Sie eine Nachrichtengruppen-IDein. Nachrichten in derselben Nachrichtengruppe werden in der Reihenfolge zugestellt, in der sie veröffentlicht werden.

   c. Geben Sie für ein FIFO-Thema eine Nachrichtendeduplizierungs-IDein. Diese ID ist optional, wenn Sie die Inhaltsbasierte Nachrichtendeduplizierung-Einstellung für das Thema aktiviert haben.

   d. (Optional) Für Mobile Push-Benachrichtigungen geben Sie einen Lebenszeitwert (TTL) in Sekunden ein. Das ist die Zeit, die einem Push-Benachrichtigungs-Service wie z. B. Apple Push-Benachrichtigungs-Service (APNs) oder Firebase Cloud Messaging (FCM) zur Verfügung steht, um die Nachricht an den Endpunkt zu übermitteln.

5. Führen Sie im Abschnitt Message body (Nachrichtentext) eine der folgenden Aktionen aus:

   a. Wählen Sie Identical payload for all delivery protocols (Identische Nutzlast für alle Bereitstellungsprotokolle) und geben Sie anschließend die Nachricht ein.

   b. Wählen Sie Custom payload for each delivery protocol (Benutzerdefinierte Nutzlast für die einzelnen Bereitstellungsprotokolle) und verwenden Sie anschließend ein JSON-Objekt, um die Nachricht zu definieren, die an die einzelnen Protokolle gesendet werden soll.

   Weitere Informationen finden Sie unter Veröffentlichung mit plattformspezifischen Payloads.

6. Fügen Sie im Abschnitt Message attributes (Nachrichtenattribute) die gewünschten Attribute hinzu, die Amazon SNS mit dem Abonnementattribut `FilterPolicy`abgleichen soll, um zu entscheiden, ob der abonnierte Endpunkt an der veröffentlichten Nachricht interessiert ist.

   a. Als Typ wählen Sie einen Attributtyp aus, z. B. String.Array.

   > **Note**
   >
   > Wenn der Attributtyp String.Array (String.Array) ist, schließen Sie das Array in eckige Klammern (`[]`) ein. Innerhalb des Arrays schließen Sie Zeichenfolgenwerte in doppelte Anführungszeichen ein. Für Zahlen oder die Schlüsselwörter `true`, `false`und `null` benötigen Sie keine Anführungszeichen.

   b. Geben Sie einen Attributnamen, wie beispielsweise `customer_interestsan`.

   c. Geben Sie einen Attributvalue, wie beispielsweise `["soccer", "rugby", "hockey"]`an.

   Wenn der Attributtyp String (Zeichenfolge), String.Array oder Number (Zahl) ist, wertet Amazon SNS das Nachrichtenattribut anhand der Filterrichtlinie (sofern vorhanden) des betreffenden Abonnements aus, bevor die Nachricht an dieses Abonnement gesendet wird, vorausgesetzt, der Geltungsbereich der Filterrichtlinie ist nicht explizit auf `MessageBody` festgelegt.

   Weitere Informationen finden Sie unter Amazon-SNS-Nachrichtenattribute.

7. Wählen Sie Publish message (Nachricht veröffentlichen) aus.

   Die Nachricht wird in dem Thema veröffentlicht. Die Konsole öffnet die Details-Seite.

So veröffentlichen Sie eine Nachricht für ein Thema mit AWS -SDK

Um ein AWS SDK verwenden zu können, müssen Sie es mit Ihren Anmeldeinformationen konfigurieren. Weitere Informationen finden Sie unter Freigegebene Konfigurations- und Anmeldeinformationsdateien im AWS -Referenzhandbuch zu SDKs und Tools.

Die folgenden Codebeispiele zeigen, wie Sie es verwenden`Publish`.

.NET

AWS SDK for .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

Veröffentlichen einer Nachricht für ein Thema.

```
    using System;
    using System.Threading.Tasks;
    using Amazon.SimpleNotificationService;
    using Amazon.SimpleNotificationService.Model;

    /// <summary>
    /// This example publishes a message to an Amazon Simple Notification
    /// Service (Amazon SNS) topic.
    /// </summary>
    public class PublishToSNSTopic
    {
        public static async Task Main()
        {
            string topicArn = "arn:aws:sns:us-
east-2:000000000000:ExampleSNSTopic";
            string messageText = "This is an example message to publish to the
 ExampleSNSTopic.";

            IAmazonSimpleNotificationService client = new
 AmazonSimpleNotificationServiceClient();

            await PublishToTopicAsync(client, topicArn, messageText);
```

```
        }

        /// <summary>
        /// Publishes a message to an Amazon SNS topic.
        /// </summary>
        /// <param name="client">The initialized client object used to publish
        /// to the Amazon SNS topic.</param>
        /// <param name="topicArn">The ARN of the topic.</param>
        /// <param name="messageText">The text of the message.</param>
        public static async Task PublishToTopicAsync(
            IAmazonSimpleNotificationService client,
            string topicArn,
            string messageText)
        {
            var request = new PublishRequest
            {
                TopicArn = topicArn,
                Message = messageText,
            };

            var response = await client.PublishAsync(request);

            Console.WriteLine($"Successfully published message ID:
{response.MessageId}");
        }
    }
```

Veröffentlichen Sie eine Nachricht zu einem Thema mit Gruppen-, Duplizierungs- und Attributoptionen.

```
    /// <summary>
    /// Publish messages using user settings.
    /// </summary>
    /// <returns>Async task.</returns>
    public static async Task PublishMessages()
    {
        Console.WriteLine("Now we can publish messages.");

        var keepSendingMessages = true;
        string? deduplicationId = null;
        string? toneAttribute = null;
```

```
        while (keepSendingMessages)
        {
            Console.WriteLine();
            var message = GetUserResponse("Enter a message to publish.", "This is
a sample message");

            if (_useFifoTopic)
            {
                Console.WriteLine("Because you are using a FIFO topic, you must
set a message group ID." +
                                  "\r\nAll messages within the same group will be
received in the order " +
                                  "they were published.");

                Console.WriteLine();
                var messageGroupId = GetUserResponse("Enter a message group ID
for this message:", "1");

                if (!_useContentBasedDeduplication)
                {
                    Console.WriteLine("Because you are not using content-based
deduplication, " +
                                      "you must enter a deduplication ID.");

                    Console.WriteLine("Enter a deduplication ID for this
message.");
                    deduplicationId = GetUserResponse("Enter a deduplication ID
for this message.", "1");
                }

                if (GetYesNoResponse("Add an attribute to this message?"))
                {
                    Console.WriteLine("Enter a number for an attribute.");
                    for (int i = 0; i < _tones.Length; i++)
                    {
                        Console.WriteLine($"\t{i + 1}. {_tones[i]}");
                    }

                    var selection = GetUserResponse("", "1");
                    int.TryParse(selection, out var selectionNumber);

                    if (selectionNumber > 0 && selectionNumber < _tones.Length)
                    {
                        toneAttribute = _tones[selectionNumber - 1];
```

```
                    }
                }

                var messageID = await SnsWrapper.PublishToTopicWithAttribute(
                    _topicArn, message, "tone", toneAttribute, deduplicationId,
 messageGroupId);

                Console.WriteLine($"Message published with id {messageID}.");
            }

            keepSendingMessages = GetYesNoResponse("Send another message?",
 false);
        }
    }
```

Wenden Sie die Benutzerauswahl auf die Veröffentlichungsaktion an.

```
    /// <summary>
    /// Publish a message to a topic with an attribute and optional deduplication
 and group IDs.
    /// </summary>
    /// <param name="topicArn">The ARN of the topic.</param>
    /// <param name="message">The message to publish.</param>
    /// <param name="attributeName">The optional attribute for the message.</
param>
    /// <param name="attributeValue">The optional attribute value for the
 message.</param>
    /// <param name="deduplicationId">The optional deduplication ID for the
 message.</param>
    /// <param name="groupId">The optional group ID for the message.</param>
    /// <returns>The ID of the message published.</returns>
    public async Task<string> PublishToTopicWithAttribute(
        string topicArn,
        string message,
        string? attributeName = null,
        string? attributeValue = null,
        string? deduplicationId = null,
        string? groupId = null)
    {
        var publishRequest = new PublishRequest()
        {
            TopicArn = topicArn,
```

```
            Message = message,
            MessageDeduplicationId = deduplicationId,
            MessageGroupId = groupId
        };

        if (attributeValue != null)
        {
            // Add the string attribute if it exists.
            publishRequest.MessageAttributes =
                new Dictionary<string, MessageAttributeValue>
                {
                    { attributeName!, new MessageAttributeValue() { StringValue =
 attributeValue, DataType = "String"} }
                };
        }

        var publishResponse = await
 _amazonSNSClient.PublishAsync(publishRequest);
        return publishResponse.MessageId;
    }
```

- Details zu API finden Sie unter Veröffentlichen in der AWS SDK for .NET -API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

```
//! Send a message to an Amazon Simple Notification Service (Amazon SNS) topic.
/*!
  \param message: The message to publish.
  \param topicARN: The Amazon Resource Name (ARN) for an Amazon SNS topic.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SNS::publishToTopic(const Aws::String &message,
```

```
                                const Aws::String &topicARN,
                                const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::SNS::Model::PublishRequest request;
    request.SetMessage(message);
    request.SetTopicArn(topicARN);

    const Aws::SNS::Model::PublishOutcome outcome = snsClient.Publish(request);

    if (outcome.IsSuccess()) {
        std::cout << "Message published successfully with id '"
                  << outcome.GetResult().GetMessageId() << "'." << std::endl;
    }
    else {
        std::cerr << "Error while publishing message "
                  << outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

Veröffentlichen Sie eine Nachricht mit einem Attribut.

```
        static const Aws::String TONE_ATTRIBUTE("tone");
        static const Aws::Vector<Aws::String> TONES = {"cheerful", "funny",
 "serious",
                                                       "sincere"};

        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::SNS::SNSClient snsClient(clientConfiguration);

        Aws::SNS::Model::PublishRequest request;
        request.SetTopicArn(topicARN);
        Aws::String message = askQuestion("Enter a message text to publish.  ");
        request.SetMessage(message);
```

```
        if (filteringMessages && askYesNoQuestion(
                "Add an attribute to this message? (y/n) ")) {
            for (size_t i = 0; i < TONES.size(); ++i) {
                std::cout << "  " << (i + 1) << ". " << TONES[i] << std::endl;
            }
            int selection = askQuestionForIntRange(
                    "Enter a number for an attribute. ",
                    1, static_cast<int>(TONES.size()));
            Aws::SNS::Model::MessageAttributeValue messageAttributeValue;
            messageAttributeValue.SetDataType("String");
            messageAttributeValue.SetStringValue(TONES[selection - 1]);
            request.AddMessageAttributes(TONE_ATTRIBUTE, messageAttributeValue);
        }

        Aws::SNS::Model::PublishOutcome outcome = snsClient.Publish(request);

        if (outcome.IsSuccess()) {
            std::cout << "Your message was successfully published." << std::endl;
        }
        else {
            std::cerr << "Error with TopicsAndQueues::Publish. "
                      << outcome.GetError().GetMessage()
                      << std::endl;

            cleanUp(topicARN,
                    queueURLS,
                    subscriptionARNS,
                    snsClient,
                    sqsClient);

            return false;
        }
```

- Details zu API finden Sie unter Veröffentlichen in der AWS SDK for C++ -API-Referenz.

CLI

AWS CLI

Beispiel 1: So veröffentlichen Sie eine Nachricht für ein Thema

Das folgende `publish`-Beispiel veröffentlicht die angegebene Nachricht im angegebenen SNS-Thema. Die Nachricht stammt aus einer Textdatei, in der Sie Zeilenumbrüche einfügen können.

```
aws sns publish \
    --topic-arn "arn:aws:sns:us-west-2:123456789012:my-topic" \
    --message file://message.txt
```

Inhalt von `message.txt`:

```
Hello World
Second Line
```

Ausgabe:

```
{
    "MessageId": "123a45b6-7890-12c3-45d6-111122223333"
}
```

Beispiel 2: So veröffentlichen Sie eine SMS-Nachricht an eine Telefonnummer

Im folgenden `publish`-Beispiel wird Nachricht `Hello world!` an Telefonnummer `+1-555-555-0100` veröffentlicht.

```
aws sns publish \
    --message "Hello world!" \
    --phone-number +1-555-555-0100
```

Ausgabe:

```
{
    "MessageId": "123a45b6-7890-12c3-45d6-333322221111"
}
```

- API-Details finden Sie unter Publish in der AWS CLI -Befehlsreferenz.

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

```
// SnsActions encapsulates the Amazon Simple Notification Service (Amazon SNS)
 actions
// used in the examples.
type SnsActions struct {
 SnsClient *sns.Client
}



// Publish publishes a message to an Amazon SNS topic. The message is then sent
 to all
// subscribers. When the topic is a FIFO topic, the message must also contain a
 group ID
// and, when ID-based deduplication is used, a deduplication ID. An optional key-
value
// filter attribute can be specified so that the message can be filtered
 according to
// a filter policy.
func (actor SnsActions) Publish(topicArn string, message string, groupId string,
 dedupId string, filterKey string, filterValue string) error {
 publishInput := sns.PublishInput{TopicArn: aws.String(topicArn), Message:
 aws.String(message)}
 if groupId != "" {
  publishInput.MessageGroupId = aws.String(groupId)
 }
 if dedupId != "" {
  publishInput.MessageDeduplicationId = aws.String(dedupId)
 }
 if filterKey != "" && filterValue != "" {
  publishInput.MessageAttributes = map[string]types.MessageAttributeValue{
```

```
   filterKey: {DataType: aws.String("String"), StringValue:
 aws.String(filterValue)},
  }
 }
 _, err := actor.SnsClient.Publish(context.TODO(), &publishInput)
 if err != nil {
  log.Printf("Couldn't publish message to topic %v. Here's why: %v", topicArn,
 err)
 }
 return err
}
```

- Details zu API finden Sie unter Veröffentlichen in der AWS SDK for Go -API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.PublishRequest;
import software.amazon.awssdk.services.sns.model.PublishResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class PublishTopic {
```

```
    public static void main(String[] args) {
        final String usage = """

                Usage:    <message> <topicArn>

                Where:
                   message - The message text to send.
                   topicArn - The ARN of the topic to publish.
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String message = args[0];
        String topicArn = args[1];
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();
        pubTopic(snsClient, message, topicArn);
        snsClient.close();
    }

    public static void pubTopic(SnsClient snsClient, String message, String
 topicArn) {
        try {
            PublishRequest request = PublishRequest.builder()
                    .message(message)
                    .topicArn(topicArn)
                    .build();

            PublishResponse result = snsClient.publish(request);
            System.out
                    .println(result.messageId() + " Message sent. Status is " +
 result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Details zu API finden Sie unter Veröffentlichen in der AWS SDK for Java 2.x -API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie den Client in einem separaten Modul und exportieren Sie ihn.

```
import { SNSClient } from "@aws-sdk/client-sns";

// The AWS Region can be provided here using the `region` property. If you leave
 it blank
// the SDK will default to the region set in your AWS config.
export const snsClient = new SNSClient({});
```

Importieren Sie das SDK- und Client-Module und rufen Sie die API auf.

```
import { PublishCommand } from "@aws-sdk/client-sns";
import { snsClient } from "../libs/snsClient.js";

/**
 * @param {string | Record<string, any>} message - The message to send. Can be a
 plain string or an object
 *                                                  if you are using the `json`
 `MessageStructure`.
 * @param {string} topicArn - The ARN of the topic to which you would like to
 publish.
 */
export const publish = async (
  message = "Hello from SNS!",
  topicArn = "TOPIC_ARN",
) => {
```

```
  const response = await snsClient.send(
    new PublishCommand({
      Message: message,
      TopicArn: topicArn,
    }),
  );
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: 'e7f77526-e295-5325-9ee4-281a43ad1f05',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   MessageId: 'xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx'
  // }
  return response;
};
```

Veröffentlichen Sie eine Nachricht zu einem Thema mit Gruppen-, Duplizierungs- und Attributoptionen.

```
  async publishMessages() {
    const message = await this.prompter.input({
      message: MESSAGES.publishMessagePrompt,
    });

    let groupId, deduplicationId, choices;

    if (this.isFifo) {
      await this.logger.log(MESSAGES.groupIdNotice);
      groupId = await this.prompter.input({
        message: MESSAGES.groupIdPrompt,
      });

      if (this.autoDedup === false) {
        await this.logger.log(MESSAGES.deduplicationIdNotice);
        deduplicationId = await this.prompter.input({
          message: MESSAGES.deduplicationIdPrompt,
        });
```

```
      }

      choices = await this.prompter.checkbox({
        message: MESSAGES.messageAttributesPrompt,
        choices: toneChoices,
      });
    }

    await this.snsClient.send(
      new PublishCommand({
        TopicArn: this.topicArn,
        Message: message,
        ...(groupId
          ? {
              MessageGroupId: groupId,
            }
          : {}),
        ...(deduplicationId
          ? {
              MessageDeduplicationId: deduplicationId,
            }
          : {}),
        ...(choices
          ? {
              MessageAttributes: {
                tone: {
                  DataType: "String.Array",
                  StringValue: JSON.stringify(choices),
                },
              },
            }
          : {}),
      }),
    );

    const publishAnother = await this.prompter.confirm({
      message: MESSAGES.publishAnother,
    });

    if (publishAnother) {
      await this.publishMessages();
    }
  }
```

- Weitere Informationen finden Sie im AWS SDK for JavaScript -Entwicklerhandbuch.
- Details zu API finden Sie unter Veröffentlichen in der AWS SDK for JavaScript -API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

```
suspend fun pubTopic(topicArnVal: String, messageVal: String) {

    val request = PublishRequest {
        message = messageVal
        topicArn = topicArnVal
    }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.publish(request)
        println("${result.messageId} message sent.")
    }
}
```

- Details zu API finden Sie unter Publish (Veröffentlichen) in der AWS -SDK-für-Kotlin-API-Referenz.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Sends a message to an Amazon SNS topic.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/
guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$message = 'This message is sent from a Amazon SNS code sample.';
$topic = 'arn:aws:sns:us-east-1:111122223333:MyTopic';

try {
    $result = $SnSclient->publish([
        'Message' => $message,
        'TopicArn' => $topic,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
```

```
}
```

- Weitere Informationen finden Sie im AWS SDK for PHP -Entwicklerhandbuch.
- Details zu API finden Sie unter Veröffentlichen in der AWS SDK for PHP -API-Referenz.

PowerShell

Tools für PowerShell

Beispiel 1: Dieses Beispiel zeigt das Veröffentlichen einer Nachricht mit einer einzigen MessageAttribute deklarierten Inline.

```
Publish-SNSMessage -TopicArn "arn:aws:sns:us-west-2:123456789012:my-topic" -
Message "Hello" -MessageAttribute
 @{'City'=[Amazon.SimpleNotificationService.Model.MessageAttributeValue]@{DataType='Strin
 StringValue ='AnyCity'}}
```

Beispiel 2: Dieses Beispiel zeigt die Veröffentlichung einer Nachricht, bei der mehrere Nachrichten im Voraus MessageAttributes deklariert wurden.

```
$cityAttributeValue = New-Object
 Amazon.SimpleNotificationService.Model.MessageAttributeValue
$cityAttributeValue.DataType = "String"
$cityAttributeValue.StringValue = "AnyCity"

$populationAttributeValue = New-Object
 Amazon.SimpleNotificationService.Model.MessageAttributeValue
$populationAttributeValue.DataType = "Number"
$populationAttributeValue.StringValue = "1250800"

$messageAttributes = New-Object System.Collections.Hashtable
$messageAttributes.Add("City", $cityAttributeValue)
$messageAttributes.Add("Population", $populationAttributeValue)

Publish-SNSMessage -TopicArn "arn:aws:sns:us-west-2:123456789012:my-topic" -
Message "Hello" -MessageAttribute $messageAttributes
```

- Einzelheiten zur API finden Sie unter In AWS Tools for PowerShell Cmdlet-Referenz veröffentlichen.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

Veröffentlichen Sie eine Nachricht mit Attributen, damit ein Abonnement basierend auf Attributen filtern kann.

```
class SnsWrapper:
    """Encapsulates Amazon SNS topic and subscription functions."""

    def __init__(self, sns_resource):
        """
        :param sns_resource: A Boto3 Amazon SNS resource.
        """
        self.sns_resource = sns_resource


    @staticmethod
    def publish_message(topic, message, attributes):
        """
        Publishes a message, with attributes, to a topic. Subscriptions can be
 filtered
        based on message attributes so that a subscription receives messages only
        when specified attributes are present.

        :param topic: The topic to publish to.
        :param message: The message to publish.
        :param attributes: The key-value attributes to attach to the message.
 Values
                           must be either `str` or `bytes`.
        :return: The ID of the message.
        """
        try:
            att_dict = {}
            for key, value in attributes.items():
                if isinstance(value, str):
```

```
                    att_dict[key] = {"DataType": "String", "StringValue": value}
                elif isinstance(value, bytes):
                    att_dict[key] = {"DataType": "Binary", "BinaryValue": value}
            response = topic.publish(Message=message, MessageAttributes=att_dict)
            message_id = response["MessageId"]
            logger.info(
                "Published message with attributes %s to topic %s.",
                attributes,
                topic.arn,
            )
        except ClientError:
            logger.exception("Couldn't publish message to topic %s.", topic.arn)
            raise
        else:
            return message_id
```

Veröffentlichen Sie eine Nachricht, die basierend auf dem Protokoll des Abonnenten unterschiedliche Formen annimmt.

```
class SnsWrapper:
    """Encapsulates Amazon SNS topic and subscription functions."""

    def __init__(self, sns_resource):
        """
        :param sns_resource: A Boto3 Amazon SNS resource.
        """
        self.sns_resource = sns_resource


    @staticmethod
    def publish_multi_message(
        topic, subject, default_message, sms_message, email_message
    ):
        """
        Publishes a multi-format message to a topic. A multi-format message takes
        different forms based on the protocol of the subscriber. For example,
        an SMS subscriber might receive a short version of the message
        while an email subscriber could receive a longer version.

        :param topic: The topic to publish to.
        :param subject: The subject of the message.
```

```
        :param default_message: The default version of the message. This version
is
                                sent to subscribers that have protocols that are
not
                                otherwise specified in the structured message.
        :param sms_message: The version of the message sent to SMS subscribers.
        :param email_message: The version of the message sent to email
subscribers.
        :return: The ID of the message.
        """
        try:
            message = {
                "default": default_message,
                "sms": sms_message,
                "email": email_message,
            }
            response = topic.publish(
                Message=json.dumps(message), Subject=subject,
MessageStructure="json"
            )
            message_id = response["MessageId"]
            logger.info("Published multi-format message to topic %s.", topic.arn)
        except ClientError:
            logger.exception("Couldn't publish message to topic %s.", topic.arn)
            raise
        else:
            return message_id
```

- Details zu API finden Sie unter Veröffentlichen in der AWS -API-Referenz zu SDK for Python (Boto3).

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

```
# Service class for sending messages using Amazon Simple Notification Service
 (SNS)
class SnsMessageSender
  # Initializes the SnsMessageSender with an SNS client
  #
  # @param sns_client [Aws::SNS::Client] The SNS client
  def initialize(sns_client)
    @sns_client = sns_client
    @logger = Logger.new($stdout)
  end

  # Sends a message to a specified SNS topic
  #
  # @param topic_arn [String] The ARN of the SNS topic
  # @param message [String] The message to send
  # @return [Boolean] true if message was successfully sent, false otherwise
  def send_message(topic_arn, message)
    @sns_client.publish(topic_arn: topic_arn, message: message)
    @logger.info("Message sent successfully to #{topic_arn}.")
    true
  rescue Aws::SNS::Errors::ServiceError => e
    @logger.error("Error while sending the message: #{e.message}")
    false
  end
end

# Example usage:
if $PROGRAM_NAME == __FILE__
  topic_arn = "SNS_TOPIC_ARN" # Should be replaced with a real topic ARN
  message = "MESSAGE"         # Should be replaced with the actual message
 content
```

```
  sns_client = Aws::SNS::Client.new
  message_sender = SnsMessageSender.new(sns_client)

  @logger.info("Sending message.")
  unless message_sender.send_message(topic_arn, message)
    @logger.error("Message sending failed. Stopping program.")
    exit 1
  end
end
```

- Weitere Informationen finden Sie im AWS SDK for Ruby -Entwicklerhandbuch.
- Details zu API finden Sie unter Veröffentlichen in der AWS SDK for Ruby -API-Referenz.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

```
async fn subscribe_and_publish(
    client: &Client,
    topic_arn: &str,
    email_address: &str,
) -> Result<(), Error> {
    println!("Receiving on topic with ARN: `{}`", topic_arn);

    let rsp = client
        .subscribe()
        .topic_arn(topic_arn)
        .protocol("email")
        .endpoint(email_address)
        .send()
        .await?;

    println!("Added a subscription: {:?}", rsp);
```

```
    let rsp = client
        .publish()
        .topic_arn(topic_arn)
        .message("hello sns!")
        .send()
        .await?;

    println!("Published message: {:?}", rsp);

    Ok(())
}
```

- Details zu API finden Sie unter Veröffentlichen in der Referenz für das AWS -SDK für Rust-API.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

```
    TRY.
        oo_result = lo_sns->publish(              " oo_result is returned for
testing purposes. "
          iv_topicarn = iv_topic_arn
          iv_message = iv_message
        ).
        MESSAGE 'Message published to SNS topic.' TYPE 'I'.
      CATCH /aws1/cx_snsnotfoundexception.
        MESSAGE 'Topic does not exist.' TYPE 'E'.
    ENDTRY.
```

- Details zu API finden Sie unter Publish (Veröffentlichen) in der API-Referenz für das AWS SDK für SAP ABAP.

Veröffentlichen großer Nachrichten mit Amazon SNS und Amazon S3

Wenn Sie große Amazon-SNS-Nachrichten veröffentlichen möchten, können Sie die Amazon SNS Extended Client Library für Java oder die Amazon SNS Extended Client Library für Python verwenden. Diese Bibliothek ist nützlich für Nachrichten, die größer als das aktuelle Maximum von 256 KB sind. Es gilt ein Maximum von 2 GB. Beide Bibliotheken speichern die tatsächliche Nutzlast in einem Amazon-S3-Bucket und veröffentlicht den Verweis des gespeicherten Amazon-S3-Objekts im Amazon-SNS-Thema. Abonnierte Amazon SQS Queues können die Amazon SQS Extended Client Library for Jav für Java Verwenden Sie, um die Nutzlasten aus Amazon S3 aufzuheben und diese abzurufen. Andere Endpunkte, wie Lambda, können die Payload-Offloading Java Common Library für AWS, um die Nutzlast zu dereferenzieren und abzurufen.

> **Note**
>
> Die Amazon SNS Extended Client Libraries sind sowohl mit Standard- als auch mit FIFO-Themen kompatibel.

Themen

- Extended Client Library für Java
- Extended Client Library für Python

Extended Client Library für Java

Themen

- Voraussetzungen
- Konfigurieren der Nachrichtenspeicherung
- Beispiel: Veröffentlichen von Nachrichten in Amazon SNS mit Payload, die in Amazon S3 gespeichert ist
- Andere Endpunktprotokolle

Voraussetzungen

Im Folgenden sind die Voraussetzungen für die Verwendung der Amazon SNS Extended Client Library für Java aufgeführt:

- Ein AWS SDK.

  Das Beispiel auf dieser Seite verwendet das AWS Java SDK. Informationen zum Installieren und Einrichten des SDK finden Sie unter Einrichten der AWS -SDK für Java im AWS SDK for Java - Entwicklerhandbuch.
- Ein AWS-Konto mit den richtigen Anmeldeinformationen.

  Um ein zu erstellen AWS-Konto, navigieren Sie zur AWS Startseite und wählen Sie dann Konto AWS erstellen aus. Folgen Sie den Anweisungen.

  Weitere Informationen zu Anmeldeinformationen finden Sie unter Einrichten von AWS Anmeldeinformationen und Region für die Entwicklung im AWS SDK for Java -Entwicklerhandbuch.
- Java 8 oder höher.
- Die erweiterte Amazon SNS Extended Client Library für Java (ist auch verfügbar von Maven) enthalten.

Konfigurieren der Nachrichtenspeicherung

Die Amazon SNS Extended Client-Bibliothek verwendet die Payload Offloading Java Common Library für AWS zum Speichern und Abrufen von Nachrichten. Sie können die folgenden Amazon S3 Optionen für die Nachrichtenspeicherung:

- Grenzwert für benutzerdefinierte Nachrichtengrößen - Nachrichten mit Nutzlasten und Attributen, die diese Größe überschreiten, werden automatisch in Amazon S3 gespeichert.
- `alwaysThroughS3`-Flag - Setzen Sie diesen Wert auf `true`, um die Speicherung aller Nachrichten-Payloads in Amazon S3 zu erzwingen. Beispiel:

```
SNSExtendedClientConfiguration snsExtendedClientConfiguration = new
SNSExtendedClientConfiguration() .withPayloadSupportEnabled(s3Client,
 BUCKET_NAME).withAlwaysThroughS3(true);
```

- Benutzerdefinierter KMS-Schlüssel - Der Schlüssel, der für die serverseitige Verschlüsselung in Ihrem Amazon S3-Bucket verwendet werden soll.

- Bucket Name - Der Name des Amazon S3-Buckets zum Speichern der Nachrichten-Payloads.

Beispiel: Veröffentlichen von Nachrichten in Amazon SNS mit Payload, die in Amazon S3 gespeichert ist

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie ein Beispielthema und eine Queue.
- Die abonnierte Queue kann nun Nachrichten vom Thema empfangen.
- Veröffentlichen Sie eine Testnachricht.

Die Nachrichtennutzlast wird in Amazon S3 gespeichert und der Verweis darauf wird veröffentlicht. Der Amazon SQS Extended Client wird zum Empfangen der Nachricht verwendet.

SDK für Java 1.x

> **Note**
>
> Auf gibt es mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

Um eine große Nachricht zu veröffentlichen, verwenden Sie die Amazon SNS Extended Client Library für Java. Die Nachricht, die Sie senden, verweist auf ein Amazon S3-Objekt, das den tatsächlichen Nachrichteninhalt enthält.

```
import com.amazon.sqs.javamessaging.AmazonSQSExtendedClient;
import com.amazon.sqs.javamessaging.ExtendedClientConfiguration;
import com.amazonaws.regions.Region;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.sns.AmazonSNS;
import com.amazonaws.services.sns.AmazonSNSClientBuilder;
import com.amazonaws.services.sns.model.CreateTopicRequest;
import com.amazonaws.services.sns.model.PublishRequest;
import com.amazonaws.services.sns.model.SetSubscriptionAttributesRequest;
import com.amazonaws.services.sns.util.Topics;
import com.amazonaws.services.sqs.AmazonSQS;
```

```
import com.amazonaws.services.sqs.AmazonSQSClientBuilder;
import com.amazonaws.services.sqs.model.CreateQueueRequest;
import com.amazonaws.services.sqs.model.ReceiveMessageResult;
import software.amazon.sns.AmazonSNSExtendedClient;
import software.amazon.sns.SNSExtendedClientConfiguration;

public class Example {

        public static void main(String[] args) {
                final String BUCKET_NAME = "extended-client-bucket";
                final String TOPIC_NAME = "extended-client-topic";
                final String QUEUE_NAME = "extended-client-queue";
                final Regions region = Regions.DEFAULT_REGION;

                // Message threshold controls the maximum message size that will be
 allowed to
                // be published
                // through SNS using the extended client. Payload of messages
 exceeding this
                // value will be stored in
                // S3. The default value of this parameter is 256 KB which is the
 maximum
                // message size in SNS (and SQS).
                final int EXTENDED_STORAGE_MESSAGE_SIZE_THRESHOLD = 32;

                // Initialize SNS, SQS and S3 clients
                final AmazonSNS snsClient =
 AmazonSNSClientBuilder.standard().withRegion(region).build();
                final AmazonSQS sqsClient =
 AmazonSQSClientBuilder.standard().withRegion(region).build();
                final AmazonS3 s3Client =
 AmazonS3ClientBuilder.standard().withRegion(region).build();

                // Create bucket, topic, queue and subscription
                s3Client.createBucket(BUCKET_NAME);
                final String topicArn = snsClient.createTopic(
                                new
 CreateTopicRequest().withName(TOPIC_NAME)).getTopicArn();
                final String queueUrl = sqsClient.createQueue(
                                new
 CreateQueueRequest().withQueueName(QUEUE_NAME)).getQueueUrl();
                final String subscriptionArn = Topics.subscribeQueue(
                                snsClient, sqsClient, topicArn, queueUrl);
```

```
                // To read message content stored in S3 transparently through SQS
extended
                // client,
                // set the RawMessageDelivery subscription attribute to TRUE
                final SetSubscriptionAttributesRequest subscriptionAttributesRequest
= new SetSubscriptionAttributesRequest();
                subscriptionAttributesRequest.setSubscriptionArn(subscriptionArn);

subscriptionAttributesRequest.setAttributeName("RawMessageDelivery");
                subscriptionAttributesRequest.setAttributeValue("TRUE");
                snsClient.setSubscriptionAttributes(subscriptionAttributesRequest);

                // Initialize SNS extended client
                // PayloadSizeThreshold triggers message content storage in S3 when
the
                // threshold is exceeded
                // To store all messages content in S3, use AlwaysThroughS3 flag
                final SNSExtendedClientConfiguration snsExtendedClientConfiguration
= new SNSExtendedClientConfiguration()
                                .withPayloadSupportEnabled(s3Client, BUCKET_NAME)

.withPayloadSizeThreshold(EXTENDED_STORAGE_MESSAGE_SIZE_THRESHOLD);
                final AmazonSNSExtendedClient snsExtendedClient = new
AmazonSNSExtendedClient(snsClient,
                                snsExtendedClientConfiguration);

                // Publish message via SNS with storage in S3
                final String message = "This message is stored in S3 as it exceeds
the threshold of 32 bytes set above.";
                snsExtendedClient.publish(topicArn, message);

                // Initialize SQS extended client
                final ExtendedClientConfiguration sqsExtendedClientConfiguration =
new ExtendedClientConfiguration()
                                .withPayloadSupportEnabled(s3Client, BUCKET_NAME);
                final AmazonSQSExtendedClient sqsExtendedClient = new
AmazonSQSExtendedClient(sqsClient,
                                sqsExtendedClientConfiguration);

                // Read the message from the queue
                final ReceiveMessageResult result =
sqsExtendedClient.receiveMessage(queueUrl);
                System.out.println("Received message is " +
result.getMessages().get(0).getBody());
```

```
        }
}
```

Andere Endpunktprotokolle

Sowohl die Amazon SNS - als auch die Amazon SQS Bibliotheken verwenden die Payload-Offloading Java Common Library für AWS, um Nachrichten-Payloads mit Amazon S3 zu speichern und abzurufen. Jeder Java-aktivierte Endpunkt (z. B. ein HTTPS-Endpunkt, der in Java implementiert ist) kann dieselbe Bibliothek verwenden, um den Nachrichteninhalt aufzuheben.

Endpunkte, die die Payload Offloading Java Common Library für nicht verwenden können, AWS können weiterhin Nachrichten mit Nutzlasten veröffentlichen, die in Amazon S3 gespeichert sind. Im Folgenden wird ein Beispiel für eine Amazon S3-Referenz gezeigt, die im obigen Codebeispiel veröffentlicht wird:

```
[
  "software.amazon.payloadoffloading.PayloadS3Pointer",
  {
    "s3BucketName": "extended-client-bucket",
    "s3Key": "xxxx-xxxxx-xxxxx-xxxxxx"
  }
]
```

Extended Client Library für Python

Themen

- Voraussetzungen
- Konfigurieren der Nachrichtenspeicherung
- Beispiel: Veröffentlichen von Nachrichten in Amazon SNS mit einer in Amazon S3 gespeicherten Nutzlast

Voraussetzungen

Im Folgenden sind die Voraussetzungen für die Verwendung der Amazon SNS Extended Client Library für Python aufgeführt:

- Ein AWS SDK.

Das Beispiel auf dieser Seite verwendet AWS Python SDK Boto3. Informationen zur Installation und Einrichtung des SDK finden Sie in der Dokumentation zum AWS -SDK für Python.

- Ein AWS-Konto mit den richtigen Anmeldeinformationen.

  Um ein zu erstellen AWS-Konto, navigieren Sie zur AWS Startseite und wählen Sie dann Konto AWS erstellen aus. Folgen Sie den Anweisungen.

  Weitere Informationen zu Anmeldeinformationen finden Sie unter Anmeldeinformationen im Entwicklerhandbuch zu AWS -SDK für Python.
- Python 3.x (oder höher) und Pip.
- Die erweiterte Amazon SNS Extended Client Library für Python (ist auch verfügbar von Maven) enthalten.

Konfigurieren der Nachrichtenspeicherung

Die folgenden Attribute sind auf Boto3Amazon SNS-Client-, Themen- und -PlatformEndpointObjekten verfügbar, um die Amazon S3-Nachrichtenspeicheroptionen zu konfigurieren.

- `large_payload_support` – der Name des Amazon-S3-Buckets zum Speichern großer Nachrichten.
- `message_size_threshold` – der Schwellenwert für das Speichern der Nachricht im großen Nachrichten-Bucket. Der Wert darf nicht kleiner als 0 oder größer als 262144 sein. Der Standardwert ist 262144.
- `always_through_s3` – bei `True` werden alle Nachrichten in Amazon S3 gespeichert. Der Standardwert ist `False`.
- `s3` – das `resource`-Objekt von Boto3 Amazon S3, das zum Speichern von Objekten in Amazon S3 verwendet wird. Verwenden Sie dies, wenn Sie die Amazon-S3-Ressource steuern möchten (z. B. eine benutzerdefinierte Konfiguration oder Anmeldeinformationen für Amazon S3). Wenn dies nicht bereits bei der ersten Verwendung festgelegt wurde, ist die Standardeinstellung `boto3.resource("s3")`.

Beispiel: Veröffentlichen von Nachrichten in Amazon SNS mit einer in Amazon S3 gespeicherten Nutzlast

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie Beispiele für ein Amazon-SNS-Thema und eine Amazon-SQS-Warteschlange.
- Die abonnierte Queue kann nun Nachrichten vom Thema empfangen.
- Veröffentlichen Sie eine Testnachricht.
- Die Nachrichtennutzlast wird in Amazon S3 gespeichert und der Verweis darauf wird veröffentlicht.
- Drucken Sie die veröffentlichte Nachricht aus der Warteschlange zusammen mit der aus Amazon S3 abgerufenen Originalnachricht aus.

Um eine große Nachricht zu veröffentlichen, verwenden Sie die Amazon SNS Extended Client Library für Python. Die Nachricht, die Sie senden, verweist auf ein Amazon-S3-Objekt, das den tatsächlichen Nachrichteninhalt enthält.

```
import boto3
import sns_extended_client
from json import loads

s3_extended_payload_bucket = "extended-client-bucket-store"
TOPIC_NAME = "---TOPIC-NAME---"
QUEUE_NAME = "---QUEUE-NAME---"

# Create an helper to fetch message from S3
def get_msg_from_s3(body):
    json_msg = loads(body)
    s3_client = boto3.client("s3")
    s3_object = s3_client.get_object(
        Bucket=json_msg[1].get("s3BucketName"), Key=json_msg[1].get("s3Key")
    )
    msg = s3_object.get("Body").read().decode()
    return msg

# Create an helper to fetch and print message SQS queue and S3
def fetch_and_print_from_sqs(sqs, queue_url):
    """Handy Helper to fetch and print message from SQS queue and S3"""
    message = sqs.receive_message(
        QueueUrl=queue_url, MessageAttributeNames=["All"], MaxNumberOfMessages=1
    ).get("Messages")[0]
    message_body = message.get("Body")
    print("Published Message: {}".format(message_body))
    print("Message Stored in S3 Bucket is: {}\n".format(get_msg_from_s3(message_body)))

# Initialize the SNS client and create SNS Topic
```

```
sns_extended_client = boto3.client("sns", region_name="us-east-1")
create_topic_response = sns_extended_client.create_topic(Name=TOPIC_NAME)
demo_topic_arn = create_topic_response.get("TopicArn")

# Create and subscribe an SQS queue to the SNS client
sqs = boto3.client("sqs")
demo_queue_url = sqs.create_queue(QueueName=QUEUE_NAME).get("QueueUrl")
demo_queue_arn = sqs.get_queue_attributes(QueueUrl=demo_queue_url,
AttributeNames=["QueueArn"])["Attributes"].get("QueueArn")
# Set the RawMessageDelivery subscription attribute to TRUE
sns_extended_client.subscribe(TopicArn=demo_topic_arn, Protocol="sqs",
Endpoint=demo_queue_arn, Attributes={"RawMessageDelivery":"true"})

sns_extended_client.large_payload_support = s3_extended_payload_bucket

# To store all messages content in S3, set always_through_s3 to True
# In the example, we set message size threshold as 32 bytes, adjust this threshold as
 per your usecase
# Message will only be uploaded to S3 when its payload size exceeded threshold
sns_extended_client.message_size_threshold = 32
sns_extended_client.publish(
    TopicArn=demo_topic_arn,
    Message="This message should be published to S3 as it exceeds the
 message_size_threshold limit",
)
# Print message stored in s3
fetch_and_print_from_sqs(sqs, demo_queue_url)
```

Ausgabe

```
Published Message:
[
    "software.amazon.payloadoffloading.PayloadS3Pointer",
    {
        "s3BucketName": "extended-client-bucket-store",
        "s3Key": "xxxx-xxxxx-xxxxx-xxxxxx"
    }
]
Message Stored in S3 Bucket is: This message should be published to S3 as it exceeds
 the message_size_threshold limit
```

Amazon-SNS-Nachrichtenattribute

Amazon SNS unterstützt die Übermittlung von Nachrichtenattributen, die Ihnen die Bereitstellung strukturierter Metadatenelemente (Zeitstempel, Geodaten, Signaturen, IDs usw.) zur Nachricht ermöglichen. Für SQS-Abonnements können maximal 10 Nachrichtenattribute gesendet werden, wenn die Übermittlung unformatierter Nachrichten aktiviert ist. Wenn mehr als 10 Nachrichtenattribute gesendet werden sollen, muss die Übermittlung unformatierter Nachrichten deaktiviert sein. Nachrichten werden mit mehr als 10 Nachrichtenattributen, die an Amazon-SQS-Abonnements mit aktiviertem Raw Message Delivery gerichtet sind, als clientseitige Fehler verworfen.

Nachrichtenattribute sind optional und separat vom Nachrichtentext, werden aber mit diesem versendet. Der Receiver kann diese Informationen verwenden, um zu entscheiden, wie die Nachricht behandelt wird, ohne vorher den Nachrichtentext verarbeiten zu müssen.

Weitere Informationen über das Senden von Nachrichten mit Attributen mithilfe der AWS Management Console oder AWS SDK for Java finden Sie im Tutorial Um Nachrichten in Amazon-SNS-Themen mithilfe von AWS Management Console zu veröffentlichen.

> **Note**
>
> Nachrichtenattribute werden nur bei der Nachrichtenstruktur "Zeichenfolge" und nicht bei "JSON" gesendet werden.

Sie können die Nachrichtenattribute auch zur Strukturierung der Push-Benachrichtigung für mobile Endpunkte verwenden. In diesem Fall werden die Nachrichtenattribute nur verwendet, um Ihnen bei der Gliederung der Push-Benachrichtigung zu helfen. Die Attribute werden nicht an den Endpunkt übermittelt, wie beim Senden von Nachrichten mit Nachrichtenattributen an Amazon SQS-Endpunkte.

Sie können Nachrichtenattribute auch verwenden, um Ihre Nachrichten filterbar für Abonnement-Filterrichtlinien zu machen. Sie können Richtlinien auf Themen-Abonnements anwenden. Wenn eine Filterrichtlinie angewendet wird, bei der der Geltungsbereich der Filterrichtlinie auf `MessageAttributes` (Standard) festgelegt ist, erhält ein Abonnement nur diejenigen Nachrichten mit Attributen, die die Richtlinie akzeptiert. Weitere Informationen finden Sie unter Amazon SNS Nachrichtenfilterung.

> **Note**
>
> Wenn Nachrichtenattribute für die Filterung verwendet werden, muss der Wert eine gültige JSON-Zeichenfolge sein. Dadurch wird sichergestellt, dass die Nachricht an ein Abonnement mit aktivierter Filterung von Nachrichtenattributen zugestellt wird.

Nachrichtenattributelemente und Validierung

Jedes Nachrichtenattribut besteht aus den folgenden Elementen:

- Name – Der Name des Nachrichtenattributs kann folgende Zeichen enthalten: A-Z, a-z, 0-9, Unterstrich (_), Bindestrich (-) und Punkt (.). Der Name darf nicht mit einem Punkt beginnen oder enden und darf nicht mehrere aufeinanderfolgende Punkte enthalten. Es wird zwischen Groß- und Kleinschreibung unterschieden. Der Name muss innerhalb der Attributnamen für die Nachricht eindeutig sein. Er kann eine Länge von bis zu 256 Zeichen umfassen. Er darf nicht mit `AWS.` oder `Amazon.` (oder eine Variante bei der Groß-/Kleinschreibung) beginnen, da diese Präfixe für die Verwendung durch Amazon Web Services reserviert sind.
- Typ – Die unterstützten Attributdatentypen für Nachrichten sind `String`, `String.Array`, `Number` und `Binary`. Der Datentyp unterliegt denselben Beschränkungen bezüglich des Inhalts wie der Nachrichtentext. Die Groß- und Kleinschreibung wird berücksichtigt und er kann eine Gesamtlänge von bis zu 256 Bytes haben. Weitere Informationen finden Sie im Abschnitt Datentypen für Nachrichtenattribute und Validierung.
- Wert – Der benutzerdefinierte Wert des Nachrichtenattributs. Für den Datentyp Zeichenfolge unterliegt das Wertattribut denselben Beschränkungen bezüglich des Inhalts wie der Nachrichtentext. Weitere Informationen finden Sie unter der Aktion Veröffentlichen in der Amazon Simple Notification Service API-Referenz.

Name, Typ und Wert dürfen nicht leer oder null sein. Dasselbe gilt für den Nachrichtentext. Alle Teile des Nachrichtenattributs einschließlich Name, Typ und Wert zählen zur Größenbeschränkung der Nachricht, die bei 256 KB liegt.

Datentypen für Nachrichtenattribute und Validierung

Über den Datentyp des Nachrichtenattributs wird festgelegt, wie die Nachrichtenattributwerte von Amazon SNS verarbeitet werden. Wenn Sie beispielsweise den Datentyp Zahl wählen, wertet Amazon SNS den Wert als Zahl aus.

Amazon SNS unterstützt die folgenden logischen Datentypen für alle Endpunkte, außer wie erwähnt:

- Zeichenfolge – Die Zeichenfolgen sind Unicode mit binärer UTF-8-Kodierung. Eine Liste von Codewerten finden Sie unter http://en.wikipedia.org/wiki/ASCII#ASCII_printable_characters.

  > **Note**
  >
  > Ersatzwerte werden in den Nachrichtenattributen nicht unterstützt. Wenn Sie beispielsweise einen Ersatzwert zur Darstellung eines Emojis verwenden, erhalten Sie den folgenden Fehler: `Invalid attribute value was passed in for message attribute`.

- String.Array – Ein Array, formatiert als Zeichenfolge, das mehrere Werte enthalten kann. Die Werte können Zeichenfolgen, Zahlen oder die Schlüsselwörter `true`, `false` und `null` sein. Für ein String.Array vom Typ „number“ oder „boolean“ sind keine Preise erforderlich. Mehrere String.Array-Werte werden durch Komma getrennt.

  Dieser Datentyp wird nicht unterstützt für AWS Lambda-Abonnements. Wenn Sie diesen Datentypen für Lambda-Endpunkte festlegen, wird er als der `String`-Datentyp in der JSON-Nutzlast angesehen, den Amazon SNS an Lambda liefert.

- Zahl – Zahlen sind positive oder negative Ganzzahlen oder Gleitkommazahlen. Der Umfang und die Genauigkeit von Zahlen reichen aus, um die meisten möglichen Werte zu umfassen, die von den Zahlentypen Integer, Float und Double normalerweise unterstützt werden. Eine Zahl kann einen Wert von -10^9 bis 10^9 haben, mit 5 Stellen Genauigkeit nach dem Komma. Nullen am Anfang und am Ende werden abgeschnitten.

  Dieser Datentyp wird nicht unterstützt für AWS Lambda-Abonnements. Wenn Sie diesen Datentypen für Lambda-Endpunkte festlegen, wird er als der `String`-Datentyp in der JSON-Nutzlast angesehen, den Amazon SNS an Lambda liefert.

- Binär – Mit den Binärtypattributen können Sie alle binären Daten speichern, z. B. komprimierte oder verschlüsselte Daten und Bilder.

Reservierte Nachrichtenattribute für mobile Push-Benachrichtigungen

Die folgende Tabelle enthält die reservierten Nachrichtenattribute für mobile Push-Benachrichtigungsservices, mit denen Sie Ihre Push-Benachrichtigung strukturieren können:

Push-Benachrichtigungsservice	Reserviertes Nachrichtenattribut
ADM	AWS.SNS.MOBILE.ADM.TTL
APNs[1]	AWS.SNS.MOBILE.APNS_MDM.TTL
	AWS.SNS.MOBILE.APNS_MDM_SANDBOX.TTL
	AWS.SNS.MOBILE.APNS_PASSBOOK.TTL
	AWS.SNS.MOBILE.APNS_PASSBOOK_SANDBOX.TTL
	AWS.SNS.MOBILE.APNS_SANDBOX.TTL
	AWS.SNS.MOBILE.APNS_VOIP.TTL
	AWS.SNS.MOBILE.APNS_VOIP_SANDBOX.TTL
	AWS.SNS.MOBILE.APNS.COLLAPSE_ID
	AWS.SNS.MOBILE.APNS.PRIORITY
	AWS.SNS.MOBILE.APNS.PUSH_TYPE
	AWS.SNS.MOBILE.APNS.TOPIC
	AWS.SNS.MOBILE.APNS.TTL
Baidu	AWS.SNS.MOBILE.BAIDU.DeployStatus
	AWS.SNS.MOBILE.BAIDU.MessageKey
	AWS.SNS.MOBILE.BAIDU.MessageType
	AWS.SNS.MOBILE.BAIDU.TTL
FCM	AWS.SNS.MOBILE.FCM.TTL
	AWS.SNS.MOBILE.GCM.TTL

Push-Benachrichtigungsservice	Reserviertes Nachrichtenattribut
macOS	`AWS.SNS.MOBILE.MACOS_SANDBOX.TTL`
	`AWS.SNS.MOBILE.MACOS.TTL`
MPNS	`AWS.SNS.MOBILE.MPNS.NotificationClass`
	`AWS.SNS.MOBILE.MPNS.TTL`
	`AWS.SNS.MOBILE.MPNS.Type`
WNS	`AWS.SNS.MOBILE.WNS.CachePolicy`
	`AWS.SNS.MOBILE.WNS.Group`
	`AWS.SNS.MOBILE.WNS.Match`
	`AWS.SNS.MOBILE.WNS.SuppressPopup`
	`AWS.SNS.MOBILE.WNS.Tag`
	`AWS.SNS.MOBILE.WNS.TTL`
	`AWS.SNS.MOBILE.WNS.Type`

[1] Apple lehnt Amazon-SNS-Benachrichtigungen ab, wenn die Nachrichtenattribute nicht ihren Anforderungen entsprechen. Weitere Informationen finden Sie unter Senden von Benachrichtigungsanfragen an APNs auf der Apple-Developer-Website.

Batch-Messaging in Amazon SNS

Was ist Batch-Messaging?

Eine Alternative zur Veröffentlichung von Nachrichten in Standard- oder FIFO-Themen in einzelnen `Publish` API-Anfragen ist die Verwendung der Amazon SNS `PublishBatch`-API, um bis zu 10 Nachrichten in einer einzigen API-Anfrage zu veröffentlichen. Durch das Versenden von Nachrichten in Batches können Sie die Kosten, die mit der Verbindung von verteilten Anwendungen

(A2A-Messaging) oder dem Senden von Benachrichtigungen an Personen (A2P-Messaging) mit Amazon SNS verbunden sind, bis zu einem Faktor von 10 reduzieren. Amazon SNS verfügt über Kontingente, wie viele Nachrichten Sie je nach Region, in der Sie tätig sind, für ein Thema pro Sekunde veröffentlichen können. Sehen Sie sich die Seite Amazon-SNS-Endpunkte und -Kontingente im Allgemeine AWS-Referenz-Leitfaden an, um weitere Informationen zu API-Kontingenten zu erhalten.

> **Note**
>
> Die Gesamtgröße aller Nachrichten, die Sie in einer einzigen `PublishBatch`-API-Anfrage senden, darf 262.144 Byte (256 KB) nicht überschreiten.
> Die `PublishBatch`-API verwendet dieselbe `Publish`-API-Aktion für IAM-Richtlinien.

Wie funktioniert Batch-Messaging?

Das Veröffentlichen von Nachrichten mit der `PublishBatch`-API ähnelt dem Veröffentlichen von Nachrichten mit der `Publish`-API. Der Hauptunterschied besteht darin, dass jeder Nachricht innerhalb einer `PublishBatch`-API-Anfrage eine eindeutige Batch-ID (bis zu 80 Zeichen) zugewiesen werden muss. Auf diese Weise kann Amazon SNS individuelle API-Antworten für jede Nachricht innerhalb eines Batches zurückgeben, um zu bestätigen, dass jede Nachricht entweder veröffentlicht wurde oder dass ein Fehler aufgetreten ist. Für Nachrichten, die in FIFO-Themen veröffentlicht werden, müssen Sie zusätzlich zur Zuweisung einer eindeutigen Batch-ID noch eine `MessageDeduplicationID` und `MessageGroupId` für jede einzelne Nachricht zuweisen.

Beispiele

Veröffentlichen eines Batches von 10 Nachrichten in einem Standardthema

```
// Imports
import com.amazonaws.services.sns.AmazonSNS;
import com.amazonaws.services.sns.model.PublishBatchRequest;
import com.amazonaws.services.sns.model.PublishBatchRequestEntry;
import com.amazonaws.services.sns.model.PublishBatchResult;
import com.amazonaws.services.sns.model.AmazonSNSException;
import java.util.List;
import java.util.stream.Collectors;

// Code
```

```
private static final int MAX_BATCH_SIZE = 10;

public static void publishBatchToTopic(AmazonSNS snsClient, String topicArn) {
    try {
        // Create the batch entries to send
        List<PublishBatchRequestEntry> entries = IntStream.range(0, MAX_BATCH_SIZE)
                .mapToObj(i -> new PublishBatchRequestEntry()
                        .withId("id" + i)
                        .withMessage("message" + i))
                .collect(Collectors.toList());

        // Create the batch request
        PublishBatchRequest request = new PublishBatchRequest()
                .withTopicArn(topicArn)
                .withPublishBatchRequestEntries(entries);

        // Publish the batch request
        PublishBatchResult publishBatchResult = snsClient.publishBatch(request);

        // Handle the successfully sent messages
        publishBatchResult.getSuccessful().forEach(publishBatchResultEntry -> {
            System.out.println("Batch Id for successful message: " +
 publishBatchResultEntry.getId());
            System.out.println("Message Id for successful message: " +
 publishBatchResultEntry.getMessageId());
        });

        // Handle the failed messages
        publishBatchResult.getFailed().forEach(batchResultErrorEntry -> {
            System.out.println("Batch Id for failed message: " +
 batchResultErrorEntry.getId());
            System.out.println("Error Code for failed message: " +
 batchResultErrorEntry.getCode());
            System.out.println("Sender Fault for failed message: " +
 batchResultErrorEntry.getSenderFault());
            System.out.println("Failure Message for failed message: " +
 batchResultErrorEntry.getMessage());
        });
    } catch (AmazonSNSException e) {
        // Handle any exceptions from the request
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

Veröffentlichen eines Batches von 10 Nachrichten in einem FIFO-Thema

```
// Imports
import com.amazonaws.services.sns.AmazonSNS;
import com.amazonaws.services.sns.model.PublishBatchRequest;
import com.amazonaws.services.sns.model.PublishBatchRequestEntry;
import com.amazonaws.services.sns.model.PublishBatchResult;
import com.amazonaws.services.sns.model.AmazonSNSException;
import java.util.List;
import java.util.stream.Collectors;

// Code
private static final int MAX_BATCH_SIZE = 10;

public static void publishBatchToFifoTopic(AmazonSNS snsClient, String topicArn) {
    try {
        // Create the batch entries to send
        List<PublishBatchRequestEntry> entries = IntStream.range(0, MAX_BATCH_SIZE)
                .mapToObj(i -> new PublishBatchRequestEntry()
                        .withId("id" + i)
                        .withMessage("message" + i)
                        .withMessageGroupId("groupId")
                        .withMessageDeduplicationId("deduplicationId" + i))
                .collect(Collectors.toList());

        // Create the batch request
        PublishBatchRequest request = new PublishBatchRequest()
                .withTopicArn(topicArn)
                .withPublishBatchRequestEntries(entries);

        // Publish the batch request
        PublishBatchResult publishBatchResult = snsClient.publishBatch(request);

        // Handle the successfully sent messages
        publishBatchResult.getSuccessful().forEach(publishBatchResultEntry -> {
            System.out.println("Batch Id for successful message: " +
 publishBatchResultEntry.getId());
            System.out.println("Message Id for successful message: " +
 publishBatchResultEntry.getMessageId());
```

```
            System.out.println("SequenceNumber for successful message: " +
 publishBatchResultEntry.getSequenceNumber());
        });

        // Handle the failed messages
        publishBatchResult.getFailed().forEach(batchResultErrorEntry -> {
            System.out.println("Batch Id for failed message: " +
 batchResultErrorEntry.getId());
            System.out.println("Error Code for failed message: " +
 batchResultErrorEntry.getCode());
            System.out.println("Sender Fault for failed message: " +
 batchResultErrorEntry.getSenderFault());
            System.out.println("Failure Message for failed message: " +
 batchResultErrorEntry.getMessage());
        });

    } catch (AmazonSNSException e) {
        // Handle any exceptions from the request
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

Amazon SNS Nachrichtenfilterung

Standardmäßig erhalten Abonnenten von Amazon-SNS-Themen jede Nachricht, die zum Thema veröffentlicht wird. Um eine Teilmenge der Nachrichten zu erhalten, müssen Abonnenten dem Themen-Abonnement eine Filterrichtlinie zuweisen.

Eine Filterrichtlinie ist ein JSON-Objekt mit Eigenschaften, die definieren, welche Nachrichten der Abonnent empfängt. Amazon SNS unterstützt Richtlinien, die sich auf die Nachrichtenattribute oder den Nachrichtentext auswirken, je nach dem Geltungsbereich der Filterrichtlinie, die Sie für das Abonnement festgelegt haben. Bei den Filterrichtlinien für den Nachrichtentext wird davon ausgegangen, dass es sich bei der Nachrichtennutzlast um ein wohlgeformtes JSON-Objekt handelt.

Wenn ein Abonnement keine Filterrichtlinie besitzt, erhält der Abonnement jede Nachricht, die für sein Thema veröffentlicht wird. Wenn Sie eine Nachricht für einem Thema mit aktiver Filterrichtlinie veröffentlichen, vergleicht Amazon SNS die Nachrichtenattribute oder den Nachrichtentext mit den Eigenschaften in den Filterrichtlinien für alle Abonnements des Themas. Wenn eines der Nachrichtenattribute oder der Eigenschaften des Nachrichtentexts übereinstimmt, sendet Amazon SNS die Nachricht an den Abonnenten. Andernfalls sendet Amazon SNS die Nachricht nicht an diesen Abonnenten.

Weitere Informationen finden Sie unter Filtern von Nachrichten, die in für Themen veröffentlicht sind..

Themen

- Filterrichtlinien für Amazon-SNS-Abonnements – Geltungsbereich
- Filterrichtlinien für Amazon-SNS-Abonnements
- Anwenden von Abonnementfilterrichtlinien
- Entfernen von Abonnementfilterrichtlinien

Filterrichtlinien für Amazon-SNS-Abonnements – Geltungsbereich

Mit dem `FilterPolicyScope`-Abonnementattribut können Sie den Filterbereich auswählen, indem Sie einen der folgenden Werte festlegen:

- `MessageAttributes` – die Filterrichtlinie wird auf die Nachrichtenattribute angewendet. Dies ist die Standardeinstellung.
- `MessageBody` – die Filterrichtlinie wird auf den Nachrichtentext angewendet.

> **Note**
>
> Wenn für eine bestehende Filterrichtlinie kein Geltungsbereich für die Filterrichtlinie definiert ist, wird der Geltungsbereich standardmäßig als `MessageAttributes` definiert.

Filterrichtlinien für Amazon-SNS-Abonnements

Mit einer Abonnementfilterrichtlinie können Sie die Eigenschaftsnamen angeben und die einzelnen Eigenschaftsnamen einer Liste von Werten zuweisen. Weitere Informationen finden Sie unter Amazon SNS Nachrichtenfilterung.

Wenn Amazon-SNS-Nachrichtenattribute oder Eigenschaften des Nachrichtentexts auf der Grundlage der Abonnementfilterrichtlinie auswertet, werden die nicht in der Richtlinie angegebenen ignoriert.

> **Important**
>
> AWS Dienste wie IAM und Amazon SNS verwenden ein verteiltes Rechenmodell, das als Eventual Consistency bezeichnet wird. Hinzufügungen oder Änderungen einer Abonnementfilterrichtlinie dauern bis zu 15 Minuten, bis sie vollständig wirksam werden.

Ein Abonnement akzeptiert eine Nachricht unter den folgenden Bedingungen:

- Wenn der Geltungsbereich der Filterrichtlinie auf `MessageAttributes` festgelegt ist, stimmt nicht jeder Eigenschaftsname in der Filterrichtlinie mit dem Namen eines Nachrichtenattributs überein. Für jeden passenden Eigenschaftsnamen in der Filterrichtlinie stimmt mindestens ein Eigenschaftswert mit dem Wert des Nachrichtenattributs überein.
- Wenn der Geltungsbereich der Filterrichtlinie auf `MessageBody` festgelegt ist, stimmt jeder Eigenschaftsname in der Filterrichtlinie mit dem Namen eines Nachrichtentexts überein. Für jeden passenden Eigenschaftsnamen in der Filterrichtlinie stimmt mindestens ein Eigenschaftswert mit dem Wert des Nachrichtentexts überein.

Amazon SNS unterstützt derzeit die folgenden Werte:

- UND-Logik

- ODER-Logik
- OR-Operator
- Schlüsselabgleich
- Exakter Abgleich mit numerischen Werten
- „Alles außer“-Abgleich für numerische Werte
- Abgleich von numerischen Wertebereichen
- Exakter Abgleich mit Zeichenfolgewerten
- „Alles außer“-Abgleich von Zeichenfolgewerten
- Zeichenfolgeabgleich unter Verwendung eines Präfixes mit dem „Alles außer“-Operator
- Zeichenfolgewert gleich bei ignorierter Groß- und Kleinschreibung
- Abgleich des Zeichenfolgewerts der IP-Adresse
- Präfix-Abgleich von Zeichenfolgewerten
- Suffix-Abgleich von Zeichenfolgewerten

Beispiel für Filterrichtlinien

Das folgende Beispiel zeigt eine Nachrichtennutzlast, zugestellt von einem Amazon-SNS-Thema, das Kundentransaktionen verarbeitet.

Das erste Beispiel enthält das `MessageAttributes`-Feld mit Attributen, die die Transaktion beschreiben:

- Interessen des Kunden
- Name des Geschäfts
- Status des Ereignisses
- Kaufpreis in USD

Da diese Nachricht das `MessageAttributes`-Feld enthält, kann jedes Themenabonnement, das eine `FilterPolicy` festlegt, die Nachricht selektiv akzeptieren oder ablehnen, sofern `FilterPolicyScope` im Abonnement auf `MessageAttributes` festgelegt ist. Weitere Informationen zur Anwendung von Attributen auf Nachrichten finden Sie unter Amazon-SNS-Nachrichtenattribute.

```
{
```

```
   "Type": "Notification",
   "MessageId": "a1b2c34d-567e-8f90-g1h2-i345j67klmn8",
   "TopicArn": "arn:aws:sns:us-east-2:123456789012:MyTopic",
   "Message": "message-body-with-transaction-details",
   "Timestamp": "2019-11-03T23:28:01.631Z",
   "SignatureVersion": "4",
   "Signature": "signature",
   "UnsubscribeURL": "unsubscribe-url",
   "MessageAttributes": {
      "customer_interests": {
         "Type": "String.Array",
         "Value": "[\"soccer\", \"rugby\", \"hockey\"]"
      },
      "store": {
         "Type": "String",
         "Value":"example_corp"
      },
      "event": {
         "Type": "String",
         "Value": "order_placed"
      },
      "price_usd": {
         "Type": "Number",
         "Value": "210.75"
      }
   }
}
```

Das folgende Beispiel zeigt dieselben Attribute, die im `Message`-Feld enthalten sind, auch als Nachrichtennutzlast oder Nachrichtentext bezeichnet. Jedes Themenabonnement, das eine `FilterPolicy` enthält, kann die Nachricht selektiv akzeptieren oder ablehnen, sofern `FilterPolicyScope` im Abonnement auf `MessageBody` festgelegt ist.

```
{
"Type": "Notification",
   "MessageId": "a1b2c34d-567e-8f90-g1h2-i345j67klmn8",
   "TopicArn": "arn:aws:sns:us-east-2:123456789012:MyTopic",
   "Message": "{
      \"customer_interests\": [\"soccer\", \"rugby\", \"hockey\"],
      \"store\": \"example_corp\",
      \"event\":\"order_placed\",
      \"price_usd\":210.75
   }",
```

```
   "Timestamp": "2019-11-03T23:28:01.631Z",
   "SignatureVersion": "4",
   "Signature": "signature",
   "UnsubscribeURL": "unsubscribe-url"
}
```

Die folgenden Filterrichtlinien akzeptieren Nachrichten auf der Grundlage ihrer Eigenschaftsnamen und Werte oder weisen sie auf dieser Grundlage ab.

Eine Richtlinie, die die Beispielnachricht akzeptiert

Die Eigenschaften in der folgenden Abonnementfilterrichtlinie entsprechen den der Beispielnachricht zugewiesenen Attributen. Beachten Sie, dass dieselbe Filterrichtlinie für `FilterPolicyScope` gilt, unabhängig davon, ob sie auf `MessageAttributes` oder `MessageBody` festgelegt ist. Jeder Abonnent wählt seinen Filterbereich entsprechend der Zusammensetzung der Nachrichten aus, die er zu dem Thema erhält.

Wenn eine einzelne Eigenschaft in dieser Richtlinie nicht einem der Nachricht zugewiesenen Attribut entsprechen, weist die Richtlinie die Nachricht ab.

```
{
   "store": ["example_corp"],
   "event": [{"anything-but": "order_cancelled"}],
   "customer_interests": [
      "rugby",
      "football",
      "baseball"
   ],
   "price_usd": [{"numeric": [">=", 100]}]
}
```

Eine Richtlinie, die die Beispielnachricht abweist

In der folgenden Abonnementfilterrichtlinie gibt es mehrere Nichtübereinstimmungen zwischen ihren Eigenschaften und den der Beispielnachricht zugewiesenen Attributen. Beispiel: Da der Eigenschaftsname `encrypted` in den Nachrichtenattributen nicht vorhanden ist, veranlasst diese Richtlinieneigenschaft die Zurückweisung der Nachricht, unabhängig von dem ihm zugewiesenen Wert.

Wenn Abweichungen auftreten, weist die Richtlinie die Nachricht zurück.

```
{
   "store": ["example_corp"],
   "event": ["order_cancelled"],
   "encrypted": [false],
   "customer_interests": [
      "basketball",
      "baseball"
   ]
}
```

Einschränkungen für Filterrichtlinien

Wenn Sie eine Filterrichtlinie für ein Amazon SNS SNS-Abonnement erstellen, ist es wichtig zu verstehen, wie die Schlüssel in der Richtlinie gezählt werden. Die wichtigsten Aspekte, die Sie beachten sollten, sind:

1. Übergeordnete Schlüssel — Die übergeordneten Schlüssel sind die Schlüssel der obersten Ebene in der Filterrichtlinie. Dies sind die Schlüssel, für die Sie Werte oder Einschränkungen angeben.
2. Attributnamen — Die übergeordneten Schlüssel werden in der Filterrichtlinie als Attributnamen betrachtet. Die Werte oder Einschränkungen, die Sie für diese Schlüssel angeben, werden auf die entsprechenden Attribute in der Nachrichtennutzlast angewendet.
3. Gültige Werte — Die für die übergeordneten Schlüssel angegebenen Werte müssen entweder eine Zeichenfolge, ein Zeichenkettenarray oder eine Zahl sein. Handelt es sich bei dem Wert um ein Objekt (z. B. ein JSON-Objekt), wird er in der Filterrichtlinie nicht als gültiger Schlüssel gezählt.

Betrachten wir das folgende Beispiel für eine Filterrichtlinie:

```
{
  "state": ["SUCCESS"],
  "severity": [{ "exists": true }],
  "message": [{ "exists": true }],
  "finding": {
    "standard_control": [{ "exists": true }],
    "region": [{ "exists": true }],
    "account": [{ "exists": true }]
  }
}
```

In diesem Beispiel werden die folgenden Schlüssel als Teil der Filterrichtlinie gezählt:

- `state`
- `severity`
- `message`
- `standard_control`
- `region`
- `account`

Das Schlüsselergebnis wird nicht gezählt, da es ein JSON-Objekt als Wert enthält und nicht eine Zeichenfolge, ein Zeichenkettenarray oder eine Zahl.

Ein weiteres Beispiel:

```
{
  "key_a": {
    "key_b": {
      "key_c": {
        "key_d": ["value_one", "value_two", "value_three", "value_four"]
      }
    },
    "key_e": {
      "key_f": ["value_one", "value_two", "value_three"]
    }
  }
}
```

In diesem Fall `key_f` werden nur die Schlüssel A `key_d` und B als Teil der Filterrichtlinie gezählt, da ihnen Werte zugewiesen sind, die entweder eine Zeichenfolge oder ein Array von Zeichenfolgen sind. Die übergeordneten Schlüssel`key_a`, und`key_b`, `key_c` werden nicht gezählt, da sie verschachtelte JSON-Objekte als Werte enthalten.

Themen

- Übliche Einschränkungen für Richtlinien
- Richtlinieneinschränkungen bei der attributbasierten Filterung
- Richtlinieneinschränkungen für die nutzlastbasierte Filterung

Übliche Einschränkungen für Richtlinien

- Zeichenkettenabgleich — Beim Zeichenkettenabgleich in der Filterrichtlinie wird beim Vergleich zwischen Groß- und Kleinschreibung unterschieden.
- Numerischer Abgleich — Für den numerischen Abgleich kann der Wert zwischen -10^9 und 10^9 (-1 Milliarde bis 1 Milliarde) liegen, wobei eine Genauigkeit von fünf Dezimalstellen besteht.
- Komplexität der Filterrichtlinie — Aufgrund der Komplexität der Filterrichtlinie darf die Gesamtkombination der Werte 150 nicht überschreiten. Um die Gesamtkombination zu berechnen, multiplizieren Sie die Anzahl der Werte in jedem Array in der Filterrichtlinie.

Betrachten Sie die folgende Beispielrichtlinie:

```
{
   "key_a": ["value_one", "value_two", "value_three"],
   "key_b": ["value_one"],
   "key_c": ["value_one", "value_two"]
}
```

In dieser Richtlinie:

- Das erste Array hat 3 Werte
- Das zweite Array hat 1 Wert
- Das dritte Array hat 2 Werte

Die Gesamtkombination wird wie folgt berechnet:

- 3 x 1 x 2 = 6

Syntax der Filterrichtlinie

Der JSON-Code der Filterrichtlinie kann Folgendes enthalten:

- Zeichenfolgen, die von Anführungszeichen umschlossen werden
- Zahlen
- Die Schlüsselwörter `true`, `false` und `null`, jeweils ohne Anführungszeichen

Wenn Sie die Amazon SNS SNS-API verwenden, müssen Sie das JSON der Filterrichtlinie als gültige UTF-8-Zeichenfolge übergeben.

Grenzwerte für die Filterrichtlinie

- Die maximale Größe einer Filterrichtlinie beträgt 256 KB.
- Standardmäßig können Sie bis zu 200 Filterrichtlinien pro Thema und 10.000 Filterrichtlinien pro AWS Konto verwenden.
- Dieses Richtlinienlimit würde nicht verhindern, dass Amazon SQS SQS-Warteschlangenabonnements mit der `Subscribe` API erstellt werden. Es schlägt jedoch fehl, wenn Sie die Filterrichtlinie an den `Subscribe`-API-Aufruf (oder den `SetSubscriptionAttributes`-API-Aufruf) anhängen.
- Zur Erhöhung dieses Kontingents können Sie AWS Service Quotas verwenden.

Richtlinieneinschränkungen bei der attributbasierten Filterung

- Attributbasierte Filterung ist die Standardoption. `FilterPolicyScope` ist im Abonnement auf `MessageAttributes` festgelegt.
- Amazon SNS akzeptiert bei der attributbasierten Filterung keine Richtlinie für verschachtelte Filter.
- Amazon SNS vergleicht die Richtlinieneigenschaften nur mit Nachrichtenattributen, die die folgenden Datentypen haben:
  - `String`
  - `String.Array`

    > **Important**
    >
    > Es wird nicht empfohlen, Objekte in Arrays zu übergeben, da dies aufgrund der Verschachtelung, die von der attributbasierten Filterung nicht unterstützt wird, zu unerwarteten Ergebnissen führen kann. Nutzlast-basierte Filterung für verschachtelte Richtlinien.

  - `Number`
- Amazon SNS ignoriert Nachrichtenattribute mit dem Datentyp `Binary`.
- Eine Filterrichtlinie darf höchstens fünf Attributnamen enthalten.

Richtlinieneinschränkungen für die nutzlastbasierte Filterung

Amazon SNS akzeptiert eine geschachtelte Filterrichtlinie für nutzlastbasierte Filterung. Um die Gesamtkombination der Werte in der Filterrichtlinie zu berechnen, multiplizieren Sie die Anzahl der Werte in jedem verschachtelten Array.

Betrachten Sie die folgende Beispielrichtlinie:

```
{
"key_a": {
    "key_b": {
        "key_c": ["value_one", "value_two", "value_three", "value_four"]
        }
    },
"key_d": {
    "key_e": ["value_one", "value_two", "value_three"]
    }
}
```

In dieser Richtlinie:

- Das erste Array hat vier Werte in einem verschachtelten Schlüssel mit drei Ebenen.
- Das zweite enthält drei Werte in einem zweistufigen verschachtelten Schlüssel.

Die Gesamtkombination wird wie folgt berechnet:

- 4 x 3 x 3 x 2 = 72

Politische Grenzen

Eine Filterrichtlinie kann maximal fünf übergeordnete Schlüssel (Schlüssel der obersten Ebene) haben. Bei einer verschachtelten Richtlinie werden nur die übergeordneten Schlüssel auf die Obergrenze von fünf Schlüsseln angerechnet.

Numerischer Bereich

Für den numerischen Abgleich in der Filterrichtlinie kann der Wert zwischen -10^9 und 10^9 (-1 Milliarde bis 1 Milliarde) liegen, mit einer Genauigkeit von fünf Stellen nach dem Komma.

Umstellung auf nutzlastbasierte Filterung

Wenn Sie von der attributbasierten (Standard) zur nutzlastbasierten Filterung wechseln möchten, müssen Sie den `FilterPolicyScope` im Abonnement auf `MessageBody` festlegen.

UND/ODER-Logik

Sie können Vorgänge mit UND/ODER-Logik verwenden, um Nachrichtenattribute abzugleichen.

Themen

UND-Logik

Sie können die UND-Logik mit mehreren Attributnamen anwenden.

Betrachten Sie folgende Richtlinie:

```
{
  "customer_interests": ["rugby"],
  "price_usd": [{"numeric": [">", 100]}]
}
```

Sie gleicht alle Nachrichtenattribute oder die Eigenschaft des Nachrichtentexts ab, bei denen der Wert `customer_interests` auf `rugby` und der Wert `price_usd` auf eine Zahl größer als 100 festgelegt ist.

> **Note**
>
> Sie können keine AND-Logik auf Werte desselben Attributs anwenden.

ODER-Logik

Sie können die ODER-Logik durch die Zuweisung mehrerer Werte zu einem Attributnamen anwenden.

Betrachten Sie folgende Richtlinie:

```
{
   "customer_interests": ["rugby", "football", "baseball"]
}
```

Sie gleicht alle Nachrichtenattribute ab, deren `customer_interests`-Wert auf `rugby`, `football`, oder `baseball` festgelegt ist.

OR-Operator

Sie können den `"$or"`-Operator verwenden, um explizit eine Filterrichtlinie zu definieren, um die OR-Beziehung zwischen mehreren Attributen in der Richtlinie auszudrücken.

Amazon SNS erkennt eine `"$or"`-Beziehung nur an, wenn die Richtlinie alle der folgenden Bedingungen erfüllt hat. Wenn all diese Bedingungen nicht erfüllt sind, wird `"$or"` wie ein regulärer Attributname behandelt, genauso wie jede andere Zeichenfolge in der Richtlinie.

- Die Regel enthält beispielsweise ein `"$or"`-Feldattribut, gefolgt von einem Array, z. B. `“$or” : []`.
- Das `"$or"`-Array enthält mindestens 2 Objekte: `"$or": [{}, {}]`.
- Keines der Objekte im `"$or"`-Array hat Feldnamen, bei denen es sich um reservierte Schlüsselwörter handelt.

Andernfalls wird `"$or"` wie ein normaler Attributname behandelt, genau wie andere Zeichenfolgen in der Richtlinie.

Die folgende Richtlinie wird nicht als OR-Beziehung analysiert, da Zahl und Präfix reservierte Schlüsselwörter sind.

```
{
   "$or": [ {"numeric" : 123}, {"prefix": "abc"} ]
}
```

Beispiele für **`OR`**-Operatoren

Standard `OR`:

```
{
  "source": [ "aws.cloudwatch" ],
  "$or": [
```

```
    { "metricName": [ "CPUUtilization" ] },
    { "namespace": [ "AWS/EC2" ] }
  ]
}
```

Die Filterlogik für diese Richtlinie lautet:

```
"source" && ("metricName" || "namespace")
```

Sie stimmt mit einem der beiden folgenden Nachrichtenattributsätze überein:

```
"source": {"Type": "String", "Value": "aws.cloudwatch"},
"metricName": {"Type": "String", "Value": "CPUUtilization"}
```

or

```
"source": {"Type": "String", "Value": "aws.cloudwatch"},
"namespace": {"Type": "String", "Value": "AWS/EC2"}
```

Sie stimmt auch mit einem der beiden folgenden Nachrichtentexte überein:

```
{
    "source": "aws.cloudwatch",
    "metricName": "CPUUtilization"
}
```

or

```
{
    "source": "aws.cloudwatch",
    "namespace": "AWS/EC2"
}
```

Richtlinieneinschränkungen, zu denen auch **OR**-Beziehungen gehören

Betrachten Sie folgende Richtlinie:

```
{
    "source": [ "aws.cloudwatch" ],
    "$or": [
```

```
      { "metricName": [ "CPUUtilization", "ReadLatency" ] },
      {
        "metricType": [ "MetricType" ] ,
        "$or" : [
          { "metricId": [ 1234, 4321 ] },
          { "spaceId": [ 1000, 2000, 3000 ] }
        ]
      }
    ]
  }
```

Die Logik dieser Richtlinie kann auch wie folgt vereinfacht werden:

```
("source" AND "metricName")
OR
("source" AND "metricType" AND "metricId")
OR
("source" AND "metricType" AND "spaceId")
```

Die Komplexitätsberechnung für Richtlinien mit OR-Beziehungen kann vereinfacht werden, indem die Summe der Kombinationskomplexitäten für jede OR-Anweisung berechnet wird.

Die Gesamtkombination wird wie folgt berechnet:

```
(source * metricName) + (source * metricType * metricId) + (source * metricType *
 spaceId)
= (1 * 2) + (1 * 1 * 2) + (1 * 1 * 3)
= 7
```

`source` hat einen Wert, `metricName` hat zwei Werte, `metricType` hat einen Wert, `metricId` hat zwei Werte und `spaceId` hat drei Werte.

Beachten Sie die folgende Richtlinie für verschachtelte Filter:

```
{
    "$or": [
      { "metricName": [ "CPUUtilization", "ReadLatency" ] },
      { "namespace": [ "AWS/EC2", "AWS/ES" ] }
    ],
    "detail" : {
      "scope" : [ "Service" ],
      "$or": [
```

```
        { "source": [ "aws.cloudwatch" ] },
        { "type": [ "CloudWatch Alarm State Change"] }
      ]
    }
  }
```

Die Logik dieser Richtlinie kann wie folgt vereinfacht werden:

```
("metricName" AND ("detail"."scope" AND "detail"."source")
OR
("metricName" AND ("detail"."scope" AND "detail"."type")
OR
("namespace" AND ("detail"."scope" AND "detail"."source")
OR
("namespace" AND ("detail"."scope" AND "detail"."type")
```

Die Berechnung der Gesamtkombinationen ist bei nicht verschachtelten Richtlinien gleich, außer dass wir die Verschachtelungsebene des Schlüssels berücksichtigen müssen.

Die Gesamtkombination wird wie folgt berechnet:

```
(2 * 2 * 2) + (2 * 2 * 2) + (2 * 2 * 2) + (2 * 2 * 2) = 32
```

`metricName` hat zwei Werte, `namespace` hat zwei Werte, `scope` ist ein zweistufiger verschachtelter Schlüssel mit einem Wert, `source` ist ein zweistufiger verschachtelter Schlüssel mit einem Wert und `type` ist ein zweistufiger verschachtelter Schlüssel mit einem Wert.

Schlüsselabgleich

Sie können den `exists`-Operator verwenden, um eingehende Nachrichten mit oder ohne angegebenen Eigenschaften in der Filterrichtlinie abzugleichen. Der `exists`-Abgleich funktioniert nur in Blattknoten. Auf Zwischenknoten funktioniert sie nicht.

- Verwenden Sie `"exists": true`, um eingehende Nachrichten abzugleichen, die die angegebene Eigenschaft enthalten. Der Schlüssel muss einen Wert haben, der nicht null ist und einen der nicht leer ist.

  Die folgende Richtlinie verwendet beispielsweise den `exists`-Operator mit dem Wert `true`:

  ```
  "store": [{"exists": true}]
  ```

Sie gleicht alle Listen mit Nachrichtenattributen ab, die den `store`-Attributschlüssel besitzen, z. B. die folgenden:

```
"store": {"Type": "String", "Value": "fans"}
"customer_interests": {"Type": "String.Array", "Value": "[\"baseball\", \"basketball
\"]"}
```

Sie stimmt auch mit einem der beiden folgenden Nachrichtentexte überein:

```
{
    "store": "fans"
    "customer_interests": ["baseball", "basketball"]
}
```

Sie stimmt jedoch nicht mit Nachrichten überein, die nicht den Attributschlüssel `store` besitzen, z. B. die folgenden:

```
"customer_interests": {"Type": "String.Array", "Value": "[\"baseball\", \"basketball
\"]"}
```

Sie stimmt auch nicht mit dem folgenden Nachrichtentext überein:

```
{
    "customer_interests": ["baseball", "basketball"]
}
```

- Verwenden Sie `"exists": false`, um eingehende Nachrichten abzugleichen, die die angegebene Eigenschaft nicht enthalten.

> **Note**
>
> `"exists": false` stimmt nur überein, wenn mindestens ein Attribut vorhanden ist. Ein leerer Satz von Attributen führt dazu, dass der Filter nicht übereinstimmt.

Die folgende Richtlinie verwendet beispielsweise den `exists`-Operator mit dem Wert `false`:

```
"store": [{"exists": false}]
```

Sie stimmt nicht mit Listen mit Nachrichtenattributen überein, die den Attributschlüssel `store` besitzen, z. B. die folgenden:

```
"store": {"Type": "String", "Value": "fans"}
"customer_interests": {"Type": "String.Array", "Value": "[\"baseball\", \"basketball
\"]"}
```

Sie stimmt auch nicht mit dem folgenden Nachrichtentext überein:

```
{
    "store": "fans"
    "customer_interests": ["baseball", "basketball"]
}
```

Sie stimmt jedoch mit allen Listen mit Nachrichtenattributen ohne den Attributschlüssel `store` überein, z. B. die folgenden:

```
"customer_interests": {"Type": "String.Array", "Value": "[\"baseball\", \"basketball
\"]"}
```

Sie stimmt auch mit dem folgenden Nachrichtentext überein:

```
{
    "customer_interests": ["baseball", "basketball"]
}
```

Abgleichen numerischer Werten

Sie können Nachrichten filtern, indem Sie numerische Werte mit Nachrichtenattributwerten oder Eigenschaftswerten für den Nachrichtentext abgleichen. Numerische Werte werden in der JSON-Richtlinie nicht von doppelten Anführungszeichen eingeschlossen. Sie können zum Filtern die folgenden numerischen Vorgänge verwenden.

> **Note**
>
> Präfixe werden nur für den Zeichenfolgeabgleich unterstützt.

Themen

Genaue Übereinstimmung

Wenn eine Richtlinieneigenschaft das Schlüsselwort `numeric` und den =-Operator enthält, stimmt sie mit allen Werten des Nachrichtenattributs und des Nachrichtentexts überein, die den gleichen Namen und den gleichen numerischen Wert besitzen.

Betrachten Sie folgende Richtlinieneigenschaft:

```
"price_usd": [{"numeric": ["=",301.5]}]
```

Sie stimmt mit einem der beiden folgenden Nachrichtenattribute überein:

```
"price_usd": {"Type": "Number", "Value": 301.5}
```

```
"price_usd": {"Type": "Number", "Value": 3.015e2}
```

Sie stimmt auch mit einem der beiden folgenden Nachrichtentexte überein:

```
{
   "price_usd": 301.5
}
```

```
{
   "price_usd": 3.015e2
}
```

„Alles außer“-Abgleich

Wenn ein Richtlinieneigenschaftswert das Schlüsselwort `anything-but` enthält, stimmt er mit allen Eigenschaftswerten für Nachrichtenattribute oder Nachrichtentexte überein, die keinen der Richtlinieneigenschaftswerte enthalten.

Betrachten Sie folgende Richtlinieneigenschaft:

```
"price": [{"anything-but": [100, 500]}]
```

Sie stimmt mit einem der beiden folgenden Nachrichtenattribute überein:

```
"price": {"Type": "Number", "Value": 101}
```

```
"price": {"Type": "Number", "Value": 100.1}
```

Sie stimmt auch mit einem der beiden folgenden Nachrichtentexte überein:

```
{
   "price": 101
}
```

```
{
   "price": 100.1
}
```

Sie stimmt zudem mit dem folgenden Nachrichtenattribut überein (da es einen Wert enthält, der nicht `100` oder `500` ist):

```
"price": {"Type": "Number.Array", "Value": "[100, 50]"}
```

Außerdem stimmt sie auch mit dem folgenden Nachrichtentext überein (da er einen Wert enthält, der weder `100` noch `500` ist):

```
{
   "price": [100, 50]
}
```

Sie stimmt jedoch nicht mit dem folgenden Nachrichtenattribut überein:

```
"price": {"Type": "Number", "Value": 100}
```

Sie stimmt auch nicht mit dem folgenden Nachrichtentext überein:

```
{
   "price": 100
}
```

Wertbereichsübereinstimmung

Zusätzlich zum =-Operator kann eine numerische Richtlinieneigenschaft die folgenden Operatoren enthalten: <, <=, > und >=.

Betrachten Sie folgende Richtlinieneigenschaft:

```
"price_usd": [{"numeric": ["<", 0]}]
```

Sie stimmt mit allen Nachrichtenattributen mit negativen numerischen Werten überein.

Betrachten Sie ein weiteres Nachrichtenattribut:

```
"price_usd": [{"numeric": [">", 0, "<=", 150]}]
```

Sie stimmt mit allen Nachrichtenattributen mit positiven Zahlen bis einschließlich 150 überein.

Abgleich von Zeichenfolgewerten

Sie können Nachrichten filtern, indem Sie Zeichenfolgewerte mit Nachrichtenattributwerten oder mit Eigenschaftswerte von Nachrichtentext abgleichen. Zeichenfolgenwerte werden in der JSON-Richtlinie von doppelten Anführungszeichen eingeschlossen. Sie können die folgenden Zeichenfolgeoperationen verwenden, um Nachrichtenattribute oder den Nachrichtentext abzugleichen.

Themen

Genaue Übereinstimmung

Eine genaue Übereinstimmung liegt vor, wenn eine Richtlinieneigenschaft einem oder mehreren Nachrichtenattributwerten entspricht.

Betrachten Sie folgende Richtlinieneigenschaft:

```
"customer_interests": ["rugby", "tennis"]
```

Sie stimmt mit den folgenden Nachrichtenattributen überein:

```
"customer_interests": {"Type": "String", "Value": "rugby"}
```

```
"customer_interests": {"Type": "String", "Value": "tennis"}
```

Sie stimmt auch mit den folgenden Nachrichtentexten überein:

```
{
   "customer_interests": "rugby"
}
```

```
{
   "customer_interests": "tennis"
}
```

Sie stimmt jedoch nicht mit dem folgenden Nachrichtenattribut überein:

```
"customer_interests": {"Type": "String", "Value": "baseball"}
```

Sie stimmt auch nicht mit dem folgenden Nachrichtentext überein:

```
{
   "customer_interests": "baseball"
}
```

„Alles außer“-Abgleich

Wenn ein Richtlinieneigenschaftswert das Schlüsselwort `anything-but` enthält, stimmt er mit allen Nachrichtenattributen oder Nachrichtentextwerten überein, die keinen der

Richtlinieneigenschaftswerte enthalten. `anything-but` kann mit `"exists": false` kombiniert werden.

Betrachten Sie folgende Richtlinieneigenschaft:

```
"customer_interests": [{"anything-but": ["rugby", "tennis"]}]
```

Sie stimmt mit einem der beiden folgenden Nachrichtenattribute überein:

```
"customer_interests": {"Type": "String", "Value": "baseball"}
```

```
"customer_interests": {"Type": "String", "Value": "football"}
```

Sie stimmt auch mit einem der beiden folgenden Nachrichtentexte überein:

```
{
   "customer_interests": "baseball"
}
```

```
{
   "customer_interests": "football"
}
```

Sie stimmt zudem mit dem folgenden Nachrichtenattribut überein (da es einen Wert enthält, der nicht `rugby` oder `tennis` ist):

```
"customer_interests": {"Type": "String.Array", "Value": "[\"rugby\", \"baseball\"]"}
```

Außerdem stimmt sie auch mit dem folgenden Nachrichtentext überein (da sie einen Wert enthält, der weder `rugby` noch `tennis` ist):

```
{
   "customer_interests": ["rugby", "baseball"]
}
```

Sie stimmt jedoch nicht mit dem folgenden Nachrichtenattribut überein:

```
"customer_interests": {"Type": "String", "Value": "rugby"}
```

Sie stimmt auch nicht mit dem folgenden Nachrichtentext überein:

```
{
   "customer_interests": ["rugby"]
}
```

Verwenden eines Präfix mit dem **`anything-but`**-Operator

Für den Zeichenfolgeabgleich können Sie auch ein Präfix mit dem `anything-but`-Operator verwenden. Die folgende Richtlinieneigenschaft beispielsweise verweigert das `order-`-Präfix:

```
"event":[{"anything-but": {"prefix": "order-"}}]
```

Es entspricht einem der beiden folgenden Attribute:

```
"event": {"Type": "String", "Value": "data-entry"}
```

```
"event": {"Type": "String", "Value": "order_number"}
```

Sie stimmt auch mit einem der beiden folgenden Nachrichtentexte überein:

```
{
   "event": "data-entry"
}
```

```
{
   "event": "order_number"
}
```

Sie stimmt jedoch nicht mit dem folgenden Nachrichtenattribut überein:

```
"event": {"Type": "String", "Value": "order-cancelled"}
```

Sie stimmt auch nicht mit dem folgenden Nachrichtentext überein:

```
{
   "event": "order-cancelled"
}
```

E-Abgleich quals-ignore-case

Wenn eine Richtlinieneigenschaft das Schlüsselwort `equals-ignore-case` enthält, stimmt sie mit allen Werten des Nachrichtenattributs oder der Texteigenschaft überein, die mit den angegebenen Zeichen beginnen.

Betrachten Sie folgende Richtlinieneigenschaft:

```
"customer_interests": [{"equals-ignore-case": "tennis"}]
```

Sie stimmt mit einem der beiden folgenden Nachrichtenattribute überein:

```
"customer_interests": {"Type": "String", "Value": "TENNIS"}
```

```
"customer_interests": {"Type": "String", "Value": "Tennis"}
```

Sie stimmt auch mit einem der beiden folgenden Nachrichtentexte überein:

```
{
    "customer_interests": "TENNIS"
}
```

```
{
    "customer_interests": "teNnis"
{
```

Abgleich von IP-Adressen

Sie können das `cidr`-Operator, um zu überprüfen, ob eine eingehende Nachricht von einer bestimmten IP-Adresse oder einem bestimmten Subnetz stammt.

Betrachten Sie folgende Richtlinieneigenschaft:

```
"source_ip":[{"cidr": "10.0.0.0/24"}]
```

Sie stimmt mit einem der beiden folgenden Nachrichtenattribute überein:

```
"source_ip": {"Type": "String", "Value": "10.0.0.0"}
```

```
"source_ip": {"Type": "String", "Value": "10.0.0.255"}
```

Sie stimmt auch mit einem der beiden folgenden Nachrichtentexte überein:

```
{
   "source_ip": "10.0.0.0"
}
```

```
{
   "source_ip": "10.0.0.255"
}
```

Sie stimmt jedoch nicht mit dem folgenden Nachrichtenattribut überein:

```
"source_ip": {"Type": "String", "Value": "10.1.1.0"}
```

Sie stimmt auch nicht mit dem folgenden Nachrichtentext überein:

```
{
   "source_ip": "10.1.1.0"
}
```

Übereinstimmung mit einem Präfix

Wenn eine Richtlinieneigenschaft das Schlüsselwort `prefix` enthält, stimmt sie mit allen Werten des Nachrichtenattributs oder der Texteigenschaft überein, die mit den angegebenen Zeichen beginnen.

Betrachten Sie folgende Richtlinieneigenschaft:

```
"customer_interests": [{"prefix": "bas"}]
```

Sie stimmt mit einem der beiden folgenden Nachrichtenattribute überein:

```
"customer_interests": {"Type": "String", "Value": "baseball"}
```

```
"customer_interests": {"Type": "String", "Value": "basketball"}
```

Sie stimmt auch mit einem der beiden folgenden Nachrichtentexte überein:

```
{
   "customer_interests": "baseball"
}
```

```
{
   "customer_interests": "basketball"
}
```

Sie stimmt jedoch nicht mit dem folgenden Nachrichtenattribut überein:

```
"customer_interests": {"Type": "String", "Value": "rugby"}
```

Sie stimmt auch nicht mit dem folgenden Nachrichtentext überein:

```
{
   "customer_interests": "rugby"
}
```

Suffixabgleich

Wenn eine Richtlinieneigenschaft das Schlüsselwort `suffix` enthält, stimmt sie mit allen Werten des Nachrichtenattributs oder der Texteigenschaft überein, die auf das angegebene Zeichen enden.

Betrachten Sie folgende Richtlinieneigenschaft:

```
"customer_interests": [{"suffix": "ball"}]
```

Sie stimmt mit einem der beiden folgenden Nachrichtenattribute überein:

```
"customer_interests": {"Type": "String", "Value": "baseball"}
```

```
"customer_interests": {"Type": "String", "Value": "basketball"}
```

Sie stimmt auch mit einem der beiden folgenden Nachrichtentexte überein:

```
{
    "customer_interests": "baseball"
}
```

```
{
    "customer_interests": "basketball"
}
```

Sie stimmt jedoch nicht mit dem folgenden Nachrichtenattribut überein:

```
"customer_interests": {"Type": "String", "Value": "rugby"}
```

Sie stimmt auch nicht mit dem folgenden Nachrichtentext überein:

```
{
    "customer_interests": "rugby"
}
```

Anwenden von Abonnementfilterrichtlinien

Sie können eine Filterrichtlinie unter Verwendung der Amazon SNS-Konsole auf ein Amazon SNS-Abonnement anwenden. Um Richtlinien programmgesteuert anzuwenden, können Sie auch die Amazon SNS-API, das AWS Command Line Interface (AWS CLI) oder ein beliebiges AWS SDK verwenden, das Amazon SNS unterstützt. Sie können auch verwenden. AWS CloudFormation

> **Important**
>
> AWS Dienste wie IAM und Amazon SNS verwenden ein verteiltes Rechenmodell, das als Eventual Consistency bezeichnet wird. Hinzufügungen oder Änderungen einer Abonnementfilterrichtlinie dauern bis zu 15 Minuten, bis sie vollständig wirksam werden.

AWS Management Console

1. Melden Sie sich bei der Amazon SNS-Konsole an.
2. Wählen Sie im Navigationsbereich Subscriptions (Abonnements) aus.
3. Wählen Sie ein Abonnement und dann Edit (Bearbeiten) aus.
4. Erweitern Sie auf der Seite Edit (Bearbeiten) den Abschnitt Subscription filter policy (Abonnementfilterrichtlinie).
5. Wählen Sie zwischen attributbasierter Filterung oder nutzlastbasierter Filterung aus.
6. Geben Sie im Feld JSON editor (JSON-Editor) den JSON-Text Ihrer Filterrichtlinie an.

7. Wählen Sie Änderungen speichern aus.

   Amazon SNS wendet Ihre Filterrichtlinie auf das Abonnement an.

AWS CLI

Um eine Filterrichtlinie mit dem AWS Command Line Interface (AWS CLI) anzuwenden, verwenden Sie den `set-subscription-attributes`Befehl, wie im folgenden Beispiel gezeigt. Wählen Sie für die Option `--attribute-name` den Wert `FilterPolicy`. Geben Sie für `--attribute-value` Ihre JSON-Richtlinie an.

```
$ aws sns set-subscription-attributes --subscription-arn arn:aws:sns: ... --
attribute-name FilterPolicy --attribute-value '{"store":["example_corp"],"event":
["order_placed"]}'
```

Um gültiges JSON für Ihre Richtlinie anzugeben, schließen Sie die Attributnamen und Werte in doppelte Anführungszeichen ein. Sie müssen auch das gesamte Richtlinienargument in Anführungszeichen einschließen. Um das Maskieren von Anführungszeichen zu vermeiden, können Sie die Richtlinie in einfache Anführungszeichen und die JSON-Namen und -Werte in doppelte Anführungszeichen setzen, wie im folgenden Beispiel gezeigt.

Wenn Sie von der attributbasierten (Standard) zur nutzlastbasierten Nachrichtenfilterung wechseln möchten, können Sie den Befehl ebenfalls verwenden. set-subscription-attributes Wählen Sie für die Option `--attribute-name` den Wert `FilterPolicyScope`. Legen Sie für `--attribute-value` die Option `MessageBody` fest.

```
$ aws sns set-subscription-attributes --subscription-arn arn:aws:sns: ... --attribute-
name FilterPolicyScope --attribute-value MessageBody
```

Um sicherzustellen, dass Ihre Filterrichtlinie angewendet wurde, verwenden Sie den Befehl `get-subscription-attributes`. Die Attribute in der Terminal-Ausgabe sollten Ihre Filterrichtlinie für den `FilterPolicy`-Schlüssel zeigen, wie im folgenden Beispiel:

```
$ aws sns get-subscription-attributes --subscription-arn arn:aws:sns: ...
{
    "Attributes": {
        "Endpoint": "endpoint . . .",
        "Protocol": "https",
        "RawMessageDelivery": "false",
```

```
        "EffectiveDeliveryPolicy": "delivery policy . . .",
        "ConfirmationWasAuthenticated": "true",
        "FilterPolicy": "{\"store\": [\"example_corp\"], \"event\": [\"order_placed
\"]}",
        "FilterPolicyScope": "MessageAttributes",
        "Owner": "111122223333",
        "SubscriptionArn": "arn:aws:sns: . . .",
        "TopicArn": "arn:aws:sns: . . ."
    }
}
```

AWS SDKs

Die folgenden Codebeispiele zeigen die Verwendung`SetSubscriptionAttributes`.

> **Important**
>
> Wenn Sie das Beispiel SDK for Java 2.x verwenden, ist die Klasse `SNSMessageFilterPolicy` nicht sofort verfügbar. Anweisungen zur Installation dieser Klasse finden Sie im Beispiel GitHub auf der Website.

CLI

AWS CLI

So legen Sie Abonnementattribute fest

Im folgenden `set-subscription-attributes`-Beispiel wird das `RawMessageDelivery`-Attribut auf ein SQS-Abonnement festgelegt.

```
aws sns set-subscription-attributes \
    --subscription-arn arn:aws:sns:us-
east-1:123456789012:mytopic:f248de18-2cf6-578c-8592-b6f1eaa877dc \
    --attribute-name RawMessageDelivery \
    --attribute-value true
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Im folgenden `set-subscription-attributes`-Beispiel wird ein `FilterPolicy`-Attribut auf ein SQS-Abonnement festgelegt.

```
aws sns set-subscription-attributes \
    --subscription-arn arn:aws:sns:us-
east-1:123456789012:mytopic:f248de18-2cf6-578c-8592-b6f1eaa877dc \
    --attribute-name FilterPolicy \
    --attribute-value "{ \"anyMandatoryKey\": [\"any\", \"of\", \"these\"] }"
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Im folgenden `set-subscription-attributes`-Beispiel wird das `FilterPolicy`-Attribut von einem SQS-Abonnement entfernt.

```
aws sns set-subscription-attributes \
    --subscription-arn arn:aws:sns:us-
east-1:123456789012:mytopic:f248de18-2cf6-578c-8592-b6f1eaa877dc \
    --attribute-name FilterPolicy \
    --attribute-value "{}"
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

- Einzelheiten zur API finden Sie SetSubscriptionAttributesin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SnsException;
import java.util.ArrayList;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
```

```
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class UseMessageFilterPolicy {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <subscriptionArn>

                Where:
                   subscriptionArn - The ARN of a subscription.

                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String subscriptionArn = args[0];
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        usePolicy(snsClient, subscriptionArn);
        snsClient.close();
    }

    public static void usePolicy(SnsClient snsClient, String subscriptionArn) {
        try {
            SNSMessageFilterPolicy fp = new SNSMessageFilterPolicy();
            // Add a filter policy attribute with a single value
            fp.addAttribute("store", "example_corp");
            fp.addAttribute("event", "order_placed");

            // Add a prefix attribute
            fp.addAttributePrefix("customer_interests", "bas");

            // Add an anything-but attribute
            fp.addAttributeAnythingBut("customer_interests", "baseball");

            // Add a filter policy attribute with a list of values
            ArrayList<String> attributeValues = new ArrayList<>();
```

```
            attributeValues.add("rugby");
            attributeValues.add("soccer");
            attributeValues.add("hockey");
            fp.addAttribute("customer_interests", attributeValues);

            // Add a numeric attribute
            fp.addAttribute("price_usd", "=", 0);

            // Add a numeric attribute with a range
            fp.addAttributeRange("price_usd", ">", 0, "<=", 100);

            // Apply the filter policy attributes to an Amazon SNS subscription
            fp.apply(snsClient, subscriptionArn);

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Einzelheiten zur API finden Sie SetSubscriptionAttributesin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

```
class SnsWrapper:
    """Encapsulates Amazon SNS topic and subscription functions."""

    def __init__(self, sns_resource):
        """
        :param sns_resource: A Boto3 Amazon SNS resource.
```

```
        """
        self.sns_resource = sns_resource


    @staticmethod
    def add_subscription_filter(subscription, attributes):
        """
        Adds a filter policy to a subscription. A filter policy is a key and a
        list of values that are allowed. When a message is published, it must
have an
        attribute that passes the filter or it will not be sent to the
subscription.

        :param subscription: The subscription the filter policy is attached to.
        :param attributes: A dictionary of key-value pairs that define the
filter.
        """
        try:
            att_policy = {key: [value] for key, value in attributes.items()}
            subscription.set_attributes(
                AttributeName="FilterPolicy",
AttributeValue=json.dumps(att_policy)
            )
            logger.info("Added filter to subscription %s.", subscription.arn)
        except ClientError:
            logger.exception(
                "Couldn't add filter to subscription %s.", subscription.arn
            )
            raise
```

- Einzelheiten zur API finden Sie SetSubscriptionAttributesin AWS SDK for Python (Boto3) API Reference.

Amazon SNS-API

Um eine Filterrichtlinie mit der Amazon SNS API anzuwenden, stellen Sie eine Anfrage an die `SetSubscriptionAttributes`-Aktion. Setzen Sie den Parameter `AttributeName` auf `FilterPolicy`, und den Parameter `AttributeValue` auf das JSON Ihrer Filterrichtlinie.

Wenn Sie von der attributbasierten (Standard) zur nutzlastbasierten Nachrichtenfilterung wechseln möchten, können Sie auch die Aktion `SetSubscriptionAttributes` verwenden. Legen Sie den `AttributeName`-Parameter auf `FilterPolicyScope` und den `AttributeValue`-Parameter auf `MessageBody` fest.

AWS CloudFormation

Um eine Filterrichtlinie anzuwenden AWS CloudFormation, verwenden Sie eine JSON- oder YAML-Vorlage, um einen Stack zu erstellen. AWS CloudFormation Weitere Informationen finden Sie unter der `FilterPolicy`Eigenschaft der `AWS::SNS::Subscription` Ressource im AWS CloudFormation Benutzerhandbuch und in der AWS CloudFormation Beispielvorlage.

1. Melden Sie sich an der AWS CloudFormation -Konsole an.
2. Wählen Sie Stapel erstellen aus.
3. Wählen Sie auf der Seite Select Template (Vorlage auswählen) die Option Upload a template to Amazon S3 (Eine Vorlage zu Amazon S3 hochladen) aus. Wählen Sie dann Ihre Datei und Next (Weiter) aus.
4. Führen Sie auf der Seite Specify DB Details (Festlegen von DB-Detail) die folgenden Schritte aus:

   a. Geben Sie für Stack name (Stack-Name) `MyFilterPolicyStack` ein.

   b. Geben Sie zum myHttpEndpointBeispiel den HTTP-Endpunkt ein, um Ihr Thema zu abonnieren.

   > **Tip**
   >
   > Wenn Sie nicht über einen HTTP-Endpunkt verfügen, erstellen Sie einen.

5. Wählen Sie auf der Seite Optionen Weiter aus.
6. Klicken Sie auf der Seite Review auf Create.

Entfernen von Abonnementfilterrichtlinien

Um das Filtern der Nachrichten, die an ein Abonnement gesendet werden, zu beenden, entfernen Sie die Filterrichtlinie für das Abonnement, indem Sie sie mit einer leeren JSON-Nachricht überschreiben. Nachdem Sie die Richtlinie entfernt haben, akzeptiert das Abonnement jede Nachricht, die ihm gesendet wird.

AWS Management Console

1. Melden Sie sich bei der Amazon SNS-Konsole an.
2. Wählen Sie im Navigationsbereich Subscriptions (Abonnements) aus.
3. Wählen Sie ein Abonnement und dann Edit (Bearbeiten) aus.
4. Auf der Seite Bearbeiten ***Beispiel1-23bc-4567-d890-ef12g3hij456*** erweitern Sie die Abonnement-Filterrichtlinie.
5. Geben Sie im Feld JSON editor (JSON-Editor) einen leeren JSON-Code für Ihre Filterrichtlinie an: `{}`
6. Wählen Sie Save Changes.

   Amazon SNS wendet Ihre Filterrichtlinie auf das Abonnement an.

AWS CLI

Um eine Filterrichtlinie mithilfe der AWS CLI zu entfernen, verwenden Sie den Befehl `set-subscription-attributes` und geben Sie für das Argument `--attribute-value` leeren JSON-Text an:

```
$ aws sns set-subscription-attributes --subscription-arn arn:aws:sns: ... --attribute-name FilterPolicy --attribute-value "{}"
```

Amazon SNS-API

Um eine Filterrichtlinie mit dem Amazon SNS API zu entfernen, stellen Sie eine Anfrage an die `SetSubscriptionAttributes`-Aktion. Setzen Sie den Parameter `AttributeName` auf `FilterPolicy` und geben Sie einen leeren JSON-Rumpf für den Parameter `AttributeValue` an.

Nachrichtendatenschutz

Themen

Was ist Nachrichtendatenschutz?

Der Nachrichtendatenschutz schützt die Daten, die in Ihren Amazon-SNS-Themen veröffentlicht werden, indem Datenschutzrichtlinien verwendet werden, um sensible Informationen zu überprüfen, zu maskieren, zu redigieren oder zu blockieren, die zwischen Anwendungen oder AWS-Services übertragen werden.

Im Rahmen des Nachrichtendatenschutzes werden Daten mithilfe von Datenkennungen in Übertragung auf persönlich identifizierbare Informationen (PII) und auf geschützte Gesundheitsdaten (PHI) gescannt. Sie können wählen, ob Sie vordefinierte (oder von Amazon SNS verwaltete) Datenkennungen (z. B. Namen, Adressen, Kreditkartennummern und Codes für verschreibungspflichtige Medikamente) verwenden möchten, oder Sie können Ihre eigenen benutzerdefinierten Datenkennungen erstellen, die auf Ihren geschäftlichen Anwendungsfall zugeschnitten sind. Mithilfe der gescannten Informationen stellt der Nachrichtendatenschutz detaillierte Überwachungsprotokolle bereit und ermöglicht es Ihnen, Maßnahmen zum Schutz dieser Daten zu ergreifen.

Der Nachrichtendatenschutz unterstützt die folgenden Aktionen, um sensible Kundeninformationen zu schützen:

- Audit – Überprüft bis zu 99 % der Daten, die in einem Amazon-SNS-Thema veröffentlicht werden. Anschließend können Sie die Ergebnisse an Amazon CloudWatch, Amazon S3 oder Amazon Data Firehose senden.
- Anonymisieren – Maskieren oder Redigieren sensibler Daten, ohne die Nachrichtenveröffentlichung zu unterbrechen.
- Deny – Blockiert die Übertragung von Daten zwischen Anwendungen und AWS-Ressourcen, wenn sensible Daten innerhalb der Nutzlast vorhanden sind.

> **Note**
>
> Amazon SNS unterstützt den Nachrichtendatenschutz nur für Standardthemen von Amazon SNS.

Weshalb sollte ich den Nachrichtendatenschutz verwenden?

Durch die Einführung des Nachrichtendatenschutzes in Ihre Governance-, Risikomanagement- und Compliance-Programme können Sie Datenschutzrichtlinien implementieren, die Ihnen helfen, Datenlecks zu erkennen und zu verhindern. Dadurch stehen Ihren Teams Tools zur Verfügung, mit denen Sie finanzielle, rechtliche und regulatorische Risiken reduzieren können, indem sie Datenschutzbestimmungen wie HIPAA, DSGVO, PCI und FedRAMP einhalten. Außerdem entlastet es Ihre Entwickler vom betrieblichen Aufwand, der mit der Entwicklung und Verwaltung Ihrer eigenen Tools zum Schutz sensibler Daten verbunden ist.

Beispielsweise können Sie mit dem Nachrichtendatenschutz eine Prüfungsrichtlinie erstellen, um festzustellen, ob eines Ihrer Systeme versehentlich sensible Daten sendet oder empfängt. Wenn Ihre Prüfungsergebnisse zeigen, dass Systeme Kreditkarteninformationen an Systeme senden, die diese nicht benötigen, können Sie eine Blockierungsrichtlinie erstellen, um die Zustellung der Daten zu verhindern.

> **Note**
>
> Amazon SNS unterstützt den Nachrichtendatenschutz nur für Standardthemen von Amazon SNS.

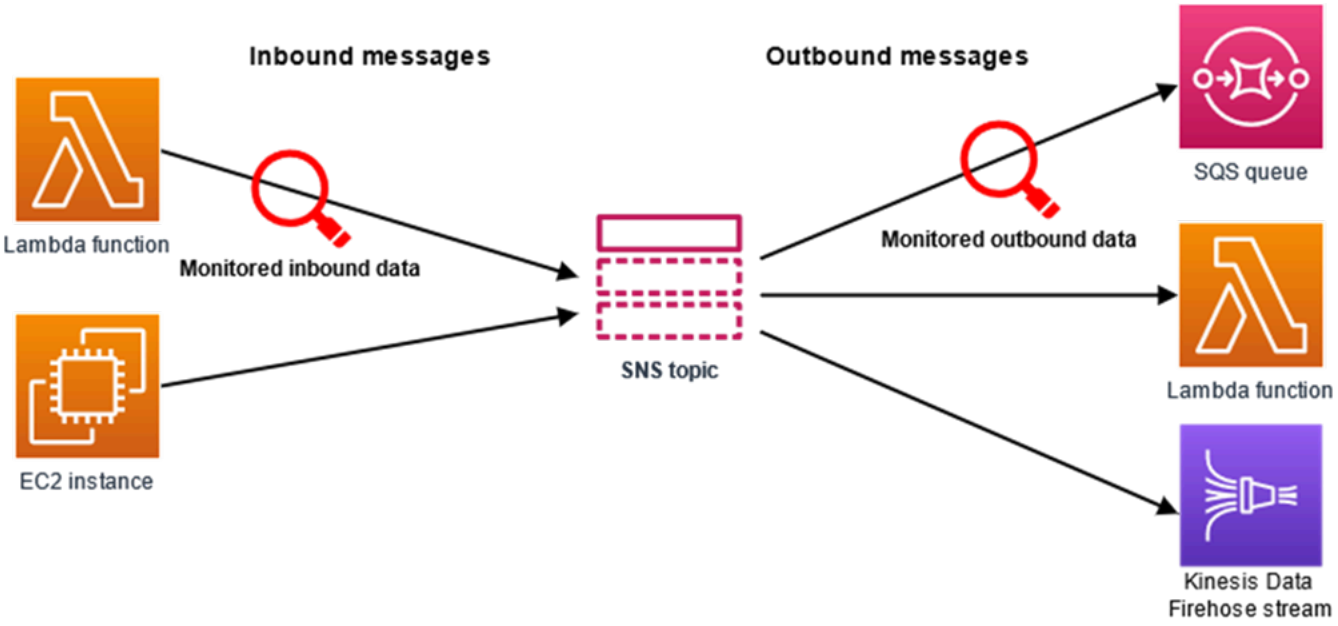
Informationen über Datenschutzrichtlinien

Themen

- Erstellen von Datenschutzrichtlinien
- Löschen von Datenschutzrichtlinien in Amazon SNS

Was sind Datenschutzrichtlinien?

Amazon SNS verwendet Datenschutzrichtlinien, um die sensiblen Daten, nach denen Sie suchen möchten, und die Maßnahmen auszuwählen, die Sie ergreifen sollten, um zu verhindern, dass diese Daten von Ihren Amazon-SNS-Themen ausgetauscht werden. Sie verwenden Datenkennungen, um die entsprechenden sensiblen Daten auszuwählen. Der Amazon-SNS-Nachrichtendatenschutz erkennt dann die sensiblen Daten mithilfe von Machine Learning und Musterabgleich. Um auf gefundene Datenkennungen zu reagieren, können Sie einen Vorgang zum Prüfen Anonymisieren oder Verweigern definieren. Mit diesen Vorgängen können Sie die gefundenen (oder nicht gefundenen) sensiblen Daten protokollieren, maskieren oder redigieren oder die Nachrichtenzustellung verweigern.

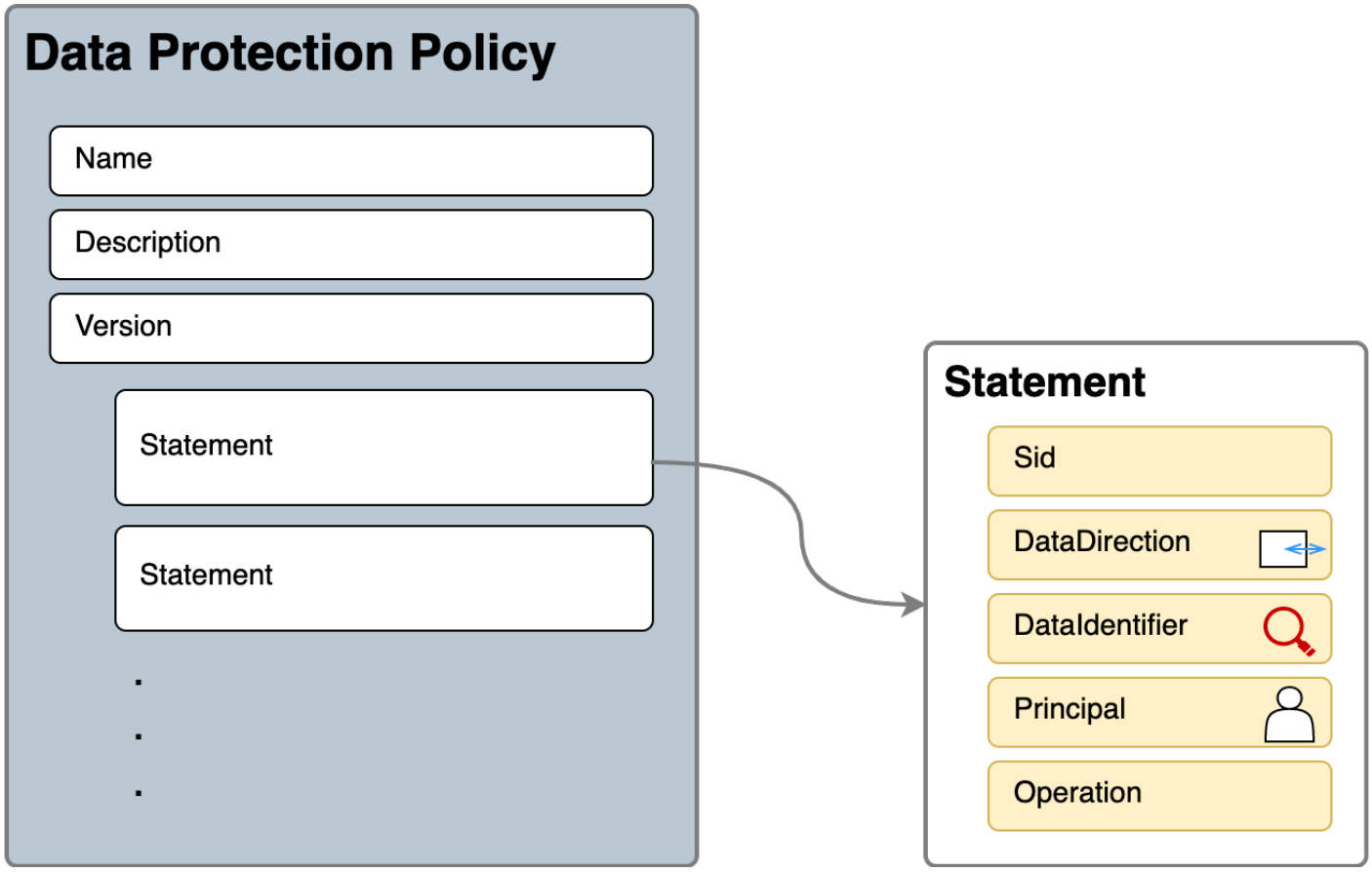


Wie ist die Datenschutzrichtlinie aufgebaut?

Wie in der folgenden Abbildung dargestellt, umfasst ein Datenschutzrichtlinien-Dokument die folgenden Elemente:

- Optionale richtlinienweite Informationen oben im Dokument.
- Eine oder mehrere einzelne Anweisungen

Jede Anweisung enthält Angaben über die jeweilige Berechtigung.



Für jedes Amazon-SNS-Thema kann nur eine Datenschutzrichtlinie definiert werden. Die Datenschutzrichtlinie kann mindestens eine Verweigerungs- oder Anonymisierungsanweisung aber nur eine Prüfungsanweisung enthalten.

JSON-Eigenschaften für die Datenschutzrichtlinie

Eine Datenschutzrichtlinie erfordert die folgenden grundlegenden Richtlinieninformationen zur Identifizierung:

- Name – der Name der Richtlinie
- Beschreibung (Optional) – die Beschreibung der Richtlinie
- Version – die Sprachversion der Richtlinie. Die aktuelle Version ist 2021-06-01.
- Anweisung – Eine Liste mit Anweisungen, die Aktionen der Datenschutzrichtlinie angeben.

```
{
  "Name": "basicPII-protection",
```

```
  "Description": "Protect basic types of sensitive data",
  "Version": "2021-06-01",
  "Statement": [
        ...
  ]
}
```

JSON-Eigenschaften für eine Richtlinienanweisung

Eine Richtlinienanweisung legt den Erkennungskontext für den Datenschutzvorgang fest.

- Sid (Optional) – der Anweisungsbezeichner
- DataDirection – „Eingehend“ (für API-Anfragen zur Veröffentlichung) oder „Ausgehend“ (für Benachrichtigungszustellungen) in Bezug auf das Amazon-SNS-Thema.
- DataIdentifier – Die sensiblen Daten, nach denen im Amazon-SNS-Thema gesucht werden soll. Zum Beispiel Name, Adresse oder Telefonnummer.
- Prinzipal – Der IAM-Prinzipal, der in dem Thema veröffentlicht ist, oder der IAM-Prinzipal, der das Thema abonniert hat.
- Vorgang – die Folgeaktion, entweder Audit (Prüfen), De-identify (Anonymisieren) (maskieren oder redigieren) oder Deny (Verweigern) (Blockieren), die das Amazon-SNS-Thema ausführt, sobald es sensible Daten findet.

```
{
    "Sid": "basicPII-inbound-protection",
    "DataDirection": "Inbound",
    "Principal": ["*"],
    "DataIdentifier": [
        "arn:aws:dataprotection::aws:data-identifier/Name",
        "arn:aws:dataprotection::aws:data-identifier/PhoneNumber-US"
    ],
    "Operation": {
        ...
    }
}
```

JSON-Eigenschaften für eine Richtlinienanweisungsoperation

Eine Richtlinienanweisung legt eine der folgenden Datenschutzoperationen fest.

- Audit (Prüfen) – Gibt Metriken und Ergebnisprotokolle aus, ohne die Veröffentlichung oder Zustellung von Nachrichten zu unterbrechen.
- Anonymisieren – maskieren oder redigieren Sie sensible Daten, ohne die Nachrichtenveröffentlichung zu unterbrechen.
- Verweigern – blockiert die Amazon-SNS-Veröffentlichungsanforderung oder verhindert die Nachrichtenzustellung.

Wie ermittle ich die IAM-Prinzipale für meine Datenschutzrichtlinie?

Der Nachrichtendatenschutz verwendet zwei IAM-Prinzipale, die mit Amazon SNS interagieren.

1. Publish-API-Prinzipal (eingehend) – Der authentifizierte IAM-Prinzipal, der die `Publish`-API von Amazon SNS aufruft.
2. Abonnementprinzipal (ausgehend) – Der authentifizierte IAM-Prinzipal, der die `Subscribe`-API während der Abonnementerstellung aufgerufen hat.

Der `SubscriptionPrincipal` ist eine öffentlich zugängliche Amazon-SNS-Abonnementeigenschaft, die über die `GetSubscriptionAttributes`-API abgerufen werden kann.

```
{
  "Attributes": {
    "SubscriptionPrincipal": "arn:aws:iam::123456789012:user/NoNameAccess",
    "Owner": "123412341234",
    "RawMessageDelivery": "true",
    "TopicArn": "arn:aws:sns:us-east-1:123412341234:PII-data-topic",
    "Endpoint": "arn:aws:sqs:us-east-1:123456789012:NoNameAccess",
    "Protocol": "sqs",
    "PendingConfirmation": "false",
    "ConfirmationWasAuthenticated": "true",
    "SubscriptionArn": "arn:aws:sns:us-east-1:123412341234:PII-data-
topic:5d8634ef-67ef-49eb-a824-4042b28d6f55"
  }
}
```

Operationen der Datenschutzrichtlinien

Im Folgenden finden Sie Beispiele für Datenschutzrichtlinien, mit denen Sie sensible Daten überprüfen und verweigern können. Ein vollständiges Tutorial, das eine Beispielanwendung enthält,

finden Sie im Blogbeitrag Introducing message data protection for Amazon SNS (Einführung in den Nachrichtendatenschutz für Amazon SNS).

Themen

- Audit-Operation
- Vorgang anonymisieren
- Deny-Operation

Audit-Operation

Die Audit-Operation führt ein Sampling der eingehenden Nachrichten des Themas aus und protokolliert die gefundenen sensiblen Daten in einem AWS-Ziel. Die Samplerate kann eine ganze Zahl zwischen 0 und 99 sein. Für diese Operation ist eine der folgenden Arten von Protokollierungszielen erforderlich:

1. FindingsDestination – Das Protokollierungsziel, wenn das Amazon SNS-Thema sensible Daten in der Nutzlast findet.
2. NoFindingsDestination – Das Protokollierungsziel, wenn das Amazon SNS-Thema keine sensiblen Daten in der Nutzlast findet.

Sie können die folgenden AWS-Services in jeder der folgenden Arten von Protokollierungszielen verwenden:

- Amazon CloudWatch Logs (optional) – Die `LogGroup` muss sich in der Themenregion befinden und der Name muss mit /aws/vendedlogs/ beginnen.
- Amazon Data Firehose (optional) – Die `DeliveryStream` muss sich in der Themenregion befinden und Direct PUT als Quelle des Bereitstellungs-Streams haben. Weitere Informationen finden Sie unter Quelle, Ziel und Name im Amazon-Data-Firehose-Entwicklerhandbuch.
- Amazon S3 (Optional) – Ein Amazon-S3-Bucket-Name. Für die Verwendung des Amazon-S3-Buckets mit aktivierter SSE-KMS-Verschlüsselung sind zusätzliche Aktionen erforderlich.

```
{
  "Operation": {
    "Audit": {
      "SampleRate": "99",
      "FindingsDestination": {
```

```
            "CloudWatchLogs": {
                "LogGroup": "/aws/vendedlogs/log-group-name"
            },
            "Firehose": {
                "DeliveryStream": "delivery-stream-name"
            },
            "S3": {
                "Bucket": "bucket-name"
            }
        },
        "NoFindingsDestination": {
            "CloudWatchLogs": {
                "LogGroup": "/aws/vendedlogs/log-group-name"
            },
            "Firehose": {
                "DeliveryStream": "delivery-stream-name"
            },
            "S3": {
                "Bucket": "bucket-name"
            }
        }
    }
  }
}
```

Erforderliche Berechtigungen bei der Angabe von Protokollierungszielen

Wenn Sie in der Datenschutzrichtlinie Ziele für die Protokollierung angeben, müssen Sie der IAM-Identitätsrichtlinie des IAM-Prinzipals, der in Amazon SNS die `PutDataProtectionPolicy`-API oder die `CreateTopic`-API mit dem `--data-protection-policy`-Parameter aufruft, die folgenden Berechtigungen hinzufügen.

Audit-Ziel	IAM-Berechtigung
Standard	logs:CreateLogDelivery
	logs:GetLogDelivery
	logs:UpdateLogDelivery
	logs:DeleteLogDelivery

Audit-Ziel	IAM-Berechtigung
	logs:ListLogDeliveries
CloudWatchLogs	logs:PutResourcePolicy logs:DescribeResourcePolicies logs:DescribeLogGroups
Firehose	iam:CreateServiceLinkedRole firehose:TagDeliveryStream
S3	s3:PutBucketPolicy s3:GetBucketPolicy Für die Verwendung des Amazon-S3-Buckets mit aktivierter SSE-KMS-Verschlüsselung sind zusätzliche Aktionen erforderlich.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "logs:CreateLogDelivery",
        "logs:GetLogDelivery",
        "logs:UpdateLogDelivery",
        "logs:DeleteLogDelivery",
        "logs:ListLogDeliveries"
      ],
      "Resource": [
        "*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "logs:PutResourcePolicy",
```

```
        "logs:DescribeResourcePolicies",
        "logs:DescribeLogGroups"
      ],
      "Resource": [
        "arn:aws:logs:region:account-id:SampleLogGroupName:*:*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "iam:CreateServiceLinkedRole",
        "firehose:TagDeliveryStream"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:PutBucketPolicy",
        "s3:GetBucketPolicy"
      ],
      "Resource": [
        "arn:aws:s3:::bucket-name"
      ]
    }
  ]
}
```

Erforderliche Schlüsselrichtlinie zur Verwendung mit SSE-KMS

Durch Verwendung eines Amazon-S3-Buckets als Protokollziel können Sie die Daten in Ihrem Bucket schützen, indem Sie entweder die serverseitige Verschlüsselung mit von Amazon S3 verwalteten Schlüsseln (SSE-S3) oder die serverseitige Verschlüsselung mit AWS KMS keys (SSE-KMS) aktivieren. Weitere Informationen finden Sie unter Schützen von Daten mithilfe serverseitiger Verschlüsselung im Amazon S3-Entwicklerhandbuch.

Wenn Sie SSE-S3 wählen, ist keine zusätzliche Konfiguration erforderlich. Amazon S3 verarbeitet den Verschlüsselungsschlüssel.

Wenn Sie sich für SSE-KMS entscheiden, müssen Sie einen vom Kunden verwalteten Schlüssel verwenden. Sie müssen die Schlüsselrichtlinie für Ihren vom Kunden verwalteten Schlüssel aktualisieren, damit das Protokollzustellungskonto in Ihren S3-Bucket schreiben kann. Weitere

Informationen zur erforderlichen Schlüsselrichtlinie für die Verwendung mit SSE-KMS finden Sie unter Serverseitige Verschlüsselung des Amazon S3-Buckets im Amazon- CloudWatch Logs-Benutzerhandbuch.

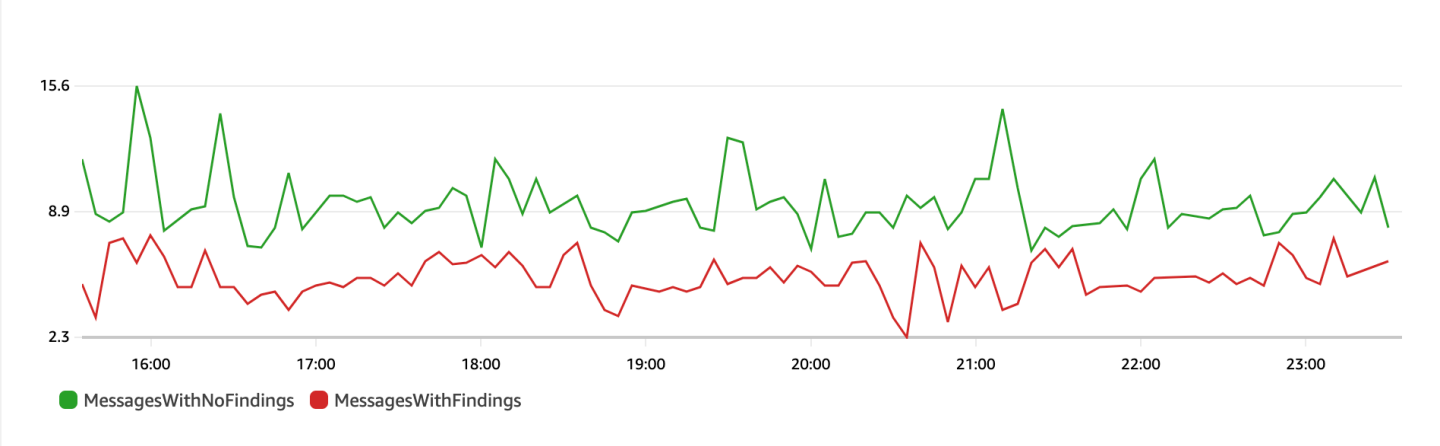
Beispiel für ein Audit-Zielprotokoll

Im folgenden Beispiel wird `callerPrincipal` verwendet, um die Quelle des sensiblen Inhalts zu identifizieren, und `messageID` als Referenz zur Überprüfung der `Publish`-API-Antwort.

```
{
  "messageId": "34d9b400-c6dd-5444-820d-fbeb0f1f54cf",
  "auditTimestamp": "2022-05-12T2:10:44Z",
  "callerPrincipal": "arn:aws:iam::123412341234:role/Publisher",
  "resourceArn": "arn:aws:sns:us-east-1:123412341234:PII-data-topic",
  "dataIdentifiers": [
    {
      "name": "Name",
      "count": 1,
      "detections": [
        {
          "start": 1,
          "end": 2
        }
      ]
    },
    {
      "name": "PhoneNumber",
      "count": 2,
      "detections": [
        {
          "start": 3,
          "end": 4
        },
        {
          "start": 5,
          "end": 6
        }
      ]
    }
  ]
}
```

Metriken für die Audit-Operation

Wenn ein Audit-Vorgang die `NoFindingsDestination` Eigenschaft `FindingsDestination` oder angegeben hat, erhalten CloudWatch die Themenbesitzer auch - `MessagesWithFindings` und -`MessagesWithNoFindings`Metriken.



Vorgang anonymisieren

Der Vorgang der Anonymisierung maskiert oder redigiert sensible Daten aus veröffentlichten oder übermittelten Nachrichten. Dieser Vorgang ist nur für eingehende und ausgehende Nachrichten verfügbar und erfordert einen der folgenden Konfigurationstypen:

- MaskConfig – Maskieren Sie mit einem unterstützten Zeichen aus der folgenden Tabelle. Beispielsweise wird die SSN: `123-45-6789` zur SSN: `###########`.

```
{
"Operation": {
    "Deidentify": {
        "MaskConfig": {
            "MaskWithCharacter": "#"
          }
    }
}
```

Unterstütztes Maskierungszeichen	Name
*	Sternchen
A–Z, a–z und 0–9	Alphanumerisch

Unterstütztes Maskierungszeichen	Name
	Leerzeichen
!	Ausrufezeichen
$	Dollarzeichen
%	Prozentzeichen
&	Kaufmännisches Und-Zeichen
()	Klammer
+	Pluszeichen
,	Komma
-	Bindestrich
.	Intervall
/\	Schrägstrich, umgekehrter Schrägstrich
#	Doppelkreuz
:	Doppelpunkt
;	Semikolon
=, <>	Gleich. kleiner oder größer als
@	At-Zeichen
[]	Klammern
^	Caret-Symbol
_	Unterstrich
`	Backtick

Unterstütztes Maskierungszeichen	Name
|	Senkrechter Balken
~	Tilde-Zeichen

- RedactConfig – Redigieren Sie, indem Sie die Daten vollständig entfernen. Beispielsweise wird SSN: `123-45-6789` zur SSN: .

```
{
"Operation": {
    "Deidentify": {
        "RedactConfig": {}
    }
}
```

Bei einer eingehenden Nachricht werden die sensiblen Daten nach dem Prüfvorgang anonymisiert, und der `SNS:Publish`-API-Aufrufer erhält den folgenden „Ungültiger Parameter“-Fehler, wenn die gesamte Nachricht vertraulich ist.

```
Error code: AuthorizationError ...
```

Deny-Operation

Die Deny-Operation unterbricht entweder die `Publish`-API-Anfrage oder die Zustellung der Nachricht, wenn die Nachricht sensible Daten enthält. Das Deny-Operationsobjekt ist leer, da es keine zusätzliche Konfiguration erfordert.

```
"Operation": {
    "Deny": {}
}
```

Bei einer eingehenden Nachricht erhält der `SNS:Publish`-API-Aufrufer einen Autorisierungsfehler.

```
Error code: AuthorizationError ...
```

Bei einer ausgehenden Nachricht stellt das Amazon-SNS-Thema die Nachricht nicht dem Abonnement zu. Wenn Sie nicht autorisierte Zustellungen verfolgen möchten, aktivieren Sie die Protokollierung des Zustellungsstatus für das Thema. Im Folgenden finden Sie ein Beispiel für ein Zustellungsstatusprotokoll:

```
{
    "notification": {
        "messageMD5Sum": "29638742ffb68b32cf56f42a79bcf16b",
        "messageId": "34d9b400-c6dd-5444-820d-fbeb0f1f54cf",
        "topicArn": "arn:aws:sns:us-east-1:123412341234:PII-data-topic",
        "timestamp": "2022-05-12T2:12:44Z"
    },
    "delivery": {
        "deliveryId": "98236591c-56aa-51ee-a5ed-0c7d43493170",
        "destination": "arn:aws:sqs:us-east-1:123456789012:NoNameAccess",
        "providerResponse": "The topic's data protection policy prohibits this message
 from being delivered to <subscription-arn>",
        "dwellTimeMs":20,
        "attempts":1,
        "statusCode": 403
    },
    "status": "FAILURE"
}
```

Beispiele für Datenschutzrichtlinien

Im Folgenden finden Sie Beispiele für Datenschutzrichtlinien, mit denen Sie sensible Daten überprüfen und ablehnen können. Ein vollständiges Tutorial, das eine Beispielanwendung enthält, finden Sie im Blogbeitrag Introducing message data protection for Amazon SNS (Einführung in den Nachrichtendatenschutz für Amazon SNS).

Themen

- Beispielrichtlinie für das Prüfen
- Beispielrichtlinie mit eingehender Anonymisierungsmaskierungsanweisung
- Beispielrichtlinie mit eingehender Anonymisierungsredigierungsanweisung
- Beispielrichtlinie mit ausgehender Anonymisierungsmaskierungsanweisung
- Beispielrichtlinie mit ausgehender Anonymisierungsredaktionsanweisung
- Beispiel für eine Richtlinie mit einer Verweigerungsanweisung für eingehende Nachrichten
- Beispiel für eine Richtlinie mit einer Verweigerungsanweisung für ausgehende Nachrichten

Beispielrichtlinie für das Prüfen

Mit Prüfungsrichtlinien können Sie bis zu 99 % der eingehenden Nachrichten prüfen und Ergebnisse an Amazon CloudWatch, Amazon Data Firehose und Amazon S3 senden.

Beispielsweise können Sie eine Prüfungsrichtlinie erstellen, um festzustellen, ob eines Ihrer Systeme versehentlich sensible Daten sendet oder empfängt. Wenn Ihre Prüfungsergebnisse zeigen, dass Systeme Kreditkarteninformationen an Systeme senden, die diese nicht benötigen, können Sie eine Datenschutzrichtlinie erstellen, um die Zustellung der Daten zu blockieren.

Im folgenden Beispiel werden 99 % der Nachrichten überprüft, die das Thema durchlaufen, indem nach Kreditkartennummern gesucht und die Ergebnisse an CloudWatch Logs, Firehose und Amazon S3 gesendet werden.

Datenschutzrichtlinie:

```
{
  "Name": "__example_data_protection_policy",
  "Description": "Example data protection policy",
  "Version": "2021-06-01",
  "Statement": [
    {
      "DataDirection": "Inbound",
      "Principal": ["*"],
      "DataIdentifier": [
        "arn:aws:dataprotection::aws:data-identifier/CreditCardNumber"
      ],
      "Operation": {
        "Audit": {
          "SampleRate": "99",
          "FindingsDestination": {
            "CloudWatchLogs": {
              "LogGroup": "<example log name>"
            },
            "Firehose": {
              "DeliveryStream": "<example stream name>"
            },
            "S3": {
              "Bucket": "<example bucket name>"
            }
          }
        }
```

```
      }
    }
  ]
}
```

Beispiel für das Format von Prüfungsergebnissen:

```
{
    "messageId": "...",
    "callerPrincipal": "arn:aws:sts::123456789012:assumed-role/ExampleRole",
    "resourceArn": "arn:aws:sns:us-east-1:123456789012:ExampleArn",
    "dataIdentifiers": [
        {
            "name": "CreditCardNumber",
            "count": 1,
            "detections": [
                { "start": 1, "end": 2 }
            ]
        }
    ],
    "timestamp": "2021-04-20T00:33:40.241Z"
}
```

Beispielrichtlinie mit eingehender Anonymisierungsmaskierungsanweisung

Das folgende Beispiel verhindert durch die Maskierung der sensiblen Daten im Nachrichteninhalt, dass ein Benutzer eine Nachricht mit `CreditCardNumber` in einem Thema veröffentlicht.

```
{
  "Name": "__example_data_protection_policy",
  "Description": "Example data protection policy",
  "Version": "2021-06-01",
  "Statement": [
    {
      "DataDirection": "Inbound",
      "Principal": [
        "arn:aws:iam::123456789012:user/ExampleUser"
      ],
      "DataIdentifier": [
        "arn:aws:dataprotection::aws:data-identifier/CreditCardNumber"
      ],
```

```
      "Operation": {
        "Deidentify": {
          "MaskConfig": {
            "MaskWithCharacter": "#"
          }
        }
      }
    }
  ]
}
```

Beispiel für eingehende Anonymisierungsmaskierungsanweisung:

```
// original message
My credit card number is 4539894458086459

// delivered message
My credit card number is ################
```

Beispielrichtlinie mit eingehender Anonymisierungsredigierungsanweisung

Das folgende Beispiel verhindert durch Redigieren der sensiblen Daten im Nachrichteninhalt, dass ein Benutzer eine Nachricht mit `CreditCardNumber` in einem Thema veröffentlicht.

```
{
  "Name": "__example_data_protection_policy",
  "Description": "Example data protection policy",
  "Version": "2021-06-01",
  "Statement": [
    {
      "DataDirection": "Inbound",
      "Principal": [
        "arn:aws:iam::123456789012:user/ExampleUser"
      ],
      "DataIdentifier": [
        "arn:aws:dataprotection::aws:data-identifier/CreditCardNumber"
      ],
      "Operation": {
        "Deidentify": {
          "RedactConfig": {}
        }
      }
```

```
    }
  ]
}
```

Beispiel für eingehende Anonymisierungs- und Redigierungsergebnisse:

```
// original message
My credit card number is 4539894458086459

// delivered message
My credit card number is
```

Beispielrichtlinie mit ausgehender Anonymisierungsmaskierungsanweisung

Das folgende Beispiel verhindert durch die Maskierung der sensiblen Daten im Nachrichteninhalt, dass ein Benutzer eine Nachricht mit `CreditCardNumber` in einem Thema veröffentlicht.

```
{
  "Name": "__example_data_protection_policy",
  "Description": "Example data protection policy",
  "Version": "2021-06-01",
  "Statement": [
    {
      "DataDirection": "Outbound",
      "Principal": [
        "arn:aws:iam::123456789012:user/ExampleUser"
      ],
      "DataIdentifier": [
        "arn:aws:dataprotection::aws:data-identifier/CreditCardNumber"
      ],
      "Operation": {
        "Deidentify": {
          "MaskConfig": {
            "MaskWithCharacter": "-"
          }
        }
      }
    }
  ]
}
```

Beispiel für Ergebnisse einer Anonymisierungsmaskierungsanweisung:

```
// original message
My credit card number is 4539894458086459

// delivered message
My credit card number is ----------------
```

Beispielrichtlinie mit ausgehender Anonymisierungsredaktionsanweisung

Das folgende Beispiel verhindert durch die Redigierung der sensiblen Daten im Nachrichteninhalt, dass ein Benutzer eine Nachricht mit `CreditCardNumber` in einem Thema veröffentlicht.

```
{
  "Name": "__example_data_protection_policy",
  "Description": "Example data protection policy",
  "Version": "2021-06-01",
  "Statement": [
    {
      "DataDirection": "Outbound",
      "Principal": [
        "arn:aws:iam::123456789012:user/ExampleUser"
      ],
      "DataIdentifier": [
        "arn:aws:dataprotection::aws:data-identifier/CreditCardNumber"
      ],
      "Operation": {
        "Deidentify": {
          "RedactConfig": {}
        }
      }
    }
  ]
}
```

Beispiel für Ergebnisse einer Anonymisierungsredaktionsanweisung:

```
// original message
My credit card number is 4539894458086459

// delivered message
My credit card number is
```

Beispiel für eine Richtlinie mit einer Verweigerungsanweisung für eingehende Nachrichten

Das folgende Beispiel verhindert, dass ein Benutzer eine Nachricht mit `CreditCardNumber` im Nachrichteninhalt in einem Thema veröffentlicht. Abgelehnte Nutzlasten in der API-Antwort haben den Statuscode „403 AuthorizationError“.

```
{
  "Name": "__example_data_protection_policy",
  "Description": "Example data protection policy",
  "Version": "2021-06-01",
  "Statement": [
    {
      "DataDirection": "Inbound",
      "Principal": [
        "arn:aws:iam::123456789012:user/ExampleUser"
      ],
      "DataIdentifier": [
        "arn:aws:dataprotection::aws:data-identifier/CreditCardNumber"
      ],
      "Operation": {
        "Deny": {}
      }
    }
  ]
}
```

Beispiel für eine Richtlinie mit einer Verweigerungsanweisung für ausgehende Nachrichten

Das folgende Beispiel verhindert, dass ein AWS-Konto Nachrichten erhält, die `CreditCardNumber` enthalten.

```
{
  "Name": "__example_data_protection_policy",
  "Description": "Example data protection policy",
  "Version": "2021-06-01",
  "Statement": [
    {
      "DataDirection": "Outbound",
      "Principal": [
```

```
        "arn:aws:iam::123456789012:user/ExampleUser"
      ],
      "DataIdentifier": [
        "arn:aws:dataprotection::aws:data-identifier/CreditCardNumber"
      ],
      "Operation": {
        "Deny": {}
      }
    }
  ]
}
```

Beispiel für Verweigerungsergebnisse für ausgehende Nachrichten, protokolliert in Amazon CloudWatch:

```
{
  "notification": {
    "messageMD5Sum": "2e8f58ff2eeed723b56b15493fbfb5a5",
    "messageId": "8747a956-ebf1-59da-b291-f2c2e4b87c9c",
    "topicArn": "arn:aws:sns:us-east-2:664555388960:test1",
    "timestamp": "2022-09-08 15:40:57.144"
  },
  "delivery": {
    "deliveryId": "6a422437-78cc-5171-ad64-7fa3778507aa",
    "destination": "arn:aws:sqs:us-east-2:664555388960:test",
    "providerResponse": "The topic's data protection policy prohibits this message from
 being delivered to <subscription arn>",
    "dwellTimeMs": 22,
    "attempts": 1,
    "statusCode": 403
  },
  "status": "FAILURE"
}
```

Erstellen von Datenschutzrichtlinien

Datenschutzrichtlinien schützen die Daten, die in Ihren Amazon-SNS-Themen veröffentlicht werden, indem sensible Informationen, die zwischen Anwendungen oder AWS-Services übertragen werden geprüft, anonymisiert (maskiert oder redigiert) und verweigert (blockiert) werden. Sie können die AWS-API, die AWS CLI, AWS CloudFormation oder die AWS Management Console verwenden, um Datenschutzrichtlinien in Amazon SNS zu erstellen. Für jedes Amazon-SNS-Thema kann nur eine

Richtlinie definiert werden. Die Datenschutzrichtlinie kann mindestens eine Anonymisierungs- und Verweigerungsanweisungen aber nur eine Prüfungsanweisung enthalten.

Themen

- Erstellen von Datenschutzrichtlinien zum Schutz von Nachrichtendaten (API)
- Erstellen von Datenschutzrichtlinien zum Schutz von Nachrichtendaten (CLI)
- Erstellen von Datenschutzrichtlinien zum Schutz von Nachrichtendaten (CloudFormation)
- Erstellen von Datenschutzrichtlinien zum Schutz von Nachrichtendaten (Konsole)
- Erstellen von Datenschutzrichtlinien zum Schutz von Nachrichtendaten (SDK)

Erstellen von Datenschutzrichtlinien zum Schutz von Nachrichtendaten (API)

Anzahl und Größe von Amazon-SNS-Ressourcen in einem AWS-Konto sind begrenzt. Weitere Informationen finden Sie unter Amazon Simple Notification Service-Endpunkte und -Kontingente.

Erstellen von Datenschutzrichtlinien (AWS-API)

Sie können eine Amazon-SNS-Datenschutzrichtlinie mit der AWS-API erstellen.

So erstellen Sie eine Datenschutzrichtlinie zusammen mit einem Amazon-SNS-Thema (AWS-API)

Verwenden Sie die `DataProtectionPolicy`-Eigenschaft eines Amazon-SNS-Standardthemas:

- `CreateTopic`

So erstellen oder rufen Sie eine Datenschutzrichtlinie für ein vorhandenes Amazon-SNS-Thema ab (AWS-API)

Rufen Sie eine der folgenden Operationen auf:

- GetDataProtectionPolicy
- PutDataProtectionPolicy

Erstellen von Datenschutzrichtlinien zum Schutz von Nachrichtendaten (CLI)

Anzahl und Größe von Amazon-SNS-Ressourcen in einem AWS-Konto sind begrenzt. Weitere Informationen finden Sie unter Amazon Simple Notification Service-Endpunkte und -Kontingente.

Erstellen von Datenschutzrichtlinien (AWS CLI)

Sie können eine Amazon-SNS-Datenschutzrichtlinie mit der AWS Command Line Interface erstellen.

So erstellen Sie eine Datenschutzrichtlinie zusammen mit einem Amazon-SNS-Thema (AWS CLI)

Verwenden Sie die folgende Option, um eine neue Datenschutzrichtlinie zusammen mit einem standardmäßigen Amazon-SNS-Thema zu erstellen:

- create-topic

So erstellen oder rufen Sie eine Datenschutzrichtlinie für ein vorhandenes Amazon-SNS-Thema ab (AWS CLI)

Rufen Sie eine der folgenden Operationen auf:

- get-data-protection-policy
- put-data-protection-policy

Erstellen von Datenschutzrichtlinien zum Schutz von Nachrichtendaten (CloudFormation)

Anzahl und Größe von Amazon-SNS-Ressourcen in einem AWS-Konto sind begrenzt. Weitere Informationen finden Sie unter Amazon Simple Notification Service-Endpunkte und -Kontingente.

Erstellen von Datenschutzrichtlinien (CloudFormation)

Sie können eine Amazon-SNS-Datenschutzrichtlinie erstellen, indem Sie AWS CloudFormation verwenden.

So erstellen Sie eine Datenschutzrichtlinie zusammen mit einem Amazon-SNS-Thema (CloudFormation)

Verwenden Sie die folgende Option, um eine neue Datenschutzrichtlinie zusammen mit einem standardmäßigen Amazon-SNS-Thema zu erstellen:

- AWS::SNS::Topic

Erstellen von Datenschutzrichtlinien zum Schutz von Nachrichtendaten (Konsole)

Anzahl und Größe von Amazon-SNS-Ressourcen in einem AWS-Konto sind begrenzt. Weitere Informationen finden Sie unter Amazon Simple Notification Service-Endpunkte und -Kontingente.

So erstellen Sie eine Datenschutzrichtlinie zusammen mit einem Amazon-SNS-Thema (Konsole)

Verwenden Sie die folgende Option, um eine neue Datenschutzrichtlinie zusammen mit einem standardmäßigen Amazon-SNS-Thema zu erstellen.

1. Melden Sie sich bei der Amazon-SNS-Konsole an.
2. Wählen Sie ein Thema aus oder erstellen Sie ein neues Thema. Weitere Informationen zum Erstellen von Themen finden Sie unter Erstellen eines Amazon-SNS-Themas.
3. Wählen Sie auf der Seite Create topic (Thema erstellen) im Abschnitt Details die Option Standard aus.

   a. Geben Sie den Namen des neuen Themas ein.

   b. (Optional) Geben Sie einen Anzeigenamen für Ihr Thema ein.
4. Erweitern Sie Data protection policy (Datenschutzrichtlinie).
5. Wählen Sie einen Configuration mode (Konfigurationsmodus) aus:

   - Basic (Basismodus) – Definieren Sie eine Datenschutzrichtlinie über ein einfaches Menü.

   - Advanced (Erweiterter Modus) – Definieren Sie eine benutzerdefinierte Datenschutzrichtlinie mithilfe von JSON.
6. (Optional) Um Ihre eigene benutzerdefinierte Datenkennung zu erstellen, erweitern Sie den Abschnitt Custom data identifier configuration (Konfiguration für benutzerdefinierte Datenkennung) und gehen Sie wie folgt vor:

   a. Geben Sie einen eindeutigen Namen für die benutzerdefinierte Datenkennung ein. Benutzerdefinierte Datenkennungen können alphanumerische Zeichen, Unterstriche (_) und Bindestriche (-) enthalten. Es werden bis zu 128 Zeichen unterstützt. Dieser Name darf nicht denselben Namen wie eine verwaltete Datenkennung haben. Eine vollständige Liste der Einschränkungen für benutzerdefinierte Datenkennungen finden Sie unter Einschränkungen für benutzerdefinierte Datenkennungen.

   b. Geben Sie einen regulären Ausdruck (RegEx) für die benutzerdefinierte Datenkennung ein. RegEx unterstützt alphanumerische Zeichen, RegEx reservierte Zeichen und

Symbole. RegEx hat eine maximale Länge von 200 Zeichen. Wenn der zu kompliziert RegEx ist, schlägt Amazon SNS den API-Aufruf fehl. Eine vollständige Liste der RegEx Einschränkungen finden Sie unter Einschränkungen für benutzerdefinierte Datenkennungen.

c. (Optional) Wählen Sie Add custom data identifier (Benutzerdefinierte Datenkennung hinzufügen), um bei Bedarf weitere Datenkennungen hinzuzufügen. Datenschutzrichtlinien unterstützen derzeit maximal 10 benutzerdefinierte Datenkennungen.

7. Wählen Sie die Anweisung(en) aus, die Sie zu Ihrer Datenschutzrichtlinie hinzufügen möchten. Sie können derselben Datenschutzrichtlinie die Anwendungstypen audit (prüfen), de-identify (anonymisieren) (maskieren oder redigieren) und deny (verweigern) hinzufügen.

   a. Add audit statement (Audit-Anweisung hinzufügen) – Konfigurieren Sie, welche sensiblen Daten überprüft, wie viel Prozent der Nachrichten auf diese Daten überprüft und wohin Überwachungsprotokolle gesendet werden sollen.

      > **Note**
      >
      > Pro Datenschutzrichtlinie oder Thema ist nur eine Audit-Anweisung zulässig.

      i. Wählen Sie data identifiers (Datenkennungen) aus, um die sensiblen Daten zu definieren, die Sie prüfen möchten.

      ii. Geben Sie unter Audit sample rate (Audit-Samplerate) den Prozentsatz der Nachrichten ein, die auf sensible Informationen überprüft werden sollen (maximal 99 %).

      iii. Wählen Sie unter Audit destination (Audit-Ziel) aus, an welche AWS-Services die Prüfergebnisse gesendet werden sollen. Geben Sie einen Zielnamen für jeden AWS-Service ein, den Sie verwenden. Sie können einen der folgenden Amazon Web Services auswählen:

         - Amazon CloudWatch – CloudWatch Logs ist die AWS Standardprotokollierungslösung. Mithilfe von - CloudWatch Protokollen können Sie Protokollanalysen mit Logs Insights durchführen (siehe Beispiele hier) und Metriken und Alarme erstellen. - CloudWatch Protokolle sind ein Ort, an dem viele - Services Protokolle veröffentlichen, wodurch es einfacher ist, alle Protokolle mit einer Lösung zu aggregieren. Weitere Informationen zu Amazon CloudWatchfinden Sie im Amazon- CloudWatch Benutzerhandbuch.

- Amazon Data Firehose – Firehose erfüllt die Anforderungen an Echtzeit-Streaming an Splunk OpenSearchund Amazon Redshift für weitere Protokollanalysen. Weitere Informationen zu Amazon Data Firehose finden Sie im Amazon-Data-Firehose-Benutzerhandbuch.
- Amazon Simple Storage Service – Amazon S3 ist ein kostengünstiges Protokollziel für Archivierungszwecke. Möglicherweise müssen Sie Protokolle für einen Zeitraum von mehreren Jahren aufbewahren. In diesem Fall können Sie Protokolle in Amazon S3 speichern, um Kosten zu sparen. Weitere Informationen zu Amazon Simple Storage Service finden Sie im Benutzerhandbuch von Amazon Simple Storage Service.

b. Add a de-identify statement (Eine Anonymisierungsanweisung hinzufügen) – konfigurieren Sie die sensiblen Daten, die Sie in der Nachricht anonymisieren möchten, ob Sie diese Daten maskieren oder redigieren möchten, und die Konten, um die Zustellung dieser Daten zu beenden.

   i. Wählen Sie bei data identifiers (Datenkennungen) die sensiblen Daten aus, die Sie prüfen möchten.

   ii. Wählen Sie bei Define this de-identify statement for (Diese Anonymisierungsanweisung definieren für) die AWS-Konten oder IAM-Prinzipale aus, für die diese Anonymisierungsanweisung gilt. Sie können sie auf alle AWS-Konten oder auf spezifische AWS oder IAM-Entitäten (Konto-Roots, Rollen oder Benutzer) anwenden, die Konto-IDs oder IAM-Entitäts-ARNs verwenden. Trennen Sie mehrere IDs oder ARNs durch ein Komma (,).

   Die folgenden IAM-Prinzipale werden unterstützt:

   - IAM account principals (IAM-Kontoprinzipale) – zum Beispiel `arn:aws:iam::AWS-account-ID:root`.
   - IAM role principals (IAM-Rollenprinzipale) – zum Beispiel `arn:aws:iam::AWS-account-ID:role/role-name`.
   - IAM user principals (IAM-Benutzerprinzipale) – zum Beispiel `arn:aws:iam::AWS-account-ID:user/user-name`.

   iii. Wählen Sie bei De-identify Option (Anonymisierungsoption) die zu prüfenden sensiblen Daten aus. Die folgenden Optionen werden unterstützt:

- Redact (Redigieren) – löscht Daten vollständig. Beispielsweise wird die E-Mail-Adresse: `classified@amazon.com` zur E-Mail-Adresse: .
- Mask (Maskieren) – ersetzt die Daten durch einzelne Zeichen. Beispielsweise wird die E-Mail-Adresse: `classified@amazon.com` zur E-Mail-Adresse: `**********`.

iv. (Optional) Fügen Sie bei Bedarf weitere de-identify-Anweisungen hinzu.

c. Add deny statement (Verweigerungsanweisung hinzufügen) – legen Sie fest, welche sensiblen Daten sich nicht in Ihrem Thema bewegen dürfen und welche Prinzipale diese Daten nicht senden dürfen.

i. Wählen Sie bei data direction (Datenrichtung) die Richtung der Nachrichten für die Ablehnungsanweisung aus:

- Inbound messages (Eingehende Nachrichten) – Wenden Sie diese Deny-Anweisung auf Nachrichten an, die an das Thema gesendet werden.
- Outbound messages (Ausgehende Nachrichten) – Wenden Sie diese Deny-Anweisung auf Nachrichten an, die das Thema an Abonnementendpunkte sendet.

ii. Wählen Sie die data identifiers (Datenkennungen) aus, um die sensiblen Daten zu definieren, die Sie verweigern möchten.

iii. Wählen Sie die IAM principals (IAM-Prinzipale) aus, für die diese Verweigerungsanweisung gilt. Sie können sie auf alle AWS-Konten oder auf spezifische AWS-Konten oder IAM-Entitäten (z. B. Konto-Roots, Rollen oder Benutzer) anwenden, die Konto-IDs oder IAM-Entitäts-ARNs verwenden. Trennen Sie mehrere IDs oder ARNs durch ein Komma (,). Die folgenden IAM-Prinzipale werden unterstützt:

- IAM-Kontoprinzipale – zum Beispiel `arn:aws:iam::AWS-account-ID:root`.
- IAM-Rollenprinzipale – zum Beispiel `arn:aws:iam::AWS-account-ID:role/role-name`.
- IAM-Benutzerprinzipale – zum Beispiel `arn:aws:iam::AWS-account-ID:user/user-name`.

iv. (Optional) Fügen Sie bei Bedarf weitere Deny-Anweisungen hinzu.

Erstellen von Datenschutzrichtlinien zum Schutz von Nachrichtendaten (SDK)

Anzahl und Größe von Amazon-SNS-Ressourcen in einem AWS-Konto sind begrenzt. Weitere Informationen finden Sie unter Amazon Simple Notification Service-Endpunkte und -Kontingente.

Erstellen von Datenschutzrichtlinien (AWS SDK)

Sie können eine Amazon-SNS-Datenschutzrichtlinie mit dem AWS SDK erstellen.

So erstellen Sie eine Datenschutzrichtlinie zusammen mit einem Amazon-SNS-Thema (AWS SDK)

Verwenden Sie die folgenden Optionen, um eine neue Datenschutzrichtlinie zusammen mit einem standardmäßigen Amazon-SNS-Thema zu erstellen:

Java

```
/**
 * For information regarding CreateTopic see this documentation topic:
 *
 * https://docs.aws.amazon.com/code-samples/latest/catalog/javav2-sns-src-main-java-com-example-sns-CreateTopic.java.html
 */

public static String createSNSTopicWithDataProtectionPolicy(SnsClient snsClient,
 String topicName, String dataProtectionPolicy) {

    try {
        CreateTopicRequest request = CreateTopicRequest.builder()
                .name(topicName)
                .dataProtectionPolicy(dataProtectionPolicy)
                .build();

        CreateTopicResponse result = snsClient.createTopic(request);
        return result.topicArn();
    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
```

JavaScript

```
// Import required AWS SDK clients and commands for Node.js
import {CreateTopicCommand } from "@aws-sdk/client-sns";
import {snsClient } from "./libs/snsClient.js";

// Set the parameters
const params = { Name: "TOPIC_NAME", DataProtectionPolicy:
 "DATA_PROTECTION_POLICY" };

const run = async () => {
  try {
    const data = await snsClient.send(new CreateTopicCommand(params));
    console.log("Success.",  data);
    return data; // For unit tests.
  } catch (err) {
    console.log("Error", err.stack);
  }
};
run();
```

So erstellen oder rufen Sie eine Datenschutzrichtlinie für ein vorhandenes Amazon-SNS-Thema ab (AWS SDK)

Verwenden Sie die folgenden Optionen, um eine neue Datenschutzrichtlinie zusammen mit einem standardmäßigen Amazon-SNS-Thema zu erstellen oder abzurufen:

Java

```
public static void putDataProtectionPolicy(SnsClient snsClient, String topicName,
 String dataProtectionPolicy) {

    try {
        PutDataProtectionPolicyRequest request =
 PutDataProtectionPolicyRequest.builder()
                .resourceArn(topicName)
                .dataProtectionPolicy(dataProtectionPolicy)
                .build();

        PutDataProtectionPolicyResponse result =
 snsClient.putDataProtectionPolicy(request);
```

```
        System.out.println("\n\nStatus was " +
 result.sdkHttpResponse().statusCode()
                + "\n\nTopic " + request.resourceArn()
                + " DataProtectionPolicy " + request.dataProtectionPolicy());
    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}


public static void getDataProtectionPolicy(SnsClient snsClient, String topicName) {

    try {
        GetDataProtectionPolicyRequest request =
 GetDataProtectionPolicyRequest.builder()
                .resourceArn(topicName)
                .build();

        GetDataProtectionPolicyResponse result =
 snsClient.getDataProtectionPolicy(request);

        System.out.println("\n\nStatus is " + result.sdkHttpResponse().statusCode()
        + "\n\nDataProtectionPolicy: \n\n" + result.dataProtectionPolicy());
    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

JavaScript

```
// Import required AWS SDK clients and commands for Node.js
import {PutDataProtectionPolicyCommand, GetDataProtectionPolicyCommand } from "@aws-
sdk/client-sns";
import {snsClient } from "./libs/snsClient.js";

// Set the parameters
const putParams = { ResourceArn: "TOPIC_ARN", DataProtectionPolicy:
 "DATA_PROTECTION_POLICY" };

const runPut = async () => {
  try {
```

```
    const data = await snsClient.send(new
 PutDataProtectionPolicyCommand(putParams));
    console.log("Success.",  data);
    return data; // For unit tests.
  } catch (err) {
    console.log("Error", err.stack);
  }
};
runPut();

// Set the parameters
const getParams = { ResourceArn: "TOPIC_ARN" };

const runGet = async () => {
  try {
    const data = await snsClient.send(new
 GetDataProtectionPolicyCommand(getParams));
    console.log("Success.",  data);
    return data; // For unit tests.
  } catch (err) {
    console.log("Error", err.stack);
  }
};
runGet();
```

Löschen von Datenschutzrichtlinien in Amazon SNS

Sie können die AWS-API, die AWS CLI, AWS CloudFormation oder die AWS Management Console verwenden, um Datenschutzrichtlinien in Amazon SNS zu löschen.

Allgemeine Informationen zu Datenschutzrichtlinien von Amazon SNS finden Sie unter Informationen über Datenschutzrichtlinien.

Anzahl und Größe von Amazon-SNS-Ressourcen für Datenschutzrichtlinien in einem AWS-Konto sind begrenzt. Weitere Informationen finden Sie unter Amazon-SNS-API-Drosselung in Allgemeine AWS-Referenz.

Themen

- Löschen von Datenschutzrichtlinien (Konsole)
- Löschen einer Datenschutzrichtlinie mit einer leeren JSON-Zeichenfolge

- Löschen einer Datenschutzrichtlinie mit der AWS CLI

Löschen von Datenschutzrichtlinien (Konsole)

So löschen Sie eine verwaltete Datenschutzrichtlinie (Konsole)

1. Melden Sie sich bei der Amazon-SNS-Konsole an.
2. Wählen Sie das Thema aus, das die Datenschutzrichtlinie enthält, die Sie löschen möchten.
3. Wählen Sie Edit (Bearbeiten) aus.
4. Erweitern Sie den Abschnitt Data protection policy (Datenschutzrichtlinie).
5. Wählen Sie Remove (Entfernen) neben der Datenschutzrichtlinie aus, die Sie entfernen möchten.
6. Wählen Sie Save Changes.

Löschen einer Datenschutzrichtlinie mit einer leeren JSON-Zeichenfolge

Sie können eine Datenschutzrichtlinie löschen, indem Sie sie auf eine leere JSON-Zeichenfolge aktualisieren.

Löschen einer Datenschutzrichtlinie mit der AWS CLI

Sie können eine Datenschutzrichtlinie mit der AWS CLI löschen.

```
//aws sns put-data-protection-policy --resource-arn topic-arn --data-protection-policy ""
```

Datenkennungen

Amazon SNS verwendet eine Kombination aus Kriterien und Techniken, einschließlich Machine Learning und Musterabgleich, um sensible Daten zu erkennen. Diese Kriterien und Techniken werden zusammenfassend als Datenkennungen bezeichnet. Sie können eine große und wachsende Liste vertraulicher Datentypen für viele Länder und Regionen erkennen. Amazon SNS verwaltete Datenkennungen bieten vorkonfigurierte Datentypen zum Schutz von Finanzdaten, persönlichen Gesundheitsinformationen (PHI) und persönlich identifizierbaren Informationen (PII). Sie können auch benutzerdefinierte Datenkennungen verwenden, um Ihre eigenen Datenkennungen zu erstellen, die auf Ihren speziellen Anwendungsfall zugeschnitten sind.

Themen

Verwenden von verwalteten Datenkennungen in Amazon SNS

Themen

Was sind verwaltete Datenkennungen?

Von Amazon SNS verwaltete Datenkennungen sind so konzipiert, dass sie eine bestimmte Art von sensiblen Daten erkennt, wie Kreditkartennummern, geheime AWS-Zugriffsschlüssel oder Passnummern für ein bestimmtes Land oder eine bestimmte Region. Beim Erstellen einer Datenschutzrichtlinie können Sie Amazon SNS so konfigurieren, dass diese Kennungen verwendet werden, um Nachrichten zu analysieren, die das Thema durchlaufen, und bei entsprechender Erkennung Maßnahmen zu ergreifen.

Amazon SNS kann mithilfe verwalteter Datenkennungen die folgenden Kategorien sensibler Daten erkennen:

- Anmeldeinformationen, wie private Schlüssel oder geheime AWS-Zugriffsschlüssel
- Gerätekennungen, wie IP-Adresse oder MAC-Adresse
- Finanzinformationen, wie Kreditkartennummern
- Gesundheitsinformationen für PHI, wie Krankenversicherungs- oder medizinische Identifikationsnummern
- Personenbezogene Daten für PII, wie Führerscheine oder Sozialversicherungsnummern

Innerhalb jeder Kategorie kann Amazon SNS mehrere Arten sensibler Daten erkennen. In den Themen in diesem Abschnitt werden die einzelnen Arten und alle relevanten Anforderungen für deren Erkennung aufgeführt und beschrieben. Für jede Art geben sie auch die eindeutige Kennung (ID) für die verwaltete Datenkennung an, die für die Erkennung der Daten konzipiert ist. Wenn Sie eine Datenschutzrichtlinie erstellen, können Sie diese ID verwenden, um die verwaltete Datenkennung für die Erkennung durch den Nachrichtendatenschutz einzuschließen.

Anforderungen an Schlüsselwörter

Um bestimmte Arten sensibler Daten zu erkennen, sucht Amazon SNS in der Nähe der Daten nach Schlüsselwörtern. Wenn dies bei einem bestimmten Datentyp der Fall ist, werden in einem nachfolgenden Thema in diesem Abschnitt spezifische Schlüsselwortanforderungen für diese Daten angegeben.

Bei Schlüsselwörtern muss die Groß- und Kleinschreibung nicht beachtet werden. Wenn ein Schlüsselwort ein Leerzeichen enthält, gleicht Amazon SNS außerdem automatisch Schlüsselwortvarianten ab, die das Leerzeichen nicht enthalten oder anstelle des Leerzeichens einen Unterstrich (_) oder einen Bindestrich (-) enthalten. In bestimmten Fällen erweitert oder kürzt Amazon SNS ein Schlüsselwort auch, um gängige Variationen des Schlüsselworts zu berücksichtigen.

Von Amazon SNS verwaltete Datenkennungen für sensible Datentypen

In der folgenden Liste werden die Arten von Anmeldeinformationen, Geräte-, Finanz- und medizinische Daten sowie persönliche Gesundheitsinformationen (PHI) aufgeführt und beschrieben, die Amazon SNS mithilfe verwalteter Datenkennungen erkennen kann. Diese gelten zusätzlich zu bestimmten Arten von Daten, die auch als persönlich identifizierbare Informationen (PII) gelten können.

Für regionsabhängige Datenkennungen sind der Kennungsname mit einem Bindestrich und die aus zwei Buchstaben bestehenden Codes (ISO 3166-1 Alpha-2) erforderlich. Zum Beispiel DriversLicense-USA.

Kennung	Kategorie	Länder/Sprachen
BankAccountNumber	Finanzanwendungen	DE, ES, FR, GB, IT
CepCode	Persönlich	BR
Cnpj	Persönlich	BR

Kennung	Kategorie	Länder/Sprachen
CpfCode	Persönlich	BR
DriversLicense	Persönlich	AT, AU, BE, BG, CA, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IT, LT, LU, LV, MT, NL, PL, PT, RO, SE, SI, SK, US
DrugEnforcementAgencyNumber	Gesundheit	US
ElectoralRollNumber	Persönlich	GB
HealthInsuranceCardNumber	Gesundheit	EU
HealthInsuranceClaimNumber	Gesundheit	US
HealthInsuranceNumber	Gesundheit	FR
HealthcareProcedureCode	Gesundheit	US
IndividualTaxIdentificationNumber	Persönlich	US
InseeCode	Persönlich	FR
MedicareBeneficiaryNumber	Gesundheit	US
NationalDrugCode	Gesundheit	US
NationalIdentificationNumber	Persönlich	DE, ES, IT
NationalInsuranceNumber	Persönlich	GB
NationalProviderId	Gesundheit	US
NhsNumber	Gesundheit	GB
NieNumber	Persönlich	ES

Kennung	Kategorie	Länder/Sprachen
NifNumber	Persönlich	ES
PassportNumber	Persönlich	CA, DE, ES, FR, GB, IT, US
PermanentResidenceNumber	Persönlich	CA
PersonalHealthNumber	Gesundheit	CA
PhoneNumber	Persönlich	BR, DE, ES, FR, GB, IT, US
PostalCode	Persönlich	CA
RgNumber	Persönlich	BR
SocialInsuranceNumber	Persönlich	CA
Ssn	Persönlich	ES, US
TaxId	Persönlich	DE, ES, FR, GB
ZipCode	Persönlich	US

Unterstützte Kennungen, die sprach-/regionsunabhängig sind

Kennung	Kategorie
Adresse	Persönlich
AwsSecretKey	Anmeldeinformationen
CreditCardExpiration	Finanzanwendungen
CreditCardNumber	Finanzanwendungen
CreditCardSecurityCode	Finanzanwendungen
EmailAddress	Persönlich

Kennung	Kategorie
IpAddress	Persönlich
LatLong	Persönlich
Name	Persönlich
OpenSshPrivateKey	Anmeldeinformationen
PgpPrivateKey	Anmeldeinformationen
PkcsPrivateKey	Anmeldeinformationen
PuttyPrivateKey	Anmeldeinformationen
VehicleIdentificationNumber	Persönlich

Sensible Datentypen: Anmeldeinformationen

In der folgenden Liste werden die Arten von Anmeldeinformationen aufgeführt und beschrieben, die Amazon SNS mithilfe verwalteter Datenkennungen erkennen kann.

Art der Erkennung	ID der verwalteten Datenkennung	Schlüsselwort erforderlich	Länder und Regionen
Geheimer AWS-Zugriffsschlüssel	AwsSecretKey	aws_secret_access_key, credentials, secret access key, secret key, set-awscredential	Any
Privater OpenSSH-Schlüssel	OpenSshPrivateKey	Nein	Any
Privater PGP-Schlüssel	PgpPrivateKey	Nein	Any

Art der Erkennung	ID der verwalteten Datenkennung	Schlüsselwort erforderlich	Länder und Regionen
Privater Public Key Cryptography Standard (PKCS)-Schlüssel	PkcsPrivateKey	Nein	Any
Privater PuTTY-Schlüssel	PuttyPrivateKey	Nein	Any

Datenkennungs-ARNs für Anmeldeinformations-Datentypen

Im Folgenden werden die Amazon-Ressourcennamen (ARNs) für die Datenkennungen aufgelistet, die Sie Ihren Datenschutzrichtlinien hinzufügen können.

ARNs der Datenkennung für Anmeldeinformationen
arn:aws:dataprotection::aws:data-identifier/AwsSecretKey
arn:aws:dataprotection::aws:data-identifier/OpenSshPrivateKey
arn:aws:dataprotection::aws:data-identifier/PgpPrivateKey
arn:aws:dataprotection::aws:data-identifier/PkcsPrivateKey
arn:aws:dataprotection::aws:data-identifier/PuttyPrivateKey

Sensible Datentypen: Geräte

In der folgenden Liste werden die Arten von Gerätekennungen aufgeführt und beschrieben, die Amazon SNS mithilfe verwalteter Datenkennungen erkennen kann.

Art der Erkennung	ID der verwalteten Datenkennung	Schlüsselwort erforderlich	Länder und Regionen
IP-Adresse	IpAddress	Nein	Any

Datenkennungs-ARNs für Gerätedatentypen

Im Folgenden werden die Amazon-Ressourcennamen (ARNs) für die Datenkennungen aufgelistet, die Sie Ihren Datenschutzrichtlinien hinzufügen können.

Gerätedatenkennungs-ARN
arn:aws:dataprotection::aws:data-identifier/IpAddress

Sensible Datentypen: Finanzdaten

In der folgenden Liste werden die Arten von Finanzdaten aufgeführt und beschrieben, die Amazon SNS mithilfe verwalteter Datenkennungen erkennen kann.

Art der Erkennung	ID der verwalteten Datenkennung	Schlüsselwort erforderlich	Zusätzliche Informationen	Länder und Regionen
Bankkontonummer	BankAccountNumber BankAccountNumber-US	Ja, siehe Schlüsselwörter für Bankkontonummern.	Dazu gehören: Internationale Bankkontonummern (IBANs), die aus bis zu 34 alphanumerischen Zeichen bestehen, einschließlich Elementen wie dem Ländercode.	Frankreich, Deutschland, Italien, Spanien, Vereinigtes Königreich
Ablaufdatum der Kreditkarte	CreditCardExpiration	exp d, exp m, exp y, expiration, expiry	–	Any

Art der Erkennung	ID der verwaltet en Datenkenn ung	Schlüsselwort erforderlich	Zusätzliche Informationen	Länder und Regionen
Daten des Kreditkarten-Magnetstreifens	CreditCar dMagneticStripe	Ja, einschlie ßlich: Kartendat en, ISO7813, MAG, Magnetstr eifen, Streifen, streichen.	Dies beinhaltet die Titel 1 und 2.	Any

Art der Erkennung	ID der verwaltet en Datenkenn ung	Schlüsselwort erforderlich	Zusätzliche Informationen	Länder und Regionen
Kreditkar tennummer	CreditCar dNumber	Kontonumm er, American Express, Amex, Bankkarte, Karte, Karten-Nr., Kartennum mer, cc #, CCN, Scheckkarte, Kredit, Kreditkar tennummer, Dankort, Debit, Debitkarte, Diners Club, Discover, Electron, Elo-Verifizierungs code, Japanese Card Bureau, JCB, Mastercar d, mc, pan, Zahlungsk ontonumme r, Zahlungsk artennummer, PCN, Union Pay, Visa	Für die Erkennung müssen die Daten eine 13- bis 19-stelli ge Sequenz sein, die der Luhn-Prüfformel entspricht und ein Standardk artennumm ernpräfix für eine der folgenden Arten von Kreditkarten verwendet : American Express, Dankort, Diner's Club, Discover, EOn, Japanese Card Bureau (JCB), Mastercar d UnionPayund Visa (Superscr ipt-Link unter 1).	Any

Art der Erkennung	ID der verwaltet en Datenkenn ung	Schlüsselwort erforderlich	Zusätzliche Informationen	Länder und Regionen
Bestätigu ngscode für die Kreditkarte	CreditCar dSecurityCode	Karten-ID , Kartenide ntifikationscode, Kartenidentifikati onsnummer , Kartensic herheitscode, Kartenval idierungscode, Kartenval idierungs nummer, Kartenübe rprüfungsdaten, Kartenübe rprüfungswert, cvc, cvc2, cvv, cvv2, elo-Bestä tigungscode	–	Any

1. Amazon SNS meldet keine Vorkommen der folgenden Sequenzen, die Kreditkartenaussteller für öffentliche Tests reserviert haben:

   122000000000003, 2222405343248877, 2222990905257051, 2223007648726984, 2223577120017656, 30569309025904, 34343434343434, 3528000700000000, 3530111333300000, 3566002020360505, 36148900647913, 36700102000000, 371449635398431, 378282246310005, 378734493671000, 38520000023237, 4012888888881881, 4111111111111111, 4222222222222, 4444333322221111, 4462030000000000, 4484070000000000, 4911830000000, 4917300800000000, 4917610000000000, 4917610000000000003, 5019717010103742, 5105105105105100, 5111010030175156, 5185540810000019, 5200828282828210, 5204230080000017,

5204740009900014, 5420923878724339, 5454545454545454, 5455330760000018, 5506900490000436, 5506900490000444, 5506900510000234, 5506920809243667, 5506922400634930, 5506927427317625, 5553042241984105, 5555553753048194, 5555555555554444, 5610591081018250, 6011000990139424, 6011000400000000, 6011111111111117, 630490017740292441, 630495060000000000, 6331101999990016, 6759649826438453, 6799990100000000019, and 76009244561.

Schlüsselwörter für Bankkontonummern

Verwenden Sie die folgenden Schlüsselwörter, um Internationale Bankkontonummern (IBANs) zu erkennen, die aus bis zu 34 alphanumerischen Zeichen bestehen, einschließlich Elementen wie dem Ländercode.

Land oder Region	Schlüsselwörter			
Frankreich	account code, account number, accountno #, accountnu mber#, bban, code bancaire, compte bancaire, customer account id, customer account number, customer bank account id, iban, numéro de compte			
Deutschland	account code, account number, accountno #, accountnu mber#, bankleitz			

Land oder Region	Schlüsselwörter			
	ahl, bban, customer account id, customer account number, customer bank account id, geheimzahl, iban, kartennummer, kontonummer, kreditkartennummer, sepa			
Italien	account code, account number, accountno #, accountnumber#, bban, codice bancario, conto bancario, customer account id, customer account number, customer bank account id, iban, numero di conto			

Land oder Region	Schlüsselwörter			
Spanien	account code, account number, accountno #, accountnu mber#, bban, código cuenta, código cuenta bancaria, cuenta cliente id, customer account ID, customer account number, customer bank account id, iban, número cuenta bancaria cliente, número cuenta cliente			
UK	account code, account number, accountno #, accountnu mber#, bban, customer account id, customer account number, customer bank account id, iban, sepa			

Land oder Region	Schlüsselwörter			
US	Bankkonto , Bankkto, Girokonto , Girokto, Depotkonto, Depositkto, Sparkonto, Sparkto			

Datenkennungs-ARNs für Finanzdatentypen

Im Folgenden werden die Amazon-Ressourcennamen (ARNs) für die Datenkennungen aufgelistet, die Sie Ihren Datenschutzrichtlinien hinzufügen können.

ARNs der Finanzdatenkennungen
arn:aws:dataprotection::aws:data-identifier/BankAccountNumber-DE
arn:aws:dataprotection::aws:data-identifier/BankAccountNumber-ES
arn:aws:dataprotection::aws:data-identifier/BankAccountNumber-FR
arn:aws:dataprotection::aws:data-identifier/BankAccountNumber-GB
arn:aws:dataprotection::aws:data-identifier/BankAccountNumber-IT
arn:aws:dataprotection::aws:data-identifier/BankAccountNumber-US
arn:aws:dataprotection::aws:data-identifier/CreditCardExpiration
arn:aws:dataprotection::aws:data-identifier/CreditCardNumber
arn:aws:dataprotection::aws:data-identifier/CreditCardSecurityCode

Sensible Datentypen: geschützte Gesundheitsdaten (PHI)

In der folgenden Liste werden die Arten von geschützten Gesundheitsdaten aufgeführt und beschrieben, die Amazon SNS mithilfe verwalteter Datenkennungen erkennen kann.

Art der Erkennung	ID der verwalteten Datenkennung	Schlüsselwort erforderlich	Länder und Regionen
Registrierungsnummer der Drug Enforcement Agency (DEA)	DrugEnforcementAgencyNumber	dea number, dea registration	US
Nummer der Krankenversicherungskarte (EHIC)	HealthInsuranceCardNumber	assicurazione sanitaria numero, carta assicurazione numero, carte d’assurance maladie, carte européenne d'assurance maladie, ceam, ehic, ehic#, finlandehicnumber#, gesundheitskarte, hälsokort, health card, health card number, health insurance card, health insurance number, insurance card number, krankenversicherungskarte, krankenversicherungsnummer, medical account number, numero conto medico, numéro d’assurance maladie, numéro de	EU

Art der Erkennung	ID der verwalteten Datenkennung	Schlüsselwort erforderlich	Länder und Regionen
		carte d’assurance, numéro de compte medical, número de cuenta médica, número de seguro de salud, número de tarjeta de seguro, sairaanhoitokortin, sairausvakuutuskortti, sairausvakuutusnumero, sjukförsäkring nummer, sjukförsäkringskort, suomi ehic-numero, tarjeta de salud, terveyskortti, tessera sanitaria assicurazione numero, versicherungsnummer	
Health Insurance Claim Number (HICN)	HealthInsuranceClaimNumber	health insurance claim number, hic no, hic no., hic number, hic#, hicn, hicn#., hicno#	US
Krankenversicherungs- oder medizinische Identifizierungsnummer	HealthInsuranceNumber	carte d'assuré social, carte vitale, insurance card	FR

Art der Erkennung	ID der verwalteten Datenkennung	Schlüsselwort erforderlich	Länder und Regionen
Standardisierte Codes für medizinische Leistungen (HCPCS)	HealthcareProcedureCode	current procedural terminology, hcpcs, healthcare common procedure coding system	US
Medicare-Versichertennummer (MBN)	MedicareBeneficiaryNumber	mbi, medicare beneficiary	US
National Drug Code (NDC)	NationalDrugCode	national drug code, ndc	US
National Provider Identifier (NPI)	NationalProviderId	hipaa, n.p.i, national provider, npi	US
Nummer des Nationalen Gesundheitsdienstes (NHS)	NhsNumber	national health service, NHS	GB
Persönliche Gesundheitsnummer (PHN)	PersonalHealthNumber	canada healthcare number, msp number, personal healthcare number, phn, soins de santé	CA

Schlüsselwörter für Krankenversicherungs- oder medizinische Identifizierungsnummern

Damit Amazon SNS verschiedene Arten von Krankenversicherungs- und medizinischen Identifizierungsnummern erkennen kann, muss sich ein Schlüsselwort in der Nähe der Nummern befinden. Dazu gehören die Nummern der Europäischen Krankenversicherungskarte (EU, Finnland), Krankenversicherungsnummern (Frankreich), Medicare-Versichertennummern (USA), nationale Versicherungsnummern (UK), NHS-Nummern (UK) und persönliche Gesundheitsnummern (Kanada).

In der folgenden Tabelle sind die Schlüsselwörter aufgeführt, die Amazon SNS für bestimmte Länder und Regionen erkennt.

Land oder Region	Schlüsselwörter
Kanada	Canada healthcare number, msp number, personal healthcare number, phn, soins de santé
EU	assicurazione sanitaria numero, carta assicurazione numero, carte d’assurance maladie, carte européenne d'assurance maladie, ceam, ehic, ehic#, finlandehicnumber#, gesundheitskarte, hälsokort, health card, health card number, health insurance card, health insurance number, insurance card number, krankenversicherungskarte, krankenversicherungsnummer, medical account number, numero conto medico, numéro d’assurance maladie, numéro de carte d’assurance, numéro de compte medical, número de cuenta médica, número de seguro de salud, número de tarjeta de seguro, sairaanhoitokortin, sairausvakuutuskortti, sairausvakuutusnumero, sjukförsäkring nummer, sjukförsäkringskort, suomi ehic-numero, tarjeta de salud, terveyskortti, tessera sanitaria assicurazione numero, versicherungsnummer
Finnland	ehic, ehic#, finland health insurance card, finlandehicnumber#, finska sjukförsäkringskort, hälsokort, health card, health card number, health insurance card, health insurance number, sairaanhoitokortin, sairaanhoitokortin, sairausvakuutuskortti, sairausvakuutusnumero, sjukförsäkring nummer, sjukförsäkringskort, suomen sairausvakuutuskortti, suomi ehic-numero, terveyskortti

Land oder Region	Schlüsselwörter
Frankreich	carte d'assuré social, carte vitale, insurance card
Vereinigtes Königreich	nationaler Gesundheitsdienst, NHS
US	mbi, medicare beneficiary

Datenkennungs-ARNs für geschützte Gesundheitsdatentypen (PHI)

Im Folgenden werden die Amazon-Ressourcennamen (ARNs) für die Datenkennungen aufgelistet, die Sie Ihren Datenschutzrichtlinien hinzufügen können.

ARNS für PHI-Datenkennungen
arn:aws:dataprotection::aws:data-identifier/DrugEnforcementAgencyNumber-US
arn:aws:dataprotection::aws:data-identifier/HealthcareProcedureCode-US
arn:aws:dataprotection::aws:data-identifier/HealthInsuranceCardNumber-EU
arn:aws:dataprotection::aws:data-identifier/HealthInsuranceClaimNumber-US
arn:aws:dataprotection::aws:data-identifier/HealthInsuranceNumber-FR
arn:aws:dataprotection::aws:data-identifier/MedicareBeneficiaryNumber-US
arn:aws:dataprotection::aws:data-identifier/NationalDrugCode-US
arn:aws:dataprotection::aws:data-identifier/NationalInsuranceNumber-GB
arn:aws:dataprotection::aws:data-identifier/NationalProviderId-US
arn:aws:dataprotection::aws:data-identifier/NhsNumber-GB
arn:aws:dataprotection::aws:data-identifier/PersonalHealthNumber-CA

Sensible Datentypen: Persönlich identifizierbare Informationen (PII)

In der folgenden Liste werden die Arten von persönlich identifizierbaren Informationen (PII) aufgeführt und beschrieben, die Amazon SNS mithilfe verwalteter Datenkennungen erkennen kann.

Art der Erkennung	ID der verwaltet en Datenkenn ung	Schlüsselwort erforderlich	Zusätzliche Informationen	Länder und Regionen
Geburtsdatum	DateOfBirth	dob, date of birth, birthdate , birth date, birthday, b-day, bday	Der Support umfasst die meisten Datumsformate, z. B. nur Ziffern und Kombinati onen aus Ziffern und Monatsnam en. Datumskom ponenten können durch Leerzeichen, Schrägstriche (/) oder Bindestri che (-) getrennt werden.	Any
Código de Endereçamento Postal (CEP)	CepCode	cep, código de endereçamento postal, codigo de endereçamento postal	–	Brasilien
Cadastro Nacional da Pessoa Jurídica (CNPJ)	Cnpj	cadastro nacional da pessoa jurídica, cadastro nacional da	–	Brasilien

Art der Erkennung	ID der verwaltet en Datenkenn ung	Schlüsselwort erforderlich	Zusätzliche Informationen	Länder und Regionen
		pessoa juridica, cnpj		
Cadastro de Pessoas Físicas (CPF)	CpfCode	Cadastro de pessoas fisicas, cadastro de pessoas físicas, cadastro de pessoa física, cadastro de pessoa fisica, cpf	–	Brasilien

Art der Erkennung	ID der verwalteten Datenkennung	Schlüsselwort erforderlich	Zusätzliche Informationen	Länder und Regionen
Identifikationsnummer des Führerscheins	DriversLicense	Ja, siehe Schlüsselwörter für Identifikationsnummern des Führerscheins.	–	Australien, Belgien, Bulgarien, Dänemark, Deutschland, Estland, Finnland, Frankreich, Griechenland, Irland, Italien, Kanada, Kroatien, Lettland, Litauen, Luxemburg, Malta, Niederlande, Österreich, Polen, Portugal, Rumänien, Schweden, Slowakei, Slowenien, Spanien, Tschechische Republik, Ungarn, USA, Vereinigtes Königreich, Zypern

Art der Erkennung	ID der verwaltet en Datenkenn ung	Schlüsselwort erforderlich	Zusätzliche Informationen	Länder und Regionen
Nummer der Wählerliste	Electoral RollNumber	electoral#, electoral #, electoralnumber, electoral number, electoralroll#, electoral roll#, electoral roll #, electoral roll no., electoral roll number, electoralrollno	–	UK
Individue lle Steuerpfl ichtigen-ID	Individua lTaxIdentification Number	Ja, siehe Schlüsselwörter für Steuerpfl ichtigenidentifika tions- und Referenzn ummern.	–	US
Nationales Institut für Statistik und Wirtschaf tsstudien (INSEE)	InseeCode	Ja, siehe Schlüsselwörter für nationale Identifik ationsnummern.	–	Frankreich

Art der Erkennung	ID der verwaltet en Datenkenn ung	Schlüsselwort erforderlich	Zusätzliche Informationen	Länder und Regionen
Nationale Identifik ationsnummern	NationalIdentifica tionNumber	Ja, siehe Schlüsselwörter für nationale Identifik ationsnummern.	Dazu gehören Kennungen des Documento Nacional de Identidad (DNI) (Spanien), Codice fiscale codes (Italien) und Personala usweisnummern (Deutsch).	Deutschland, Italien, Spanien
Landesver sicherung snummer (NINO)	NationalI nsuranceNumber	insurance no., insurance number, insurance #, national insurance number, nationali nsurance# , nationali nsurancen umber, nin, nino	–	UK
Número de identidad de extranjero (NIE)	NieNumber	Ja, siehe Schlüsselwörter für Steuerpfl ichtigenidentifika tions- und Referenzn ummern.	–	Spanien

Art der Erkennung	ID der verwaltet en Datenkenn ung	Schlüsselwort erforderlich	Zusätzliche Informationen	Länder und Regionen
Número de Identificación Fiscal (NIF)	NifNumber	Ja, siehe Schlüsselwörter für Steuerpfl ichtigenidentifika tions- und Referenzn ummern.	–	Spanien
Passnummer	PassportNumber	Ja, siehe Schlüsselwörter für Passnumme rn.	–	Kanada, Frankreich, Deutschland, Italien, Spanien, Vereinigtes Königreich, USA

Art der Erkennung	ID der verwaltet en Datenkenn ung	Schlüsselwort erforderlich	Zusätzliche Informationen	Länder und Regionen
Ständige Wohnsitzn ummer	Permanent Residence Number	carte résident permanent , numéro carte résident permanent, numéro résident permanent , permanent resident card, permanent resident card number, permanent resident no, permanent resident no., permanent resident number, pr no, pr no., pr non, pr number, résident permanent no., résident permanent non	–	Kanada

Art der Erkennung	ID der verwaltet en Datenkenn ung	Schlüsselwort erforderlich	Zusätzliche Informationen	Länder und Regionen
Phone number (Telefonnummer)	PhoneNumber	Brasilien: Zu den Schlüsselwörtern gehören auch: cel, celular, fone, móvel, número residencial, numero residenci al, telefone Andere: cell, contact, fax, fax number, mobile, phone, phone number, tel, telephone , telephone number	Dazu gehören gebührenfreie Nummern in den USA und Faxnummer n. Wenn sich ein Schlüssel wort in der Nähe der Daten befindet, muss die Nummer keine Landesvor wahl enthalten . Wenn sich ein Schlüssel wort nicht in der Nähe der Daten befindet, muss die Nummer eine Landesvorwahl enthalten.	Brasilien , Kanada, Frankreich, Deutschland, Italien, Spanien, Vereinigtes Königreich, USA
Postal Code (Postleitzahl)	PostalCode	No	–	Kanada
Registro Geral (RG)	RgNumber	Ja, siehe Schlüsselwörter für nationale Identifik ationsnummern.	–	Brasilien

Art der Erkennung	ID der verwaltet en Datenkenn ung	Schlüsselwort erforderlich	Zusätzliche Informationen	Länder und Regionen
Sozialver sicherung snummer (SIN)	SocialIns uranceNumber	canadian id, numéro d'assurance sociale, social insurance number, sin	–	Kanada
Sozialver sicherung snummer (SSN)	Ssn	Spanien – número de la seguridad social, social security no., social security no. número de la seguridad social, social security number, socialsec urityno#, ssn, ssn# USA: social security, ss#, ssn	–	Spanien, USA
Steuerpfl ichtigen-Identifik ationsnummer oder Referenzn ummer	TaxId	Ja, siehe Schlüsselwörter für Steuerpfl ichtigenidentifika tions- und Referenzn ummern.	Dazu gehören TIN (Frankreich), Steueridentifikati onsnummer (Deutschland), CIF (Spanien) und TRN, UTR (UK).	Frankreich, Deutschla nd, Spanien, Vereinigtes Königreich

Art der Erkennung	ID der verwaltet en Datenkenn ung	Schlüsselwort erforderlich	Zusätzliche Informationen	Länder und Regionen
US-Postleitzahl	ZipCode	zip code, zip+4	–	US
Postanschrift	Address	No	Obwohl ein Schlüsselwort nicht erforderl ich ist, muss die Adresse bei der Erkennung den Namen einer Stadt oder eines Ortes sowie einen Zip-Code oder eine Postleitzahl enthalten.	Australie n, Kanada, Frankreich, Deutschland, Italien, Spanien, Vereinigtes Königreich, USA
E-Mail-Adresse	EmailAddress	Email, Email-Adr esse, E-Mail, E-Mail-Adresse	–	Any

Art der Erkennung	ID der verwaltet en Datenkenn ung	Schlüsselwort erforderlich	Zusätzliche Informationen	Länder und Regionen
Koordinaten des globalen Positioni erungssystems (GPS)	LatLong	coordinate, coordinates, lat long, latitude longitude, location, position	Amazon SNS kann GPS-Koordinaten erkennen, wenn die Breiten- und Längengra dkoordinaten paarweise gespeichert sind und im DD-Format (Dezimalstelle, Grad) vorliegen, z. B. 41,948614 , -87,655311. Nicht unterstüt zt werden Koordinaten im DDM-Format (Grad, Dezimalst elle, Minuten), z. B. 41°56.916 8'N 87°39.318 7'W, oder im DMS-Format (Grad, Minuten, Sekunden), z. B. 41°56'55.0104"N 87°39'19. 1196"W.	Any

Art der Erkennung	ID der verwaltet en Datenkenn ung	Schlüsselwort erforderlich	Zusätzliche Informationen	Länder und Regionen
Vollständiger Name	Name	No	Amazon SNS kann nur vollständ ige Namen erkennen. Unterstützt werden nur lateinische Zeichensätze.	Any
Fahrgeste llnummern (VIN)	VehicleIdentificat ionNumber	Fahrgeste llnummer, niv, numarul de identificare, numarul seriei de sasiu, serie sasiu, numer VIN, Número de Identificação do Veículo, Número de Identificación de Automóvil es, numéro d'identification du véhicule, vehicle identification number, vin, VIN numeris	Amazon SNS kann VINs erkennen, die aus einer 17-stelligen Sequenz bestehen und den Standards ISO 3779 und 3780 entsprech en. Diese Standards wurden für den weltweiten Einsatz konzipier t.	Any

Schlüsselwörter für Identifikationsnummern des Führerscheins

Damit Amazon SNS verschiedene Arten von Führerschein-Identifikationsnummern erkennen kann, muss sich ein Schlüsselwort in der Nähe der Nummern befinden. In der folgenden Tabelle sind die Schlüsselwörter aufgeführt, die Amazon SNS für bestimmte Länder und Regionen erkennt.

Land oder Region	Schlüsselwörter
Australien	dl# dl:, dl :, dlno# driver licence, driver license, driver permit, drivers lic., drivers licence, driver's licence, drivers license, driver's license, drivers permit, driver's permit, drivers permit number, driving licence, driving license, driving permit
Österreich	führerschein, fuhrerschein, führerschein republik österreich, fuhrerschein republik osterreich
Belgien	fuehrerschein, fuehrerschein- nr, fuehrersc heinnummer, fuhrerschein, führerschein, fuhrerschein- nr, führerschein- nr, fuhrersch einnummer, führerscheinnummer, numéro permis conduire, permis de conduire, rijbewijs, rijbewijsnummer
Bulgarien	превозно средство, свидетелство за управление на моторно, свидетелство за управление на мпс, сумпс, шофьорска книжка
Kanada	dl#, dl:, dlno#, driver licence, driver licences, driver license, driver licenses, driver permit, drivers lic., drivers licence, driver's licence, drivers licences, driver's licences, drivers license, driver's license, drivers licenses, driver's licenses, drivers permit, driver's permit,

Land oder Region	Schlüsselwörter
	drivers permit number, driving licence, driving license, driving permit, permis de conduire
Kroatien	vozačka dozvola
Zypern	άδεια οδήγησης
Tschechische Republik	číslo licence, číslo licence řidiče, číslo řidičskéh o průkazu, ovladače lic., povolení k jízdě, povolení řidiče, řidiči povolení, řidičský prúkaz, řidičský průkaz
Dänemark	kørekort, kørekortnummer
Estland	juhi litsentsi number, juhiloa number, juhiluba, juhiluba number
Finnland	ajokortin numero, ajokortti, förare lic., körkort, körkort nummer, kuljettaja lic., permis de conduire
Frankreich	permis de conduire
Deutschland	fuehrerschein, fuehrerschein- nr, fuehrersc heinnummer, fuhrerschein, führerschein, fuhrerschein- nr, führerschein- nr, fuhrersch einnummer, führerscheinnummer
Griechenland	δεια οδήγησης, adeia odigisis
Ungarn	illesztőprogramok lic, jogosítvány, jogsi, licencszám, vezető engedély, vezetői engedély
Irland	ceadúnas tiomána
Italien	patente di guida, patente di guida numero, patente guida, patente guida numero

Land oder Region	Schlüsselwörter
Lettland	autovadītāja apliecība, licences numurs, vadītāja apliecība, vadītāja apliecības numurs, vadītāja atļauja, vadītāja licences numurs, vadītāji lic.
Litauen	vairuotojo pažymėjimas
Luxemburg	fahrerlaubnis, führerschäin
Malta	liċenzja tas-sewqan
Niederlande	permis de conduire, rijbewijs, rijbewijsnummer
Polen	numer licencyjny, prawo jazdy, zezwolenie na prowadzenie
Portugal	carta de condução, carteira de habilitação, carteira de motorist, carteira habilitação, carteira motorist, licença condução, licença de condução, número de licença, número licença, permissão condução, permissão de condução
Rumänien	numărul permisului de conducere, permis de conducere
Slowakei	číslo licencie, číslo vodičského preukazu, ovládače lic., povolenia vodičov, povolenie jazdu, povolenie na jazdu, povolenie vodiča, vodičský preukaz
Slowenien	vozniško dovoljenje

Land oder Region	Schlüsselwörter
Spanien	carnet conducer, el carnet de conducer, licencia conducer, licencia de manejo, número carnet conducer, número de carnet de conducer, número de permiso conducer, número de permiso de conducer, número licencia conducer, número permiso conducer, permiso conducción, permiso conducer, permiso de conducción
Schweden	ajokortin numero, dlno# ajokortti, drivere lic., förare lic., körkort, körkort nummer, körkortsn ummer, kuljettajat lic.
UK	dl#, dl:, dlno#, driver licence, driver licences, driver license, driver licenses, driver permit, drivers lic., drivers licence, driver's licence, drivers licences, driver's licences, drivers license, driver's license, drivers licenses, driver's licenses, drivers permit, driver's permit, drivers permit number, driving licence, driving license, driving permit
US	dl#, dl:, dlno#, driver licence, driver licences, driver license, driver licenses, driver permit, drivers lic., drivers licence, driver's licence, drivers licences, driver's licences, drivers license, driver's license, drivers licenses, driver's licenses, drivers permit, driver's permit, drivers permit number, driving licence, driving license, driving permit

Schlüsselwörter für nationale Identifikationsnummern

Damit Amazon SNS verschiedene Arten von nationalen Identifikationsnummern erkennen kann, muss sich ein Schlüsselwort in der Nähe der Nummern befinden. Dazu gehören Kennungen des

Documento Nacional de Identidad (DNI) (Spanien), Codes des französischen Nationalen Instituts für Statistik und Wirtschaftsstudien (INSEE), deutsche Personalausweisnummern und Registro Geral (RG)-Nummern (Brasilien).

In der folgenden Tabelle sind die Schlüsselwörter aufgeführt, die Amazon SNS für bestimmte Länder und Regionen erkennt.

Land oder Region	Schlüsselwörter
Brasilien	registro geral, rg
Frankreich	assurance sociale, carte nationale d’identit é, cni, code sécurité sociale, French social security number, fssn#, insee, insurance number, national id number, nationalid#, numéro d'assurance, sécurité sociale, sécurité sociale non., sécurité sociale numéro, social, social security, social security number, socialsecuritynumber, ss#, ssn, ssn#
Deutschland	ausweisnummer, id number, identification number, identity number, insurance number, personal id, personalausweis
Italien	codice fiscal, dati anagrafici, ehic, health card, health insurance card, p. iva, partita i.v.a., personal data, tax code, tessera sanitaria
Spanien	dni, dni#, dninúmero#, documento nacional de identidad, identidad único, identidadúnico#, insurance number, national identification number, national identity, nationalid#, nationali dno#, número nacional identidad, personal identification number, personal identity no, unique identity number, uniqueid#

Schlüsselwörter für Passnummern

Damit Amazon SNS verschiedene Arten von Passnummern erkennen kann, muss sich ein Schlüsselwort in der Nähe der Nummern befinden. In der folgenden Tabelle sind die Schlüsselwörter aufgeführt, die Amazon SNS für bestimmte Länder und Regionen erkennt.

Land oder Region	Schlüsselwörter
Kanada	passeport, passeport#, passport, passport#, passportno, passportno#
Frankreich	numéro de passeport, passeport, passeport #, passeport #, passeportn °, passeport n °, passeportNon, passeport non
Deutschland	ausstellungsdatum, ausstellungsort, geburtsdatum, passport, passports, reisepass, reisepass–nr, reisepassnummer
Italien	italian passport number, numéro passeport, numéro passeport italien, passaporto, passaporto italiana, passaporto numero, passport number, repubblica italiana passaporto
Spanien	españa pasaporte, libreta pasaporte, número pasaporte, pasaporte, passport, passport book, passport no, passport number, spain passport
UK	passeport #, passeport n °, passeportNon, passeport non, passeportn °, passport #, passport no, passport number, passport#, passportid
US	passport, travel document

Schlüsselwörter für Steuerpflichtigenidentifikations- und Referenznummern

Damit Amazon SNS verschiedene Arten von Steuerpflichtigenidentifikations- und Referenznummern erkennen kann, muss sich ein Schlüsselwort in der Nähe der Nummern befinden. In der folgenden Tabelle sind die Schlüsselwörter aufgeführt, die Amazon SNS für bestimmte Länder und Regionen erkennt.

Land oder Region	Schlüsselwörter
Brasilien	cadastro de pessoa física, cadastro de pessoa fisica, cadastro de pessoas físicas, cadastro de pessoas fisicas, cadastro nacional da pessoa jurídica, cadastro nacional da pessoa juridica, cnpj, cpf
Frankreich	numéro d'identification fiscale, tax id, tax identification number, tax number, tin, tin#
Deutschland	identifikationsnummer, steuer id, steueride ntifikationsnummer, steuernummer, tax id, tax identification number, tax number
Spanien	cif, cif número, cifnúmero#, nie, nif, número de contribuyente, número de identidad de extranjero, número de identificación fiscal, número de impuesto corporativo, personal tax number, tax id, tax identification number, tax number, tin, tin#
UK	paye, tax id, tax id no., tax id number, tax identification, tax identification#, tax no., tax number, tax reference, tax#, taxid#, temporary reference number, tin, trn, unique tax reference, unique taxpayer reference, utr
US	Individuelle Steuerzahler-Identifikationsnummer (ITIN)

Datenkennungs-ARNS für persönlich identifizierbare Informationen (PII)

In der folgenden Tabelle werden die Amazon-Ressourcennamen (ARNs) für die Datenkennungen aufgelistet, die Sie Ihren Datenschutzrichtlinien hinzufügen können.

ARNS für PII-Datenkennungen
arn:aws:dataprotection::aws:data-identifier/Address
arn:aws:dataprotection::aws:data-identifier/CepCode-BR
arn:aws:dataprotection::aws:data-identifier/Cnpj-BR
arn:aws:dataprotection::aws:data-identifier/CpfCode-BR
arn:aws:dataprotection::aws:data-identifier/DateOfBirth
arn:aws:dataprotection::aws:data-identifier/DriversLicense-AT
arn:aws:dataprotection::aws:data-identifier/DriversLicense-AU
arn:aws:dataprotection::aws:data-identifier/DriversLicense-BE
arn:aws:dataprotection::aws:data-identifier/DriversLicense-BG
arn:aws:dataprotection::aws:data-identifier/DriversLicense-CA
arn:aws:dataprotection::aws:data-identifier/DriversLicense-CY
arn:aws:dataprotection::aws:data-identifier/DriversLicense-CZ
arn:aws:dataprotection::aws:data-identifier/DriversLicense-DE
arn:aws:dataprotection::aws:data-identifier/DriversLicense-DK
arn:aws:dataprotection::aws:data-identifier/DriversLicense-EE
arn:aws:dataprotection::aws:data-identifier/DriversLicense-ES
arn:aws:dataprotection::aws:data-identifier/DriversLicense-FI
arn:aws:dataprotection::aws:data-identifier/DriversLicense-FR

ARNS für PII-Datenkennungen
arn:aws:dataprotection::aws:data-identifier/DriversLicense-GB
arn:aws:dataprotection::aws:data-identifier/DriversLicense-GR
arn:aws:dataprotection::aws:data-identifier/DriversLicense-HR
arn:aws:dataprotection::aws:data-identifier/DriversLicense-HU
arn:aws:dataprotection::aws:data-identifier/DriversLicense-IE
arn:aws:dataprotection::aws:data-identifier/DriversLicense-IT
arn:aws:dataprotection::aws:data-identifier/DriversLicense-LT
arn:aws:dataprotection::aws:data-identifier/DriversLicense-LU
arn:aws:dataprotection::aws:data-identifier/DriversLicense-LV
arn:aws:dataprotection::aws:data-identifier/DriversLicense-MT
arn:aws:dataprotection::aws:data-identifier/DriversLicense-NL
arn:aws:dataprotection::aws:data-identifier/DriversLicense-PL
arn:aws:dataprotection::aws:data-identifier/DriversLicense-PT
arn:aws:dataprotection::aws:data-identifier/DriversLicense-RO
arn:aws:dataprotection::aws:data-identifier/DriversLicense-SE
arn:aws:dataprotection::aws:data-identifier/DriversLicense-SI
arn:aws:dataprotection::aws:data-identifier/DriversLicense-SK
arn:aws:dataprotection::aws:data-identifier/DriversLicense-US
arn:aws:dataprotection::aws:data-identifier/ElectoralRollNumber-GB
arn:aws:dataprotection::aws:data-identifier/EmailAddress

ARNS für PII-Datenkennungen
arn:aws:dataprotection::aws:data-identifier/IndividualTaxIdentificationNumber-US
arn:aws:dataprotection::aws:data-identifier/InseeCode-FR
arn:aws:dataprotection::aws:data-identifier/LatLong
arn:aws:dataprotection::aws:data-identifier/Name
arn:aws:dataprotection::aws:data-identifier/NationalIdentificationNumber-DE
arn:aws:dataprotection::aws:data-identifier/NationalIdentificationNumber-ES
arn:aws:dataprotection::aws:data-identifier/NationalIdentificationNumber-IT
arn:aws:dataprotection::aws:data-identifier/NieNumber-ES
arn:aws:dataprotection::aws:data-identifier/NifNumber-ES
arn:aws:dataprotection::aws:data-identifier/PassportNumber-CA
arn:aws:dataprotection::aws:data-identifier/PassportNumber-DE
arn:aws:dataprotection::aws:data-identifier/PassportNumber-ES
arn:aws:dataprotection::aws:data-identifier/PassportNumber-FR
arn:aws:dataprotection::aws:data-identifier/PassportNumber-GB
arn:aws:dataprotection::aws:data-identifier/PassportNumber-IT
arn:aws:dataprotection::aws:data-identifier/PassportNumber-US
arn:aws:dataprotection::aws:data-identifier/PermanentResidenceNumber-CA
arn:aws:dataprotection::aws:data-identifier/PhoneNumber-BR
arn:aws:dataprotection::aws:data-identifier/PhoneNumber-DE
arn:aws:dataprotection::aws:data-identifier/PhoneNumber-ES

ARNS für PII-Datenkennungen
arn:aws:dataprotection::aws:data-identifier/PhoneNumber-FR
arn:aws:dataprotection::aws:data-identifier/PhoneNumber-GB
arn:aws:dataprotection::aws:data-identifier/PhoneNumber-IT
arn:aws:dataprotection::aws:data-identifier/PhoneNumber-US
arn:aws:dataprotection::aws:data-identifier/PostalCode-CA
arn:aws:dataprotection::aws:data-identifier/RgNumber-BR
arn:aws:dataprotection::aws:data-identifier/SocialInsuranceNumber-CA
arn:aws:dataprotection::aws:data-identifier/Ssn-ES
arn:aws:dataprotection::aws:data-identifier/Ssn-US
arn:aws:dataprotection::aws:data-identifier/TaxId-DE
arn:aws:dataprotection::aws:data-identifier/TaxId-ES
arn:aws:dataprotection::aws:data-identifier/TaxId-FR
arn:aws:dataprotection::aws:data-identifier/TaxId-GB
arn:aws:dataprotection::aws:data-identifier/VehicleIdentificationNumber
arn:aws:dataprotection::aws:data-identifier/ZipCode-US

Verwenden von benutzerdefinierten Datenkennungen in Amazon SNS

Themen

Was sind benutzerdefinierte SNS-Datenkennungen?

Mit benutzerdefinierten Datenkennungen (Custom data identifiers; CDIs) können Sie Ihre eigenen benutzerdefinierten regulären Ausdrücke definieren, die in Ihrer Datenschutzrichtlinie verwendet werden können. Mithilfe von benutzerdefinierten Datenkennungen können Sie gezielt auf geschäftsspezifische Anwendungsfälle mit persönlich identifizierbaren Informationen (PII) abzielen, die verwaltete Datenkennungen nicht bieten können. Sie können beispielsweise eine benutzerdefinierte Datenkennung verwenden, um nach unternehmensspezifischen Mitarbeiter-IDs zu suchen. Benutzerdefinierte Datenkennungen können in Verbindung mit verwalteten Datenkennungen verwendet werden.

Verwenden benutzerdefinierter Datenkennungen in Ihrer Datenschutzrichtlinie

Die folgende Datenschutzrichtlinie weist das Amazon SNS-Thema an, Nutzlasten zu erkennen, die unternehmensspezifische Mitarbeiter-IDs enthalten, und diese IDs dann mit dem Hash-Symbol (#) zu maskieren.

1. Erstellen Sie einen `Configuration`-Block innerhalb Ihrer Datenschutzrichtlinie.
2. Geben Sie eine `Name` als benutzerdefinierte Datenkennung ein. Beispiel: **`EmployeeId`**
3. Geben Sie eine `Regex` als benutzerdefinierte Datenkennung ein. Beispiel: **`EID-\d{9}-US`**
4. Beziehen Sie sich in einer Richtlinienerklärung auf die folgende benutzerdefinierte Datenkennung.

```
{
 "Name": "__example_data_protection_policy",
 "Description": "Example data protection policy",
 "Version": "2021-06-01",
 "Configuration": {
   "CustomDataIdentifier": [
     {"Name": "EmployeeId", "Regex": "EID-\d{9}-US"}
   ]
 },
 "Statement": [
   {
     "DataDirection": "Inbound",
     "Principal": ["*"],
     "DataIdentifier": [
        "EmployeeId"
     ],
     "Operation": {
```

```
        "Deidentify": {
          "MaskConfig": {
            "MaskWithCharacter": "#"
          }
        }
      }
    }
  ]
}
```

5. (Optional) Fügen Sie dem `Configuration`-Block bei Bedarf weitere benutzerdefinierte Datenkennungen hinzu. Datenschutzrichtlinien unterstützen derzeit maximal 10 benutzerdefinierte Datenkennungen.

Einschränkungen für benutzerdefinierte Datenkennungen

Benutzerdefinierte Datenkennungen von Amazon SNS haben die folgenden Einschränkungen:

- Datenschutzrichtlinien unterstützen derzeit maximal 10 benutzerdefinierte Datenkennungen.
- Die Namen von benutzerdefinierten Datenkennungen sind auf maximal 128 Zeichen beschränkt. Folgende Zeichen werden unterstützt:
  - Alphanumerisch: (a-zA-Z0-9)
  - Sonderzeichen: („_“ | „-“)
- RegEx ist auf maximal 200 Zeichen beschränkt. Folgende Zeichen werden unterstützt:
  - Alphanumerisch: (a-zA-Z0-9)
  - Sonderzeichen: („_“ | „#“ | „=“ | „@“ | „/“ | „;“ | „,“ | „-“ | „ “)
  - RegEx reservierte Zeichen: („^“ | „$“ | „?“ | „[“ | „]“ | „{“ | „}“ | „|“ | „\\“ | „*“ | „+“ | „.“)
- Benutzerdefinierte Datenkennungen können nicht denselben Namen wie verwaltete Datenkennungen haben.
- Benutzerdefinierte Datenkennungen müssen in jeder Datenschutzrichtlinie für jedes Amazon SNS-Thema angegeben werden.

Nachrichtenzustellung von Amazon SNS

In diesem Abschnitt wird beschrieben, wie die Nachrichtenzustellung funktioniert.

Themen

Übermittlung unformatierter Nachrichten Amazon SNS

Um zu verhindern, dass Amazon Data Firehose -, Amazon SQS - und HTTP/S-Endpunkte die JSON-Formatierung von Nachrichten verarbeiten, ermöglicht Amazon SNS die Zustellung von Rohnachrichten:

- Wenn Sie die Zustellung von Rohnachrichten für Amazon Data Firehose- oder Amazon SQS SQS-Endpunkte aktivieren, werden alle Amazon SNS SNS-Metadaten aus der veröffentlichten Nachricht entfernt und die Nachricht wird unverändert gesendet.
- Wenn Sie die Übermittlung unformatierter Nachrichten für HTTP/S-Endpunkte aktivieren, wird der HTTP-Header `x-amz-sns-rawdelivery` mit seinem auf `true` gesetzten Wert zur Nachricht hinzugefügt. Damit wird angezeigt, dass die Nachricht ohne JSON-Formatierung veröffentlicht wurde.
- Wenn Sie die Übermittlung unformatiertem Nachrichten für HTTP/S-Endpunkte aktivieren, werden der Nachrichtentext, die Client-IP und die erforderlichen Header zugestellt. Wenn Sie Nachrichtenattribute angeben, werden diese nicht gesendet.
- Wenn Sie die Übermittlung von Rohnachrichten für Firehose-Endpunkte aktivieren, wird der Nachrichtentext zugestellt. Wenn Sie Nachrichtenattribute angeben, werden diese nicht gesendet.

Um die Zustellung von Rohnachrichten mithilfe eines AWS SDK zu aktivieren, müssen Sie die `SetSubscriptionAttribute` API-Aktion verwenden und den Wert des `RawMessageDelivery` Attributs auf setzen. `true`

Aktivieren der Übermittlung unformatierter Nachrichten mit der AWS Management Console

1. Melden Sie sich bei der Amazon-SNS-Konsole an.
2. Wählen Sie im Navigationsbereich Topics (Themen) aus.
3. Wählen Sie auf der Seite Themen ein Thema aus, das einen Firehose-, Amazon SQS- oder HTTP/S-Endpunkt abonniert hat.
4. Wählen Sie auf der ***MyTopic***Seite im Abschnitt Abonnement ein Abonnement aus und klicken Sie auf Bearbeiten.
5. Wählen Sie auf der Seite Bearbeiten ***EXAMPLE1-23bc-4567-d890-ef12g3hij456*** im Abschnitt Details Übermittlung von unformatierten Nachrichten aktivieren.
6. Wählen Sie Save Changes (Änderungen speichern).

Nachrichtenformat – Beispiele

In den folgenden Beispielen wird dieselbe Nachricht zweimal an dieselbe Amazon SQS Warteschlange gesendet. Der einzige Unterschied besteht darin, dass die unformatierte Nachrichtenübermittlung für die erste Nachricht deaktiviert und für die zweite aktiviert ist.

- Übermittlung von unformatierten Nachrichten ist deaktiviert


```
{
  "Type": "Notification",
  "MessageId": "dc1e94d9-56c5-5e96-808d-cc7f68faa162",
  "TopicArn": "arn:aws:sns:us-east-2:111122223333:ExampleTopic1",
  "Subject": "TestSubject",
  "Message": "This is a test message.",
  "Timestamp": "2021-02-16T21:41:19.978Z",
  "SignatureVersion": "1",
  "Signature":
 "FMG5tlZhJNHLHUXvZgtZzlk24FzVa7oX0T4P03neeXw8ZEXZx6z35j2FOTuNYShn2h0bKNC/
zLTnMyIxEzmi2X1shOBWsJHkrW2xkR58ABZF+4uWHEE73yDVR4SyYAikP9jstZzDRm
+bcVs8+T0yaLiEGLrIIIL4esi1llhIkgErCuy5btPcWXBdio2fpCRD5x9oR6gmE/
rd5O7lX1c1uvnv4r1Lkk4pqP2/iUfxFZva1xLSRvgyfm6D9hNklVyPfy
```

```
+7TalMD0lzmJuOrExtnSIbZew3foxgx8GT+lbZkLd0ZdtdRJlIyPRP44eyq78sU0Eo/
LsDr0Iak4ZDpg8dXg==",
  "SigningCertURL": "https://sns.us-east-2.amazonaws.com/
SimpleNotificationService-010a507c1833636cd94bdb98bd93083a.pem",
  "UnsubscribeURL": "https://sns.us-east-2.amazonaws.com/?
Action=Unsubscribe&SubscriptionArn=arn:aws:sns:us-
east-2:111122223333:ExampleTopic1:e1039402-24e7-40a3-a0d4-797da162b297"
}
```

- Übermittlung von unformatierten Nachrichten ist aktiviert

```
This is a test message.
```

Nachrichtenattribute und Zustellung von Rohnachrichten für Amazon SQS SQS-Abonnements

Amazon SNS unterstützt die Übermittlung von Nachrichtenattributen, mit denen Sie strukturierte Metadatenelemente wie Zeitstempel, Geodaten, Signaturen und Kennungen zur Nachricht bereitstellen können. Bei Amazon SQS SQS-Abonnements mit aktivierter Raw Message Delivery können maximal 10 Nachrichtenattribute gesendet werden. Um mehr als 10 Nachrichtenattribute zu senden, müssen Sie Raw Message Delivery deaktivieren. Amazon SNS verwirft jedoch Nachrichten mit mehr als 10 Nachrichtenattributen, die an Amazon SQS SQS-Abonnements mit aktivierter Raw Message Delivery gerichtet sind, und behandelt sie als clientseitige Fehler.

Senden von Amazon SNS-Nachrichten an eine Amazon SQS-Warteschlange in einem anderen Konto

In diesem Dokument wird beschrieben, wie Sie eine Benachrichtigung zu einem Amazon SNS-Thema mit einem oder mehreren Abonnements für Amazon SQS-Warteschlangen in einem anderen Konto veröffentlichen. Sie richten das Thema und die Warteschlangen so ein, als wenn sie sich im selben Konto befinden würden (siehe Verteilung an Amazon-SQS-Warteschlangen). Der größte Unterschied besteht in der Handhabung der Abonnementbestätigung, die davon abhängt, wie Sie die Warteschlange für das Thema abonnieren.

Es empfiehlt sich, möglichst die Schritte zu befolgen, auf die im Abschnitt Abonnement erstellt durch Warteschlangeneigentümer verwiesen wird, da die Bestätigung automatisch erfolgt, wenn der Eigentümer der Warteschlange das Abonnement erstellt.

> **Note**
>
> Wenn die Amazon-SQS-Warteschlange ein hohes Nachrichtenvolumen enthält, empfehlen wir, dass der Warteschlangenbesitzer das Abonnement erstellt.

Themen

Abonnement erstellt durch Warteschlangeneigentümer

Das Konto, das die Amazon SQS-Warteschlange erstellt hat, ist der Besitzer der Warteschlange. Wenn der Eigentümer der Warteschlange ein Abonnement erstellt, muss das Abonnement nicht bestätigt werden. Die Warteschlange beginnt mit dem Empfang von Benachrichtigungen aus dem Thema, sobald die Aktion `Subscribe` abgeschlossen ist. Damit der Warteschlangeneigentümer das Thema des Themeneigentümers abonnieren kann, muss der Themeneigentümer die Kontoberechtigung des Warteschlangeneigentümers zum Aufrufen der Aktion `Subscribe` für das Thema erteilen.

Schritt 1: So legen Sie die Themen-Richtlinie mithilfe der AWS Management Console fest

1. Melden Sie sich bei der Amazon-SNS-Konsole an.
2. Wählen Sie im Navigationsbereich Topics (Themen) aus.
3. Wählen Sie ein Thema und dann Edit (Bearbeiten) aus.
4. Erweitern Sie auf der Seite Edit ***MyTopic*** (MyTopic bearbeiten) den Abschnitt Access policy (Access-Richtlinie) .
5. Geben Sie die folgende Richtlinie ein:

```
{
   "Statement": [
      {
         "Effect": "Allow",
```

```
         "Principal": {
            "AWS": "111122223333"
         },
         "Action": "sns:Subscribe",
         "Resource": "arn:aws:sns:us-east-2:123456789012:MyTopic"
      }
   ]
}
```

Diese Richtlinie gewährt Konto `111122223333` die Berechtigung zum Anruf von `sns:Subscribe` über `MyTopic` in Konto `123456789012`.

Ein Benutzer mit den Anmeldeinformationen für das Konto `111122223333` kann `MyTopic` abonnieren. Diese Berechtigung ermöglicht es der Konto-ID, die Berechtigung an ihren IAM-Benutzer/ihre IAM-Rolle zu delegieren. Nur das Root-Konto oder die Administratorbenutzer dürfen `sns:Subscribe` aufrufen. Der IAM-Benutzer/die IAM-Rolle muss ebenfalls `sns:subscribe` aufrufen, damit seine/ihre Warteschlange abonniert werden kann.

6. Wählen Sie Änderungen speichern.

   Ein Benutzer mit den Anmeldeinformationen für das Konto `111122223333` kann MyTopic abonnieren.

Schritt 2: So fügen Sie einer Amazon SQS-Warteschlange zu einem Thema in einem anderen AWS-Konto mithilfe der AWS Management Console ein Abonnement hinzu

Bevor Sie beginnen, stellen Sie sicher, dass Sie über die ARNs für Ihr Thema und Ihre Warteschlange verfügen und dass Sie dem Thema die Berechtigung erteilt haben, Nachrichten an die Warteschlange zu senden.

1. Melden Sie sich bei der Amazon-SQS-Konsole an.
2. Wählen Sie im Navigationsbereich Queues (Warteschlangen) aus.
3. Wählen Sie in der Liste der Warteschlangen die Warteschlange aus, für die das Amazon-SNS-Thema abonniert werden soll.
4. Wählen Sie Subscribe to Amazon SNS topic (Amazon-SNS-Thema abonnieren).
5. Wählen Sie im Menü Specify an Amazon SNS topic available for this queue (Ein für dieses Warteschlangenmenü verfügbares Amazon-SNS-Thema angeben) das SNS topic (SNS-Thema) für Ihre Warteschlange aus.

6. Wählen Sie Enter Amazon SNS topic ARN (Den ARN des Amazon-SNS-Themas eingeben) aus und geben Sie dann den Amazon Ressouren Name (ARN, Amazon-Ressourcenname) ein.
7. Wählen Sie Speichern.

> **Note**
>
> - Zur Kommunikation mit dem Service muss die Warteschlange über Berechtigungen für Amazon SNS verfügen.
> - Da Sie der Besitzer der Warteschlange sind, müssen Sie das Abonnement nicht bestätigen.

Ein Benutzer, der nicht der Warteschlangeneigentümer ist, erstellt ein Abonnement

Jeder Benutzer, der ein Abonnement erstellt, aber nicht der Besitzer der Warteschlange ist, muss das Abonnement bestätigen.

Wenn Sie die `Subscribe`-Aktion verwenden, sendet Amazon SNS eine Abonnementbestätigung an die Warteschlange. Das Abonnement wird in der Amazon-SNS-Konsole angezeigt, wobei die Abonnement-ID auf Pending Confirmation (Ausstehende Bestätigung) festgelegt ist.

Zur Bestätigung des Abonnements muss ein Benutzer mit der Berechtigung zum Lesen von Nachrichten aus der Warteschlange die URL zur Bestätigung des Abonnements abrufen und der Eigentümer des Abonnements muss das Abonnement unter Verwendung dieser URL bestätigen. Solange das Abonnement nicht bestätigt ist, werden keine Benachrichtigungen, die zum Thema veröffentlicht wurden, an die Warteschlange gesendet. Sie können das Abonnement über die Amazon-SQS-Konsole oder mit der `ReceiveMessage`-Aktion bestätigen.

> **Note**
>
> Bevor Sie einen Endpunkt für das Thema abonnieren, stellen Sie sicher, dass die Warteschlange Nachrichten aus dem Thema empfangen kann. legen Sie dazu die `sqs:SendMessage`-Berechtigung für die Warteschlange fest. Weitere Informationen finden Sie unter Schritt 2: Erteilen der Berechtigung für das Amazon SNS-Thema zum Senden von Nachrichten an die Amazon-SQS-Warteschlange.

Schritt 1: So fügen Sie einer Amazon-SQS-Warteschlange ein Abonnement zu einem Thema in einem anderen AWS-Konto mithilfe der AWS Management Console hinzu

Bevor Sie beginnen, stellen Sie sicher, dass Sie über die ARNs für Ihr Thema und Ihre Warteschlange verfügen und dass Sie dem Thema die Berechtigung erteilt haben, Nachrichten an die Warteschlange zu senden.

1. Melden Sie sich bei der Amazon SNS-Konsole an.
2. Wählen Sie im Navigationsbereich Subscriptions (Abonnements) aus.
3. Wählen Sie auf der Seite Subscriptions (Abonnements) die Option Create subscription (Abonnement erstellen)aus.
4. Führen Sie auf der Seite Create subscription (Abonnement erstellen) im Abschnitt Details die folgenden Schritte aus:

   a. Geben Sie unter Topic ARN (Themen-ARN) den ARN des Themas ein.
   b. Für Protokoll wählen Sie Amazon SQS.
   c. Geben Sie als Endpoint (Endpunkt) den ARN der Warteschlange ein.
   d. Klicken Sie auf Create subscription (Abonnement erstellen).

> **Note**
>
> - Zur Kommunikation mit dem Service muss die Warteschlange über Berechtigungen für Amazon SNS verfügen.

Nachfolgend finden Sie eine Beispielrichtlinienanweisung, die es dem Amazon-SNS-Thema erlaubt, eine Nachricht an die Amazon-SQS-Warteschlange zu senden.

```
{
   "Sid": "Stmt1234",
   "Effect": "Allow",
   "Principal": "*",
   "Action": "sqs:SendMessage",
   "Resource": "arn:aws:sqs:us-west-2:111111111111:QueueName",
   "Condition": {
      "ArnEquals": {
         "aws:SourceArn": "arn:aws:sns:us-west-2:555555555555:TopicName"
      }
```

```
    }
}
```

Schritt 2: So bestätigen Sie ein Abonnement mithilfe der AWS Management Console

1. Melden Sie sich bei der Amazon-SQS-Konsole an.
2. Wählen Sie die Warteschlange aus, für die ein Abonnement für das Thema aussteht.
3. Wählen Sie Send and receive messages (Nachrichten senden und empfangen) und dann Poll for messages (Nachrichten abfragen) aus.

   Eine Nachricht mit der Abonnementbestätigung wird in der Warteschlange empfangen.
4. Führen Sie in der Spalte Body (Textkörper) die folgenden Schritte aus:

   a. Wählen Sie More Details (Weitere Details) aus.

   b. Suchen Sie im Dialogfeld Message Details (Nachrichtendetails) den Wert für SubscribeURL. Dies ist Ihr Abonnement-Link (Beispiel unten). Weitere Informationen zur Validierung von API-Token finden Sie unter `ConfirmSubscription` in der Amazon-SNS-API-Referenz.

   ```
   https://sns.us-west-2.amazonaws.com/?
   Action=ConfirmSubscription&TopicArn=arn:aws:sns:us-
   east-2:123456789012:MyTopic&Token=2336412f37fb...
   ```

   c. Notieren Sie sich den Link zur Bestätigung des Abonnements. Die URL muss vom Eigentümer der Warteschlange an den Eigentümer des Abonnements weitergegeben werden. Der Eigentümer des Abonnements muss die URL in die Amazon-SNS-Konsole eingeben.
5. Melden Sie sich als Eigentümer des Abonnements bei der Amazon-SNS-Konsole an. Die Bestätigung erfolgt durch den Eigentümer des Abonnements.
6. Wählen Sie das relevante Thema aus.
7. Wählen Sie das relevante Abonnement in der Tabelle für Abonnementangebote des Themas aus. Es ist als „Pending confirmation“ (Bestätigung ausstehend) gekennzeichnet.
8. Wählen Sie Confirm subscription (Abonnement bestätigen) aus.
9. Ein Modal wird angezeigt, das den Link zur Bestätigung des Abonnements anfordert. Fügen Sie den Link zur Bestätigung des Abonnements ein.
10. Wählen Sie im Modal Confirm subscription (Abonnement bestätigen) aus.

    Eine XML-Antwort wird angezeigt, zum Beispiel:

```
<ConfirmSubscriptionResponse>
   <ConfirmSubscriptionResult>
      <SubscriptionArn>arn:aws:sns:us-east-2:123456789012:MyTopic:1234a567-
bc89-012d-3e45-6fg7h890123i</SubscriptionArn>
   </ConfirmSubscriptionResult>
   <ResponseMetadata>
      <RequestId>abcd1efg-23hi-jkl4-m5no-p67q8rstuvw9</RequestId>
   </ResponseMetadata>
</ConfirmSubscriptionResponse>
```

Die abonnierte Warteschlange kann nun Nachrichten vom Thema empfangen.

11. (Optional) Wenn Sie das Themenabonnement in der Amazon-SNS-Konsole anzeigen, können Sie sehen, dass die Meldung Pending Confirmation (Ausstehende Bestätigung) in der Spalte Subscription ID (Abonnement-ID) durch den Abonnement-ARN ersetzt wurde.

Wie erzwinge ich, dass ein Abonnement eine Authentifizierung bei Abmeldeanfragen erfordert?

Der Eigentümer des Abonnements muss das `AuthenticateOnUnsubscribe`-Flag bei der Bestätigung des Abonnements auf „true“ (wahr) festlegen.

- `AuthenticateOnUnsubscribe` wird automatisch auf „true“ (wahr) festgelegt, wenn der Eigentümer der Warteschlange das Abonnement erstellt.
- `AuthenticateOnUnsubscribe` kann nicht auf „true“ (wahr) festgelegt werden, wenn der Link zur Abonnementbestätigung ohne Authentifizierung aufgerufen wird.

Senden von Amazon SNS-Nachrichten an eine Amazon SQS-Warteschlange oder AWS Lambda -Funktion in einer anderen Region

Amazon SNS unterstützt regionsübergreifende Lieferungen sowohl für Regionen, die standardmäßig aktiviert sind, als auch für Opt-In-Regionen. Für die aktuelle Liste von AWS -Regionen, die Amazon SNS unterstützt, einschließlich Opt-in-Regionen, finden Sie unter Endpunkte und Kontingente für Amazon-Simple-Notification-Service-Endpunkte in der Allgemeine Amazon Web Services-Referenz.

Amazon SNS unterstützt die regionsübergreifende Übermittlung von Benachrichtigungen an Amazon SQS Queues und an AWS Lambda -Funktionen. Wenn es sich bei einer der Regionen um eine Opt-in-Region handelt, müssen Sie in der Richtlinie der abonnierten Ressource einen anderen Amazon SNS Dienstprinzipal angeben.

Der Amazon-SNS-Abonnementbefehl muss im Zielkonto in der Region ausgeführt werden, in der Amazon SNS gehostet wird. Wenn sich Amazon SNS beispielsweise im Konto „A“ in der Region us-east-1 und die Lambda-Funktion im Konto „B“ in der Region us-east-2 befindet, muss der Abonnement-CLI-Befehl in Konto „A“ in der Region us-east-1 ausgeführt werden.

Opt-in-Regionen

Amazon SNS unterstützt die folgenden Opt-in-Regionen:

Name der Region	Region
Region Afrika (Kapstadt)	af-south-1
Region Asien-Pazifik (Hongkong)	ap-east-1
Region Asien-Pazifik (Hyderabad)	ap-south-2
Region Asien-Pazifik (Jakarta)	ap-southeast-3
Region Asien-Pazifik (Melbourne)	ap-southeast-4
Region Asien-Pazifik (Osaka)	ap-northeast-3
Region Europa (Mailand)	eu-south-1
Region Europa (Spanien)	eu-south-2
Region Europa (Zürich)	eu-central-2
Region Israel (Tel Aviv)	il-central-1
Region Naher Osten (Bahrain)	me-south-1
Region Naher Osten (UAE)	me-central-1

Informationen zum Aktivieren einer Opt-In-Region finden Sie unter Verwalten von - AWS Regionen im Allgemeine Amazon Web Services-Referenz.

Wenn Sie Amazon SNS zum Übermitteln von Nachrichten aus Opt-in-Regionen an standardmäßig aktivierte Regionen verwenden, müssen Sie die für die Queue erstellte Ressourcenrichtlinie ändern. Ersetzen Sie den Prinzipal `sns.amazonaws.com` durch `sns.<opt-in-region>.amazonaws.com`. Beispiel:

- Wenn Sie beispielsweise eine Amazon-SQS-Warteschlange in USA Ost (Nord-Virginia) für ein Amazon-SNS-Thema in Asien-Pazifik (Hongkong) abonnieren möchten, ändern Sie den Prinzipal in der Warteschlangenrichtlinie in `sns.ap-east-1.amazonaws.com`. Zu den Opt-In-Regionen gehören alle Regionen, die nach dem 20. März 2019 eingeführt wurden, darunter Asien-Pazifik (Hongkong), Asien-Pazifik (Jakarta), Naher Osten (Bahrain), Europa (Mailand) und Afrika (Kapstadt). Regionen, die vor dem 20. März 2019 gestartet wurden, sind standardmäßig aktiviert.

  Regionsübergreifende Zustellungsunterstützung für Amazon SQS

Regionsübergreifende Zustellungsart	Unterstützt/Nicht unterstützt
Standardmäßig aktivierte Region zu Opt-In-Region	Unterstützt mit `sns.<opt-in-region>.amazonaws.com` im Service-Prinzipal für die Warteschlange
Opt-In-Region zu standardmäßig aktivierter Region	Unterstützt mit `sns.<opt-in-region>.amazonaws.com` im Service-Prinzipal für die Warteschlange
Opt-In Region zu Opt-In-Region	Nicht unterstützt

  Im Folgenden finden Sie ein Beispiel für eine Zugriffsrichtlinienanweisung, die es einem Amazon SNS-Thema in einer Opt-in-Region (af-south-1) ermöglicht, an eine Amazon SQS-Warteschlange in einer -enabled-by-default Region (us-east-1) zu liefern. Es enthält die notwendige regionalisierte Service-Prinzipal-Konfiguration unter dem Pfad `Statement/Principal/Service`.

```
{
```

```
  "Version": "2008-10-17",
  "Id": "__default_policy_ID",
  "Statement": [
    {
      "Sid": "allow_sns_arn:aws:sns:af-south-1:111111111111:source_topic_name",
      "Effect": "Allow",
      "Principal": {
        "Service": "sns.af-south-1.amazonaws.com"
      },
      "Action": "SQS:SendMessage",
      "Resource": "arn:aws:sqs:us-east-1:111111111111:destination_queue_name",
      "Condition": {
        "ArnLike": {
          "aws:SourceArn": "arn:aws:sns:af-south-1:111111111111:source_topic_name"
        }
      }
    },
    ...
  ]
}
```

- Um eine - AWS Lambda Funktion in USA Ost (Nord-Virginia) für ein Amazon SNS-Thema in Asien-Pazifik (Hongkong) zu abonnieren, ändern Sie den Prinzipal in der AWS Lambda Funktionsrichtlinie in `sns.ap-east-1.amazonaws.com`. Zu den Opt-In-Regionen gehören alle Regionen, die nach dem 20. März 2019 eingeführt wurden, darunter Asien-Pazifik (Hongkong), Asien-Pazifik (Jakarta), Naher Osten (Bahrain), Europa (Mailand) und Afrika (Kapstadt). Regionen, die vor dem 20. März 2019 gestartet wurden, sind standardmäßig aktiviert.

  Unterstützung der regionsübergreifenden Bereitstellung an AWS Lambda

Regionsübergreifende Zustellungsart	Unterstützt/Nicht unterstützt	
Standardmäßig aktivierte Region zu Opt-In-Region	Nicht unterstützt	
Opt-In-Region zu standardmäßig aktivierter Region	Unterstützt mithilfe von `sns.<opt-in-region>.amazonaws.com` im Service-Principal für die Lambda-Funktion	

Regionsübergreifende Zustellungsart	Unterstützt/Nicht unterstützt	
Opt-In Region zu Opt-In-Region	Nicht unterstützt	

Zustellungsstatus von Amazon SNS Nachrichtenübermittmittungen

Amazon SNS unterstützt das Protokollieren des Zustellungsstatus von Benachrichtigungen, die an Themen gesendet wurden, mithilfe der folgenden Amazon SNS-Endpunkte:

- HTTP
- Amazon Data Firehose
- AWS Lambda
- Plattformanwendungsendpunkt
- Amazon Simple Queue Service

Nachdem Sie die Attribute für den Nachrichtenzustellungsstatus konfiguriert haben, werden Protokolleinträge für Nachrichten, die an Themenabonnenten gesendet wurden, an CloudWatch Logs gesendet. Die Protokollierung des Zustellungsstatus von Nachrichten verhilft Ihnen zu besseren betrieblichen Einblicken, wie beispielsweise den Folgenden:

- Kenntnis, ob eine Mitteilung an den Amazon SNS-Endpunkt gesendet wurde.
- Ermittlung der Antwort, die vom Amazon SNS-Endpunkt an Amazon SNS gesendet wurde.
- Ermittlung der Leerlaufzeit für die Nachricht (das Zeitfenster zwischen dem Zeitstempel der Veröffentlichung bis direkt vor der Übergabe an einen Amazon SNS-Endpunkt).

Um Themenattribute für den Nachrichtenzustellungsstatus zu konfigurieren, können Sie die AWS Management Console AWS Software Development Kits (SDKs), die Abfrage-API oder AWS CloudFormation verwenden.

Themen

- Konfigurieren der Protokollierung des Zustellungsstatus mithilfe von AWS Management Console
- Konfiguration der Protokollierung des Lieferstatus mithilfe der AWS SDKs

- AWS SDK-Beispiele zur Konfiguration von Themenattributen
- Konfigurieren der Protokollierung des Zustellungsstatus mithilfe von AWS CloudFormation

Konfigurieren der Protokollierung des Zustellungsstatus mithilfe von AWS Management Console

1. Melden Sie sich bei der Amazon SNS-Konsole an.
2. Wählen Sie im Navigationsbereich Topics (Themen) aus.
3. Wählen Sie auf der Seite Topics (Themen) ein Thema und anschließend Edit (Bearbeiten) aus.
4. Erweitern Sie auf der *MyTopic*Seite Bearbeiten den Abschnitt Protokollierung des Zustellungsstatus.
5. Wählen Sie die Protokolle aus, für die Sie den Zustellungsstatus protokollieren möchten, z. B. AWS Lambda.
6. Geben Sie die Erfolgsrate ein (den Prozentsatz der erfolgreichen Nachrichten, für die Sie CloudWatch Protokolle erhalten möchten).
7. Führen Sie im Unterabschnitt IAM roles (IAM-Rollen) einen der folgenden Schritte aus:

   - Um eine vorhandene Service-Rolle aus Ihrem Konto auszuwählen, wählen Sie Use existing service role (Vorhandene Service-Rolle verwenden aus und geben Sie dann IAM-Rollen für erfolgreiche und fehlgeschlagene Übermittlungen an.

   - Um eine neue Service-Rolle in Ihrem Konto zu erstellen, wählen Sie Create new service role (Neue Service-Rolle erstellen) und dann Create new roles (Neue Rollen erstellen) aus, um die IAM-Rollen für erfolgreiche und fehlgeschlagene Übermittlungen in der IAM-Konsole zu definieren.

     Um Amazon SNS Schreibzugriff auf die Nutzung von CloudWatch Logs in Ihrem Namen zu gewähren, wählen Sie Allow.
8. Wählen Sie Änderungen speichern aus.

   Sie können jetzt die CloudWatch Protokolle mit dem Status der Nachrichtenzustellung anzeigen und analysieren. Weitere Informationen zur Verwendung CloudWatch finden Sie in der CloudWatchDokumentation.

Konfiguration der Protokollierung des Lieferstatus mithilfe der AWS SDKs

Die AWS SDKs bieten APIs in mehreren Sprachen für die Verwendung von Nachrichtenzustellungsstatusattributen mit Amazon SNS.

Themenattribute

Sie können für den Zustellungsstatus der Nachrichten die folgenden Werte für die Themenattributnamen verwenden:

HTTP

- `HTTPSuccessFeedbackRoleArn` – zeigt den Status der erfolgreichen Nachrichtenzustellung für ein Amazon-SNS-Thema an, das einen HTTP-Endpunkt abonniert hat.
- `HTTPSuccessFeedbackSampleRate` – zeigt den Prozentsatz der erfolgreichen Nachrichten an, die für ein Amazon-SNS-Thema abgefragt werden müssen, das einen HTTP-Endpunkt abonniert hat.
- `HTTPFailureFeedbackRoleArn` – zeigt den Status der fehlerhaften Nachrichtenzustellung für ein Amazon-SNS-Thema an, das einen HTTP-Endpunkt abonniert hat.

Amazon Data Firehose

- `FirehoseSuccessFeedbackRoleArn` – zeigt den Status der erfolgreichen Nachrichtenzustellung für ein Amazon-SNS-Thema an, das einen Amazon-Kinesis-Data-Firehose-Endpunkt abonniert hat.
- `FirehoseSuccessFeedbackSampleRate` – zeigt den Prozentsatz der erfolgreichen Nachrichten an, die für ein Amazon-SNS-Thema abgefragt werden müssen, das einen Amazon-Kinesis-Data-Firehose-Endpunkt abonniert hat.
- `FirehoseFailureFeedbackRoleArn` – zeigt den Status der fehlerhaften Nachrichtenzustellung für ein Amazon-SNS-Thema an, das einen Amazon-Kinesis-Data-Firehose-Endpunkt abonniert hat.

AWS Lambda

- `LambdaSuccessFeedbackRoleArn` – zeigt den Status der erfolgreichen Nachrichtenzustellung für ein Amazon-SNS-Thema an, das einen Lambda-Endpunkt abonniert hat.

- `LambdaSuccessFeedbackSampleRate` – zeigt den Prozentsatz der erfolgreichen Nachrichten an, die für ein Amazon-SNS-Thema abgefragt werden müssen, das einen Lambda-Endpunkt abonniert hat.
- `LambdaFailureFeedbackRoleArn` – zeigt den Status der fehlerhaften Nachrichtenzustellung für ein Amazon-SNS-Thema an, das einen Lambda-Endpunkt abonniert hat.

Plattformanwendungs-Endpunkt

- `ApplicationSuccessFeedbackRoleArn`— Zeigt den Status der erfolgreichen Nachrichtenzustellung für ein Amazon SNS SNS-Thema an, das einen AWS Anwendungsendpunkt abonniert hat.
- `ApplicationSuccessFeedbackSampleRate`— Gibt den Prozentsatz erfolgreicher Nachrichten an, die für ein Amazon SNS SNS-Thema, das einen AWS Anwendungsendpunkt abonniert hat, als Stichprobe verwendet werden sollen.
- `ApplicationFailureFeedbackRoleArn`— Zeigt den Status der fehlgeschlagenen Nachrichtenzustellung für ein Amazon SNS SNS-Thema an, das einen AWS Anwendungsendpunkt abonniert hat.

> **Note**
>
> Zusätzlich zum Konfigurieren von Themenattributen für den Zustellungsstatus von Benachrichtigungen, die an Amazon SNS-Anwendungsendpunkte gesendet werden, können Sie auch Anwendungsattribute für den Zustellungsstatus von Push-Benachrichtigungen einrichten, die an Push-Benachrichtigungsservices gesendet werden. Weitere Informationen finden Sie unter Verwendung von Amazon SNS-Anwendungsattributen für den Status von Nachrichtenübermittlungen.

Amazon SQS

- `SQSSuccessFeedbackRoleArn` – zeigt den Status der erfolgreichen Nachrichtenzustellung für ein Amazon-SNS-Thema an, das einen Amazon-SQS-Endpunkt abonniert hat.
- `SQSSuccessFeedbackSampleRate` – zeigt den Prozentsatz der erfolgreichen Nachrichten an, die für ein Amazon-SNS-Thema abgefragt werden müssen, das einen Amazon–SQS-Endpunkt abonniert hat.

- `SQSFailureFeedbackRoleArn` – zeigt den Status der fehlerhaften Nachrichtenzustellung für ein Amazon-SNS-Thema an, das einen Amazon–SQS-Endpunkt abonniert hat.

> **Note**
>
> Die `<ENDPOINT>FailureFeedbackRoleArn` Attribute `<ENDPOINT>SuccessFeedbackRoleArn` und werden verwendet, um Amazon SNS Schreibzugriff auf die Nutzung von CloudWatch Logs in Ihrem Namen zu gewähren. Das Attribut `<ENDPOINT>SuccessFeedbackSampleRate` dient zum Festlegen des Prozentsatzes der Samplerate (0-100) der erfolgreich zugestellten Nachrichten. Nachdem Sie das `<ENDPOINT>FailureFeedbackRoleArn` Attribut konfiguriert haben, werden bei allen fehlgeschlagenen Nachrichtenzustellungen CloudWatch Protokolle generiert.

AWS SDK-Beispiele zur Konfiguration von Themenattributen

Die folgenden Codebeispiele zeigen die Verwendung`SetTopicAttributes`.

CLI

AWS CLI

So legen Sie ein Attribut für ein Thema fest

Im folgenden `set-topic-attributes`-Beispiel wird das `DisplayName`-Attribute für das angegebene Thema festgelegt.

```
aws sns set-topic-attributes \
    --topic-arn arn:aws:sns:us-west-2:123456789012:MyTopic \
    --attribute-name DisplayName \
    --attribute-value MyTopicDisplayName
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

- Einzelheiten zur API finden Sie SetTopicAttributesin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SetTopicAttributesRequest;
import software.amazon.awssdk.services.sns.model.SetTopicAttributesResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class SetTopicAttributes {

    public static void main(String[] args) {
        final String usage = """

                Usage:    <attribute> <topicArn> <value>

                Where:
                   attribute - The attribute action to use. Valid parameters are:
 Policy | DisplayName | DeliveryPolicy .
                   topicArn - The ARN of the topic.\s
                   value - The value for the attribute.
                """;

        if (args.length < 3) {
            System.out.println(usage);
            System.exit(1);
```

```
        }

        String attribute = args[0];
        String topicArn = args[1];
        String value = args[2];

        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        setTopAttr(snsClient, attribute, topicArn, value);
        snsClient.close();
    }

    public static void setTopAttr(SnsClient snsClient, String attribute, String
 topicArn, String value) {
        try {
            SetTopicAttributesRequest request =
 SetTopicAttributesRequest.builder()
                    .attributeName(attribute)
                    .attributeValue(value)
                    .topicArn(topicArn)
                    .build();

            SetTopicAttributesResponse result =
 snsClient.setTopicAttributes(request);
            System.out.println(
                    "\n\nStatus was " + result.sdkHttpResponse().statusCode() +
 "\n\nTopic " + request.topicArn()
                            + " updated " + request.attributeName() + " to " +
 request.attributeValue());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Einzelheiten zur API finden Sie SetTopicAttributesin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie den Client in einem separaten Modul und exportieren Sie ihn.

```
import { SNSClient } from "@aws-sdk/client-sns";

// The AWS Region can be provided here using the `region` property. If you leave
 it blank
// the SDK will default to the region set in your AWS config.
export const snsClient = new SNSClient({});
```

Importieren Sie das SDK- und Client-Module und rufen Sie die API auf.

```
import { SetTopicAttributesCommand } from "@aws-sdk/client-sns";
import { snsClient } from "../libs/snsClient.js";

export const setTopicAttributes = async (
  topicArn = "TOPIC_ARN",
  attributeName = "DisplayName",
  attributeValue = "Test Topic",
) => {
  const response = await snsClient.send(
    new SetTopicAttributesCommand({
      AttributeName: attributeName,
      AttributeValue: attributeValue,
      TopicArn: topicArn,
    }),
  );
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: 'd1b08d0e-e9a4-54c3-b8b1-d03238d2b935',
```

```
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   }
  // }
  return response;
};
```

- Weitere Informationen finden Sie im AWS SDK for JavaScript -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie SetTopicAttributesin der AWS SDK for JavaScript API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

```
suspend fun setTopAttr(attribute: String?, topicArnVal: String?, value: String?)
 {

    val request = SetTopicAttributesRequest {
        attributeName = attribute
        attributeValue = value
        topicArn = topicArnVal
    }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        snsClient.setTopicAttributes(request)
        println("Topic ${request.topicArn} was updated.")
    }
}
```

- Einzelheiten zur API finden Sie SetTopicAttributesin der API-Referenz zum AWS SDK für Kotlin.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Configure the message delivery status attributes for an Amazon SNS Topic.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/
guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);
$attribute = 'Policy | DisplayName | DeliveryPolicy';
$value = 'First Topic';
$topic = 'arn:aws:sns:us-east-1:111122223333:MyTopic';

try {
    $result = $SnSclient->setTopicAttributes([
        'AttributeName' => $attribute,
        'AttributeValue' => $value,
        'TopicArn' => $topic,
```

```
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- Einzelheiten zur API finden Sie SetTopicAttributesin der AWS SDK for PHP API-Referenz.

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

```
# Service class to enable an SNS resource with a specified policy
class SnsResourceEnabler
  # Initializes the SnsResourceEnabler with an SNS resource client
  #
  # @param sns_resource [Aws::SNS::Resource] The SNS resource client
  def initialize(sns_resource)
    @sns_resource = sns_resource
    @logger = Logger.new($stdout)
  end

  # Sets a policy on a specified SNS topic
  #
  # @param topic_arn [String] The ARN of the SNS topic
  # @param resource_arn [String] The ARN of the resource to include in the policy
  # @param policy_name [String] The name of the policy attribute to set
  def enable_resource(topic_arn, resource_arn, policy_name)
    policy = generate_policy(topic_arn, resource_arn)
    topic = @sns_resource.topic(topic_arn)

    topic.set_attributes({
                           attribute_name: policy_name,
```

```
                          attribute_value: policy
                        })
    @logger.info("Policy #{policy_name} set successfully for topic
 #{topic_arn}.")
  rescue Aws::SNS::Errors::ServiceError => e
    @logger.error("Failed to set policy: #{e.message}")
  end

  private

  # Generates a policy string with dynamic resource ARNs
  #
  # @param topic_arn [String] The ARN of the SNS topic
  # @param resource_arn [String] The ARN of the resource
  # @return [String] The policy as a JSON string
  def generate_policy(topic_arn, resource_arn)
    {
      Version: "2008-10-17",
      Id: "__default_policy_ID",
      Statement: [{
                   Sid: "__default_statement_ID",
                   Effect: "Allow",
                   Principal: { "AWS": "*" },
                   Action: ["SNS:Publish"],
                   Resource: topic_arn,
                   Condition: {
                     ArnEquals: {
                       "AWS:SourceArn": resource_arn
                     }
                   }
                 }]
    }.to_json
  end
end

# Example usage:
if $PROGRAM_NAME == __FILE__
  topic_arn = "MY_TOPIC_ARN"     # Should be replaced with a real topic ARN
  resource_arn = "MY_RESOURCE_ARN" # Should be replaced with a real resource ARN
  policy_name = "POLICY_NAME"    # Typically, this is "Policy"

  sns_resource = Aws::SNS::Resource.new
  enabler = SnsResourceEnabler.new(sns_resource)
```

```
  enabler.enable_resource(topic_arn, resource_arn, policy_name)
end
```

- Weitere Informationen finden Sie im AWS SDK for Ruby -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie SetTopicAttributesin der AWS SDK for Ruby API-Referenz.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

```
    TRY.
        lo_sns->settopicattributes(
            iv_topicarn = iv_topic_arn
            iv_attributename  = iv_attribute_name
            iv_attributevalue = iv_attribute_value
        ).
        MESSAGE 'Set/updated SNS topic attributes.' TYPE 'I'.
      CATCH /aws1/cx_snsnotfoundexception.
        MESSAGE 'Topic does not exist.' TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie SetTopicAttributesin der API-Referenz zum AWS SDK für SAP ABAP.

Konfigurieren der Protokollierung des Zustellungsstatus mithilfe von AWS CloudFormation

Verwenden Sie zur Konfiguration `DeliveryStatusLogging` der Verwendung eine JSON- oder YAML-Vorlage AWS CloudFormation, um einen AWS CloudFormation Stack zu erstellen. Weitere Informationen finden Sie unter der `DeliveryStatusLogging` Eigenschaft der `AWS::SNS::Topic`

Ressource im AWS CloudFormation Benutzerhandbuch. Im Folgenden finden Sie Beispiele für AWS CloudFormation Vorlagen in JSON und YAML, mit denen Sie ein neues Thema erstellen oder ein vorhandenes Thema mit allen `DeliveryStatusLogging` Attributen für das Amazon SQS SQS-Protokoll aktualisieren können.

JSON

```
"Resources": {
    "MySNSTopic" : {
        "Type" : "AWS::SNS::Topic",
        "Properties" : {
            "TopicName" : "TestTopic",
            "DisplayName" : "TEST",
            "SignatureVersion" : "2",
            "DeliveryStatusLogging" : [{
                "Protocol": "sqs",
                "SuccessFeedbackSampleRate": "45",
                "SuccessFeedbackRoleArn": "arn:aws:iam::123456789012:role/
SNSSuccessFeedback_test1",
                "FailureFeedbackRoleArn": "arn:aws:iam::123456789012:role/
SNSFailureFeedback_test2"
            }]
        }
    }
}
```

YAML

```
Resources:
  MySNSTopic:
    Type: AWS::SNS::Topic
    Properties:
      TopicName:TestTopic
      DisplayName:TEST
      SignatureVersion:2
      DeliveryStatusLogging:
       - Protocol: sqs
         SuccessFeedbackSampleRate: 45
         SuccessFeedbackRoleArn: arn:aws:iam::123456789012:role/
SNSSuccessFeedback_test1
         FailureFeedbackRoleArn: arn:aws:iam::123456789012:role/
SNSFailureFeedback_test2
```

Wiederholungsversuche bei der Nachrichtenzustellung Amazon SNS

Amazon SNS definiert eine Zustellungsrichtlinie für jedes Zustellungsprotokoll. Die Zustellungsrichtlinie definiert, wie Amazon SNS die Zustellung von Nachrichten bei serverseitigen Fehlern (wenn das den abonnierten Endpunkt hostenden System nicht verfügbar ist) wiederholt. Wenn die Zustellungsrichtlinie erschöpft ist, stoppt Amazon SNS den erneuten Versand und verwirft die Nachricht, es sei denn, dem Abonnement ist eine Warteschlange für unzustellbare Nachrichten angehängt. Weitere Informationen finden Sie unter Amazon SNS Queues für unzustellbare Nachrichten (DLQs).

Themen

Zustellungsprotokolle und -richtlinien

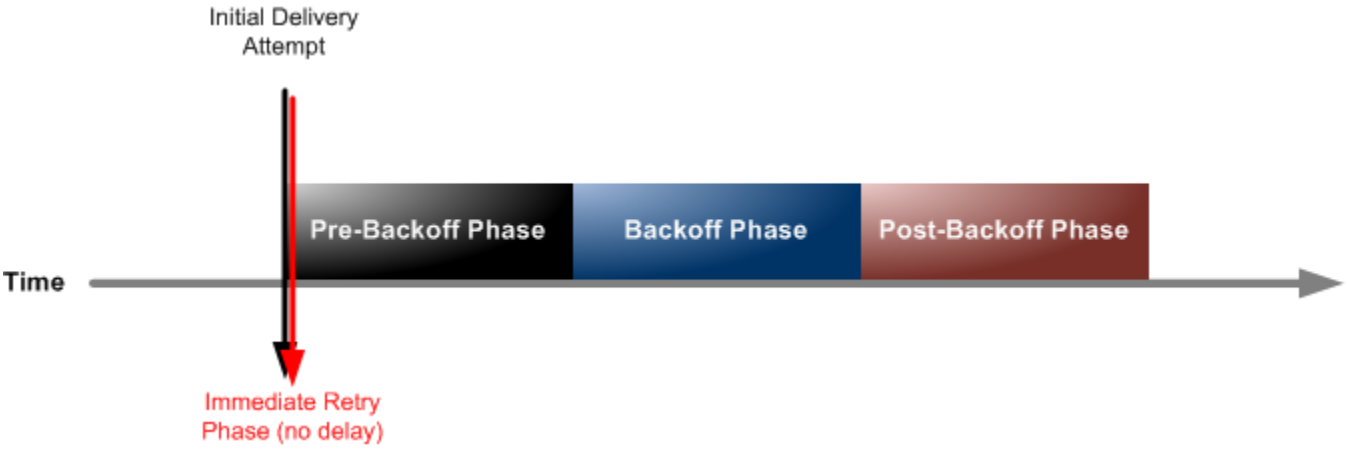
> **Note**
>
> - Mit Ausnahme von HTTP/S können Sie Amazon SNS-definierte Zustellungsrichtlinien nicht ändern. Nur HTTP/S unterstützt benutzerdefinierte Richtlinien. Siehe Erstellen einer HTTP/S-Zustellungsrichtlinie.
> - Amazon SNS wendet Jittering bei Zustellungswiederholungen an. Weitere Informationen finden Sie im Beitrag Exponential Backoff and Jitter im AWS Architecture Blog.
> - Die Gesamtdauer der Richtlinienwiederholung für einen HTTP/S-Endpunkt darf nicht mehr als 3.600 Sekunden betragen. Dies ist ein hartes Limit, das nicht erhöht werden kann.

Endpunkttyp	Zustellungsprotokolle	Phase „Sofortiger Wiederholungsversuch (keine Verzögerung)“	Phase „Pre-Backoff“	Phase „Backoff“	Phase „Post-Backoff“	Gesamtzahl der Versuche
AWS verwaltete Endpunkte	Amazon Data Firehose[1]	3-mal, ohne Verzögerung	2-mal, mit 1 Sekunde Verzögerung	10-mal, mit exponentiellem Backoff von 1 Sekunde bis 20 Sekunden	100.000-mal, mit 20 Sekunden Verzögerung	100.015-mal, über 23 Tage
	AWS Lambda					
	Amazon SQS					
Vom Kunden verwaltete Endpunkte	SMTP	0-mal, ohne Verzögerung	2-mal, mit 10 Sekunden Verzögerung	10-mal, mit exponentiellem Backoff von 10 Sekunden bis 600 Sekunden (10 Minuten)	38-Mal, mit 600 Sekunden (10 Minuten) Verzögerung	50 Versuche, über 6 Stunden
	SMS					
	Mobile Push					

[1] Für Drosselungsfehler mit dem Firehose-Protokoll verwendet Amazon SNS dieselben Lieferrichtlinien wie für vom Kunden verwaltete Endgeräte.

Stufen von Zustellungsrichtlinien

Das folgende Diagramm zeigt die Phasen einer Zustellungsrichtlinie.



Jede Zustellungsrichtlinie umfasst vier Phasen.

1. Phase „Sofortiger Wiederholungsversuch (keine Verzögerung) – Diese Phase erfolgt unmittelbar nach dem ersten Zustellungsversuch. Es gibt keine Verzögerung zwischen den Wiederholungen in dieser Phase.
2. Pre-Backoff-Phase – Diese Phase folgt direkt auf die Retry-Phase. Amazon SNS verwendet diese Phase für eine Reihe von Wiederholungsversuchen, bevor eine Backoff-Funktion angewendet wird. Diese Phase gibt die Anzahl der Wiederholungsversuche und die Verzögerung dazwischen an.
3. Backoff-Phase – Diese Phase steuert die Verzögerung zwischen den Wiederholungsversuchen mithilfe der Funktion „retry-backoff“. Diese Phase legt eine Mindestverzögerung, eine Maximalverzögerung und eine retry-backoff-Funktion fest, die definiert, wie schnell die Verzögerung vom Mindest- zum Maximalwert steigt. Die backoff-Funktion kann arithmetisch, exponentiell, geometrisch oder linear sein.
4. Post-Backoff-Phase – Diese Phase folgt auf die Backoff-Phase. Sie gibt die Anzahl der Wiederholungsversuche und die Verzögerung dazwischen an. Dies ist die letzte Phase.

Erstellen einer HTTP/S-Zustellungsrichtlinie

Mit einer Zustellungsrichtlinie und deren vier Phasen können Sie festlegen, wie Amazon SNS die Zustellung von Nachrichten an HTTP/S-Endpunkte wiederholt. Mit Amazon SNS können Sie die standardmäßige Wiederholungsrichtlinie für HTTP-Endpunkte außer Kraft setzen, wenn Sie beispielsweise die Richtlinie basierend auf der Kapazität Ihres HTTP-Servers anpassen möchten.

Sie können Ihre HTTP/S-Zustellungsrichtlinie als JSON-Objekt auf Abonnement- oder Themenebene festlegen. Wenn Sie die Richtlinie auf Themenebene definieren, gilt sie für

alle HTTP/S-Abonnements, die dem Thema zugeordnet sind. Wenn Sie die Lieferrichtlinie auf Abonnementebene festlegen möchten, können Sie entweder die `Subscribe`- oder die `SetSubscriptionAttributes`-API-Aktion verwenden. Wenn Sie die Bereitstellungsrichtlinie auf Themenebene festlegen möchten, können Sie entweder die `CreateTopic`- oder die `SetTopicAttributes`-API-Aktion verwenden. Alternativ können Sie die Ressource auch in Ihren Vorlagen verwenden. AWS::SNS::Subscription AWS CloudFormation

Sie sollten Ihre Zustellungsrichtlinie entsprechend der Kapazität Ihres HTTP/S-Servers anpassen. Sie können die Richtlinie als Themen- oder Abonnementattribut festlegen. Wenn alle HTTP/S-Abonnements in Ihrem Thema auf denselben HTTP/S-Server weisen, wird empfohlen, die Zustellungsrichtlinie als Themenattribut festzulegen, damit sie für alle HTTP/S-Abonnements im Thema gültig bleibt. Andernfalls müssen Sie für jedes HTTP/S-Abonnement in Ihrem Thema eine Zustellungsrichtlinie erstellen, je nach Kapazität des HTTP/S-Servers, auf den die Richtlinie weist.

Sie können in der Anforderungsrichtlinie auch den Content-Type-Header festlegen, um den Medientyp der Benachrichtigung anzugeben. Standardmäßig sendet Amazon SNS alle Benachrichtigungen an HTTP/S-Endpunkte, wobei der Inhaltstyp auf `text/plain; charset=UTF-8` festgelegt ist. Mit Amazon SNS können Sie die Standardanforderungsrichtlinie außer Kraft setzen. Weitere Informationen zum unterstützten `headerContentType` und zu den Einschränkungen finden Sie in der nachfolgenden Tabelle.

Das folgende JSON-Objekt stellt eine Zustellungsrichtlinie dar, mit der Amazon SNS angewiesen wird, einen fehlgeschlagenen HTTP/S-Zustellungsversuch wie folgt zu wiederholen:

1. 3-mal sofort in der Phase ohne Verzögerung
2. 2-mal (mit 1 Sekunde Verzögerung) in der Pre-Backoff-Phase
3. 10-mal (mit exponentiellem Backoff von 1 Sekunde bis 60 Sekunden)
4. 35-mal (mit 60 Sekunden Verzögerung) in der Post-Backoff-Phase

Amazon SNS unternimmt insgesamt 50 Versuche, bevor die Nachricht verworfen wird.
Um die Nachricht zu behalten, nachdem die in der Zustellungsrichtlinie angegebenen Wiederholungsversuche erschöpft sind, konfigurieren Sie Ihr Abonnement so, dass unzustellbare Nachrichten in eine Dead-Letter-Warteschlange (Warteschlange für unzustellbare Nachrichten; DLQ) verschoben werden. Weitere Informationen finden Sie unter Amazon SNS Queues für unzustellbare Nachrichten (DLQs).

> **Note**
>
> Diese Zustellungsrichtlinie weist Amazon SNS außerdem an, die Anzahl der Zustellungen auf maximal 10 pro Sekunde zu drosseln mit der `maxReceivesPerSecond` Eigenschaft. Diese Selbstdrosselungsrate kann dazu führen, dass mehr Nachrichten veröffentlicht (eingehender Verkehr) als zugestellt (ausgehender Verkehr) werden. Wenn mehr eingehender Datenverkehr als ausgehender Datenverkehr vorhanden ist, kann Ihr Abonnement einen großen Nachrichtenrückstand ansammeln, was zu einer hohen Nachrichtenzustellungslatenz führen kann. Stellen Sie in Ihren Zustellungsrichtlinien sicher, dass Sie einen Wert für`maxReceivesPerSecond` festlegen, der sich nicht negativ auf Ihre Arbeitslast auswirkt.

> **Note**
>
> Diese Bereitstellungsrichtlinie überschreibt die standardmäßige Inhaltstyp für die HTTP/S-Benachrichtigung zu `application/json`.

```
{
    "healthyRetryPolicy": {
        "minDelayTarget": 1,
        "maxDelayTarget": 60,
        "numRetries": 50,
        "numNoDelayRetries": 3,
        "numMinDelayRetries": 2,
        "numMaxDelayRetries": 35,
        "backoffFunction": "exponential"
    },
    "throttlePolicy": {
        "maxReceivesPerSecond": 10
    },
    "requestPolicy": {
        "headerContentType": "application/json"
    }
}
```

Die Bereitstellungsrichtlinie umfasst eine Wiederholungs-, eine Drosselungs- und eine Anforderungsrichtlinie. Insgesamt gibt es 9 Attribute in einer Zustellungsrichtlinie.

Richtlinie	Beschreibung	Constraint
`minDelayTarget`	Die Mindestverzögerung bei einem Wiederholungsversuch. Einheit: Sekunden	1 bis zur Maximalverzögerung Standard: 20
`maxDelayTarget`	Die Maximalverzögerung bei einem Wiederholungsversuch. Einheit: Sekunden	Mindestverzögerung auf 3.600 Standard: 20
`numRetries`	Die Gesamtzahl der Wiederholungsversuche, einschließlich sofortiger Wiederholungsversuche sowie, Pre-Backoff-, Backoff- und Post-Backoff-Wiederholungsversuchen.	0 bis 100 Standard: 3
`numNoDelayRetries`	Die Anzahl der Wiederholungen, die sofort durchgeführt werden sollen, ohne Verzögerung zwischen ihnen.	0 oder höher Standard: 0
`numMinDelayRetries`	Die Anzahl der Wiederholungsversuche in der Pre-Backoff-Phase mit der angegebenen Mindestverzögerung dazwischen.	0 oder höher Standard: 0
`numMaxDelayRetries`	Die Anzahl der Wiederholungsversuche in der Post-Backoff-Phase mit der Maximalverzögerung dazwischen.	0 oder höher Standard: 0
`backoffFunction`	Das Modell für Backoff zwischen Wiederholungsversuchen.	Eine von vier Optionen: • Arithmetisch

Richtlinie	Beschreibung	Constraint
		• Exponentiell • Geometrisch • Linear Standard: linear
`maxReceivesPerSecond`	Die maximale Anzahl der Zustellungen pro Sekunde pro Abonnement.	1 oder höher Standard: Keine Drosselung

Richtlinie	Beschreibung	Constraint
`headerContentType`	Der Inhaltstyp der Benachrichtigung, die an HTTP/S-Endpunkte gesendet wird.	Wenn die Anforderungsrichtlinie nicht definiert ist, wird der Inhaltstyp standardmäßig auf `text/plain; charset=UTF-8` festgelegt. Wenn die Bereitstellung von Rohnachrichten für ein Abonnement deaktiviert ist (Standard) oder wenn die Bereitstellungsrichtlinie auf Themenebene definiert ist, werden die Header-Inhaltstypen `application/json` und `text/plain` unterstützt. Wenn die Bereitstellung von Rohnachrichten für ein Abonnement aktiviert ist, werden die folgenden Inhaltstypen unterstützt: • text/css • text/csv • text/html • text/plain • text/xml • application/atom+xml • application/json • Anwendung/octet-stream • application/soap+xml • Anwendung/ x-www-form-urlencoded

Richtlinie	Beschreibung	Constraint
		• application/xhtml+xml • application/xml

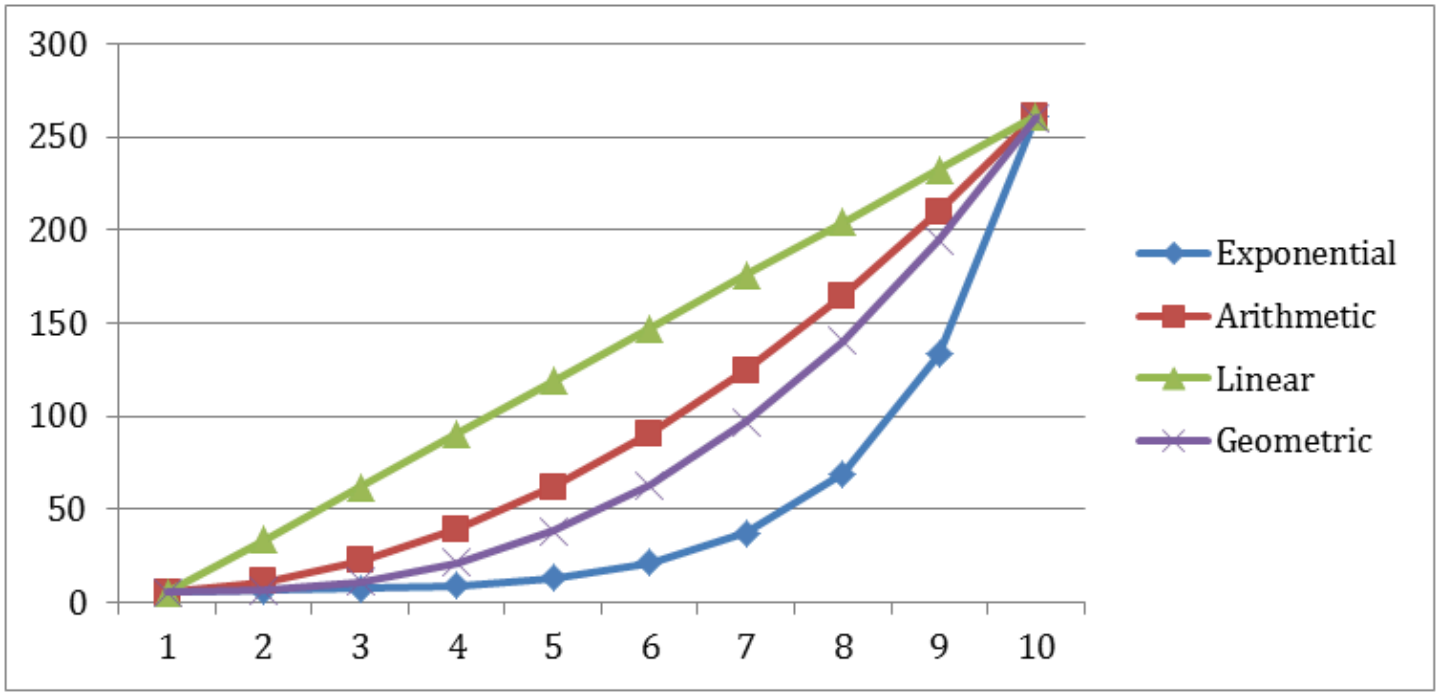
Amazon SNS verwendet die folgende Formel, um die Anzahl der Wiederholungsversuche in der Backoff-Phase zu berechnen:

```
numRetries - numNoDelayRetries - numMinDelayRetries - numMaxDelayRetries
```

Sie können die Häufigkeit der Wiederholungsversuche in der Backoff-Phase mithilfe von drei Parametern steuern.

- `minDelayTarget` – Definiert die Verzögerung, die mit dem ersten Wiederholungsversuch in der Backoff-Phase verbunden ist.
- `maxDelayTarget` – Definiert die Verzögerung, die mit dem letzten Wiederholungsversuch in der Backoff-Phase verbunden ist.
- `backoffFunction` – Definiert den Algorithmus, den Amazon SNS zum Berechnen der Abstände im Zusammenhang mit allen Wiederholversuchen zwischen dem ersten und letzten Wiederholungsversuch in der Backoff-Phase verwendet. Sie können eine von vier retry-backoff-Funktionen verwenden.

Das folgende Diagramm zeigt, wie sich jede Wiederholungs-Backoff-Funktion auf die Verzögerung auswirkt, die mit Wiederholungen während der Backoff-Phase verbunden ist: Eine Zustellungsrichtlinie, bei der die Gesamtzahl der Wiederholungen auf 10 festgelegt ist, die minimale Verzögerung auf 5 Sekunden und die maximale Verzögerung auf 260 Sekunden festgelegt ist. Die vertikale Achse stellt die Verzögerung in Sekunden für jede der 10 Wiederholungen dar. Die horizontale Achse stellt die Anzahl der Wiederholungsversuche vom ersten bis zum zehnten Versuch dar.



Amazon SNS Queues für unzustellbare Nachrichten (DLQs)

Eine Queue für unzustellbare Nachrichten ist eine Amazon-SQS-Queue, an die ein Amazon SNS Abonnement Nachrichten senden kann, die nicht erfolgreich an Abonnenten gesendet werden konnten. Nachrichten, die aufgrund von Clientfehlern oder Serverfehlern nicht zugestellt werden können, werden in der Queue für unzustellbare Nachrichten zur weiteren Analyse oder erneuten Verarbeitung gespeichert. Weitere Informationen finden Sie unter Konfigurieren einer Amazon-SNS-Warteschlange für unzustellbare Nachrichten für ein Abonnement und Wiederholungsversuche bei der Nachrichtenzustellung Amazon SNS.

> **Note**
>
> - Das Amazon SNS Abonnement und die Amazon SQS Queue müssen unter demselben AWS-Konto und derselben Region laufen.
> - Für ein FIFO-Thema können Sie eine Amazon-SQS-Warteschlange als Warteschlange für unzustellbare Nachrichten für das Amazon-SNS-Abonnement verwenden. FIFO-Themenabonnements verwenden FIFO-Warteschlangen und Standardthemenabonnements verwenden Standardwarteschlangen.
> - Wenn Sie eine verschlüsselte Amazon-SQS-Warteschlange als Warteschlange für unzustellbare Nachrichten nutzen möchten, müssen Sie einen benutzerdefinierten KMS-

Schlüssel mit einer Schlüsselrichtlinie verwenden, die dem Amazon-SNS-Service-Prinzipal Zugriff auf AWS KMS-API-Aktionen gewährt. Weitere Informationen finden Sie unter Verschlüsselung im Ruhezustand in dieser Anleitung und Schutz von Amazon SQS Daten mithilfe serverseitiger Verschlüsselung (SSE) undAWS KMS im Amazon Simple Queue Service-Entwicklerhandbuch.

Themen

Warum schlagen Nachrichtenzustellungen fehl?

Im Allgemeinen schlägt die Nachrichtenzustellung fehl, wenn Amazon SNS aufgrund eines clientseitigen oder serverseitigen Fehlers nicht auf einen abonnierten Endpunkt zugegriffen werden kann. Wenn Amazon SNS einen clientseitigen Fehler erhält oder weiterhin einen serverseitigen Fehler für eine Nachricht erhält, die über die Anzahl der Wiederholungsversuche hinausgeht, die in der entsprechenden Wiederholungsrichtlinie angegeben ist, verwirft Amazon SNS die Nachricht — es sei denn, es ist eine Queue für unzustellbare Nachrichten an das Abonnement angehängt. Fehlgeschlagene Zustellungen ändern den Status Ihrer Abonnements nicht. Weitere Informationen finden Sie unter Wiederholungsversuche bei der Nachrichtenzustellung Amazon SNS.

Client-seitige Fehler

Client-seitige Fehler können auftreten, wenn Amazon SNS über veraltete Abonnementmetadaten verfügt. Diese Fehler treten im Allgemeinen auf, wenn ein Besitzer den Endpunkt löscht (z. B. eine Lambda-Funktion, mit der ein Thema abonniert wird) oder wenn ein Besitzer die dem abonnierten Endpunkt zugewiesene Richtlinie so ändert, dass Amazon SNS keine Nachrichten an den Endpunkt zugestellt werden können. Amazon SNS versucht nicht, eine durch einen clientseitigen Fehler verursachte fehlerhafte Nachrichtenzustellung zu wiederholen. Amazon SNS versucht keine erneute Nachrichtenzustellung, die aufgrund eines clientseitigen Fehlers fehlschlägt.

Serverseitige Fehler

Serverseitige Fehler können auftreten, wenn das für den abonnierten Endpunkt verantwortliche System nicht verfügbar ist oder eine Ausnahme zurückgibt, die angibt, dass eine gültige Anfrage nicht von Amazon SNS verarbeitet werden kann. Bei serverseitigen Fehlern versucht Amazon SNS die fehlgeschlagenen Zustellungen mithilfe einer linearen oder exponentiellen Backoff-Funktion zu wiederholen. Bei serverseitigen Fehlern, die durch von AWS verwaltete und von Amazon SQS oder AWS Lambda gesicherte Endpunkte verursacht werden, wird die Zustellung innerhalb von 23 Tagen bis zu 100.015-mal von Amazon SNS wiederholt.

Von Kunden verwaltete Endpunkte (wie HTTP, SMTP, SMS oder mobile Pushnachrichten) können ebenfalls serverseitige Fehler verursachen. Amazon SNS versucht auch die Zustellung an diese Endgerätetypen erneut. Obwohl HTTP-Endpunkte von Kunden definierte Wiederholungsrichtlinien unterstützen, legt Amazon SNS eine interne Zustellungsrichtlinie für SMTP-, SMS- und mobile Push-Endpunkte auf 50-mal innerhalb von 6 Stunden fest.

Funktionsweise von Queues für unzustellbare Nachrichten

Eine Queue für unzustellbare Nachrichten wird einem Amazon SNS-Abonnement (nicht einem Thema) zugewiesen, da Nachrichtenzustellungen auf Abonnementebene stattfinden. Auf diese Weise können Sie den ursprünglichen Zielendpunkt für jede Nachricht leichter identifizieren.

Bei einer Queue für unzustellbare Nachrichten, die einem Amazon SNS-Abonnement zugeordnet ist, handelt es sich um eine gewöhnliche Amazon SQS-Queue. Weitere Informationen zum Aufbewahrungszeitraum für Nachrichten finden Sie unter Kontingente im Zusammenhang mit Nachrichten in der Amazon Simple Queue Service-Entwicklerrichtlinie. Sie können den Aufbewahrungszeitraum für Nachrichten mithilfe der `SetQueueAttributes` Amazon SQS-API-Aktion anpassen. Um Ihre Anwendungen widerstandsfähiger zu machen, empfehlen wir Ihnen, die maximale Aufbewahrungsfrist für Queues für unzustellbare Nachrichten auf 14 Tage festzulegen.

Wie werden Nachrichten in eine Queue für unzustellbare Nachrichten verschoben?

Ihre Nachrichten werden mithilfe einer Richtlinie für erneute Ausführung in eine Queue für unzustellbare Nachrichten verschoben. Eine Richtlinie für erneute Ausführung ist ein JSON-Objekt, das auf den ARN der Queue für unzustellbare Nachrichten verweist. Das `deadLetterTargetArn`-Attribut gibt den ARN an. Der ARN muss auf eine Amazon SQS-Queue in dem AWS-Konto und der Region Ihres Amazon SNS-Abonnement verweisen. Weitere Informationen finden Sie unter Konfigurieren einer Amazon-SNS-Warteschlange für unzustellbare Nachrichten für ein Abonnement.

Das folgende JSON-Objekt ist eine Beispielrichtlinie für eine Richtlinie für erneute Ausführung, die einem SNS-Abonnement zugewiesen ist.

```
{
  "deadLetterTargetArn": "arn:aws:sqs:us-east-2:123456789012:MyDeadLetterQueue"
}
```

Wie kann ich Nachrichten aus einer Queue für unzustellbare Nachrichten verschieben?

Sie können Nachrichten auf zwei Arten aus einer Queue für unzustellbare Nachrichten verschieben:

- Vermeiden Sie Schreiben von Amazon SQS-Verbraucherlogik: legen Sie Ihre Queue für unzustellbare Nachrichten als Ereignisquelle für die Lambda-Funktion fest, um Ihre Queue für unzustellbare Nachrichten zu leeren.
- Amazon SQS-Verbraucherlogik schreiben : Verwenden Sie die Amazon SQS-API, das AWS-SDK oder AWS CLI, um benutzerdefinierte Verbraucherlogik zum Abrufen, Verarbeiten und Löschen der Nachrichten in der Queue für unzustellbare Nachrichten zu schreiben.

Wie kann ich Queues für unzustellbare Nachrichten überwachen und protokollieren?

Mit Amazon CloudWatch-Metriken können Sie die Queues für unzustellbare Nachrichten, die Ihren Amazon SNS-Abonnements zugeordnet sind, überwachen. Alle Amazon SQS-Queues senden CloudWatch-Metriken in einminütigen Intervallen aus. Weitere Informationen finden Sie unter Verfügbare CloudWatch-Metriken für Amazon SQS im Amazon Simple Queue Service-Entwicklerhandbuch. Alle Amazon SNS-Abonnements mit Queues für unzustellbare Nachrichten geben auch CloudWatch-Metriken aus. Weitere Informationen finden Sie unter Überwachung von Amazon-SNS-Themen mit CloudWatch.

Mithilfe von CloudWatch-Metriken und Alarmen können Sie sich über Aktivitäten in Ihren Queues für unzustellbare Nachrichten benachrichtigen lassen. Wenn Sie beispielsweise erwarten, dass die Queue für unzustellbare Nachrichten immer leer ist, können Sie einen CloudWatch-Alarm für die `NumberOfMessagesSent` Metrik erstellen. Sie können den Schwellenwert für den Alarm auf 0 festlegen und ein Amazon-SNS-Thema angeben, um benachrichtigt zu werden, wenn der Alarm ausgelöst wird. Dieses Amazon-SNS-Thema kann Ihre Alarmbenachrichtigung an jeden Endpunkttyp (z. B. E-Mail-Adresse, Telefonnummer oder mobile Pager-App) zustellen.

Mit CloudWatch Logs können Sie die Ausnahmen untersuchen, die dazu führen, dass Amazaon-SNS-Zustellungen fehlschlagen, und dass Nachrichten an Queues für unzustellbare Nachrichten gesendet werden. Amazon SNS kann sowohl erfolgreiche als auch fehlgeschlagene Zustellungen in CloudWatch protokollieren. Amazon SNS kann sowohl erfolgreiche als auch fehlgeschlagene Lieferungen in CloudWatch protokollieren. Weitere Informationen finden Sie unter Zustellungsstatus von Amazon SNS Nachrichtenübermittmittungen.

Konfigurieren einer Amazon-SNS-Warteschlange für unzustellbare Nachrichten für ein Abonnement

Eine Queue für unzustellbare Nachrichten ist eine Amazon-SQS-Queue, an die ein Amazon SNS Abonnement Nachrichten senden kann, die nicht erfolgreich an Abonnenten gesendet werden konnten. Nachrichten, die aufgrund von Clientfehlern oder Serverfehlern nicht zugestellt werden können, werden in der Queue für unzustellbare Nachrichten zur weiteren Analyse oder erneuten Verarbeitung gespeichert. Weitere Informationen finden Sie unter Amazon SNS Queues für unzustellbare Nachrichten (DLQs) und Wiederholungsversuche bei der Nachrichtenzustellung Amazon SNS.

Auf dieser Seite wird gezeigt, wie Sie das AWS Management Console, ein AWS SDK, das und verwenden können, AWS CloudFormation um eine Warteschlange für unzustellbare Briefe für ein Amazon SNS SNS-Abonnement zu konfigurieren. AWS CLI

> **Note**
>
> Für ein FIFO-Thema können Sie eine Amazon-SQS-Warteschlange als Warteschlange für unzustellbare Nachrichten für das Amazon-SNS-Abonnement verwenden. FIFO-Themen-Abonnements verwenden FIFO-Warteschlangen, und Standardthemenabonnements verwenden Standardwarteschlangen.

Voraussetzungen

Führen Sie die folgenden Voraussetzungen aus, bevor Sie eine Queue für unzustellbare Nachrichten konfigurieren:

1. Erstellen Sie ein Amazon-SNS-Thema mit dem Namen `MyTopic`.
2. Erstellen einer Amazon-SQS-Queuebenannt`MyEndpoint`verwenden, um als Endpunkt für das Amazon SNS Abonnement zu verwenden.

3. (Überspringen für AWS CloudFormation) Abonnieren Sie die Warteschlange für das Thema.
4. Erstellen Sie eine weitere Amazon–SQS-Warteschlange mit dem Namen `MyDeadLetterQueue`, die als Queue für das Amazon-SNS-Abonnement dienen soll.
5. Um dem Amazon SNS-Prinzipal Zugriff auf die Amazon SQS-API-Aktion zu gewähren, legen Sie die folgende Queue-Richtlinie für `MyDeadLetterQueue` fest.

```
{
  "Statement": [{
    "Effect": "Allow",
    "Principal": {
      "Service": "sns.amazonaws.com"
    },
    "Action": "SQS:SendMessage",
    "Resource": "arn:aws:sqs:us-east-2:123456789012:MyDeadLetterQueue",
    "Condition": {
      "ArnEquals": {
        "aws:SourceArn": "arn:aws:sns:us-east-2:123456789012:MyTopic"
      }
    }
  }]
}
```

Themen

- Um eine Warteschlange für unzustellbare Briefe für ein Amazon SNS SNS-Abonnement zu konfigurieren, verwenden Sie AWS Management Console
- So konfigurieren Sie mithilfe eines SDK eine Warteschlange für unzustellbare Briefe für ein Amazon SNS SNS-Abonnement AWS
- Um eine Warteschlange für unzustellbare Briefe für ein Amazon SNS SNS-Abonnement zu konfigurieren, verwenden Sie AWS CLI
- Um eine Warteschlange für unzustellbare Briefe für ein Amazon SNS SNS-Abonnement zu konfigurieren, verwenden Sie AWS CloudFormation

Um eine Warteschlange für unzustellbare Briefe für ein Amazon SNS SNS-Abonnement zu konfigurieren, verwenden Sie AWS Management Console

Bevor Sie mit diesem Tutorial beginnen, müssen Sie die Voraussetzungen erfüllen.

1. Melden Sie sich bei der Amazon-SQS-Konsole an.
2. Erstellen Sie eine Amazon-SQS-Warteschlange oder verwenden Sie eine vorhandene Queue und notieren Sie sich den ARN der Queue auf deren Details-Registerkarte, z. B.:

   ```
   arn:aws:sqs:us-east-2:123456789012:MyDeadLetterQueue
   ```

3. Melden Sie sich bei der Amazon SNS-Konsole an.
4. Wählen Sie im Navigationsbereich Subscriptions (Abonnements) aus.
5. Wählen Sie auf der Seite Subscriptions (Abonnements) ein vorhandenes Abonnement und dann Edit (Bearbeiten) aus.
6. Erweitern Sie auf der Seite Bearbeiten Sie ***1234a567-bc89-012d-3e45-6fg7h890123i*** den Abschnitt Redrive policy (dead-letter queue) Richtlinie für erneute Ausführung (Queue für unzustellbare Nachrichten) und führen Sie dann die folgenden Schritte aus:

   a. Wählen Sie Aktiviert.

   b. Geben Sie den ARN einer Amazon SQS Queue an.

7. Wählen Sie Änderungen speichern aus.

   Ihr Abonnement ist für die Verwendung einer Queue für unzustellbare Nachrichten konfiguriert.

So konfigurieren Sie mithilfe eines SDK eine Warteschlange für unzustellbare Briefe für ein Amazon SNS SNS-Abonnement AWS

Bevor Sie dieses Beispiel ausführen, müssen Sie sicherstellen, dass Sie dieVoraussetzungenerfüllen.

Um ein AWS SDK verwenden zu können, müssen Sie es mit Ihren Anmeldeinformationen konfigurieren. Weitere Informationen finden Sie unter Freigegebene Konfigurations- und Anmeldeinformationsdateien im AWS -Referenzhandbuch zu SDKs und Tools.

Das folgende Codebeispiel zeigt, wie Sie es verwenden`SetSubscriptionAttributesRedrivePolicy`.

Java

SDK für Java 1.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

```
// Specify the ARN of the Amazon SNS subscription.
String subscriptionArn =
    "arn:aws:sns:us-east-2:123456789012:MyEndpoint:1234a567-
bc89-012d-3e45-6fg7h890123i";

// Specify the ARN of the Amazon SQS queue to use as a dead-letter queue.
String redrivePolicy =
    "{\"deadLetterTargetArn\":\"arn:aws:sqs:us-
east-2:123456789012:MyDeadLetterQueue\"}";

// Set the specified Amazon SQS queue as a dead-letter queue
// of the specified Amazon SNS subscription by setting the RedrivePolicy
 attribute.
SetSubscriptionAttributesRequest request = new SetSubscriptionAttributesRequest()
    .withSubscriptionArn(subscriptionArn)
    .withAttributeName("RedrivePolicy")
    .withAttributeValue(redrivePolicy);
sns.setSubscriptionAttributes(request);
```

Um eine Warteschlange für unzustellbare Briefe für ein Amazon SNS SNS-Abonnement zu konfigurieren, verwenden Sie AWS CLI

Bevor Sie mit diesem Tutorial beginnen, müssen Sie die Voraussetzungen erfüllen.

1. Installieren und Konfigurieren der AWS CLI. Weitere Informationen finden Sie im AWS Command Line Interface -Benutzerhandbuch.
2. Verwenden Sie den folgenden -Befehl.

```
aws sns set-subscription-attributes \
```

```
--subscription-arn arn:aws:sns:us-east-2:123456789012:MyEndpoint:1234a567-
bc89-012d-3e45-6fg7h890123i
--attribute-name RedrivePolicy
--attribute-value "{\"deadLetterTargetArn\": \"arn:aws:sqs:us-
east-2:123456789012:MyDeadLetterQueue\"}"
```

Um eine Warteschlange für unzustellbare Briefe für ein Amazon SNS SNS-Abonnement zu konfigurieren, verwenden Sie AWS CloudFormation

Bevor Sie mit diesem Tutorial beginnen, müssen Sie die Voraussetzungen erfüllen.

1. Kopieren Sie den JSON-Code in eine Datei mit dem Namen `MyDeadLetterQueue.json`.

```
{
  "Resources": {
    "mySubscription": {
      "Type" : "AWS::SNS::Subscription",
      "Properties" : {
        "Protocol": "sqs",
        "Endpoint": "arn:aws:sqs:us-east-2:123456789012:MyEndpoint",
        "TopicArn": "arn:aws:sns:us-east-2:123456789012:MyTopic",
        "RedrivePolicy": {
          "deadLetterTargetArn":
            "arn:aws:sqs:us-east-2:123456789012:MyDeadLetterQueue"
        }
      }
    }
  }
}
```

2. Melden Sie sich an der AWS CloudFormation -Konsole an.
3. Klicken Sie auf der Seite Select Template (Vorlage auswählen) auf die Option Upload a template to Amazon S3 (Eine Vorlage zu Amazon S3 hochladen), wählen Sie dann Ihre `MyDeadLetterQueue.json`-Datei und anschließend Next (Weiter) aus.
4. Geben Sie auf der Seite Specify Details (Details angeben) `MyDeadLetterQueue` als Stack Name (Stack-Name) ein und wählen Sie Next (Weiter) aus.
5. Wählen Sie auf der Seite Optionen Weiter aus.
6. Klicken Sie auf der Seite Review auf Create.

AWS CloudFormation beginnt mit der Erstellung des **`MyDeadLetterQueue`** Stacks und zeigt den Status CREATE_IN_PROGRESS an. Wenn der Vorgang abgeschlossen ist, wird der Status CREATE_COMPLETE AWS CloudFormation angezeigt.

Amazon SNS Nachrichtenarchivierung, -wiederholung und -analyse

Amazon SNS-Standardthemen unterstützen die Nachrichtenarchivierung über Amazon Data Firehose. Sie können Benachrichtigungen an Firehose-Bereitstellungsdatenströme verteilen, mit denen Sie Benachrichtigungen an von Firehose unterstützte Speicher- und Analyseziele senden können, einschließlich Amazon Simple Storage Service (Amazon S3), Amazon Redshift und mehr.

Amazon SNS FIFO-Themen unterstützen ein direktes Nachrichtenarchiv ohne Code, mit dem Themeninhaber Nachrichten, die zu einem Thema veröffentlicht wurden, bis zu 365 Tage lang speichern (oder archivieren) können. Für Themen mit einem aktiven `ArchivePolicy` können Abonnenten dann eine `ReplayPolicy` erstellen, um die archivierten Nachrichten an einen abonnierten Endpunkt abzurufen (oder zu wiederholen). Für weitere Informationen zu dieser Funktion siehe Archivierung und Wiederholung von Nachrichten für FIFO-Themen.

Features	Standardthemen	FIFO-Themen
Nachrichtenarchivierung	Fanout zu Firehose-Bereitstellungsdatenströmen	Nachrichtenarchivierung für Besitzer von FIFO-Themen
Wiederholung der Nachricht	Die Wiederholung von Standardthemen ist keine integrierte Funktion. Viele Kunden erstellen ihre eigenen Nachrichten anhand ihres Nachrichtenarchivs.	Nachrichtenwiederholung für Subscriber des FIFO-Themas

Verwenden von Amazon SNS für Anwendung-zu-Anwendung-Messaging (A2A)

Dieser Abschnitt enthält Informationen zur Verwendung von Amazon SNS für Anwendung-zu-Anwendung-Messaging mit Abonnenten.

Themen

Fanout zu Firehose-Bereitstellungsdatenströmen

Sie können Amazon-Data-Firehose-Bereitstellungsdatenströme für Amazon SNS-Themen abonnieren, mit denen Sie Benachrichtigungen an zusätzliche Speicher- und Analyseendpunkte senden können. In einem Amazon SNS-Thema veröffentlichte Nachrichten werden an den abonnierten Firehose-Bereitstellungs-Stream gesendet und an das Ziel übermittelt, wie in Firehose konfiguriert. Ein Abonnementinhaber kann bis zu fünf Firehose-Bereitstellungsdatenströme für ein Amazon SNS-Thema abonnieren. Jeder Firehose-Bereitstellungsdatenstrom verfügt über ein Standardkontingent für Anforderungen und Durchsatz pro Sekunde. Dieses Limit könnte dazu führen, dass mehr Nachrichten veröffentlicht werden (eingehender Datenverkehr) als zugestellt (ausgehender Datenverkehr). Wenn mehr eingehender Datenverkehr als ausgehender Datenverkehr vorhanden ist, kann Ihr Abonnement einen großen Nachrichtenrückstand ansammeln, was zu einer hohen Nachrichtenzustellungslatenz führt. Sie können eine Erhöhung des Kontingents auf der Grundlage der Veröffentlichungsrate anfordern, um negative Auswirkungen auf Ihren Workload zu vermeiden.

Über Firehose-Bereitstellungsdatenströme können Sie Amazon SNS-Benachrichtigungen an Amazon Simple Storage Service (Amazon S3), Amazon Redshift, Amazon OpenSearch Service (OpenSearch Service) und an Drittanbieter wie Datadog, New Relic, MongoDB und Splunk verteilen.

Sie können diese Funktion beispielsweise verwenden, um Nachrichten, die an ein Thema gesendet wurden, dauerhaft in einem Amazon S3 Bucket zu Compliance, Archivierung oder anderen Zwecken zu speichern. Erstellen Sie dazu einen Firehose-Bereitstellungsdatenstrom mit einem S3-Bucket-Ziel und abonnieren Sie diesen Bereitstellungsdatenstrom für das Amazon SNS-Thema. Um beispielsweise eine Analyse von Nachrichten durchzuführen, die an ein Amazon SNS-Thema gesendet werden, erstellen Sie einen Bereitstellungsdatenstrom mit einem - OpenSearch Service-Indexziel. Sie können dann den Firehose-Bereitstellungsdatenstrom für das Amazon SNS-Thema abonnieren.

Amazon SNS unterstützt auch die Protokollierung des Nachrichtenzustellungsstatus für Benachrichtigungen, die an Firehose-Endpunkte gesendet werden. Weitere Informationen finden Sie unter Zustellungsstatus von Amazon SNS Nachrichtenübermittmittungen.

Themen

- Voraussetzungen für das Abonnieren von Firehose-Bereitstellungsdatenströmen für Amazon SNS-Themen
- Abonnieren eines Firehose-Bereitstellungsdatenstroms für ein Amazon SNS-Thema
- Arbeiten mit Bereitstellungsdatenstrom-Zielen
- Anwendungsbeispiel für Nachrichtenarchivierung und -analyse

Voraussetzungen für das Abonnieren von Firehose-Bereitstellungsdatenströmen für Amazon SNS-Themen

Um einen Amazon-Data-Firehose-Bereitstellungsdatenstrom für ein SNS-Thema zu abonnieren, AWS-Konto muss Ihr über Folgendes verfügen:

- Ein Standard–SNS-Thema. Weitere Informationen finden Sie unter Erstellen Sie ein Amazon SNS-Thema.
- Ein Firehose-Bereitstellungs-Stream. Weitere Informationen finden Sie unter Erstellen eines Amazon-Data-Firehose-Bereitstellungsdatenstroms und Gewähren von Anwendungszugriff auf Ihre Firehose-Ressourcen im Amazon-Data-Firehose-Entwicklerhandbuch.
- Eine AWS Identity and Access Management (IAM) -Rolle, die dem Amazon-SNS-Dienstprinzipal vertraut und über die Berechtigung zum Schreiben in den Bereitstellungsdatenstrom verfügt. Geben Sie als den Amazon-Ressourcennamen (ARN) dieser Rolle als`SubscriptionRoleARN`Wenn Sie das Abonnement erstellen. Amazon SNS übernimmt

diese Rolle, die es Amazon SNS ermöglicht, Datensätze im Firehose-Bereitstellungsdatenstrom abzulegen.

Hier ein Beispiel für eine Berechtigungsrichtlinie.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "firehose:DescribeDeliveryStream",
        "firehose:ListDeliveryStreams",
        "firehose:ListTagsForDeliveryStream",
        "firehose:PutRecord",
        "firehose:PutRecordBatch"
      ],
      "Resource": [
        "arn:aws:firehose:us-east-1:111111111111:deliverystream/firehose-sns-
delivery-stream"
      ],
      "Effect": "Allow"
    }
  ]
}
```

Um volle Berechtigungen für die Verwendung von Firehose zu erteilen, können Sie auch die verwaltete AWS Richtlinie verwenden`AmazonKinesisFirehoseFullAccess`. Um strengere Berechtigungen für die Verwendung von Firehose bereitzustellen, können Sie auch Ihre eigene Richtlinie erstellen. Die Richtlinie muss mindestens die Berechtigung zum Ausführen des `PutRecord`-Vorgangs auf einem bestimmten Bereitstellungsdatenstrom haben.

In allen Fällen müssen Sie auch die Vertrauensstellung bearbeiten, um den Amazon SNS-Serviceprinzipal einzubeziehen. Beispielsweise:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "sns.amazonaws.com"
```

```
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

Weitere Informationen zum Erstellen einer Rolle finden Sie unter Creating a role to delegate permissions to an AWS service (Erstellen einer Rolle zum Delegieren von Berechtigungen an einen AWS-Service) im IAM-Benutzerhandbuch.

Nachdem Sie diese Anforderungen erfüllt haben, können Sie Abonnieren des Bereitstellungsdatenströme für das SNS-Thema auswählen.

Abonnieren eines Firehose-Bereitstellungsdatenstroms für ein Amazon SNS-Thema

Um Amazon SNS-Benachrichtigungen an Amazon-Data-Firehose-Bereitstellungsdatenströme zu senden, stellen Sie zunächst sicher, dass Sie alle Voraussetzungen erfüllt haben. Eine Liste der unterstützten Endpunkte finden Sie unter Endpunkte und Kontingente von Amazon Data Firehose im Allgemeine Amazon Web Services-Referenz.

So abonnieren Sie einen Firehose-Bereitstellungsdatenstrom für ein Thema

1. Melden Sie sich bei der Amazon SNS-Konsole an.
2. Wählen Sie im Navigationsbereich Subscriptions aus.
3. Wählen Sie auf der Seite Subscriptions (Abonnements) die Option Create subscription (Abonnement erstellen)aus.
4. Führen Sie auf der Seite Create subscription (Abonnement erstellen) im Abschnitt Details die folgenden Schritte aus:
   a. Für Thema ARN wählen Sie den Amazon-Ressourcennamen (ARN) eines Standard-Themas aus.
   b. Wählen Sie für Protokoll die Option Firehose aus.
   c. Wählen Sie für Endpunkt den ARN eines Firehose-Bereitstellungs-Streams aus, der Benachrichtigungen von Amazon SNS empfangen kann.
   d. Geben Sie für Abonnementrollen-ARN den ARN der AWS Identity and Access Management (IAM)-Rolle an, die Sie zum Schreiben in Firehose-Bereitstellungsdatenströme erstellt

haben. Weitere Informationen finden Sie unter Voraussetzungen für das Abonnieren von Firehose-Bereitstellungsdatenströmen für Amazon SNS-Themen.

e. (Optional) Um Amazon SNS-Metadaten aus veröffentlichten Nachrichten zu entfernen, wählen Sie Aktivieren der Übermittlung unformatierter Nachrichten aus. Weitere Informationen finden Sie unter Übermittlung unformatierter Nachrichten Amazon SNS.

5. (Optional) Um eine Filterrichtlinie zu konfigurieren, erweitern Sie den Abschnitt Abonnement-Filterrichtlinie. Weitere Informationen finden Sie unter Filterrichtlinien für Amazon-SNS-Abonnements.
6. (Optional) Um eine Warteschlange für unzustellbare Nachrichten für das Abonnement zu konfigurieren, erweitern Sie den Abschnitt Redrive-Richtlinie (Warteschlange für unzustellbare Nachrichten). Weitere Informationen finden Sie unter Amazon SNS Queues für unzustellbare Nachrichten (DLQs).
7. Wählen Sie Create subscription (Abonnement erstellen) aus.

Die Konsole erstellt das Abonnement und öffnet die Seite Details des Abonnements.

Arbeiten mit Bereitstellungsdatenstrom-Zielen

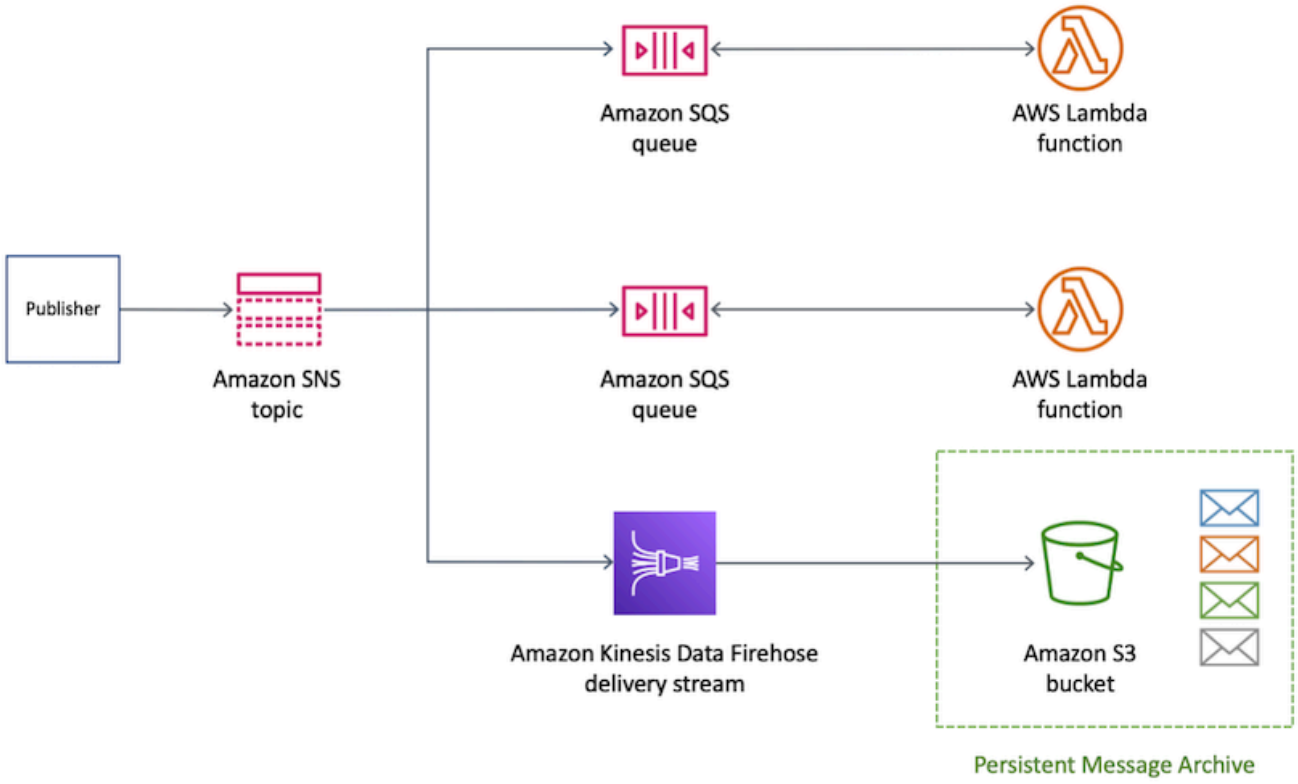
Über Amazon-Data-Firehose-Bereitstellungsdatenströme können Sie Nachrichten an zusätzliche Endpunkte senden. In diesem Abschnitt wird beschrieben, wie Sie mit unterstützten Zielen arbeiten.

Themen

- Amazon-S3-Ziele
- OpenSearch Serviceziele
- Ziel von Amazon Redshift
- HTTP-Ziele

Amazon-S3-Ziele

Dieser Abschnitt enthält Informationen zu Amazon-Data-Firehose-Bereitstellungsdatenströmen, die Daten in Amazon Simple Storage Service (Amazon S3) veröffentlichen.



Themen

- Archiviertes Nachrichtenformat für Amazon S3-Ziele
- Analysieren von Nachrichten für Amazon-S3-Ziele

Archiviertes Nachrichtenformat für Amazon S3-Ziele

Das folgende Beispiel zeigt eine Amazon-SNS-Benachrichtigung, die an einen Amazon Simple Storage Service (Amazon S3) Bucket gesendet wurde, wobei ein Einzug zur Lesbarkeit verwendet wird.

> **Note**
>
> In diesem Beispiel ist die unformatierte Nachrichtenzustellung für die veröffentlichte Nachricht deaktiviert. Wenn die Zustellung von unformatierten Nachrichten deaktiviert ist, fügt Amazon SNS der Nachricht JSON-Metadaten hinzu, einschließlich der folgenden Eigenschaften:
>
> - `Type`
> - `MessageId`

- TopicArn
- Subject
- Timestamp
- UnsubscribeURL
- MessageAttributes

Weitere Informationen zu „Raw Message Delivery“ finden Sie unter Übermittlung unformatierter Nachrichten Amazon SNS.

```
{
    "Type": "Notification",
    "MessageId": "719a6bbf-f51b-5320-920f-3385b5e9aa56",
    "TopicArn": "arn:aws:sns:us-east-1:333333333333:my-kinesis-test-topic",
    "Subject": "My 1st subject",
    "Message": "My 1st body",
    "Timestamp": "2020-11-26T23:48:02.032Z",
    "UnsubscribeURL": "https://sns.us-east-1.amazonaws.com/?
Action=Unsubscribe&SubscriptionArn=arn:aws:sns:us-east-1:333333333333:my-kinesis-test-
topic:0b410f3c-ee5e-49d8-b59b-3b4aa6d8fcf5",
    "MessageAttributes": {
        "myKey1": {
            "Type": "String",
            "Value": "myValue1"
        },
        "myKey2": {
            "Type": "String",
            "Value": "myValue2"
        }
    }
 }
```

Das folgende Beispiel zeigt drei SNS-Nachrichten, die über einen Amazon-Data-Firehose-Bereitstellungsdatenstrom an denselben Amazon S3-Bucket gesendet werden. Die Pufferung wird berücksichtigt, und Zeilenumbrüche trennen die Nachrichten.

```
{"Type":"Notification","MessageId":"d7d2513e-6126-5d77-
bbe2-09042bd0a03a","TopicArn":"arn:aws:sns:us-east-1:333333333333:my-
kinesis-test-topic","Subject":"My 1st subject","Message":"My 1st
```

```
 body","Timestamp":"2020-11-27T00:30:46.100Z","UnsubscribeURL":"https://
sns.us-east-1.amazonaws.com/?Action=Unsubscribe&SubscriptionArn=arn:aws:sns:us-
east-1:313276652360:my-kinesis-test-topic:0b410f3c-ee5e-49d8-
b59b-3b4aa6d8fcf5","MessageAttributes":{"myKey1":
{"Type":"String","Value":"myValue1"},"myKey2":{"Type":"String","Value":"myValue2"}}}
{"Type":"Notification","MessageId":"0c0696ab-7733-5bfb-b6db-
ce913c294d56","TopicArn":"arn:aws:sns:us-east-1:333333333333:my-
kinesis-test-topic","Subject":"My 2nd subject","Message":"My 2nd
 body","Timestamp":"2020-11-27T00:31:22.151Z","UnsubscribeURL":"https://
sns.us-east-1.amazonaws.com/?Action=Unsubscribe&SubscriptionArn=arn:aws:sns:us-
east-1:313276652360:my-kinesis-test-topic:0b410f3c-ee5e-49d8-
b59b-3b4aa6d8fcf5","MessageAttributes":{"myKey1":{"Type":"String","Value":"myValue1"}}}
{"Type":"Notification","MessageId":"816cd54d-8cfa-58ad-91c9-8d77c7d173aa","TopicArn":"arn:aws:s
east-1:333333333333:my-kinesis-test-topic","Subject":"My 3rd subject","Message":"My
 3rd body","Timestamp":"2020-11-27T00:31:39.755Z","UnsubscribeURL":"https://
sns.us-east-1.amazonaws.com/?Action=Unsubscribe&SubscriptionArn=arn:aws:sns:us-
east-1:313276652360:my-kinesis-test-topic:0b410f3c-ee5e-49d8-b59b-3b4aa6d8fcf5"}
```


Analysieren von Nachrichten für Amazon-S3-Ziele

Auf dieser Seite wird beschrieben, wie Amazon SNS-Nachrichten analysiert werden, die über Amazon-Data-Firehose-Bereitstellungsdatenströme an Amazon Simple Storage Service (Amazon S3)-Ziele gesendet werden.

So analysieren Sie SNS-Nachrichten, die über Firehose-Bereitstellungsdatenströme an Amazon S3-Ziele gesendet werden

1. Konfigurieren Sie Ihre Amazon-S3-Ressourcen. Anweisungen finden Sie unter Erstellen eines Buckets im Benutzerhandbuch zu Amazon Simple Storage Service und Arbeiten mit Amazon-S3-Buckets im Benutzerhandbuch zu Amazon Simple Storage Service.
2. Konfigurieren Sie Ihren Bereitstellungsdatenstrom. Anweisungen finden Sie unter Wählen Sie Amazon S3 als Ihr Ziel im Amazon-Data-Firehose-Entwicklerhandbuch.
3. Verwenden von Amazon Athena zum Abfragen der Amazon-S3-Objekte unter Verwendung von Standard-SQL. Weitere Informationen finden Sie unter Erste Schritte im Benutzerhandbuch zu Amazon Athena.

Beispielabfrage

Nehmen Sie für dieses Beispiel Folgendes an:

- Nachrichten werden in der `notifications`Tabelle im `default` Schema gespeichert.

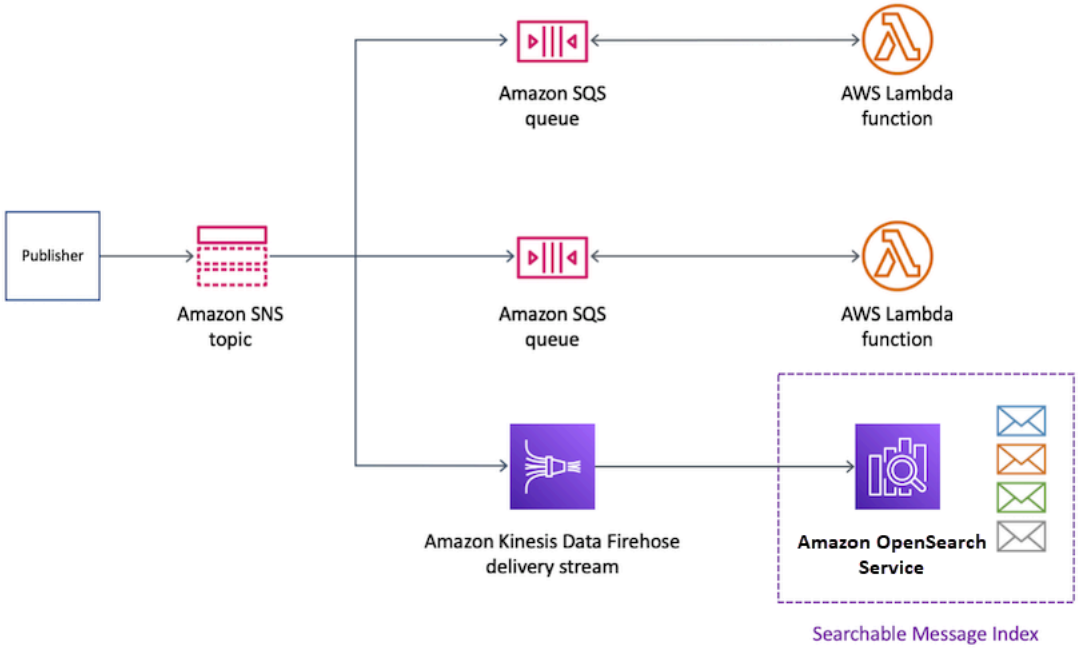
- Die `notifications` Tabelle enthält eine `timestamp` Spalte mit dem Typ `string`.

Die folgende Abfrage gibt alle SNS-Nachrichten zurück, die im angegebenen Datumsbereich empfangen wurden:

```
SELECT *
FROM default.notifications
WHERE from_iso8601_timestamp(timestamp) BETWEEN TIMESTAMP '2020-12-01 00:00:00' AND
 TIMESTAMP '2020-12-02 00:00:00';
```

OpenSearch Serviceziele

Dieser Abschnitt enthält Informationen zu Amazon-Data-Firehose-Bereitstellungsdatenströmen, die Daten in Amazon OpenSearch Service (OpenSearch Service) veröffentlichen.



Themen

- Archiviertes Nachrichtenformat in OpenSearch-Service-Indizes
- Analysieren von Nachrichten für OpenSearch Serviceziele

Archiviertes Nachrichtenformat in OpenSearch-Service-Indizes

Das folgende Beispiel zeigt eine Amazon-SNS-Benachrichtigung, die an einen Amazon-OpenSearch-Service(OpenSearch Service)-Index mit dem Namen `my-index` gesendet wurde. Dieser Index

verfügt über ein Zeitfilterfeld auf der Registerkarte `Timestamp`. Die SNS-Benachrichtigung wird in der `_source`-Eigenschaft der Nutzlast angezeigt.

> **Note**
>
> In diesem Beispiel ist die unformatierte Nachrichtenzustellung für die veröffentlichte Nachricht deaktiviert. Wenn die Zustellung von unformatierten Nachrichten deaktiviert ist, fügt Amazon SNS der Nachricht JSON-Metadaten hinzu, einschließlich der folgenden Eigenschaften:
>
> - `Type`
> - `MessageId`
> - `TopicArn`
> - `Subject`
> - `Timestamp`
> - `UnsubscribeURL`
> - `MessageAttributes`
>
> Weitere Informationen zu „Raw Message Delivery“ finden Sie unter Übermittlung unformatierter Nachrichten Amazon SNS.

```
{
  "_index": "my-index",
  "_type": "_doc",
  "_id": "4961310096311132320325040540219328379477388655098593280 2.0",
  "_version": 1,
  "_score": null,
  "_source": {
    "Type": "Notification",
    "MessageId": "bf32e294-46e3-5dd5-a6b3-bad65162e136",
    "TopicArn": "arn:aws:sns:us-east-1:111111111111:my-topic",
    "Subject": "Sample subject",
    "Message": "Sample message",
    "Timestamp": "2020-12-02T22:29:21.189Z",
    "UnsubscribeURL": "https://sns.us-east-1.amazonaws.com/?
Action=Unsubscribe&SubscriptionArn=arn:aws:sns:us-east-1:111111111111:my-
topic:b5aa9bc1-9c3d-452b-b402-aca2cefc63c9",
    "MessageAttributes": {
```

```
      "my_attribute": {
        "Type": "String",
        "Value": "my_value"
      }
    }
  },
  "fields": {
    "Timestamp": [
      "2020-12-02T22:29:21.189Z"
    ]
  },
  "sort": [
    1606948161189
  ]
}
```

Analysieren von Nachrichten für OpenSearch Serviceziele

Auf dieser Seite wird beschrieben, wie Amazon SNS-Nachrichten analysiert werden, die über Amazon-Data-Firehose-Bereitstellungsdatenströme an Amazon OpenSearch -Service-(OpenSearch Service)-Ziele gesendet werden.

So analysieren Sie SNS-Nachrichten, die über Firehose-Bereitstellungsdatenströme an OpenSearch Serviceziele gesendet werden

1. Konfigurieren Sie Ihre OpenSearch Service-Ressourcen. Anweisungen finden Sie unter Erste Schritte mit Amazon OpenSearch Service im Amazon- OpenSearch Service-Entwicklerhandbuch.
2. Konfigurieren Sie Ihren Bereitstellungsdatenstrom. Anweisungen finden Sie unter Wählen Sie OpenSearch Service für Ihr Ziel im Amazon-Data-Firehose-Entwicklerhandbuch.
3. Führen Sie eine Abfrage mit OpenSearch Serviceabfragen und Kibana aus. Weitere Informationen finden Sie unter Schritt 3: Suchen von Dokumenten in einer - OpenSearch Service-Domain und Kibana im Amazon- OpenSearch Service-Entwicklerhandbuch.

Beispielabfrage

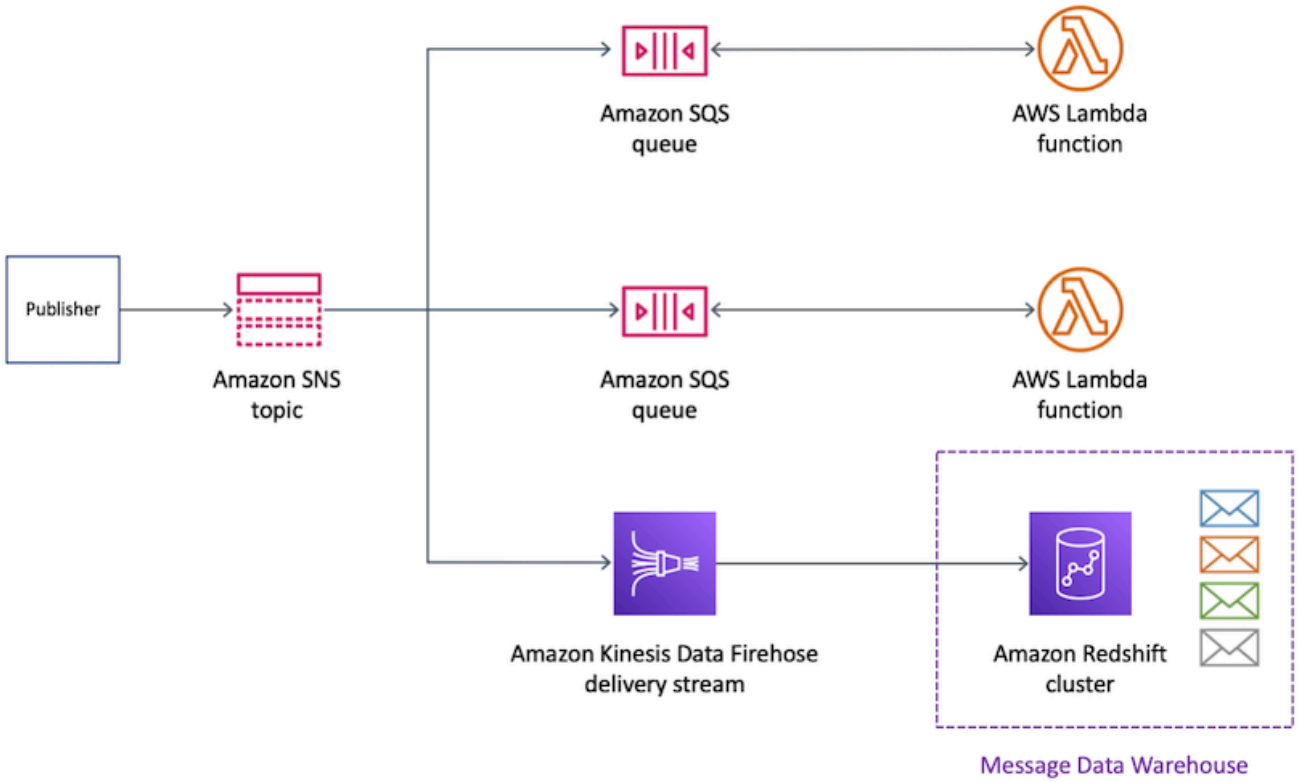
Im folgenden Beispiel werden die `my-index` Indizes für alle SNS-Nachrichten abgeragt, die im angegebenen Datumsbereich empfangen werden:

```
POST https://search-my-domain.us-east-1.es.amazonaws.com/my-index/_search
```

```
{
  "query": {
    "bool": {
      "filter": [
        {
          "range": {
            "Timestamp": {
              "gte": "2020-12-08T00:00:00.000Z",
              "lte": "2020-12-09T00:00:00.000Z",
              "format": "strict_date_optional_time"
            }
          }
        }
      ]
    }
  }
}
```

Ziel von Amazon Redshift

In diesem Abschnitt wird beschrieben, wie Sie Amazon SNS-Benachrichtigungen an einen Amazon-Data-Firehose-Bereitstellungsdatenstrom verteilen, der Daten in Amazon Redshift veröffentlicht. Mit dieser Konfiguration können Sie eine Verbindung zur Amazon-Redshift-Datenbank herstellen und ein SQL-Abfragetool verwenden, um aus der Datenbank die Amazon-SNS-Nachrichten abzufragen, die bestimmte Kriterien erfüllen.



Themen

- Archiv-Tabellenstruktur für Amazon-Redshift-Ziele
- Analysieren von Nachrichten für Amazon Redshift

Archiv-Tabellenstruktur für Amazon-Redshift-Ziele

Für Amazon-Redshift-Endpunkte werden veröffentlichte Amazon-SNS-Nachrichten als Zeilen in einer Tabelle archiviert. Im Folgenden wird ein Beispiel gezeigt.

> **Note**
>
> In diesem Beispiel ist die unformatierte Nachrichtenzustellung für die veröffentlichte Nachricht deaktiviert. Wenn die Zustellung von unformatierten Nachrichten deaktiviert ist, fügt Amazon SNS der Nachricht JSON-Metadaten hinzu, einschließlich der folgenden Eigenschaften:
>
> - `Type`
> - `MessageId`
> - `TopicArn`

- `Subject`
- `Message`
- `Timestamp`
- `UnsubscribeURL`
- `MessageAttributes`

Weitere Informationen zu „Raw Message Delivery“ finden Sie unter Übermittlung unformatierter Nachrichten Amazon SNS.
Obwohl Amazon SNS der Nachricht mithilfe der in dieser Liste angezeigten Groß-/Kleinschreibung Eigenschaften hinzufügt, werden Spaltennamen in Amazon-Redshift-Tabellen in Kleinbuchstaben angezeigt. Um die JSON-Metadaten für den Amazon-Redshift-Endpunkt zu transformieren, können Sie den SQL `COPY`-Befehl nutzen. Weitere Informationen finden Sie unter Beispiele für die COPY-Operation aus JSON und Laden von JSON-Daten unter Verwendung der Option „auto ignorecase“ im Entwicklerhandbuch für Amazon Redshift Datenbanken.

type	messageid	topicarn	subject	Nachricht	timestamp	unsubscribeurl	messageattributes
Benachrichtigung	ea544832-a0d8-581d-9275-108243c46103	arn:aws:sns:us-east-1:111111111111:my-topic	Beispiel-Betreff	Beispiel-Nachricht	2020-12-02T00:33:32.272Z	https://sns.us-east-1.amazonaws.com/?Action=Unsubscribe&SubscriptionArn=arn:aws:sns:us-east-1:111111111111:m	{\"my_attribute\":{\"Type\":\"String\",\"Value\":\"my_value\"}}

type	messageid	topicarn	subject	Nachricht	timestamp	unsubscribeurl	messageattributes
						y-topic:326deeeb-cbf4-45da-b92b-ca77a247813b	
Benachrichtigung	ab124832-a0d8-581d-9275-108243c46114	arn:aws:sns:us-east-1:111111111111:my-topic	Beispiel-Betreff 2	Beispielnachricht 2	2020-12-03T00:18:11.129Z	https://sns.us-east-1.amazonaws.com/?Action=Unsubscribe&SubscriptionArn=arn:aws:sns:us-east-1:111111111111:my-topic:326deeeb-cbf4-45da-b92b-ca77a247813b	{\"my_attribute2\":{\"Type\":\"String\",\"Value\":\"my_value\"}}

type	messageid	topicarn	subject	Nachricht	timestamp	unsubscribeurl	messageattributes
Benachrichtigung	ce644832-a0d8-581d-9275-108243c46125	arn:aws:sns:us-east-1:111111111111:my-topic	Beispiel-Betreff 3	Beispielnachricht 3	2020-12-09T00:08:44.405Z	https://sns.us-east-1.amazonaws.com/?Action=Unsubscribe&SubscriptionArn=arn:aws:sns:us-east-1:111111111111:my-topic:326deeeb-cbf4-45da-b92b-ca77a247813b	{\"my_attribute3\":{\"Type\":\"String\",\"Value\":\"my_value\"}}

Weitere Informationen zum Verteilen von Benachrichtigungen an Amazon-Redshift-Endpunkte finden Sie unter Ziel von Amazon Redshift.

Analysieren von Nachrichten für Amazon Redshift

Auf dieser Seite wird beschrieben, wie Amazon SNS-Nachrichten analysiert werden, die über Amazon-Data-Firehose-Bereitstellungsdatenströme an Amazon-Redshift-Ziele gesendet werden.

So analysieren Sie SNS-Nachrichten, die über Firehose-Bereitstellungsdatenströme an Amazon-Redshift-Ziele gesendet werden

1. Konfigurieren Sie Ihre Amazon-Redshift-Ressourcen. Detaillierte Anweisungen finden Sie unter Erste Schritte mit Amazon Redshift im Handbuch für erste Schritte mit Amazon Redshift.
2. Konfigurieren Sie Ihren Bereitstellungsdatenstrom. Anweisungen finden Sie unter Auswählen von Amazon Redshift für Ihr Ziel im Amazon-Data-Firehose-Entwicklerhandbuch.
3. Ausführen einer Abfrage Weitere Informationen finden Sie unter Abfragen einer Datenbank mit dem Abfrage-Editor im Amazon-Redshift-Verwaltungshandbuch.

Beispielabfrage

Nehmen Sie für dieses Beispiel Folgendes an:

- Nachrichten werden in der `notifications`-Tabelle im Standardschema `public` gespeichert.
- Die `Timestamp`-Eigenschaft aus der SNS-Nachricht ist in der Spalte der Tabelle `timestamp` mit dem Spaltendatentyp `timestamptz`gespeichert.

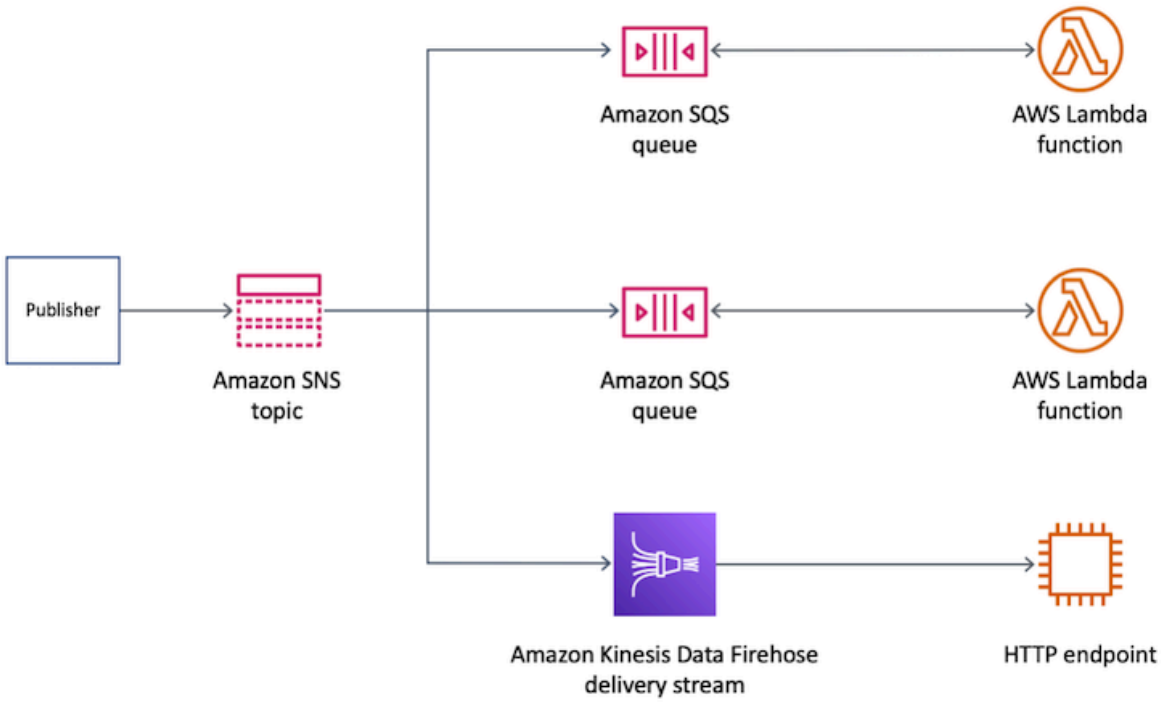
> **Note**
>
> Um die JSON-Metadaten für den Amazon-Redshift-Endpunkt zu transformieren, können Sie den SQL `COPY`-Befehl nutzen. Weitere Informationen finden Sie unter Beispiele für die COPY-Operation aus JSON und Laden von JSON-Daten unter Verwendung der Option „auto ignorecase“ im Entwicklerhandbuch für Amazon Redshift Datenbanken.

Die folgende Abfrage gibt alle SNS-Nachrichten zurück, die im angegebenen Datumsbereich empfangen wurden:

```
SELECT *
FROM public.notifications
WHERE timestamp > '2020-12-01T09:00:00.000Z' AND timestamp <
 '2020-12-02T09:00:00.000Z';
```

HTTP-Ziele

Dieser Abschnitt enthält Informationen zu Amazon-Data-Firehose-Bereitstellungsdatenströmen, die Daten auf HTTP-Endpunkten veröffentlichen.



Themen

- Format der übermittelten Nachricht für HTTP-Ziele

Format der übermittelten Nachricht für HTTP-Ziele

Im Folgenden finden Sie ein Beispiel für einen HTTP-POST-Anforderungstext von Amazon SNS, den ein Amazon-Data-Firehose-Bereitstellungsdatenstrom an den HTTP-Endpunkt senden kann. Die SNS-Benachrichtigung ist als Base64-Nutzlast in der Eigenschaft `records` kodiert.

> **Note**
>
> In diesem Beispiel ist die unformatierte Nachrichtenzustellung für die veröffentlichte Nachricht deaktiviert. Weitere Informationen zu „Raw Message Delivery“ finden Sie unter Übermittlung unformatierter Nachrichten Amazon SNS.

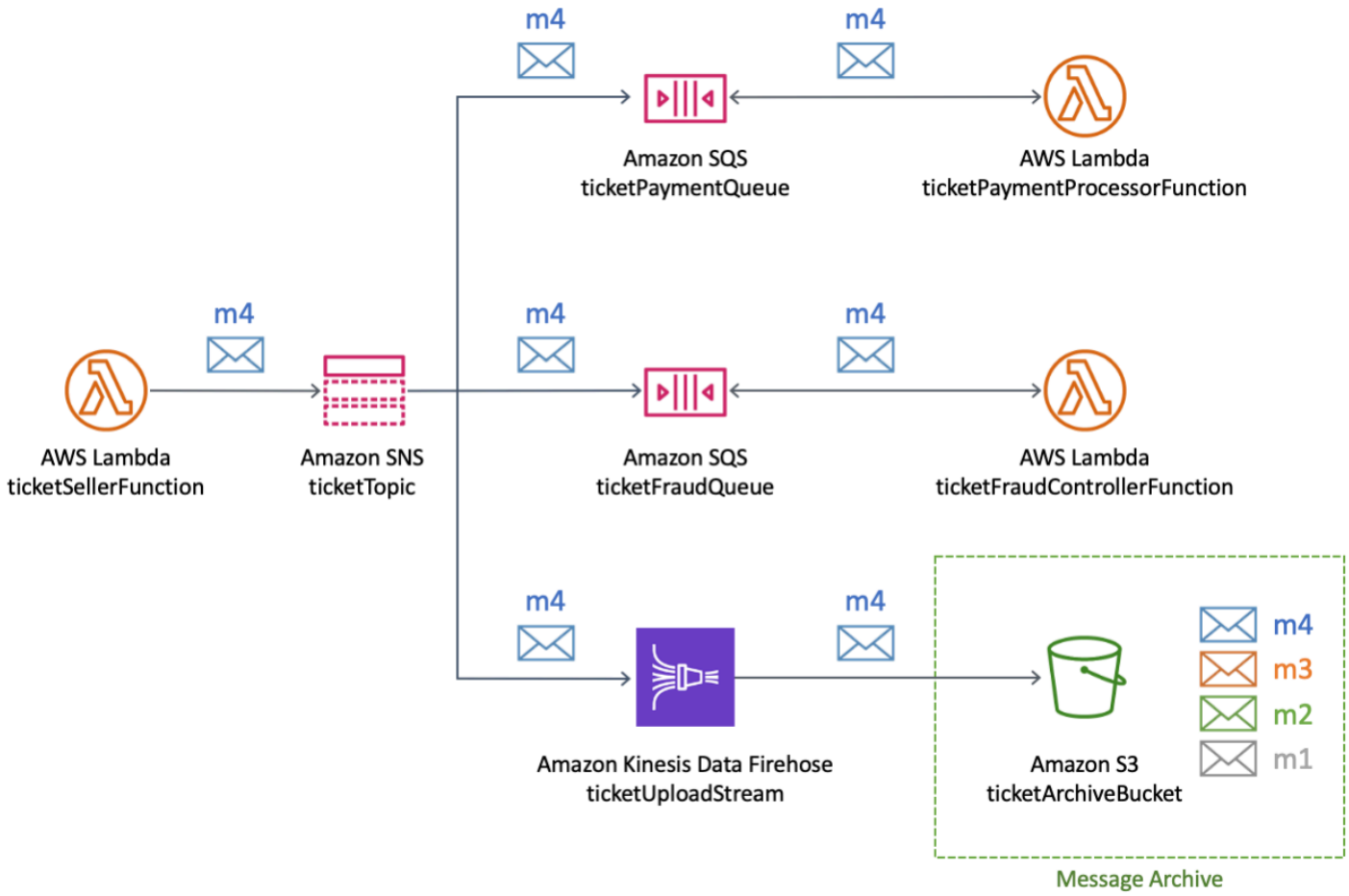
```
"body": {
    "requestId": "ebc9e8b2-fce3-4aef-a8f1-71698bf8175f",
    "timestamp": 1606255960435,
```

```
    "records": [
      {
        "data":
"eyJUeXBlIjoiTm90aWZpY2F0aW9uIiwiTWVzc2FnZUlkIjoiMjFkMmUzOGQtMmNhYi01ZjYxLTliYTItYmJiYWFhYzg0M
      }
    ]
  }
```

Anwendungsbeispiel für Nachrichtenarchivierung und -analyse

Dieser Abschnitt enthält ein Tutorial zu einem allgemeinen Anwendungsfall für die Archivierung und Analyse von Amazon-SNS-Nachrichten.

Die Einstellung dieses Anwendungsfalles ist eine Flugticket-Plattform, die in einem regulierten Umfeld betrieben wird. Die Plattform unterliegt einem Compliance-Rahmen, der das Unternehmen verpflichtet, alle Ticketverkäufe mindestens fünf Jahre lang zu archivieren. Um das Compliance-Ziel bei der Datenaufbewahrung zu erreichen, abonniert das Unternehmen einen Amazon-Data-Firehose-Bereitstellungsdatenstrom für ein vorhandenes SNS-Thema. Das Ziel für den Bereitstellungsdatenstrom ist ein Amazon Simple Storage Service (Amazon S3) Bucket. Mit dieser Konfiguration werden alle im SNS-Thema veröffentlichten Ereignisse im Amazon S3 Bucket archiviert. Das folgende Diagramm illustriert die Architektur dieser Konfiguration.



Um Analysen durchzuführen und Einblicke in den Ticketverkauf zu erhalten, führt das Unternehmen SQL-Abfragen mithilfe von Amazon Athena aus. Zum Beispiel kann das Unternehmen abfragen, um mehr über die beliebtesten Reiseziele und die häufigsten Flyer zu erfahren.

Um AWS-Ressourcen für diesen Anwendungsfall zu erstellen, können Sie den AWS Management Console oder eine AWS CloudFormation-Vorlage verwenden.

Themen

- Erstellen der anfänglichen Ressourcen
- Erstellen des Firehose-Bereitstellungsdatenstroms
- Abonnieren des Firehose-Bereitstellungsdatenstroms für das Amazon SNS-Thema
- Testen und Abfragen der Konfiguration
- Bearbeiten einer AWS CloudFormation-Vorlage

Erstellen der anfänglichen Ressourcen

Auf dieser Seite wird beschrieben, wie Sie die folgenden Ressourcen für den Anwendungsfall für die Nachrichtenarchivierung und -analyse erstellen:

- Ein Amazon Simple Storage Service (Amazon S3)-Bucket
- Zwei Amazon-Simple-Queue-Service-(Amazon-SQS)-Warteschlangen
- Amazon-SNS-Thema
- Ein Amazon-SNS-Thema mit zwei Abonnements zu Amazon-SQS-Themen

So erstellen Sie die anfänglichen Ressourcen

1. Erstellung eines Amazon S3 Buckets

   a. Öffnen Sie die Amazon S3-Konsole.
   b. Wählen Sie Create Bucket (Bucket erstellen) aus.
   c. Geben Sie für Bucket name einen eindeutigen Namen ein. Behalten Sie die anderen Felder als Standardwerte bei.
   d. Wählen Sie Create Bucket (Bucket erstellen).

   Weitere Informationen über Amazon S3 Buckets finden Sie unter Erstellen eines Buckets im Handbuch für den Amazon Simple Storage Service und Arbeiten mit Amazon-S3-Buckets im Handbuch für den Amazon Simple Storage Service.
2. Erstellen Sie die beiden Amazon-SQS-Warteschlangen:

   a. Öffnen Sie die Amazon-SQS-Konsole.
   b. Wählen Sie Create queue (Warteschlange erstellen) aus.
   c. Wählen Sie unter Type (Typ) die Option Standard aus.
   d. Geben Sie unter Name **`ticketPaymentQueue`** ein.
   e. Unter Zugriffsrichtlinie, für Wählen Sie die Methode, wählen Sie Advanced aus.
   f. Fügen Sie unter Policy Document für JSON Folgendes ein:

```
{
  "Version": "2008-10-17",
  "Statement": [
    {
```

```
      "Effect": "Allow",
      "Principal": {
        "Service": "sns.amazonaws.com"
      },
      "Action": "sqs:SendMessage",
      "Resource": "*",
      "Condition": {
        "ArnEquals": {
          "aws:SourceArn": "arn:aws:sns:us-east-1:123456789012:ticketTopic"
        }
      }
    }
  ]
}
```

Ersetzen Sie in dieser Zugriffsrichtlinie die AWS-Konto Zahl (*123456789012*) mit Ihrer eigenen und ändern Sie die AWSRegion (*us-east-1*) entsprechend.

g. Wählen Sie Create queue (Warteschlange erstellen) aus.

h. Wiederholen Sie diese Schritte, um eine zweite SQS-Warteschlange namens zu erstellen **`ticketFraudQueue`**.

Weitere Informationen zum Erstellen von SQS-Warteschlangen finden Sie unter Erstellen einer Amazon-SQS-Warteschlange (Konsole) im Entwicklungshandbuch für Amazon-Simple-Warteschlangen.

3. Erstellen eines SNS-Themas

a. Öffnen Sie die Seite Themen der Amazon-SNS-Konsole.

b. Wählen Sie Create topic aus.

c. UnterDetails, für Typ, wählen Sie Standard aus.

d. Geben Sie unter Name **`ticketTopic`** ein.

e. Wählen Sie Create topic aus.

Weitere Informationen zum Erstellen von SNS-Themen finden Sie in Erstellen Sie ein Amazon SNS-Thema.

4. Abonnieren Sie die SQS-Warteschlangen in das SNS-Thema.

a. In der Amazon-SNS-Konsole, wählen Sie auf der Detailseite zu TicketTopic das ThemaErstellen eines Abonnements aus.
b. Unter Details, fürProtocol (Protokoll), wählen Sie Amazon SQS aus.
c. FürEndpunkt wählen Sie den Amazon-Ressourcennamen (ARN) der Warteschlange TicketPaymentQueue.
d. Klicken Sie auf Create subscription (Abonnement erstellen).
e. Wiederholen Sie diese Schritte, um ein zweites Abonnement mithilfe des ARN der Warteschlange TicketFraudQueue.

   Weitere Informationen zum Abonnieren von SNS-Themen finden Sie unter Abonnieren eines Amazon-SNS-Themas. Sie können SQS-Warteschlangen auch von der Amazon-SQS-Konsole aus in SNS-Themen abonnieren. Weitere Informationen finden Sie unter Tutorial: Abonnieren einer Amazon-SQS-Warteschlange zu einem Amazon-SNS-Thema (Konsole) im Amazon-Simple-Queue-Service-Entwicklerhandbuch.

Sie haben die anfänglichen Ressourcen für diesen Anwendungsfall erstellt. Geben Sie ein, um fortzufahren Erstellen des Firehose-Bereitstellungsdatenstroms.

Erstellen des Firehose-Bereitstellungsdatenstroms

Auf dieser Seite wird beschrieben, wie Sie den Amazon-Data-Firehose-Bereitstellungsdatenstrom für den Anwendungsfall für die Nachrichtenarchivierung und -analyse erstellen.

So erstellen Sie den Firehose-Bereitstellungs-Stream

1. Öffnen Sie die Amazon-Kinesis-Service-Konsole.
2. Wählen Sie Firehose und dann Bereitstellungsdatenstrom erstellen aus.
3. Klicken Sie auf der Seite Neuer Bereitstellungsdatenstrom fürName eines Bereitstellungsdatenstroms den Wert **`ticketUploadStream`** und klicken Sie danach auf Weiter.
4. Klicken Sie auf der Seite Process records (Datensätze verarbeiten) auf Weiter.
5. Führen Sie auf der Seite Choose a Destination (Ziel auswählen) folgende Schritte durch:

   a. Für Ziel, wählen Sie Amazon S3 aus.
   b. Unter S3-Ziel, für S3 bucket wählen Sie den S3-Bucket aus, den Sie anfänglich erstellthaben.

c. Wählen Sie Weiter aus.

6. Klicken Sie auf der Seite Konfigurieren der Einstellungen für S3-Pufferbedingungen wie folgt:

   - Für Puffer-Größe geben Sie **1** ein.
   - Für Puffer-Interval geben Sie **60** ein.

   Wenn Sie diese Werte für den Amazon S3-Puffer verwenden, können Sie die Konfiguration schnell testen. Der erste Pufferbedingung, die erfüllt wird, bewirkt, dass dem S3-Bucket die Daten bereitgestellt werden.

7. Klicken Sie auf der Seite Konfigurieren der Einstellungen für Berechtigungen, wählen Sie aus, ob Sie eine AWS Identity and Access Management-Rolle (IAM) mit den erforderlichen Berechtigungen automatisch zuweisen. Wählen Sie anschließend Weiter.
8. Klicken Sie auf der Seite Prüfen, wählen Sie Erstellen von Bereitstellungsdatenstrom aus.
9. Wählen Sie auf der Seite Kinesis-Data-Firehose-Bereitstellungsdatenströme den Bereitstellungsdatenstrom aus, den Sie gerade erstellt haben (ticketUploadStream). Notieren Sie auf der Registerkarte Details den Amazon-Ressourcennamen (ARN) des Datenstroms für später.

Weitere Informationen zum Erstellen von Bereitstellungsdatenströmen finden Sie unter Erstellen eines Amazon-Data-Firehose-Bereitstellungsdatenstroms im Amazon-Data-Firehose-Entwicklerhandbuch. Weitere Informationen zum Erstellen einer IAM-Rolle finden Sie unter Erstellen einer Rolle zum Delegieren von Berechtigungen an einen AWS-Service im IAM-Benutzerhandbuch.

Sie haben den Firehose-Bereitstellungsdatenstrom mit den erforderlichen Berechtigungen erstellt. Geben Sie Abonnieren des Firehose-Bereitstellungsdatenstroms für das Amazon SNS-Thema ein, um fortzufahren.

Abonnieren des Firehose-Bereitstellungsdatenstroms für das Amazon SNS-Thema

Auf dieser Seite wird beschrieben, wie Sie das Folgende für den Anwendungsfall für die Nachrichtenarchivierung und -analyse erstellen:

- Die AWS Identity and Access Management (IAM)-Rolle, die es dem Amazon SNS-Abonnement ermöglicht, Datensätze im Amazon-Data-Firehose-Bereitstellungsdatenstrom zu speichern
- Das Firehose-Bereitstellungs-Stream-Abonnement für das SNS-Thema

So erstellen Sie die IAM-Rolle für das Amazon-SNS-Abonnement

1. Öffnen Sie die Seite Roles (Rollen) in der IAM-Konsole.
2. Wählen Sie Create role (Rolle erstellen) aus.
3. Wählen Sie unter Select type of trusted entity (Typ der vertrauenswürdigen Entität wählen) die Option AWS service (Service) aus.
4. Für Auswahl eines Anwendungsfalls, wählen Sie SNS aus. Wählen Sie dann Next: Permissions.
5. Wählen Sie Next: Tags (Weiter: Tags) aus.
6. Klicken Sie auf Weiter: Prüfen.
7. Geben Sie auf der Seite Review (Überprüfen) für Role name (Rollenname) **`ticketUploadStreamSubscriptionRole`** ein. Wählen Sie dann Create Role.
8. Wenn die Rolle erstellt wurde, wählen Sie ihren Namen (ticketUploadStreamSubscriptionRole).
9. Wählen Sie auf der Seite Summary die Option Add inline policy aus.
10. Wählen Sie auf der Seite Create policy die Registerkarte JSON aus, kopieren Sie dann die folgende JSON-Richtlinie und fügen Sie sie in das Textfeld ein.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "firehose:DescribeDeliveryStream",
                "firehose:ListDeliveryStreams",
                "firehose:ListTagsForDeliveryStream",
                "firehose:PutRecord",
                "firehose:PutRecordBatch"
            ],
            "Resource": [
                "arn:aws:firehose:us-east-1:123456789012:deliverystream/
ticketUploadStream"
            ],
            "Effect": "Allow"
        }
    ]
}
```

Ersetzen Sie in dieser Richtlinie die AWS-KontoZahl (*123456789012*) mit Ihrer eigenen, und ändern Sie die AWSRegion (*us-east-1*) entsprechend.

11. Wählen Sie Richtlinie prüfen.
12. Geben Sie auf der Seite Create policy (Richtlinie erstellen) für Name **`FirehoseSnsPolicy`** ein. Wählen Sie dann Richtlinie erstellen aus.
13. Auf der Seite Übersicht der Rolle, notieren Sie die ARN der Rolle für später.

Weitere Informationen zum Erstellen einer IAM-Rolle finden Sie unter Erstellen einer Rolle zum Delegieren von Berechtigungen an einen AWS-Service im IAM-Benutzerhandbuch.

So abonnieren Sie den Firehose-Bereitstellungsdatenstrom für das SNS-Thema

1. Öffnen Sie die Seite Themen der Amazon-SNS-Konsole.
2. Wählen Sie auf der Seite Subscriptions (Abonnements) die Option Create subscription (Abonnement erstellen) aus.
3. Wählen Sie unter Details für Protokoll die Option Amazon Data Firehose aus.
4. Geben Sie für Endpunkt den Amazon-Ressourcennamen (ARN) des ticketUploadStream Bereitstellungsdatenstroms ein, den Sie zuvor erstellt haben. Geben Sie z. B. ei **`arn:aws:firehose:us-east-1:123456789012:deliverystream/ticketUploadStream`**.
5. Geben Sie für Abonnementrollen-ARN den ARN der ticketUploadStreamSubscriptionRole IAM-Rolle ein, die Sie zuvor erstellt haben. Geben Sie z. B. ei **`arn:aws:iam::123456789012:role/ticketUploadStreamSubscriptionRole`**.
6. Wählen Sie Aktivieren der Übermittlung unformatierter Nachrichten aus.
7. Wählen Sie Create subscription (Abonnement erstellen) aus.

Sie haben die IAM-Rolle und das SNS-Themenabonnement erstellt. Geben Sie Testen und Abfragen der Konfiguration ein, um fortzufahren.

Testen und Abfragen der Konfiguration

Auf dieser Seite wird beschrieben, wie Sie den Anwendungsfall für die Nachrichtenarchivierung und -analyse testenm indem Sie eine Nachricht im Amazon-SNS-Thema veröffentlichen. Die Anweisungen enthalten eine Beispielabfrage, die Sie ausführen und an Ihre eigenen Bedürfnisse anpassen können.

So testen Sie die Konfiguration

1. Öffnen Sie die Seite Themen der Amazon-SNS-Konsole.

2. Wählen Sie das Symbol **`ticketTopic`** Thema.
3. Wählen Sie Publish message (Nachricht veröffentlichen) aus.
4. Geben Sie auf der Seite Nachricht im Thema veröffentlichen für den Nachrichtentext Folgendes ein. Fügen Sie am Ende der Nachricht einen Zeilenumbruch hinzu.

```
{"BookingDate":"2020-12-15","BookingTime":"2020-12-15
 04:15:05","Destination":"Miami","FlyingFrom":"Vancouver","TicketNumber":"abcd1234"}
```

   Behalten Sie alle anderen Optionen als Standardwerte bei.
5. Wählen Sie Publish message (Nachricht veröffentlichen) aus.

   Weitere Informationen zum Veröffentlichen von Nachrichten finden Sie unter Amazon-SNS-Nachrichten veröffentlichen.
6. Öffnen Sie nach dem Übermittlungsstromintervall von 60 Sekunden das Dialogfeld Konsole für Amazon Simple Storage Service (Amazon S3) und wählen Sie den Amazon S3 Bucket aus, den SieInital erstellt haben.

   Die veröffentlichte Nachricht wird im Bucket angezeigt.

Abfragen der Daten

1. Öffnen Sie die Amazon-Athena-Konsole.
2. Ausführen einer Abfrage

   Angenommen, dass die `notifications`-Tabelle im `default`-Schema folgende Daten enthält:

```
{"BookingDate":"2020-12-15","BookingTime":"2020-12-15
 04:15:05","Destination":"Miami","FlyingFrom":"Vancouver","TicketNumber":"abcd1234"}
{"BookingDate":"2020-12-15","BookingTime":"2020-12-15
 11:30:15","Destination":"Miami","FlyingFrom":"Omaha","TicketNumber":"efgh5678"}
{"BookingDate":"2020-12-15","BookingTime":"2020-12-15
 3:30:10","Destination":"Miami","FlyingFrom":"NewYork","TicketNumber":"ijkl9012"}
{"BookingDate":"2020-12-15","BookingTime":"2020-12-15
 12:30:05","Destination":"Delhi","FlyingFrom":"Omaha","TicketNumber":"mnop3456"}
```

   Führen Sie die folgende Abfrage aus, um das oberste Ziel zu finden:

```
SELECT destination
```

```
FROM default.notifications
GROUP BY destination
ORDER BY count(*) desc
LIMIT 1;
```

Führen Sie zum Abfragen nach Tickets aus, die in einem bestimmten Datums- und Zeitbereich verkauft wurden, eine Abfrage wie die folgende aus:

```
SELECT *
FROM default.notifications
WHERE bookingtime
  BETWEEN TIMESTAMP '2020-12-15 10:00:00'
  AND TIMESTAMP '2020-12-15 12:00:00';
```

Sie können beide Beispielabfragen an Ihre eigenen Bedürfnisse anpassen. Weitere Informationen über die Verwendung von Athena zum Ausführen von Abfragen finden Sie unter Erste Schritte im Benutzerhandbuch zu Amazon Athena.

Bereinigen

Um zu vermeiden, dass nach dem Testen Nutzungsgebühren entstehen, löschen Sie die folgenden Ressourcen, die Sie während des Lernprogramms erstellt haben:

- Amazon SNS-Abonnement
- Amazon SNS-Thema
- Amazon-Simple-Queue-Service-(Amazon-SQS)-Warteschlangen
- Amazon S3-Bucket
- Bereitstellungsdatenstrom für Amazon Data Firehose
- AWS Identity and Access Management (IAM)-Rollen und -Richtlinien

Bearbeiten einer AWS CloudFormation-Vorlage

Um die Bereitstellung von Amazon SNS Anwendungsfall für die Nachrichtenarchivierung und -analyse zu automatisieren, können Sie die folgende YAML-Vorlage verwenden:

```
---
AWSTemplateFormatVersion: '2010-09-09'
Description: Template for creating an SNS archiving use case
```

```
Resources:
  ticketUploadStream:
    DependsOn:
    - ticketUploadStreamRolePolicy
    Type: AWS::KinesisFirehose::DeliveryStream
    Properties:
      S3DestinationConfiguration:
        BucketARN: !Sub 'arn:${AWS::Partition}:s3:::${ticketArchiveBucket}'
        BufferingHints:
          IntervalInSeconds: 60
          SizeInMBs: 1
        CompressionFormat: UNCOMPRESSED
        RoleARN: !GetAtt ticketUploadStreamRole.Arn
  ticketArchiveBucket:
    Type: AWS::S3::Bucket
  ticketTopic:
    Type: AWS::SNS::Topic
  ticketPaymentQueue:
    Type: AWS::SQS::Queue
  ticketFraudQueue:
    Type: AWS::SQS::Queue
  ticketQueuePolicy:
    Type: AWS::SQS::QueuePolicy
    Properties:
      PolicyDocument:
        Statement:
          Effect: Allow
          Principal:
            Service: sns.amazonaws.com
          Action:
            - sqs:SendMessage
          Resource: '*'
          Condition:
            ArnEquals:
              aws:SourceArn: !Ref ticketTopic
      Queues:
        - !Ref ticketPaymentQueue
        - !Ref ticketFraudQueue
  ticketUploadStreamSubscription:
    Type: AWS::SNS::Subscription
    Properties:
      TopicArn: !Ref ticketTopic
      Endpoint: !GetAtt ticketUploadStream.Arn
      Protocol: firehose
```

```
      SubscriptionRoleArn: !GetAtt ticketUploadStreamSubscriptionRole.Arn
  ticketPaymentQueueSubscription:
    Type: AWS::SNS::Subscription
    Properties:
      TopicArn: !Ref ticketTopic
      Endpoint: !GetAtt ticketPaymentQueue.Arn
      Protocol: sqs
  ticketFraudQueueSubscription:
    Type: AWS::SNS::Subscription
    Properties:
      TopicArn: !Ref ticketTopic
      Endpoint: !GetAtt ticketFraudQueue.Arn
      Protocol: sqs
  ticketUploadStreamRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: '2012-10-17'
        Statement:
        - Sid: ''
          Effect: Allow
          Principal:
            Service: firehose.amazonaws.com
          Action: sts:AssumeRole
  ticketUploadStreamRolePolicy:
    Type: AWS::IAM::Policy
    Properties:
      PolicyName: FirehoseticketUploadStreamRolePolicy
      PolicyDocument:
        Version: '2012-10-17'
        Statement:
        - Effect: Allow
          Action:
          - s3:AbortMultipartUpload
          - s3:GetBucketLocation
          - s3:GetObject
          - s3:ListBucket
          - s3:ListBucketMultipartUploads
          - s3:PutObject
          Resource:
          - !Sub 'arn:aws:s3:::${ticketArchiveBucket}'
          - !Sub 'arn:aws:s3:::${ticketArchiveBucket}/*'
      Roles:
      - !Ref ticketUploadStreamRole
```

```
  ticketUploadStreamSubscriptionRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: '2012-10-17'
        Statement:
        - Effect: Allow
          Principal:
            Service:
            - sns.amazonaws.com
          Action:
          - sts:AssumeRole
      Policies:
      - PolicyName: SNSKinesisFirehoseAccessPolicy
        PolicyDocument:
          Version: '2012-10-17'
          Statement:
          - Action:
            - firehose:DescribeDeliveryStream
            - firehose:ListDeliveryStreams
            - firehose:ListTagsForDeliveryStream
            - firehose:PutRecord
            - firehose:PutRecordBatch
            Effect: Allow
            Resource:
            - !GetAtt ticketUploadStream.Arn
```

Aufrufen von Lambda-Funktionen

Amazon-SNS-Funktionen und AWS Lambda sind integriert. Somit können Sie Lambda-Funktionen mit Amazon SNS-Benachrichtigungen aufrufen. Wenn eine Nachricht in einem SNS-Thema veröffentlicht wird, das eine Lambda-Funktion abonniert hat, wird die Lambda-Funktion mit der Nutzlast der veröffentlichten Nachricht aufgerufen. Die Lambda-Funktion erhält die Nutzlast der Nachricht als Eingabeparameter und kann die Informationen in der Nachricht bearbeiten, die Nachricht in anderen SNS-Themen veröffentlichen oder die Nachricht an andere AWS-Services senden.

Amazon SNS unterstützt auch die Attribute für den Nachrichtenübertragungsstatus für Benachrichtigungen, die an Lambda-Endpunkte gesendet wurden. Weitere Informationen finden Sie unter Zustellungsstatus von Amazon SNS Nachrichtenübermittmittungen.

Prerequisites

Zum Aufrufen von Lambda-Funktionen mithilfe von Amazon SNS-Benachrichtigungen, gehen Sie wie folgt vor:

- Lambda-Funktion
- Amazon SNS-Thema

Weitere Informationen zum Erstellen einer Lambda-Funktion zur Verwendung mit Amazon SNS finden Sie unter Verwenden von Lambda mit Amazon SNS. Informationen zum Erstellen eines Amazon SNS-Themas finden Sie unter Create a topic (Erstellen eines Themas).

Wenn Sie Amazon SNS zum Übermitteln von Nachrichten aus Opt-in-Regionen an standardmäßig aktivierte Regionen verwenden, müssen Sie die in der AWS Lambda-Funktion erstellte Richtlinie ändern, indem Sie den Prinzipal `sns.amazonaws.com` durch `sns.<opt-in-region>.amazonaws.com` ersetzen.

Wenn Sie beispielsweise eine Lambda-Funktion in USA Ost (Nord-Virginia) für ein SNS-Thema in Asien-Pazifik (Hongkong) abonnieren möchten, ändern Sie den Prinzipal in der AWS Lambda-Funktionsrichtlinie in `sns.ap-east-1.amazonaws.com`. Zu den Opt-in-Regionen gehören alle Regionen, die nach dem 20. März 2019 gestartet wurden, darunter Asien-Pazifik (Hongkong), Naher Osten (Bahrain), EU (Mailand) und Afrika (Kapstadt). Regionen, die vor dem 20. März 2019 gestartet wurden, sind standardmäßig aktiviert.

> **Note**
>
> AWS unterstützen keine regionenübergreifende Bereitstellung an AWS Lambda aus einer Region, die standardmäßig für eine Opt-In-Region aktiviert ist. Darüber hinaus wird die regionenübergreifende Weiterleitung von SNS-Nachrichten von Opt-in-Regionen an andere Opt-in-Regionen nicht unterstützt.

Abonnieren eines Themas mit einer Lambda-Funktion

1. Melden Sie sich bei der Amazon-SNS-Konsole an.
2. Wählen Sie im Navigationsbereich Topics (Themen) aus.
3. Wählen Sie auf der Seite Topics (Themen) ein Thema aus.

4. Wählen Sie im Abschnitt Subscriptions (Abonnements) die Option Create subscription (Abonnement erstellen) aus.
5. Führen Sie auf der Seite Create subscription (Abonnement erstellen) im Abschnitt Details die folgenden Schritte aus:

   a. Überprüfen Sie den gewählten Topic ARN (Themen-ARN).

   b. Wählen Sie unter Protocol (Protokoll) die Option AWS Lambda aus.

   c. Geben Sie für Endpoint (Endpunkt) den ARN einer Funktion ein.

   d. Wählen Sie Create subscription.

Wenn eine Nachricht in einem SNS-Thema veröffentlicht wird, das eine Lambda-Funktion abonniert hat, wird die Lambda-Funktion mit der Nutzlast der veröffentlichten Nachricht aufgerufen. Weitere Informationen zur Verwendung von AWS Lambda mit Amazon SNS, einschließlich eines Tutorials, finden Sie unter AWS Lambda mit Amazon SNS benutzen.

Verteilung an Amazon-SQS-Warteschlangen

Amazon SNS arbeitet eng mit Amazon-Simple-Warteschlangen-Service (Amazon SQS) zusammen. Beide Services bieten verschiedene Vorteile für Entwickler. Amazon SNS ermöglicht Anwendungen zeitkritische Nachrichten über einen Push-Mechanismus an mehrere Abonnenten versenden, sodass diese nicht mehr in regelmäßigen Abständen selbst nach Updates sehen müssen. Amazon SQS ist ein Nachrichten-Warteschlangenservice, der von verschiedenen Anwendungen verwendet wird, um Nachrichten über ein Polling-Modell auszutauschen. Mit dem Service kann das Senden und Empfangen von Komponenten entkoppelt werden, ohne dass jede Komponente gleichzeitig verfügbar sein muss. Mithilfe von Amazon SNS und Amazon SQS, können Nachrichten einerseits an Anwendungen übermittelt werden, für die eine direkte Benachrichtigung über Ereignisse erforderlich ist, und gleichzeitig für andere Anwendungen zur späteren Bearbeitung in eine Amazon-SQS-Warteschlange gestellt werden.

Wenn Sie eine Amazon-SQS-Warteschlange für ein Amazon-SNS-Thema abonnieren, können Sie eine Nachricht zum Thema veröffentlichen und Amazon SNS sendet eine Amazon-SQS-Nachricht an die abonnierte Warteschlange. Die Amazon-SQS-Nachricht enthält den Betreff und die Nachricht, die zum Thema veröffentlicht wurde, sowie Metadaten über die Nachricht in einem JSON-Dokument. Die Amazon-SQS-Nachricht sieht ähnlich aus wie das folgende JSON-Dokument.

```
{
```

```
   "Type" : "Notification",
   "MessageId" : "63a3f6b6-d533-4a47-aef9-fcf5cf758c76",
   "TopicArn" : "arn:aws:sns:us-west-2:123456789012:MyTopic",
   "Subject" : "Testing publish to subscribed queues",
   "Message" : "Hello world!",
   "Timestamp" : "2012-03-29T05:12:16.901Z",
   "SignatureVersion" : "1",
   "Signature" : "EXAMPLEnTrFPa3...",
   "SigningCertURL" : "https://sns.us-west-2.amazonaws.com/SimpleNotificationService-
f3ecfb7224c7233fe7bb5f59f96de52f.pem",
   "UnsubscribeURL" : "https://sns.us-west-2.amazonaws.com/?
Action=Unsubscribe&SubscriptionArn=arn:aws:sns:us-west-2:123456789012:MyTopic:c7fe3a54-
ab0e-4ec2-88e0-db410a0f2bee"
}
```

Abonnieren einer Amazon-SQS-Warteschlange für ein Amazon-SNS-Thema

Um ein Amazon-SNS-Thema zu aktivieren, damit es Nachrichten an eine Amazon-SQS-Warteschlange sendet, führen Sie einen der folgenden Schritte aus:

- Verwenden der Amazon-SQS-Konsole, was den Prozess vereinfacht. Weitere Informationen finden Sie unter Abbonieren einer Amazon-SQS-Warteschlange zu einem Amazon-SNS-Thema im Amazon-Simple-Warteschlangen-Service-Entwicklerhandbuch.
- Dazu gehen Sie wie folgt vor:
  1. Rufen Sie den Amazon-Ressourcennamen (ARN) der Warteschlange ab, an die Sie Nachrichten senden möchten, sowie das Thema, zu dem Sie die Warteschlange abonnieren möchten.
  2. Erteilen Sie die Berechtigung `sqs:SendMessage` für das Amazon-SNS-Thema, damit es Nachrichten an die Warteschlange senden kann.
  3. Abonnieren der Warteschlange für das Amazon-SNS-Thema.
  4. Gewähren Sie IAM-Benutzern oder AWS-Konten die entsprechenden Berechtigungen zur Veröffentlichung im Amazon-SNS-Thema und zum Lesen von Nachrichten in der Amazon-SQS-Warteschlange.
  5. Testen Sie die Ergebnisse, indem Sie eine Nachricht zu dem Thema veröffentlichen und die Nachricht von der Warteschlange lesen.

Weitere Informationen zur Einrichtung eines Themas zum Senden von Nachrichten an eine Warteschlange in einem anderen AWS-Konto finden Sie unter Senden von Amazon SNS-Nachrichten an eine Amazon SQS-Warteschlange in einem anderen Konto.

Eine AWS CloudFormation-Vorlage, die ein Thema erstellt, das Nachrichten an zwei Warteschlangen sendet, finden Sie unter Verwenden einer AWS CloudFormation-Vorlage zum Erstellen eines Themas, das Nachrichten Amazon-SQS-Warteschlanges sendet.

Schritt 1: Abrufen des ARN der Warteschlange und des Themas

Beim Abonnieren einer Warteschlange zu Ihrem Thema benötigen Sie eine Kopie des ARN für die Warteschlange. Wenn Sie eine Berechtigung für das Thema erteilen, um Nachrichten an die Warteschlange zu senden, benötigen Sie auch eine Kopie des ARN für das Thema.

Zum Abrufen des ARN der Warteschlange können Sie die Amazon-SQS-Konsole oder die API-Aktion GetQueueAttributes verwenden.

So rufen Sie den ARN der Warteschlange von der Amazon-SQS-Konsole ab

1. Melden Sie sich bei AWS Management Console an und öffnen Sie die Amazon-SQS-Konsole unter https://console.aws.amazon.com/sqs/.
2. Aktivieren Sie das Kontrollkästchen für die Warteschlange, deren ARN Sie abrufen möchten.
3. Kopieren Sie auf der Registerkarte Details den ARN-Wert, sodass Sie ihn zum Abonnieren des Amazon-SNS-Themas verwenden können.

Um den ARN des Themas abzurufen, können Sie die `sns-get-topic-attributes` Amazon-SNS-Konsole, den Befehl oder die `GetQueueAttributes` API-Aktion verwenden.

So rufen Sie den ARN des Themas von der Amazon-SNS-Konsole ab

1. Melden Sie sich bei der Amazon-SNS-Konsole an.
2. Wählen Sie im Navigationsbereich das Thema aus, dessen ARN Sie abrufen möchten.
3. Kopieren Sie im Bereich Details (Themendetails) den Wert unter Topic ARN (Themen-ARN), um damit die Berechtigung für das Amazon-SNS-Thema zum Senden von Nachrichten an die Warteschlange zu erteilen.

Schritt 2: Erteilen der Berechtigung für das Amazon SNS-Thema zum Senden von Nachrichten an die Amazon-SQS-Warteschlange

Damit ein Amazon-SNS-Thema Nachrichten an eine Warteschlange senden kann, müssen Sie eine Richtlinie für die Warteschlange einrichten, die es dem Amazon-SNS-Thema erlaubt, die Aktion `sqs:SendMessage` auszuführen.

Bevor Sie eine Warteschlange für ein Thema abonnieren, benötigen Sie ein Thema und eine Warteschlange. Wenn Sie noch kein Thema bzw. keine Warteschlange erstellt haben, erstellen Sie diese jetzt. Weitere Informationen finden Sie unter Erstellen eines Themas und Erstellen einer Warteschlange im Entwicklerhandbuch für den Amazon-Simple-Warteschlangenservice.

Zum Einrichten einer Richtlinie für eine Warteschlange können Sie die Amazon-SQS-Konsole oder die API-Aktion SetQueueAttributes verwenden. Bevor Sie beginnen, stellen Sie sicher, dass Sie über den ARN für das Thema verfügen, das Nachrichten an die Warteschlange senden soll. Wenn Sie eine Warteschlange für mehrere Themen abonnieren, muss Ihre Richtlinie ein `Statement`-Element für jedes Thema enthalten.

So richten Sie eine SendMessage-Richtlinie für eine Warteschlange mithilfe der Amazon-SQS-Konsole ein

1. Melden Sie sich bei AWS Management Console an und öffnen Sie die Amazon-SQS-Konsole unter https://console.aws.amazon.com/sqs/.
2. Aktivieren Sie das Kontrollkästchen für die Warteschlange, deren Richtlinien Sie einrichten möchten. Wählen Sie die Registerkarte Access policy und danach Edit.
3. In derZugriffsrichtliniedefinieren Sie, wer auf Ihre Warteschlange zugreifen kann.

   - Fügen Sie eine Bedingung hinzu, die die Aktion für das Thema zulässt.
   - Legen Sie `Principal` als den Amazon-SNS-Service fest, wie im folgenden Beispiel gezeigt.
   - Verwenden Sie den globale Konditionsschlüssel von `aws:SourceArn` oder den `aws:SourceAccount` zum Schutz vor dem Szenario Confused Deputy. Legen Sie zur Verwendung dieses Bedingungsschlüssels den Wert auf den ARN Ihres Themas fest. Wenn Ihre Warteschlange mehrere Themen abonniert hat, können Sie stattdessen `aws:SourceAccount` verwenden.

   Die Richtlinie erlaubt MyTopic, Nachrichten an MyQueue zu senden.

```
{
```

```
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "sns.amazonaws.com"
      },
      "Action": "sqs:SendMessage",
      "Resource": "arn:aws:sqs:us-east-2:123456789012:MyQueue",
      "Condition": {
        "ArnEquals": {
          "aws:SourceArn": "arn:aws:sns:us-east-2:123456789012:MyTopic"
        }
      }
    }
  ]
}
```

Schritt 3: Abonnieren der Warteschlange für das Amazon-SNS-Thema

Um Nachrichten über ein Thema an eine Warteschlange zu senden, müssen Sie die Warteschlange für das Amazon-SNS-Thema abonnieren. Sie geben die Warteschlange durch ihre ARN an. Um ein Thema zu abonnieren, können Sie die Amazon-SNS-Konsole, den `sns-subscribe`-CLI-Befehl oder die `Subscribe`-API-Aktion verwenden. Bevor Sie beginnen, stellen Sie sicher, dass Sie über den ARN für die Warteschlange verfügen, die Sie abonnieren möchten.

1. Melden Sie sich bei der Amazon SNS-Konsole an.
2. Wählen Sie im Navigationsbereich Topics (Themen) aus.
3. Wählen Sie auf der Seite Topics (Themen) ein Thema aus.
4. Wählen Sie auf der Seite ***MeinThema*** unter Subscriptions (Abonnements) die Option Create subscription (Abonnement erstellen) aus.
5. Führen Sie auf der Seite Create subscription (Abonnement erstellen) im Abschnitt Details die folgenden Schritte aus:

   a. Überprüfen Sie den Topic ARN (Themen-ARN).

   b. FürProtocol (Protokoll), wählen Sie Amazon SQSaus.

   c. Geben Sie für Endpoint (Endpunkt) den ARN einer Amazon-SQS-Warteschlange ein.

   d. Klicken Sie auf Create subscription (Abonnement erstellen).

Wenn das Abonnement bestätigt wurde, wird die Subscription ID (Abonnement-ID) für Ihr neues Abonnement angezeigt. Wenn der Eigentümer der Warteschlange das Abonnement erstellt, wird das Abonnement automatisch bestätigt und es sollte nahezu umgehend aktiv sein.

Normalerweise abonnieren Sie Ihre eigene Warteschlange für ein eigenes Thema in Ihrem Konto. Sie können aber auch eine Warteschlange aus einem anderen Konto für Ihr Thema abonnieren. Wenn der Benutzer, der das Abonnement erstellt, nicht der Eigentümer der Warteschlange ist (z. B. wenn ein Benutzer von Konto A eine Warteschlange von Konto B für ein Thema in Konto A erstellt), muss das Abonnement bestätigt werden. Weitere Informationen zum Abonnieren einer Warteschlange von einem anderen Konto und zum Bestätigen des Abonnements finden Sie unter Senden von Amazon SNS-Nachrichten an eine Amazon SQS-Warteschlange in einem anderen Konto.

Schritt 4: Erteilen von Berechtigungen an Benutzer für die entsprechenden Themen- und Warteschlangenaktionen

Sie sollten AWS Identity and Access Management (IAM) verwenden, damit es nur geeigneten Benutzern gestattet ist, Veröffentlichungen im Amazon-SNS-Thema vorzunehmen und Nachrichten von der Amazon-SQS-Warteschlange zu lesen bzw. zu löschen. Weitere Informationen zum Steuern von Aktionen für Themen und Warteschlangen für IAM-Benutzer finden Sie unter Verwenden von identitätsbasierten Richtlinien mit Amazon SNS, und Identity and Access Management in Amazon SQS im Amazon Simple Warteschlange Service-Entwicklerhandbuch.

Es gibt zwei Möglichkeiten, den Zugriff auf ein Thema oder eine Warteschlange zu steuern:

- Hinzufügen einer Richtlinie zu einem IAM-Benutzer oder einer IAM-Gruppe. Die einfachste Möglichkeit, Benutzern Berechtigungen für Themen oder Warteschlangen zu gewähren, besteht darin, eine Gruppe zu erstellen, dieser die entsprechende Richtlinie hinzuzufügen und anschließend der Gruppe Benutzer hinzuzufügen. Es ist wesentlich einfacher, Benutzer einer Gruppe hinzufügen und zu entfernen, um die Richtlinien für einzelne Benutzer im Auge zu behalten.
- Hinzufügen einer Richtlinie zum Thema oder zur Warteschlange. Wenn Sie Berechtigungen für ein Thema oder eine Warteschlange in einem anderen AWS-Konto erteilen möchten, können Sie dies nur durch Hinzufügen einer Richtlinie tun, deren Prinzipal das AWS-Konto ist, für das Sie die Berechtigungen vergeben möchten.

Sie sollten die erste Methode für die meisten Fällen verwenden (Anwenden von Richtlinien für Gruppen und Verwalten von Berechtigungen für Benutzer durch Hinzufügen oder Entfernen der entsprechenden Benutzer zu den Gruppen). Wenn Sie Berechtigungen für einen Benutzer in einem anderen Konto erteilen müssen, sollten Sie die zweite Methode verwenden.

Hinzufügen einer Richtlinie zu einem IAM-Benutzer oder einer IAM-Gruppe

Wenn Sie die folgende Richtlinie einem IAM-Benutzer oder einer IAM-Gruppe hinzugefügt hätten, würden Sie diesem Benutzer bzw. den Mitgliedern dieser Gruppe die Berechtigung erteilen, die Aktion `sns:Publish` für das Thema „MyTopic” auszuführen.

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "sns:Publish",
      "Resource": "arn:aws:sns:us-east-2:123456789012:MyTopic"
    }
  ]
}
```

Wenn Sie die folgende Richtlinie einem IAM-Benutzer oder einer IAM-Gruppe hinzugefügt hätten, würden Sie diesem Benutzer bzw. den Mitgliedern dieser Gruppe die Berechtigung erteilen, die Aktionen `sqs:ReceiveMessage` und `sqs:DeleteMessage` für die Warteschlangen „MyQueue1" und „MyQueue2" auszuführen.

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "sqs:ReceiveMessage",
        "sqs:DeleteMessage"
      ],
      "Resource": [
        "arn:aws:sqs:us-east-2:123456789012:MyQueue1",
        "arn:aws:sqs:us-east-2:123456789012:MyQueue2"
      ]
    }
  ]
}
```

Hinzufügen einer Richtlinie zu einem Thema oder einer Warteschlange

Die folgenden Beispielrichtlinien zeigen, wie weitere Kontoberechtigungen für ein Thema und eine Warteschlange erteilt werden.

> **Note**
>
> Wenn Sie einem anderen AWS-Konto Zugriff auf eine Ressource in Ihrem Konto erteilen, haben auch IAM-Benutzer, die über Berechtigungen auf Admin-Ebene (Platzhalter-Zugriff) verfügen, Zugriff auf diese Ressource. Allen anderen IAM-Benutzern des anderen Kontos wird der Zugriff auf Ihre Ressource automatisch verweigert. Wenn Sie möchten, dass bestimmte IAM-Benutzer dieses AWS-Konto Zugriff auf Ihre Ressource haben, muss ein IAM-Benutzer mit Administratorrechten diesen IAM-Benutzern die Berechtigungen für die Ressource übertragen. Weitere Informationen zur kontenübergreifenden Berechtigungsübertragung finden Sie unter Aktivieren des kontenübergreifenden Zugriffs im Leitfaden zur Verwendung von IAM.

Wenn Sie die folgende Richtlinie dem Thema „MeinThema" im Konto 123456789012 hinzugefügt hätten, würden Sie dem Konto 111122223333 die Berechtigung erteilen, die Aktion `sns:Publish` für dieses Thema auszuführen.

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "111122223333"
      },
      "Action": "sns:Publish",
      "Resource": "arn:aws:sns:us-east-2:123456789012:MyTopic"
    }
  ]
}
```

Wenn Sie die folgende Richtlinie der Warteschlange „MyQueue" im Konto 123456789012 hinzugefügt hätten, würden Sie dem Konto 111122223333 die Berechtigung erteilen, die Aktionen `sqs:ReceiveMessage` und `sqs:DeleteMessage` für diese Warteschlange auszuführen.

```
{
```

```
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "111122223333"
      },
      "Action": [
        "sqs:DeleteMessage",
        "sqs:ReceiveMessage"
      ],
      "Resource": [
        "arn:aws:sqs:us-east-2:123456789012:MyQueue"
      ]
    }
  ]
}
```

Schritt 5: Testen der Warteschlangen-Abonnements des Themas

Sie können die Warteschlange-Abonnements für ein Thema testen, indem Sie eine Veröffentlichung für das Thema vornehmen und die Nachricht anzeigen, die das Thema an die Warteschlange sendet.

So veröffentlichen Sie über die Amazon-SNS-Konsole zu einem Thema

1. Melden Sie sich mit den Anmeldeinformationen des AWS-Konto oder IAM-Benutzers mit der Berechtigung zum Veröffentlichen in das Thema bei der AWS Management Console an und öffnen Sie die Amazon-SNS-Konsole unter https://console.aws.amazon.com/sns/.
2. Wählen Sie im Navigationsbereich das Thema und Publish to Topic (Zu Thema veröffentlichen) aus.
3. Geben Sie in das Feld Subject (Betreff) einen Betreff ein (z. B. **`Testing publish to queue`**). Geben Sie in das Feld Message (Nachricht) Text ein (z. B. **`Hello world!`**) und wählen Sie Publish Message (Nachricht veröffentlichen) aus. Die folgende Meldung wird angezeigt: Ihre Nachricht wurde erfolgreich veröffentlicht.

So zeigen Sie die Nachricht von dem Thema mithilfe der Amazon-SQS-Konsole an

1. Melden Sie sich mit den Anmeldeinformationen des AWS-Konto oder IAM-Benutzers mit der Berechtigung zum Anzeigen von Nachrichten in der Warteschlange bei der AWS Management Console an und öffnen Sie die Amazon-SQS-Konsole unter https://console.aws.amazon.com/sqs/.

2. Wählen Sie eine Warteschlange, die das Thema abonniert hat.
3. Wählen Sie Send and receive messages (Nachrichten senden und empfangen) und dann Poll for messages (Nachrichten abfragen) aus. Es wird eine Nachricht des Typs Notification (Benachrichtigung) angezeigt.
4. Wählen Sie in der Spalte Bodydie Option More Details. Das Feld Message Details (Nachrichtendetails) enthält ein JSON-Dokument mit dem Betreff und der Nachricht, die Sie zum Thema veröffentlicht haben. Die Nachricht sieht ähnlich aus wie das folgende JSON-Dokument.

```
{
  "Type" : "Notification",
  "MessageId" : "63a3f6b6-d533-4a47-aef9-fcf5cf758c76",
  "TopicArn" : "arn:aws:sns:us-west-2:123456789012:MyTopic",
  "Subject" : "Testing publish to subscribed queues",
  "Message" : "Hello world!",
  "Timestamp" : "2012-03-29T05:12:16.901Z",
  "SignatureVersion" : "1",
  "Signature" : "EXAMPLEnTrFPa3...",
  "SigningCertURL" : "https://sns.us-west-2.amazonaws.com/
SimpleNotificationService-f3ecfb7224c7233fe7bb5f59f96de52f.pem",
  "UnsubscribeURL" : "https://sns.us-west-2.amazonaws.com/?
Action=Unsubscribe&SubscriptionArn=arn:aws:sns:us-
west-2:123456789012:MyTopic:c7fe3a54-ab0e-4ec2-88e0-db410a0f2bee"
}
```

5. Klicken Sie auf Close (Schließen). Sie haben erfolgreich eine Nachricht zu einem Thema veröffentlicht, das Benachrichtigungen an eine Warteschlange sendet.

Verwenden einer AWS CloudFormation-Vorlage zum Erstellen eines Themas, das Nachrichten Amazon-SQS-Warteschlanges sendet

Mit AWS CloudFormation können Sie eine Vorlagendatei verwenden, um eine Sammlung von AWS-Ressourcen als eine Einheit zu erstellen und zu konfigurieren. Dieser Abschnitt enthält eine Beispielvorlage, die das Bereitstellen von Themen, die für Warteschlangen veröffentlicht werden, vereinfacht. Die Vorlagen übernehmen für Sie die Einrichtung und führen folgende Schritte aus: Erstellen von zwei Warteschlangen, Erstellen eines Themas mit Abonnements für die Warteschlangen, Hinzufügen einer Richtlinie zu den Warteschlangen, sodass das Thema Nachrichten an die Warteschlangen senden kann, und Erstellen von IAM-Benutzern und -Gruppen, um den Zugriff auf diese Ressourcen zu steuern.

Weitere Informationen zur Bereitstellung von AWS-Ressourcen mithilfe einer AWS CloudFormation-Vorlage finden Sie unter Erste Schritte im AWS CloudFormation-Benutzerhandbuch.

Verwenden einer AWS CloudFormation-Vorlage zum Einrichten von Themen und Warteschlangen innerhalb eines AWS-Konto

Die Beispielvorlage erstellt ein Amazon-SNS-Thema, das Nachrichten an zwei Amazon-SQS-Warteschlanges senden kann, mit den entsprechenden Berechtigungen für die Mitglieder einer IAM-Gruppe für die Veröffentlichung zum Thema und einer weiteren zum Lesen von Nachrichten aus den Warteschlangen. Die Vorlage erstellt außerdem IAM-Benutzer, die jeder Gruppe hinzugefügt werden.

Sie kopieren den Inhalt der Vorlage in eine Datei. Sie können die Vorlage auch von der AWSCloudFormation-Vorlagenseite herunterladen. Wählen Sie auf der Vorlagenseite Durchsuchen Sie Beispielvorlagen nach AWS-Service und klicken Sie auf und danach auf Amazon Simple Warteschlange Service.

"MySNSTopic" wird eingerichtet, um eine Veröffentlichung für zwei abonnierte Endpunkte vorzunehmen, die zwei Amazon-SQS-Warteschlanges („MyQueue1" und „MyQueue2") sind. "MyPublishTopicGroup" ist eine „IAM-Gruppe, deren Mitglieder die Berechtigung zum Veröffentlichen zu „MySNSTopic" mit der API-Aktion Publish oder dem Befehl sns-publish besitzen. Die Vorlage erstellt die IAM-Benutzer „MyPublishUser" und „MyQueueUser" und gibt ihnen Anmeldeprofile und Zugriffsschlüssel. Der Benutzer, der einen Stack mit dieser Vorlage erstellt, gibt die Passwörter für die Anmeldeprofile als Eingabeparameter an. Die Vorlage erstellt die Zugriffsschlüssel für die beiden &IAM;-Benutzer mit "MyPublishUserKey" und "MyQueueUserKey". "AddUserToMyPublishTopicGroup" fügt den Benutzer "MyPublishUser" zur Gruppe "MyPublishTopicGroup" hinzu, sodass der Benutzer über die Berechtigungen verfügt, die der Gruppe zugewiesen sind.

"MyRDMessageQueueGroup" ist eine IAM;-Gruppe, deren Mitglieder über die Berechtigung zum Lesen und Löschen von Nachrichten aus den beiden Amazon-SQS-Warteschlanges mithilfe der API-Aktionen ReceiveMessage und DeleteMessage verfügen. "AddUserToMyQueueGroup" fügt den Benutzer "MyQueueUser" zur Gruppe "MyRDMessageQueueGroup" hinzu, sodass der Benutzer über die Berechtigungen verfügt, die der Gruppe zugewiesen sind. "MyQueuePolicy" weist die Berechtigung für "MySNSTopic" zum Veröffentlichen seiner Benachrichtigungen an die zwei Warteschlangen zu.

Die folgende Auflistung zeigt den Inhalt der AWS CloudFormation-Vorlage.

```
{
```

```
 "AWSTemplateFormatVersion" : "2010-09-09",

 "Description" : "AWS CloudFormation Sample Template SNSToSQS: This Template creates
an SNS topic that can send messages to
 two SQS queues with appropriate permissions for one IAM user to publish to the topic
and another to read messages from the queues.
 MySNSTopic is set up to publish to two subscribed endpoints, which are two SQS queues
(MyQueue1 and MyQueue2). MyPublishUser is an IAM user
 that can publish to MySNSTopic using the Publish API. MyTopicPolicy assigns that
permission to MyPublishUser. MyQueueUser is an IAM user
 that can read messages from the two SQS queues. MyQueuePolicy assigns those
permissions to MyQueueUser. It also assigns permission for
 MySNSTopic to publish its notifications to the two queues. The template creates
access keys for the two IAM users with MyPublishUserKey
 and MyQueueUserKey. ***Warning*** you will be billed for the AWS resources used if
you create a stack from this template.",

 "Parameters": {
   "MyPublishUserPassword": {
     "NoEcho": "true",
     "Type": "String",
     "Description": "Password for the IAM user MyPublishUser",
     "MinLength": "1",
     "MaxLength": "41",
     "AllowedPattern": "[a-zA-Z0-9]*",
     "ConstraintDescription": "must contain only alphanumeric characters."
   },
   "MyQueueUserPassword": {
     "NoEcho": "true",
     "Type": "String",
     "Description": "Password for the IAM user MyQueueUser",
     "MinLength": "1",
     "MaxLength": "41",
     "AllowedPattern": "[a-zA-Z0-9]*",
     "ConstraintDescription": "must contain only alphanumeric characters."
   }
 },


 "Resources": {
   "MySNSTopic": {
     "Type": "AWS::SNS::Topic",
     "Properties": {
       "Subscription": [{
```

```
          "Endpoint": {
            "Fn::GetAtt": ["MyQueue1", "Arn"]
          },
          "Protocol": "sqs"
        },
        {
          "Endpoint": {
            "Fn::GetAtt": ["MyQueue2", "Arn"]
          },
          "Protocol": "sqs"
        }
      ]
    }
  },
  "MyQueue1": {
    "Type": "AWS::SQS::Queue"
  },
  "MyQueue2": {
    "Type": "AWS::SQS::Queue"
  },
  "MyPublishUser": {
    "Type": "AWS::IAM::User",
    "Properties": {
      "LoginProfile": {
        "Password": {
          "Ref": "MyPublishUserPassword"
        }
      }
    }
  },
  "MyPublishUserKey": {
    "Type": "AWS::IAM::AccessKey",
    "Properties": {
      "UserName": {
        "Ref": "MyPublishUser"
      }
    }
  },
  "MyPublishTopicGroup": {
    "Type": "AWS::IAM::Group",
    "Properties": {
      "Policies": [{
        "PolicyName": "MyTopicGroupPolicy",
        "PolicyDocument": {
```

```
            "Statement": [{
              "Effect": "Allow",
              "Action": [
                "sns:Publish"
              ],
              "Resource": {
                "Ref": "MySNSTopic"
              }
            }]
          }
        }]
      }
    },
    "AddUserToMyPublishTopicGroup": {
      "Type": "AWS::IAM::UserToGroupAddition",
      "Properties": {
        "GroupName": {
          "Ref": "MyPublishTopicGroup"
        },
        "Users": [{
          "Ref": "MyPublishUser"
        }]
      }
    },
    "MyQueueUser": {
      "Type": "AWS::IAM::User",
      "Properties": {
        "LoginProfile": {
          "Password": {
            "Ref": "MyQueueUserPassword"
          }
        }
      }
    },
    "MyQueueUserKey": {
      "Type": "AWS::IAM::AccessKey",
      "Properties": {
        "UserName": {
          "Ref": "MyQueueUser"
        }
      }
    },
    "MyRDMessageQueueGroup": {
      "Type": "AWS::IAM::Group",
```

```
      "Properties": {
        "Policies": [{
          "PolicyName": "MyQueueGroupPolicy",
          "PolicyDocument": {
            "Statement": [{
              "Effect": "Allow",
              "Action": [
                "sqs:DeleteMessage",
                "sqs:ReceiveMessage"
              ],
              "Resource": [{
                  "Fn::GetAtt": ["MyQueue1", "Arn"]
                },
                {
                  "Fn::GetAtt": ["MyQueue2", "Arn"]
                }
              ]
            }]
          }
        }]
      }
    },
    "AddUserToMyQueueGroup": {
      "Type": "AWS::IAM::UserToGroupAddition",
      "Properties": {
        "GroupName": {
          "Ref": "MyRDMessageQueueGroup"
        },
        "Users": [{
          "Ref": "MyQueueUser"
        }]
      }
    },
    "MyQueuePolicy": {
      "Type": "AWS::SQS::QueuePolicy",
      "Properties": {
        "PolicyDocument": {
          "Statement": [{
            "Effect": "Allow",
            "Principal": {
              "Service": "sns.amazonaws.com"
            },
            "Action": ["sqs:SendMessage"],
            "Resource": "*",
```

```
            "Condition": {
              "ArnEquals": {
                "aws:SourceArn": {
                  "Ref": "MySNSTopic"
                }
              }
            }
          }]
        },
        "Queues": [{
          "Ref": "MyQueue1"
        }, {
          "Ref": "MyQueue2"
        }]
      }
    }
  },
  "Outputs": {
    "MySNSTopicTopicARN": {
      "Value": {
        "Ref": "MySNSTopic"
      }
    },
    "MyQueue1Info": {
      "Value": {
        "Fn::Join": [
          " ",
          [
            "ARN:",
            {
              "Fn::GetAtt": ["MyQueue1", "Arn"]
            },
            "URL:",
            {
              "Ref": "MyQueue1"
            }
          ]
        ]
      }
    },
    "MyQueue2Info": {
      "Value": {
        "Fn::Join": [
          " ",
```

```
          [
            "ARN:",
            {
              "Fn::GetAtt": ["MyQueue2", "Arn"]
            },
            "URL:",
            {
              "Ref": "MyQueue2"
            }
          ]
        ]
      }
    },
    "MyPublishUserInfo": {
      "Value": {
        "Fn::Join": [
          " ",
          [
            "ARN:",
            {
              "Fn::GetAtt": ["MyPublishUser", "Arn"]
            },
            "Access Key:",
            {
              "Ref": "MyPublishUserKey"
            },
            "Secret Key:",
            {
              "Fn::GetAtt": ["MyPublishUserKey", "SecretAccessKey"]
            }
          ]
        ]
      }
    },
    "MyQueueUserInfo": {
      "Value": {
        "Fn::Join": [
          " ",
          [
            "ARN:",
            {
              "Fn::GetAtt": ["MyQueueUser", "Arn"]
            },
            "Access Key:",
```

```
            {
              "Ref": "MyQueueUserKey"
            },
            "Secret Key:",
            {
              "Fn::GetAtt": ["MyQueueUserKey", "SecretAccessKey"]
            }
          ]
        ]
      }
    }
  }
}
```

Verteilung zu HTTP(S)-Endpunkten

Sie können Amazon SNS verwenden, um Benachrichtigungen an einzelne oder mehrere HTTP- oder HTTPS-Endpunkte zu senden. Wenn Sie ein Thema für einen Endpunkt abonniert haben, können Sie eine Benachrichtigung zu diesem Thema veröffentlichen Amazon SNS sendet eine HTTP POST-Anforderung zur Zustellung des Inhalts der Benachrichtigung an den registrierten Endpunkt. Wenn Sie den Endpunkt abonniert haben, können Sie auswählen, ob Amazon SNS HTTP oder HTTPS zum Senden der POST-Anforderung an den Endpunkt verwenden soll. Wenn Sie HTTPS verwenden, können Sie die Vorteile der Amazon SNS-Unterstützung für folgende Aktionen nutzen:

- Server Name Indication (SNI, Servernamensanzeige) - Auf diese Weise kann Amazon SNS; HTTPS-Endpunkte unterstützen, die SNI benötigen. Das kann z. B. ein Server sein, der mehrere Zertifikate für das Hosten mehrerer Domänen benötigt. Weitere Informationen zur SNI (Servernamensanzeige) finden Sie unter Server Name Indication.

- Standard- und Digest-Zugriffsauthentifizierung— Auf diese Weise können Sie in der HTTPS-URL einen Benutzernamen und ein Kennwort für die HTTP POST-Anforderung angeben, z. B.`https://user:password@domain.com`oder`https://user@domain.com` Der Benutzername und das Kennwort sind über die SSL-Verbindung verschlüsselt, die bei der Verwendung von HTTPS aufgebaut wird. Nur der Domänenname wird als Klartext gesendet. Weitere Informationen zu Basic and Digest Access Authentication finden Sie unter RFC-2617.

  > **Important**
  >
  > Amazon SNS unterstützt derzeit keine privaten HTTP(S)-Endpunkte.

HTTPS-URLs können von der Amazon SNS-API-Aktion `GetSubscriptionAttributes` nur für Prinzipale abgerufen werden, denen Sie API-Zugriff gewährt haben.

> **Note**
>
> Die Client-Service muss die `HTTP/1.1 401 Unauthorized`-Header-Antwort unterstützen.

Die Anforderung enthält den Betreff und die Nachricht, die zum Thema veröffentlicht wurde, sowie Metadaten zur Benachrichtigung in einem JSON-Dokument. Die Anforderung sieht ähnlich aus wie die folgende HTTP POST-Anforderung: Weitere Informationen über den HTTP-Header und das JSON-Format des Anforderungstextes finden Sie unter HTTP/HTTPS-Header und JSON-Format für eine HTTP/HTTPS-Benachrichtigung.

```
POST / HTTP/1.1
    x-amz-sns-message-type: Notification
    x-amz-sns-message-id: da41e39f-ea4d-435a-b922-c6aae3915ebe
    x-amz-sns-topic-arn: arn:aws:sns:us-west-2:123456789012:MyTopic
    x-amz-sns-subscription-arn: arn:aws:sns:us-
west-2:123456789012:MyTopic:2bcfbf39-05c3-41de-beaa-fcfcc21c8f55
    Content-Length: 761
    Content-Type: text/plain; charset=UTF-8
    Host: ec2-50-17-44-49.compute-1.amazonaws.com
    Connection: Keep-Alive
    User-Agent: Amazon Simple Notification Service Agent

{
  "Type" : "Notification",
  "MessageId" : "da41e39f-ea4d-435a-b922-c6aae3915ebe",
  "TopicArn" : "arn:aws:sns:us-west-2:123456789012:MyTopic",
  "Subject" : "test",
  "Message" : "test message",
  "Timestamp" : "2012-04-25T21:49:25.719Z",
  "SignatureVersion" : "1",
  "Signature" :
 "EXAMPLElDMXvB8r9R83tGoNn0ecwd5UjllzsvSvbItzfaMpN2nk5HVSw7XnOn/49IkxDKz8YrlH2qJXj2iZB0Zo2O71c4
  "SigningCertURL" : "https://sns.us-west-2.amazonaws.com/SimpleNotificationService-
f3ecfb7224c7233fe7bb5f59f96de52f.pem",
```

```
   "UnsubscribeURL" : "https://sns.us-west-2.amazonaws.com/?
Action=Unsubscribe&SubscriptionArn=arn:aws:sns:us-
west-2:123456789012:MyTopic:2bcfbf39-05c3-41de-beaa-fcfcc21c8f55"
}
```

Themen

- Abonnieren eines HTTP/S endpoint-Endpunkts zu einem Thema
- Überprüfen der Signaturen von Amazon-SNS-Nachrichten
- Analysieren von Nachrichtenformaten

Abonnieren eines HTTP/S endpoint-Endpunkts zu einem Thema

Auf den Seiten in diesem Abschnitt wird beschrieben, wie HTTP/S-Endpunkte für Amazon SNS-Themen abonniert werden.

Themen

- Schritt 1: Stellen Sie sicher, dass Ihr Endpunkt Amazon-SNS-Nachrichten verarbeiten kann
- Schritt 2: Abonnieren des HTTP-/HTTPS-Endpunkts zum Amazon-SNS-Thema
- Schritt 3: Bestätigen des Abonnements
- Schritt 4: Festlegen der Bereitstellungsrichtlinie für das Abonnement (optional)
- Schritt 5: Geben Sie Benutzern die Berechtigungen für Veröffentlichungen zum Thema (optional)
- Schritt 6: Senden von Nachrichten an den HTTP-/HTTPS-Endpunkt

Schritt 1: Stellen Sie sicher, dass Ihr Endpunkt Amazon-SNS-Nachrichten verarbeiten kann

Bevor Sie ein Thema für Ihren HTTP- oder HTTPS-Endpunkt abonnieren, müssen Sie sicherstellen, dass der HTTP- oder HTTPS-Endpunkt über ausreichend Kapazität zur Durchführung von HTTP POST-Anforderungen verfügt, die Amazon SNS zum Senden der Abonnementbestätigungs- und Benachrichtigungsmitteilungen verwendet. In der Regel bedeutet dies das Erstellen und Bereitstellen einer Webanwendung, die HTTP-Anforderungen von Amazon SNS verarbeitet (z. B. ein Java Servlet, wenn Ihr Endpunkt-Host Linux mit Apache und Tomcat ausführt). Wenn Sie einen HTTP-Endpunkt abonnieren, sendet Amazon SNS eine Anforderung für die Abonnementbestätigung an diesen Endpunkt. Der Endpunkt muss beim Erstellen des Abonnements bereit für den Empfang und die Verarbeitung der Anforderung sein, da Amazon SNS diese zu diesem Zeitpunkt sendet. Amazon SNS

sendet solange keine Benachrichtigungen an diesen Endpunkt, bis Sie das Abonnement bestätigen. Sobald Sie das Abonnement bestätigen, sendet Amazon-SNS-Benachrichtigungen an den Endpunkt, wenn eine Nachricht zum abonnierten Thema veröffentlicht wird.

Einrichten des Endpunkts für die Verarbeitung von Abonnementsbestätigungen und Benachrichtigungsmitteilungen

1. Der Code sollte die HTTP-Header der HTTP POST-Anforderungen lesen, die Amazon SNS an Ihrem Endpunkt sendet. Ihr Code sollte im Header-Feld den Nachrichtentyp abrufen `x-amz-sns-message-type`, den Ihnen Amazon SNS gesendet hat. Im Header finden Sie den Nachrichtentyp, ohne dass Sie den Text der HTTP-Anforderung analysieren müssen. Diese beiden Typen müssen Sie verarbeiten: `SubscriptionConfirmation` und `Notification`. Die `UnsubscribeConfirmation` Mitteilung wird nur verwendet, wenn das Abonnement des Themas gelöscht wird.

   Details zum HTTP-Header finden Sie unter HTTP/HTTPS-Header. Die folgende HTTP POST-Anforderung ist ein Beispiel für eine Abonnement-Bestätigungsnachricht.

```
POST / HTTP/1.1
    x-amz-sns-message-type: SubscriptionConfirmation
    x-amz-sns-message-id: 165545c9-2a5c-472c-8df2-7ff2be2b3b1b
    x-amz-sns-topic-arn: arn:aws:sns:us-west-2:123456789012:MyTopic
    Content-Length: 1336
    Content-Type: text/plain; charset=UTF-8
    Host: example.com
    Connection: Keep-Alive
    User-Agent: Amazon Simple Notification Service Agent

{
  "Type" : "SubscriptionConfirmation",
  "MessageId" : "165545c9-2a5c-472c-8df2-7ff2be2b3b1b",
  "Token" : "2336412f37f...",
  "TopicArn" : "arn:aws:sns:us-west-2:123456789012:MyTopic",
  "Message" : "You have chosen to subscribe to the topic arn:aws:sns:us-
west-2:123456789012:MyTopic.\nTo confirm the subscription, visit the SubscribeURL
 included in this message.",
  "SubscribeURL" : "https://sns.us-west-2.amazonaws.com/?
Action=ConfirmSubscription&TopicArn=arn:aws:sns:us-
west-2:123456789012:MyTopic&Token=2336412f37...",
  "Timestamp" : "2012-04-26T20:45:04.751Z",
  "SignatureVersion" : "1",
```

```
  "Signature" : "EXAMPLEpH+...",
  "SigningCertURL" : "https://sns.us-west-2.amazonaws.com/
SimpleNotificationService-f3ecfb7224c7233fe7bb5f59f96de52f.pem"
}
```

2. Der Code sollte das JSON-Dokument im Text der HTTP POST-Anfrage und im text/plain vom Inhaltstyp analysieren, um die Name-Wert-Paare zu lesen, aus denen die Amazon-SNS-Nachricht besteht. Verwenden Sie einen JSON-Parser, der das Umwandeln der Escape-Sequenzen von Steuerzeichen in die entsprechenden ASCII-Zeichenwerte durchführt (z. B. Konvertieren von \n in einen Zeilenumbruch). Sie können einen vorhandenen JSON-Parser wie den Jackson JSON Processor verwenden oder einen eigenen schreiben. Zum Senden des Textes in den Betreff- und Nachrichtenfeldern im gültigen JSON-Format muss Amazon SNS einige Steuerzeichen in Escape-Sequenzen umwandeln, die in das JSON-Dokument integriert werden können. Wenn Sie das JSON-Dokument im Text der POST-Anforderung am Endpunkt empfangen, müssen Sie die Escape-Zeichen wieder in die ursprünglichen Zeichenwerte zurückkonvertieren, wenn Sie eine exakte Darstellung des ursprünglichen Betreffs und der zum Thema veröffentlichten Nachrichten sehen möchten. Dies ist besonders wichtig, wenn Sie die Signatur einer Benachrichtigung überprüfen möchten, da die Signatur die Benachrichtigung und den Betreff in der ursprünglichen Form als Teil der zu signierenden Zeichenfolge verwendet.
3. Der Code sollte die Authentizität einer Benachrichtigung, die Abonnementbestätigung oder die Bestätigungsnachricht für die Beendigung des Abonnements überprüfen, die Amazon SNS gesendet hat. Unter Verwendung der Informationen in der Amazon SNS-Nachricht kann der Endpunkt die Signatur neu erstellen. Sie können dann den Inhalt der Nachricht überprüfen, wenn Ihre Signatur mit der Signatur übereinstimmt, die Amazon SNS zusammen mit der Nachricht gesendet hat. Weitere Informationen zum Überprüfen der Signatur einer Nachricht finden Sie unter Überprüfen der Signaturen von Amazon-SNS-Nachrichten.
4. Basierend auf dem Typ, der im Header-Feld `x-amz-sns-message-type` angegeben ist, sollte der Code das JSON-Dokument im Text der HTTP-Anforderung lesen und die Mitteilung verarbeiten. Hier finden Sie die Richtlinien für die beiden primären Nachrichtentypen:

   SubscriptionConfirmation

   Lesen Sie den Wert `SubscribeURL` und rufen Sie die URL auf. Um das Abonnement zu bestätigen und Benachrichtigungen am Endpunkt zu empfangen, müssen Sie die `SubscribeURL`URL aufrufen (z. B. indem Sie eine HTTP GET-Anforderung an die URL senden). Sehen Sie sich das Beispiel der HTTP-Anforderung im vorherigen Schritt an, um zu sehen, wie `SubscribeURL` aussieht. Weitere Informationen zum Format der `SubscriptionConfirmation`-Nachricht finden Sie unter JSON-Format für die HTTP/

HTTPS-Abonnement-Bestätigung. Wenn Sie die URL aufrufen, erhalten Sie eine Antwort zurück, die wie das folgende XML-Dokument aussieht. Das Dokument gibt den Abonnement-ARN für den Endpunkt innerhalb des `ConfirmSubscriptionResult`-Elements zurück.

```
<ConfirmSubscriptionResponse xmlns="http://sns.amazonaws.com/doc/2010-03-31/">
   <ConfirmSubscriptionResult>
      <SubscriptionArn>arn:aws:sns:us-
west-2:123456789012:MyTopic:2bcfbf39-05c3-41de-beaa-fcfcc21c8f55</
SubscriptionArn>
   </ConfirmSubscriptionResult>
   <ResponseMetadata>
      <RequestId>075ecce8-8dac-11e1-bf80-f781d96e9307</RequestId>
   </ResponseMetadata>
</ConfirmSubscriptionResponse>
```

Alternativ zum Aufrufen der `SubscribeURL` können Sie das Abonnement mit der ConfirmSubscription-Aktion bestätigen. Dazu muss das `Token` auf den entsprechenden Wert in der `SubscriptionConfirmation`-Nachricht festgelegt sein. Wenn Sie möchten, dass nur der Eigentümer des Themas und der Abonnent den Endpunkt abmelden können sollen, rufen Sie die `ConfirmSubscription`-Aktion mit einer AWS-Signatur auf.

Benachrichtigung

Lesen Sie die Werte für `Subject` und `Message`, um die Benachrichtigungsinformation zu erhalten, die zum Thema veröffentlicht wurde.

Weitere Informationen über das Format der `Notification`-Nachricht finden Sie unter HTTP/HTTPS-Header. Die folgende HTTP POST-Anforderung ist ein Beispiel für eine Benachrichtigung, die an den Endpunkt example.com gesendet wurde.

```
POST / HTTP/1.1
    x-amz-sns-message-type: Notification
    x-amz-sns-message-id: 22b80b92-fdea-4c2c-8f9d-bdfb0c7bf324
    x-amz-sns-topic-arn: arn:aws:sns:us-west-2:123456789012:MyTopic
    x-amz-sns-subscription-arn: arn:aws:sns:us-
west-2:123456789012:MyTopic:c9135db0-26c4-47ec-8998-413945fb5a96
    Content-Length: 773
    Content-Type: text/plain; charset=UTF-8
    Host: example.com
    Connection: Keep-Alive
    User-Agent: Amazon Simple Notification Service Agent
```

```
{
  "Type" : "Notification",
  "MessageId" : "22b80b92-fdea-4c2c-8f9d-bdfb0c7bf324",
  "TopicArn" : "arn:aws:sns:us-west-2:123456789012:MyTopic",
  "Subject" : "My First Message",
  "Message" : "Hello world!",
  "Timestamp" : "2012-05-02T00:54:06.655Z",
  "SignatureVersion" : "1",
  "Signature" : "EXAMPLEw6JRN...",
  "SigningCertURL" : "https://sns.us-west-2.amazonaws.com/
SimpleNotificationService-f3ecfb7224c7233fe7bb5f59f96de52f.pem",
  "UnsubscribeURL" : "https://sns.us-west-2.amazonaws.com/?
Action=Unsubscribe&SubscriptionArn=arn:aws:sns:us-
west-2:123456789012:MyTopic:c9135db0-26c4-47ec-8998-413945fb5a96"
}
```

5. Stellen Sie sicher, dass der Endpunkt auf die HTTP-POST-Nachricht von Amazon SNS mit dem entsprechenden Statuscode antwortet. Die Verbindung wird in 15 Sekunden beendet. Wenn Ihr Endpunkt nicht reagiert, bevor die Verbindung endet oder wenn Ihr Endpunkt einen Statuscode außerhalb des Bereichs 200-4xx zurückgibt, behandelt Amazon SNS die Zustellung der Nachricht als fehlgeschlagen.
6. Stellen Sie sicher, dass Ihr Code wiederholte Nachrichtenzustellungen von Amazon SNS verarbeiten kann. Wenn Amazon SNS keine erfolgreiche Antwort von Ihrem Endpunkt erhält, versucht der Dienst, die Nachricht erneut zuzustellen. Dies gilt für alle Nachrichten, einschließlich der Abonnement-Bestätigungsnachricht. Wenn die erste Zustellung der Nachricht fehlschlägt, versucht Amazon SNS bis zu drei Mal eine erneute Zustellung mit einer Verzögerung von 20 Sekunden zwischen den fehlgeschlagenen Versuchen.

> **Note**
>
> Bei dieser Nachrichtenanfrage erfolgt ein Timeout nach 15 Sekunden. Das bedeutet, wenn die Nachrichtenzustellung wegen einer Zeitüberschreitung nicht zugestellt werden kann, startet Amazon SNS nach etwa 35 Sekunden einen erneuten Zustellversuch. Sie können eine andere Zustellungsrichtlinie für den Endpunkt festlegen.

Amazon SNS verwendet das Header-Feld `x-amz-sns-message-id`, um jede zu einem Amazon-SNS-Thema veröffentlichte Nachricht eindeutig zu identifizieren. Durch das Vergleichen

der Nachrichten-IDs, die Sie mit den eingehenden Nachrichten verarbeitet haben, können Sie ermitteln, ob die Nachricht ein Wiederholungsversuch ist.

7. Wenn Sie einen HTTPS-Endpunkt abonnieren, stellen Sie sicher, dass Ihr Endpunkt ein Serverzertifikat von einer vertrauenswürdigen Zertifizierungsstelle (CA) aufweist. Amazon SNS sendet nur Nachrichten an HTTPS-Endpunkte, die über ein Serverzertifikat mit der Signatur einer für Amazon SNS vertrauenswürdigen CA verfügt.
8. Stellen Sie den Code bereit, den Sie für den Empfang von Amazon-SNS-Nachrichten erstellt haben. Wenn Sie den Endpunkt abonnieren, muss dieser mindestens die Abonnement-Bestätigungsnachricht empfangen können.

Schritt 2: Abonnieren des HTTP-/HTTPS-Endpunkts zum Amazon-SNS-Thema

Um Nachrichten über ein Thema an einen HTTP- oder HTTPS-Endpunkt zu senden, müssen Sie den Endpunkt für das Amazon SNS-Thema abonnieren. Sie geben den Endpunkt über seine URL an. Zum Abonnieren eines Themas können Sie die Amazon-SNS-Konsole, den Befehl sns-subscribe oder die API-Aktion Subscribe verwenden. Bevor Sie beginnen, stellen Sie sicher, dass Sie die URL für den Endpunkt kennen, den Sie abonnieren möchten, und dass der Endpunkt Bestätigungen und Benachrichtigungsmitteilungen wie in Schritt 1 beschrieben empfangen kann.

Abonnieren eines Themas auf einen HTTP- oder HTTPS-Endpunkt mit der Amazon-SNS-Konsole

1. Melden Sie sich bei der Amazon-SNS-Konsole an.
2. Wählen Sie im Navigationsbereich Topics (Themen) aus.
3. Wählen Sie Abonnement erstellen aus.
4. Wählen Sie in der Dropdown-Liste Protocol (Protokoll) entweder HTTP oder HTTPS aus.
5. Kopieren Sie in das Feld Endpoint (Endpunkt) die URL für den Endpunkt, an den das Thema Nachrichten senden soll, und wählen Sie dann Create Subscription (Abonnement erstellen).
6. Die Bestätigungsmeldung wird angezeigt. Klicken Sie auf Close (Schließen).

   Die Subscription ID (Abonnement-ID) des neuen Abonnements zeigt "PendingConfirmation" an. Wenn Sie das Abonnement bestätigen, zeigt Subscription ID (Abonnement-ID) die Abonnement-ID an.

Schritt 3: Bestätigen des Abonnements

Nachdem Sie einen Endpunkt abonniert haben, sendet Amazon SNS eine Abonnement-Bestätigungsnachricht an den Endpunkt. Sie sollten bereits über Code verfügen, der die Aktionen auf dem Endpunkt durchführt, wie unter Schritt 1 beschrieben. Der Code am Endpunkt muss vor allem den Wert `SubscribeURL` aus der Abonnement-Bestätigungsnachricht abrufen und entweder die in `SubscribeURL` angegebene Position direkt aufrufen oder so verfügbar machen, dass Sie diese `SubscribeURL` manuell (z. B. mit einem Webbrowser) aufrufen können. Amazon SNS sendet keine Nachrichten an den Endpunkt, bis das Abonnement bestätigt wird. Wenn Sie `SubscribeURL` aufrufen, enthält die Antwort ein XML-Dokument, das seinerseits ein `SubscriptionArn`-Element enthält, mit dem der ARN für das Abonnement angegeben wird. Sie können die Amazon-SNS-Konsole auch verwenden, um zu überprüfen, ob das Abonnement bestätigt ist: Das Feld Subscription ID (Abonnement-ID) zeigt die ARN für das Abonnement anstelle des `PendingConfirmation`-Werts an, der beim ersten Hinzufügen des Abonnements angezeigt wurde.

Schritt 4: Festlegen der Bereitstellungsrichtlinie für das Abonnement (optional)

Wenn die erste Zustellung der Nachricht fehlschlägt, versucht Amazon SNS bis zu drei Mal eine erneute Zustellung mit einer Verzögerung von 20 Sekunden zwischen den fehlgeschlagenen Versuchen. Wie in Schritt 1 erläutert, sollte Ihr Endpunkt über Code verfügen, der wiederholte Nachrichten verarbeiten kann. Indem Sie die Zustellungsrichtlinie für ein Thema oder ein Abonnement festlegen, können Sie die Häufigkeit und das Intervall steuern, die Amazon SNS für wiederholte Zustellungen fehlgeschlagener Nachrichten verwendet. Sie können den Inhaltstyp für Ihre HTTP/S-Benachrichtigungen auch in `DeliveryPolicy` angeben. Weitere Informationen finden Sie unter Erstellen einer HTTP/S-Zustellungsrichtlinie.

Schritt 5: Geben Sie Benutzern die Berechtigungen für Veröffentlichungen zum Thema (optional)

Standardmäßig hat der Eigentümer des Themas die Berechtigung zur Veröffentlichung des Themas. Um anderen Benutzern oder Anwendungen eine Veröffentlichung zum Thema zu ermöglichen, sollten Sie AWS Identity and Access Management (IAM) verwenden, um die Veröffentlichungsberechtigung für das Thema zu verleihen. Weitere Informationen über die Berechtigungen für Amazon SNS-Aktionen finden Sie unter Verwenden von identitätsbasierten Richtlinien mit Amazon SNS.

Es gibt zwei Möglichkeiten, den Zugriff auf ein Thema zu steuern:

- Hinzufügen einer Richtlinie zu einem IAM-Benutzer oder einer IAM-Gruppe. Die einfachste Möglichkeit, Benutzern Berechtigungen für Themen zu gewähren, besteht darin, eine Gruppe zu

erstellen, dieser die entsprechende Richtlinie hinzuzufügen und anschließend der Gruppe Benutzer hinzuzufügen. Es ist wesentlich einfacher, Benutzer einer Gruppe hinzufügen und zu entfernen, um die Richtlinien für einzelne Benutzer im Auge zu behalten.

- Hinzufügen einer Richtlinie zum Thema: Wenn Sie Berechtigungen für ein Thema einem anderen AWS-Konto gewähren möchten, können Sie dies nur durch Hinzufügen einer Richtlinie erreichen, deren Prinzipal das AWS-Konto ist, für das Sie die Berechtigungen vergeben möchten.

Sie sollten die erste Methode für die meisten Fällen verwenden (Anwenden von Richtlinien für Gruppen und Verwalten von Berechtigungen für Benutzer durch Hinzufügen oder Entfernen der entsprechenden Benutzer zu den Gruppen). Wenn Sie Berechtigungen für einen Benutzer in einem anderen Konto erteilen müssen, verwenden Sie die zweite Methode.

Wenn Sie die folgende Richtlinie einem IAM-Benutzer oder einer IAM-Gruppe hinzugefügt hätten, würden Sie diesem Benutzer bzw. den Mitgliedern dieser Gruppe die Berechtigung erteilen, die Aktion `sns:Publish` für das Thema „MyTopic” auszuführen.

```
{
  "Statement":[{
    "Sid":"AllowPublishToMyTopic",
    "Effect":"Allow",
    "Action":"sns:Publish",
    "Resource":"arn:aws:sns:us-east-2:123456789012:MyTopic"
  }]
}
```

Die folgende Beispielrichtlinie zeigt, wie weitere Kontoberechtigungen für ein Thema erteilt werden.

> **Note**
>
> Wenn Sie einem anderen AWS-Konto Zugriff auf eine Ressource in Ihrem Konto erteilen, haben auch AWS-Konto-Benutzer, die über Berechtigungen auf Admin-Ebene (Platzhalter-Zugriff) verfügen, Zugriff auf diese Ressource. Allen anderen IAM-Benutzern des anderen Kontos wird der Zugriff auf Ihre Ressource automatisch verweigert. Wenn Sie möchten, dass bestimmte IAM-Benutzer dieses AWS-Konto Zugriff auf Ihre Ressource haben, muss ein IAM-Benutzer mit Administratorrechten diesen IAM-Benutzern die Berechtigungen für die Ressource übertragen. Weitere Informationen zur kontenübergreifenden Berechtigungsübertragung finden Sie unter Aktivieren des kontenübergreifenden Zugriffs im Leitfaden zur Verwendung von IAM.

Wenn Sie die folgende Richtlinie dem Thema „MeinThema" im Konto 123456789012 hinzugefügt hätten, würden Sie dem Konto 111122223333 die Berechtigung erteilen, die Aktion `sns:Publish` für dieses Thema auszuführen.

```
{
  "Statement":[{
    "Sid":"Allow-publish-to-topic",
    "Effect":"Allow",
      "Principal":{
        "AWS":"111122223333"
      },
    "Action":"sns:Publish",
    "Resource":"arn:aws:sns:us-east-2:123456789012:MyTopic"
  }]
}
```

Schritt 6: Senden von Nachrichten an den HTTP-/HTTPS-Endpunkt

Sie können eine Nachricht an die Abonnements eines Themas senden, indem Sie für das Thema veröffentlichen. Um zu einem Thema zu veröffentlichen, können Sie die Amazon-SNS-Konsole, den `sns-publish` CLI-Befehl oder die API-Aktion `Publish` verwenden.

Falls Sie Schritt 1 befolgt haben, sollte der bereitgestellte Code am Endpunkt die Benachrichtigung verarbeiten.

So veröffentlichen Sie über die Amazon-SNS-Konsole zu einem Thema

1. Melden Sie sich mit den Anmeldeinformationen des AWS-Konto oder IAM-Benutzers mit der Berechtigung zum Veröffentlichen in das Thema bei der AWS Management Console an und öffnen Sie die Amazon-SNS-Konsole unter https://console.aws.amazon.com/sns/.
2. Wählen Sie im Navigationsbereich Topics (Themen) und dann ein Thema aus.
3. Wählen Sie die Schaltfläche Publish message (Nachricht veröffentlichen).
4. Geben Sie im Feld Subject (Betreff) einen Betreff ein (z. B. **`Testing publish to my endpoint`**).
5. Geben Sie im Feld Message Text (Nachrichtentext) Text ein (z. B. **`Hello world!`**) und wählen Sie Publish message (Nachricht veröffentlichen).

   Die folgende Meldung wird angezeigt: Ihre Nachricht wurde erfolgreich veröffentlicht.

Überprüfen der Signaturen von Amazon-SNS-Nachrichten

Um die Echtheit einer Nachricht zu überprüfen, die von Amazon SNS an Ihren HTTP-Endpunkt gesendet wurde, können Sie die Nachrichtensignatur heranziehen. Es gibt zwei Fälle, in denen wir empfehlen, die Echtheit der Nachricht zu überprüfen. Erstens, wenn Amazon SNS eine Nachricht an Ihren HTTP-Endpunkt sendet mit dem Hinweis, dass Sie ein Thema abonniert haben. Zweitens, wenn Amazon SNS Ihnen nach Ausführung der API-Aktion `Subscribe` oder `Unsubscribe` eine Bestätigungsnachricht an Ihren HTTP-Endpunkt sendet.

Führen Sie die folgenden Schritte aus, wenn Sie die von Amazon SNS gesendeten Nachrichten überprüfen:

- Verwenden Sie immer HTTPS, wenn Sie das Zertifikat von Amazon SNS erhalten.
- Überprüfen Sie die Authentizität des Zertifikats.
- Überprüfen Sie, ob das Zertifikat von Amazon SNS empfangen wurde.
- Wenn möglich, verwenden Sie eines der unterstützten AWS-SDKs für Amazon SNS, um Nachrichten zu validieren und zu überprüfen.
- Vergewissern Sie sich, dass die Amazon-SNS-Nachrichten von Ihrem gewünschten `TopicArn` empfangen wurden.

Amazon SNS unterstützt zwei Versionen der Nachrichtensignatur:

- `SignatureVersion1`: Amazon SNS erstellt die Signatur basierend auf dem SHA1-Hash der Nachricht.
- `SignatureVersion2`: Amazon SNS erstellt die Signatur basierend auf dem SHA256-Hash der Nachricht.

So konfigurieren Sie die Version der Nachrichtensignatur in Amazon-SNS-Themen

Standardmäßig verwenden Amazon-SNS-Themen `SignatureVersion` 1. Zum Auswählen des Hashing-Algorithmus für Ihr Amazon-SNS-Thema, also entweder `SignatureVersion` 1 (SHA1) oder `SignatureVersion` 2 (SHA256), können Sie die API-Aktion `SetTopicAttributes` verwenden.

Im folgenden Codebeispiel wird veranschaulicht, wie Sie das Themenattribut `SignatureVersion` mit der AWS CLI festlegen:

```
aws sns set-topic-attributes \
    --topic-arn arn:aws:sns:us-east-2:123456789012:MyTopic \
    --attribute-name SignatureVersion \
    --attribute-value 2
```

So überprüfen Sie die Signatur einer Amazon-SNS-Nachricht bei der Verwendung von abfragebasierten HTTP-Anfragen

1. Extrahieren Sie die Name-Wert-Paare aus dem JSON-Dokument der HTTP POST-Anfrage, die Amazon SNS an den Endpunkt gesendet hat. Sie werden mit den Werten einiger Name-Wert-Paare die zu signierende Zeichenfolge erstellen. Wenn Sie die Signatur einer Amazon-SNS-Nachricht überprüfen, ist es wichtig, dass Sie die Escape-Steuerzeichen zu den ursprünglichen Zeichenwerten `Message` und `Subject` konvertieren. Diese Werte müssen in ihren ursprünglichen Formaten vorliegen, wenn Sie sie als Teil der zu signierenden Zeichenfolge verwenden. Weitere Informationen zur Analyse eines JSON-Dokuments finden Sie unter Schritt 1: Stellen Sie sicher, dass Ihr Endpunkt Amazon-SNS-Nachrichten verarbeiten kann.

   Die `SignatureVersion` gibt an, welche Signaturversion von Amazon SNS zum Generieren der Signatur der Nachricht verwendet wurde. Mit der Signaturversion können Sie die Anforderungen für das Erstellen der Signatur bestimmen. Bei Benachrichtigungen unterstützt Amazon SNS derzeit die Signaturversionen 1 und 2. Dieser Abschnitt enthält die Schritte zum Überprüfen einer Signatur anhand dieser Signaturversionen.
2. Rufen Sie das X.509-Zertifikat ab, das Amazon SNS zum Signieren der Nachricht verwendet hat. Der Wert `SigningCertURL` zeigt den Speicherort des X.509-Zertifikats, mit dem die digitale Signatur für die Nachricht erstellt wurde. Rufen Sie das Zertifikat von diesem Speicherort ab.
3. Extrahieren Sie den öffentlichen Schlüssel aus dem Zertifikat. Der von `SigningCertURL` angegebene öffentliche Schlüssel aus dem Zertifikat wird zur Überprüfung von Authentizität und Integrität der Nachricht verwendet.
4. Bestimmen des Nachrichtentyps. Das Format der zu signierenden Zeichenfolge hängt vom Nachrichtentyp ab, den der Wert `Type` bezeichnet.
5. Erstellen Sie die zu signierende Zeichenfolge. Die zu signierende Zeichenfolge ist eine durch Zeilenumbruchzeichen getrennte Liste von spezifischen Name-Wert-Paaren aus der Nachricht. Jedes Name-Wert-Paar wird zuerst mit dem Namen, gefolgt von einem Zeilenumbruchzeichen, dann gefolgt vom Wert dargestellt und endet mit einem Zeilenumbruchzeichen. Die Name-Wert-Paare müssen in einer Byte-Sortierreihenfolge aufgelistet werden.

Je nach Nachrichtentyp muss die zu signierende Zeichenfolge die folgenden Name-Wert-Paare aufweisen:

Benachrichtigung

Die Benachrichtigungen müssen die folgenden Name-Wert-Paare enthalten:

```
Message
MessageId
Subject (if included in the message)
Timestamp
TopicArn
Type
```

Das folgende Beispiel ist eine zu signierende Zeichenfolge für eine `Notification`.

```
Message
My Test Message
MessageId
4d4dc071-ddbf-465d-bba8-08f81c89da64
Subject
My subject
Timestamp
2019-01-31T04:37:04.321Z
TopicArn
arn:aws:sns:us-east-2:123456789012:s4-MySNSTopic-1G1WEFCOXTC0P
Type
Notification
```

SubscriptionConfirmation und UnsubscribeConfirmation

`SubscriptionConfirmation`- und `UnsubscribeConfirmation`-Nachrichten müssen die folgenden Name-Wert-Paare enthalten:

```
Message
MessageId
SubscribeURL
Timestamp
Token
TopicArn
```

```
Type
```

Das folgende Beispiel ist eine zu signierende Zeichenfolge für eine `SubscriptionConfirmation`.

```
Message
My Test Message
MessageId
3d891288-136d-417f-bc05-901c108273ee
SubscribeURL
https://sns.us-east-2.amazonaws.com/?
Action=ConfirmSubscription&TopicArn=arn:aws:sns:us-east-2:123456789012:s4-
MySNSTopic-1G1WEFCOXTC0P&Token=233...
Timestamp
2019-01-31T19:25:13.719Z
Token
233...
TopicArn
arn:aws:sns:us-east-2:123456789012:s4-MySNSTopic-1G1WEFCOXTC0P
Type
SubscriptionConfirmation
```

6. Dekodieren Sie den `Signature`-Wert aus dem Base64-Format. Die Nachricht stellt die Signatur im `Signature`-Wert zu, der als Base64 kodiert ist. Bevor Sie den Signaturwert mit der berechneten Signatur vergleichen, stellen Sie sicher, dass Sie den `Signature`-Wert aus dem Base64-Format decodieren, damit Sie die Werte im selben Format vergleichen.
7. Generieren Sie den abgeleiteten Hash-Wert der Amazon-SNS-Nachricht. Senden Sie die Amazon-SNS-Nachricht im kanonischen Format an denselben Hash-Algorithmus, der zum Generieren der Signatur verwendet wurde.
   a. Wenn `SignatureVersion` 1 ist, verwenden Sie SHA1 als Hash-Algorithmus.
   b. Wenn `SignatureVersion` 2 ist, verwenden Sie SHA256 als Hash-Algorithmus.
8. Generieren Sie den festgestellten Hash-Wert der Amazon-SNS-Nachricht. Der festgestellte Hash-Wert ist das Ergebnis, das sich aus der Verwendung des öffentlichen Schlüsselwerts (aus Schritt 3) für die Entschlüsselung der mit der Amazon-SNS-Nachricht übermittelten Signatur ergibt.
9. Überprüfen Sie Authentizität und Integrität der Amazon-SNS-Nachricht. Vergleichen Sie den abgeleiteten Hash-Wert (aus Schritt 7) mit dem festgestellten Hash-Wert (aus Schritt 8). Wenn die Werte identisch sind, kann der Empfänger sicher sein, dass die Nachricht während der

Übertragung nicht geändert wurde und die Nachricht von Amazon SNS stammen muss. Wenn die Werte nicht identisch sind, sollte der Empfänger der Nachricht nicht vertrauen.

Analysieren von Nachrichtenformaten

Amazon SNS verwendet die folgenden Formate:

Themen

HTTP/HTTPS-Header

Sobald Amazon SNS eine Abonnementbestätigungs-, Benachrichtigungs- oder Abmeldebestätigungsnachricht an HTTP/HTTPS-Endpunkte sendet, sendet es eine POST-Nachricht mit einer Reihe an Amazon–SNS-spezifischen Header-Werten. Sie können Header-Werte für Aufgaben wie das Identifizieren des Nachrichtentyps verwenden, ohne den JSON-Nachrichtentext zu analysieren, um den `Type`Wert zu lesen. Standardmäßig sendet Amazon SNS alle Benachrichtigungen an HTTP/S-Endpunkte, wobei `Content-Type` auf `text/plain; charset=UTF-8` festgelegt ist. Informationen zur Auswahl eines `Content-Type`, der nicht text/plain (Standard) ist, finden Sie unter `headerContentType` in Erstellen einer HTTP/S-Zustellungsrichtlinie.

`x-amz-sns-message-type`

Nachrichtentyp Die möglichen Werte sind `SubscriptionConfirmation`, `Notification` und `UnsubscribeConfirmation`.

`x-amz-sns-message-id`

Ein universell eindeutiger Bezeichner (UUID), der für jede veröffentlichte Benachrichtigung eindeutig ist. Für eine Benachrichtigung, die Amazon SNS während eines Wiederholversuchs sendet, wird die Nachrichten-ID der ursprünglichen Nachricht verwendet.

`x-amz-sns-topic-arn`

Amazon Resource Name (ARN) des Themas, in dem die Nachricht veröffentlicht wurde.

`x-amz-sns-subscription-arn`

Die ARN für das Abonnement für diesen Endpunkt.

Der folgende „HTTP POST“-Header ist ein Beispiel für einen Header für eine `Notification`-Benachrichtigung, die an einen HTTP-Endpunkt gesendet wurde.

```
POST / HTTP/1.1
x-amz-sns-message-type: Notification
x-amz-sns-message-id: 165545c9-2a5c-472c-8df2-7ff2be2b3b1b
x-amz-sns-topic-arn: arn:aws:sns:us-west-2:123456789012:MyTopic
x-amz-sns-subscription-arn: arn:aws:sns:us-
west-2:123456789012:MyTopic:2bcfbf39-05c3-41de-beaa-fcfcc21c8f55
Content-Length: 1336
Content-Type: text/plain; charset=UTF-8
Host: myhost.example.com
Connection: Keep-Alive
User-Agent: Amazon Simple Notification Service Agent
```

JSON-Format für die HTTP/HTTPS-Abonnement-Bestätigung

Nachdem Sie einen HTTP/HTTPS-Endpunkt abonniert haben, sendet Amazon SNS eine Abonnement-Bestätigung an den HTTP/HTTPS-Endpunkt. Diese Nachricht enthält einen `SubscribeURL`-Wert, den Sie besuchen müssen, um das Abonnement zu bestätigen (alternativ können Sie den `Token`-Wert mit der Aktion `ConfirmSubscription` verwenden).

> **Note**
>
> Amazon SNS sendet keine Benachrichtigungen an diesen Endpunkt, solange das Abonnement nicht bestätigt wurde

Bei der Abonnement-Bestätigung handelt es sich um eine POST-Nachricht mit einem Nachrichtentext, der ein JSON-Dokument mit den folgenden Name-Wert-Paaren enthält.

`Type`

Nachrichtentyp Für eine Abonnement-Bestätigung ist der Typ `SubscriptionConfirmation`.

`MessageId`

Ein universell eindeutiger Bezeichner (UUID), der für jede veröffentlichte Benachrichtigung eindeutig ist. Für eine Nachricht, die Amazon SNS während eines Wiederholversuchs sendet, wird die Nachrichten-ID der ursprünglichen Nachricht verwendet.

`Token`

Ein Wert, den Sie mit der Aktion `ConfirmSubscription` verwenden können, um das Abonnement zu bestätigen. Sie können aber auch einfach nur den Parameter besuchen `SubscribeURL`.

`TopicArn`

Amazon Resource Name (ARN) des Themas, für das dieser Endpunkt abonniert wurde.

`Message`

Eine Zeichenfolge, die die Nachricht beschreibt. Bei einer Abonnement-Bestätigung sieht diese Zeichenfolge wie folgt aus:

```
You have chosen to subscribe to the topic arn:aws:sns:us-
east-2:123456789012:MyTopic.\nTo confirm the subscription, visit the SubscribeURL
 included in this message.
```

`SubscribeURL`

Die URL, die Sie besuchen müssen, um das Abonnement zu bestätigen. Alternativ können Sie auch den Parameter `Token` mit der Aktion `ConfirmSubscription`verwenden, um das Abonnement zu bestätigen.

`Timestamp`

Die Uhrzeit (GMT), zu der die Abonnement-Bestätigung gesendet wurde.

`SignatureVersion`

Version der verwendeten Amazon-SNS-Signatur.

- Wenn `SignatureVersion` 1 lautet, ist `Signature` eine Base64-kodierte `SHA1withRSA`-Signatur der Werte `Message`, `MessageId`, `Type`, `Timestamp` und `TopicArn`.

- Wenn `SignatureVersion` 2 lautet, ist `Signature` eine Base64-kodierte `SHA256withRSA`-Signatur der Werte `Message`, `MessageId`, `Type`, `Timestamp` und `TopicArn`.

`Signature`

Eine Base64-kodierte `SHA1withRSA`- oder `SHA256withRSA`-Signatur der Werte `Message`, `MessageId`, `Type`, `Timestamp` und `TopicArn`.

`SigningCertURL`

Die URL, die den Zugriff auf das Zertifikat ermöglicht, das zum Signieren der Nachricht verwendet wurde.

Die folgende „HTTP POST“-Nachricht ist ein Beispiel für eine `SubscriptionConfirmation`-Benachrichtigung, die an einen HTTP-Endpunkt gesendet wurde.

```
POST / HTTP/1.1
x-amz-sns-message-type: SubscriptionConfirmation
x-amz-sns-message-id: 165545c9-2a5c-472c-8df2-7ff2be2b3b1b
x-amz-sns-topic-arn: arn:aws:sns:us-west-2:123456789012:MyTopic
Content-Length: 1336
Content-Type: text/plain; charset=UTF-8
Host: myhost.example.com
Connection: Keep-Alive
User-Agent: Amazon Simple Notification Service Agent

{
  "Type" : "SubscriptionConfirmation",
  "MessageId" : "165545c9-2a5c-472c-8df2-7ff2be2b3b1b",
  "Token" : "2336412f37...",
  "TopicArn" : "arn:aws:sns:us-west-2:123456789012:MyTopic",
  "Message" : "You have chosen to subscribe to the topic arn:aws:sns:us-
west-2:123456789012:MyTopic.\nTo confirm the subscription, visit the SubscribeURL
 included in this message.",
  "SubscribeURL" : "https://sns.us-west-2.amazonaws.com/?
Action=ConfirmSubscription&TopicArn=arn:aws:sns:us-
west-2:123456789012:MyTopic&Token=2336412f37...",
  "Timestamp" : "2012-04-26T20:45:04.751Z",
  "SignatureVersion" : "1",
  "Signature" : "EXAMPLEpH
+DcEwjAPg8O9mY8dReBSwksfg2S7WKQcikcNKWLQjwu6A4VbeS0QHVCkhRS7fUQvi2egU3N858fiTDN6bkkOxYDVrY0Ad8L
  "SigningCertURL" : "https://sns.us-west-2.amazonaws.com/SimpleNotificationService-
f3ecfb7224c7233fe7bb5f59f96de52f.pem"
```

```
}
```

JSON-Format für eine HTTP/HTTPS-Benachrichtigung

Wenn Amazon SNS eine Benachrichtigung an einen abonnierten HTTP- oder HTTPS-Endpunkt sendet, enthält der Nachrichtentext der POST-Nachricht, die an den Endpunkt gesendet wurde, ein JSON-Dokument mit den folgenden Name-Wert-Paaren.

`Type`

Nachrichtentyp Für eine Bestätigung ist der Typ `Notification`.

`MessageId`

Ein universell eindeutiger Bezeichner (UUID), der für jede veröffentlichte Benachrichtigung eindeutig ist. Für eine Benachrichtigung, die Amazon SNS während eines Wiederholversuchs sendet, wird die Nachrichten-ID der ursprünglichen Nachricht verwendet.

`TopicArn`

Amazon Resource Name (ARN) des Themas, in dem die Nachricht veröffentlicht wurde.

`Subject`

Der Parameter `Subject`, der bei der Veröffentlichung der Benachrichtigung im Thema angegeben wurde.

> **Note**
>
> Dieser Parameter ist optional. Wenn kein `Subject` angegeben wurde, wird dieses Name-Wert-Paar nicht in diesem JSON-Dokument angezeigt.

`Message`

Der Nachrichtenwert `Message`, der bei der Veröffentlichung der Benachrichtigung im Thema angegeben wurde.

`Timestamp`

Uhrzeit (GMT), zu der die Benachrichtigung veröffentlicht wurde.

`SignatureVersion`

Version der verwendeten Amazon-SNS-Signatur.

- Wenn `SignatureVersion` 1 lautet, ist `Signature` eine Base64-kodierte `SHA1withRSA`-Signatur der Werte `Message`, `MessageId`, `Subject` (sofern vorhanden), `Type`, `Timestamp` und `TopicArn`.
- Wenn `SignatureVersion` 2 lautet, ist `Signature` eine Base64-kodierte `SHA256withRSA`-Signatur der Werte `Message`, `MessageId`, `Subject` (sofern vorhanden), `Type`, `Timestamp` und `TopicArn`.

`Signature`

Eine Base64-kodierte `SHA1withRSA`- oder `SHA256withRSA`-Signatur der Werte `Message`, `MessageId`, `Subject` (sofern vorhanden), `Type`, `Timestamp` und `TopicArn`.

`SigningCertURL`

Die URL, die den Zugriff auf das Zertifikat ermöglicht, das zum Signieren der Nachricht verwendet wurde.

`UnsubscribeURL`

Eine URL, die Sie verwenden können, um den Endpunkt aus diesem Thema abzumelden. Wenn Sie diese URL besuchen, meldet Amazon SNS den Endpunkt ab und es werden keine weiteren Benachrichtigungen an diesen Endpunkt gesendet.

Die folgende „HTTP POST“-Nachricht ist ein Beispiel für eine `Notification`-Benachrichtigung, die an einen HTTP-Endpunkt gesendet wurde.

```
POST / HTTP/1.1
x-amz-sns-message-type: Notification
x-amz-sns-message-id: 22b80b92-fdea-4c2c-8f9d-bdfb0c7bf324
x-amz-sns-topic-arn: arn:aws:sns:us-west-2:123456789012:MyTopic
x-amz-sns-subscription-arn: arn:aws:sns:us-
west-2:123456789012:MyTopic:c9135db0-26c4-47ec-8998-413945fb5a96
Content-Length: 773
Content-Type: text/plain; charset=UTF-8
Host: myhost.example.com
Connection: Keep-Alive
User-Agent: Amazon Simple Notification Service Agent

{
  "Type" : "Notification",
  "MessageId" : "22b80b92-fdea-4c2c-8f9d-bdfb0c7bf324",
  "TopicArn" : "arn:aws:sns:us-west-2:123456789012:MyTopic",
```

```
  "Subject" : "My First Message",
  "Message" : "Hello world!",
  "Timestamp" : "2012-05-02T00:54:06.655Z",
  "SignatureVersion" : "1",
  "Signature" : "EXAMPLEw6JRN...",
  "SigningCertURL" : "https://sns.us-west-2.amazonaws.com/SimpleNotificationService-
f3ecfb7224c7233fe7bb5f59f96de52f.pem",
  "UnsubscribeURL" : "https://sns.us-west-2.amazonaws.com/?
Action=Unsubscribe&SubscriptionArn=arn:aws:sns:us-
west-2:123456789012:MyTopic:c9135db0-26c4-47ec-8998-413945fb5a96"
  }
}
```

JSON-Format für die HTTP/HTTPS-Abonnement-Abmeldung

Nachdem ein HTTP/HTTPS-Endpunkt aus einem Thema abgemeldet wurde, sendet Amazon SNS eine Abmelde-Bestätigung an den Endpunkt.

Bei der Abmelde-Bestätigung handelt es sich um eine POST-Nachricht mit einem Nachrichtentext, der ein JSON-Dokument mit den folgenden Name-Wert-Paaren enthält.

`Type`

Nachrichtentyp Für eine Abmelde-Bestätigung ist der Typ `UnsubscribeConfirmation`.

`MessageId`

Ein universell eindeutiger Bezeichner (UUID), der für jede veröffentlichte Benachrichtigung eindeutig ist. Für eine Nachricht, die Amazon SNS während eines Wiederholversuchs sendet, wird die Nachrichten-ID der ursprünglichen Nachricht verwendet.

`Token`

Ein Wert, den Sie mit der Aktion `ConfirmSubscription` verwenden können, um das Abonnement erneut zu bestätigen. Sie können aber auch einfach nur den Parameter besuchen `SubscribeURL`.

`TopicArn`

Amazon Resource Name (ARN) des Themas, von dem dieser Endpunkt abgemeldet wurde.

`Message`

Eine Zeichenfolge, die die Nachricht beschreibt. Bei einer Abmelde-Bestätigung sieht diese Zeichenfolge wie folgt aus:

```
You have chosen to deactivate subscription arn:aws:sns:us-
east-2:123456789012:MyTopic:2bcfbf39-05c3-41de-beaa-fcfcc21c8f55.\nTo cancel this
 operation and restore the subscription, visit the SubscribeURL included in this
 message.
```

`SubscribeURL`

Die URL, die Sie besuchen müssen, um das Abonnement erneut zu bestätigen. Alternativ können Sie auch den Parameter `Token` mit der Aktion `ConfirmSubscription`verwenden, um das Abonnement erneut zu bestätigen.

`Timestamp`

Uhrzeit (GMT), zu der die Abmelde-Bestätigung gesendet wurde.

`SignatureVersion`

Version der verwendeten Amazon-SNS-Signatur.

- Wenn `SignatureVersion` 1 lautet, ist `Signature` eine Base64-kodierte `SHA1withRSA`-Signatur der Werte `Message`, `MessageId`, `Type`, `Timestamp` und `TopicArn`.
- Wenn `SignatureVersion` 2 lautet, ist `Signature` eine Base64-kodierte `SHA256withRSA`-Signatur der Werte `Message`, `MessageId`, `Type`, `Timestamp` und `TopicArn`.

`Signature`

Eine Base64-kodierte `SHA1withRSA`- oder `SHA256withRSA`-Signatur der Werte `Message`, `MessageId`, `Type`, `Timestamp` und `TopicArn`.

`SigningCertURL`

Die URL, die den Zugriff auf das Zertifikat ermöglicht, das zum Signieren der Nachricht verwendet wurde.

Die folgende „HTTP POST“-Nachricht ist ein Beispiel für eine `UnsubscribeConfirmation`-Benachrichtigung, die an einen HTTP-Endpunkt gesendet wurde.

```
POST / HTTP/1.1
x-amz-sns-message-type: UnsubscribeConfirmation
x-amz-sns-message-id: 47138184-6831-46b8-8f7c-afc488602d7d
x-amz-sns-topic-arn: arn:aws:sns:us-west-2:123456789012:MyTopic
x-amz-sns-subscription-arn: arn:aws:sns:us-
west-2:123456789012:MyTopic:2bcfbf39-05c3-41de-beaa-fcfcc21c8f55
Content-Length: 1399
```

```
Content-Type: text/plain; charset=UTF-8
Host: myhost.example.com
Connection: Keep-Alive
User-Agent: Amazon Simple Notification Service Agent

{
  "Type" : "UnsubscribeConfirmation",
  "MessageId" : "47138184-6831-46b8-8f7c-afc488602d7d",
  "Token" : "2336412f37...",
  "TopicArn" : "arn:aws:sns:us-west-2:123456789012:MyTopic",
  "Message" : "You have chosen to deactivate subscription arn:aws:sns:us-
west-2:123456789012:MyTopic:2bcfbf39-05c3-41de-beaa-fcfcc21c8f55.\nTo cancel this
 operation and restore the subscription, visit the SubscribeURL included in this
 message.",
  "SubscribeURL" : "https://sns.us-west-2.amazonaws.com/?
Action=ConfirmSubscription&TopicArn=arn:aws:sns:us-
west-2:123456789012:MyTopic&Token=2336412f37fb6...",
  "Timestamp" : "2012-04-26T20:06:41.581Z",
  "SignatureVersion" : "1",
  "Signature" : "EXAMPLEHXgJm...",
  "SigningCertURL" : "https://sns.us-west-2.amazonaws.com/SimpleNotificationService-
f3ecfb7224c7233fe7bb5f59f96de52f.pem"
}
```

JSON-Format für die Zustellungsrichtlinie SetSubscriptionAttributes

Wenn Sie eine Anforderung an die Aktion `SetSubscriptionAttributes` senden und den Parameter `AttributeName` auf den Wert `DeliveryPolicy` festlegen, muss der Wert des Parameters `AttributeValue` ein gültiges JSON-Objekt sein. Im folgenden Beispiel wird die Zustellungsrichtlinie auf insgesamt 5 Wiederholversuche festgelegt.

```
http://sns.us-east-2.amazonaws.com/
?Action=SetSubscriptionAttributes
&SubscriptionArn=arn%3Aaws%3Asns%3Aus-east-2%3A123456789012%3AMy-Topic
%3A80289ba6-0fd4-4079-afb4-ce8c8260f0ca
&AttributeName=DeliveryPolicy
&AttributeValue={"healthyRetryPolicy":{"numRetries":5}}
...
```

Verwenden Sie das folgende JSON-Format für den Wert des Parameters `AttributeValue`.

```
{
```

```
    "healthyRetryPolicy" : {
        "minDelayTarget" :  int,
        "maxDelayTarget" : int,
        "numRetries" : int,
        "numMaxDelayRetries" : int,
        "backoffFunction" : "linear|arithmetic|geometric|exponential"
    },
    "throttlePolicy" : {
        "maxReceivesPerSecond" : int
    },
    "requestPolicy" : {
        "headerContentType" : "text/plain | application/json | application/xml"
    }
}
```

Weitere Informationen zur Aktion `SetSubscriptionAttribute` finden Sie unter SetSubscriptionAttributes in der Amazon-Simple-Notification-Service-API-Referenz. Weitere Hinweise zu den unterstützten HTTP-Inhaltstyp-Headern finden Sie unter Erstellen einer HTTP/S-Zustellungsrichtlinie.

JSON-Format für die Zustellungsrichtlinie SetTopicAttributes

Wenn Sie eine Anforderung an die Aktion `SetTopicAttributes` senden und den Parameter `AttributeName` auf den Wert `DeliveryPolicy` festlegen, muss der Wert des Parameters `AttributeValue` ein gültiges JSON-Objekt sein. Im folgenden Beispiel wird die Zustellungsrichtlinie auf insgesamt 5 Wiederholversuche festgelegt.

```
http://sns.us-east-2.amazonaws.com/
?Action=SetTopicAttributes
&TopicArn=arn%3Aaws%3Asns%3Aus-east-2%3A123456789012%3AMy-Topic
&AttributeName=DeliveryPolicy
&AttributeValue={"http":{"defaultHealthyRetryPolicy":{"numRetries":5}}}
...
```

Verwenden Sie das folgende JSON-Format für den Wert des Parameters `AttributeValue`.

```
{
    "http" : {
        "defaultHealthyRetryPolicy" : {
            "minDelayTarget":  int,
            "maxDelayTarget": int,
            "numRetries": int,
```

```
            "numMaxDelayRetries": int,
            "backoffFunction": "linear|arithmetic|geometric|exponential"
        },
        "disableSubscriptionOverrides" : Boolean,
        "defaultThrottlePolicy" : {
            "maxReceivesPerSecond" : int
        },
        "defaultRequestPolicy" : {
            "headerContentType" : "text/plain | application/json | application/xml"
        }
    }
}
```

Weitere Informationen zur`SetTopicAttribute`-Aktion finden Sie unter SetTopicAttributes im Amazon Simple Notification Service-API-Referenz. Weitere Hinweise zu den unterstützten HTTP-Inhaltstyp-Headern finden Sie unter Erstellen einer HTTP/S-Zustellungsrichtlinie.

Fanout zu AWS Event Fork Pipelines

Für die Ereignisarchivierung und -analyse empfiehlt Amazon SNS jetzt, seine native Integration mit Amazon Data Firehose zu verwenden. Sie können Firehose-Bereitstellungsdatenströme für SNS-Themen abonnieren, mit denen Sie Benachrichtigungen an Archivierungs- und Analyseen dpunkte wie Amazon Simple Storage Service (Amazon S3)-Buckets, Amazon-Redshift-Tabellen, Amazon OpenSearch Service (OpenSearch Service) und mehr senden können. Die Verwendun g von Amazon SNS mit Firehose-Bereitstellungsdatenströmen ist eine vollständig verwaltete und Codeless-Lösung, für die Sie keine -AWS LambdaFunktionen verwenden müssen. Weitere Informationen finden Sie unter Fanout zu Firehose-Bereitstellungsdatenströmen.

Sie können Amazon SNS zum Entwickeln ereignisgesteuerter Anwendungen verwenden, die Abonnenten-Services für die automatische Ausführung von Aufgaben als Antwort auf von Herausgeber-Services ausgelöste Ereignisse verwenden. Dieses Architekturmuster kann eine bessere Wiederverwendbarkeit, Interoperabilität und Skalierbarkeit von Services unterstützen. Es kann jedoch arbeitsaufwändig sein, die Verarbeitung von Ereignissen auf Pipelines aufzuteilen, die gemeinsame Anforderungen an die Ereignisbehandlung erfüllen, wie Speicherung, Backup, Suche, Analytik und Replay von Events.

Um die Entwicklung Ihrer ereignisgesteuerten Anwendungen zu beschleunigen, können Sie Pipelines für die Ereignisbehandlung abonnieren, die von AWS Event Fork Pipelines zu Amazon SNS-Themen

unterstützt werden. AWS Event Fork Pipelines ist eine Suite von verschachtelten Open-Source-Anwendungen, basierend auf einem AWS Serverless Application Model (AWS-SAM), die Sie direkt in der AWS Event Fork Pipelines Suite (wählen Sie Anzeigen von Apps, die benutzerdefinierte IAM-Rollen oder Ressourcenrichtlinien erstellen) in Ihrem AWS-Konto verfügbar machen können.

Einen AWS Anwendungsfall für Event Fork Pipelines finden Sie unter Bereitstellen und Testen der AWS Event Fork Pipelines – Beispielanwendung.

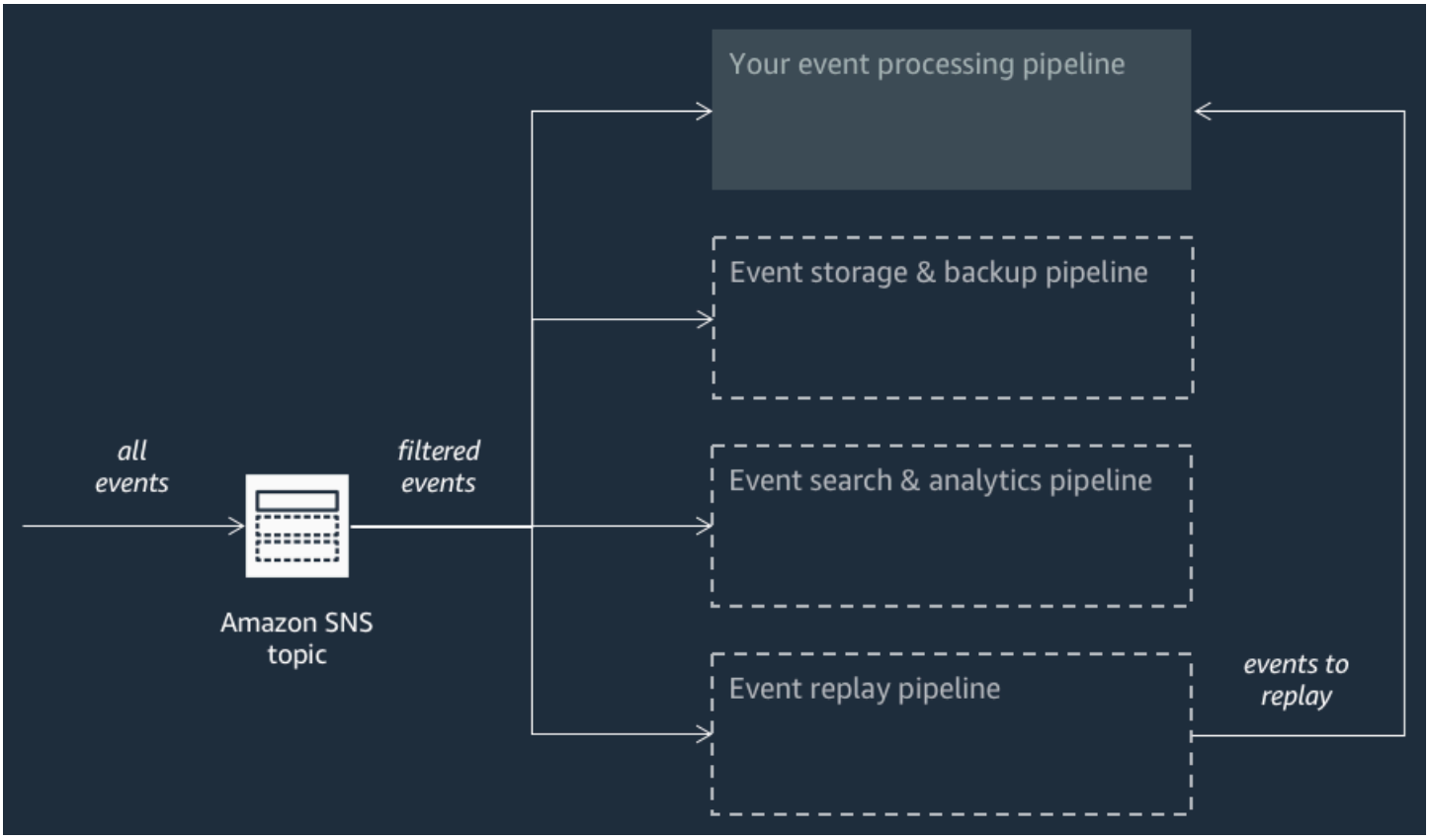
Themen

Wie AWS Event Fork Pipelines funktionieren

AWS Event Fork Pipelines haben ein Serverless Design. Es stellt jedoch auch eine Suite verschachtelter Serverless Anwendungen auf der Basis von AWS SAM dar (die Sie direkt aus AWS Serverless Application Repository (AWS SAR) Ihrem AWS-Konto bereitstellen können, um Ihre ereignisgesteuerten Plattformen zu ergänzen). Sie können diese verschachtelten Anwendungen einzeln bereitstellen wie von Ihrer Architektur gefordert.

Themen

Das folgende Diagramm zeigt eine AWS-Anwendung der Event Fork Pipelines, die von drei verschachtelten Anwendungen ergänzt wird. Sie können jede Pipeline aus der AWS Event Fork Pipelines Suite unabhängig voneinander in AWS SAR bereitstellen, wie von Ihrer Architektur gefordert.



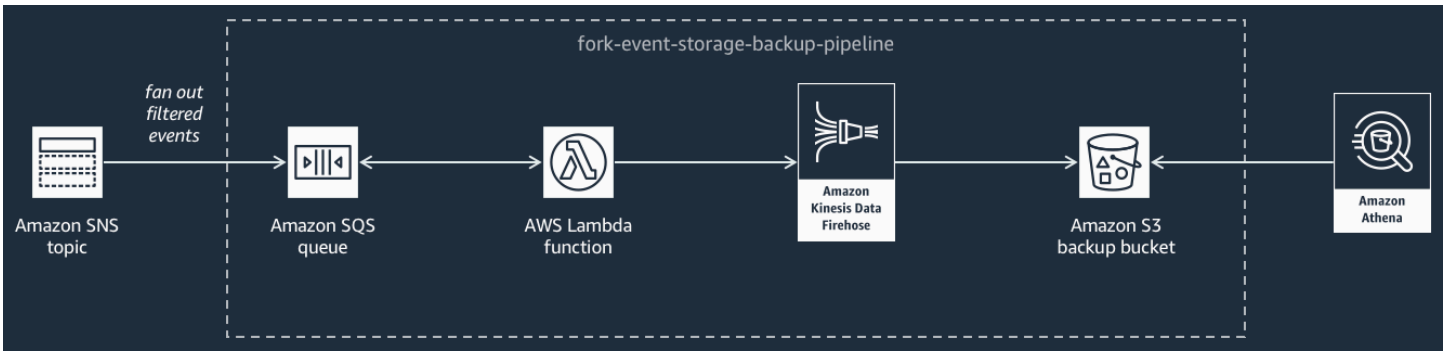
Jede Pipeline hat dasselbe Amazon SNS-Thema abonniert, um Ereignisse parallel verarbeiten zu können, wenn diese Ereignisse zum Thema veröffentlicht werden. Jede Pipeline ist unabhängig und Sie können eine eigene Abonnementfilterrichtlinie festlegen. So ist es möglich, dass eine Pipeline nur eine Teilmenge der Ereignisse verarbeitet - die von Interesse (anstelle aller zum Thema veröffentlichter Ereignisse).

> **Note**
>
> Da Sie die drei AWS Event Fork Pipelines neben Ihren regulären Pipelines für die Ereignisverarbeitung platzieren (die Ihr Amazon SNS-Thema möglicherweise bereits abonniert haben), müssen Sie keinen Abschnitt Ihres aktuellen Nachrichtenherausgebers ändern, um in Ihren vorhandenen Workloads AWS Event Fork Pipelines nutzen zu können.

Die Pipeline für die Speicherung und Sicherung von Ereignissen

Das folgende Diagramm zeigt die Pipeline für die Speicherung und Sicherung von Ereignissen. Sie können für diese Pipeline ein Abonnement Ihres Amazon SNS-Themas erstellen, um die Ereignisse in Ihrem System automatisch zu sichern.

Diese Pipeline besteht aus einer Amazon SQS-Warteschlange, die die vom Amazon SNS-Thema bereitgestellten Ereignisse puffert, einer -AWS LambdaFunktion, die diese Ereignisse automatisch in der Warteschlange abfragt und sie in einen Amazon-Data-Firehose-Stream überträgt, und einem Amazon S3-Bucket, der die vom Stream geladenen Ereignisse dauerhaft sichert.

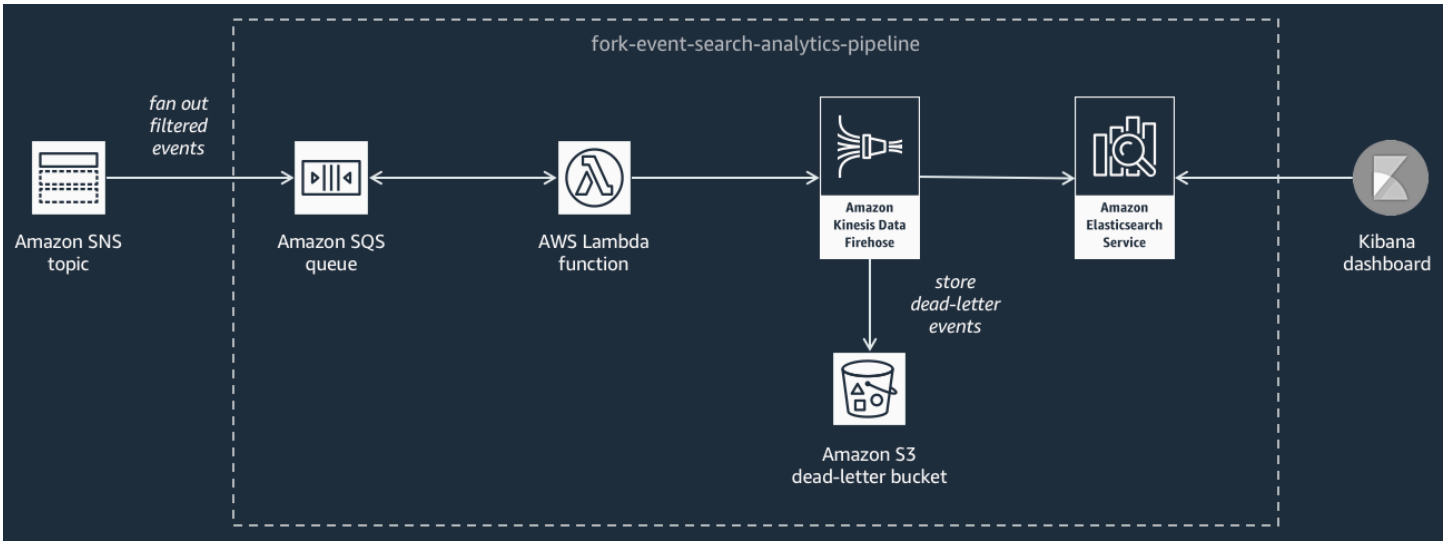


Um das Verhalten Ihres Firehose-Streams zu optimieren, können Sie ihn so konfigurieren, dass Ihre Ereignisse gepuffert, transformiert und komprimiert werden, bevor sie in den Bucket geladen werden. Während die Ereignisse geladen werden, können Sie Amazon Athena verwenden, um im Bucket SQL-Standardabfragen zu stellen. Sie können die Pipeline auch für die Wiederverwendung eines vorhandenen Amazon S3-Buckets konfigurieren oder einen neuen Bucket erstellen.

Die Pipeline für die Suche und Analyse von Ereignissen

Das folgende Diagramm zeigt die Pipeline für die Suche und Analyse von Ereignissen. Sie können für diese Pipeline ein Abonnement Ihres Amazon SNS-Themas erstellen, um die Ereignisse in Ihrem System in einer Suchdomäne zu indizieren und anschließend zu analysieren.

Diese Pipeline besteht aus einer Amazon SQS-Warteschlange, die die vom Amazon SNSThema bereitgestellten Ereignisse puffert, einer -AWS LambdaFunktion, die Ereignisse aus der Warteschlange abfragt und sie in einen Amazon-Data-Firehose-Stream überträgt, einer Amazon-OpenSearch Service-Domain, die die vom Firehose-Stream geladenen Ereignisse indiziert, und einem Amazon S3-Bucket, in dem die Ereignisse für unzustellbare Nachrichten gespeichert werden, die nicht in der Suchdomain indiziert werden können.



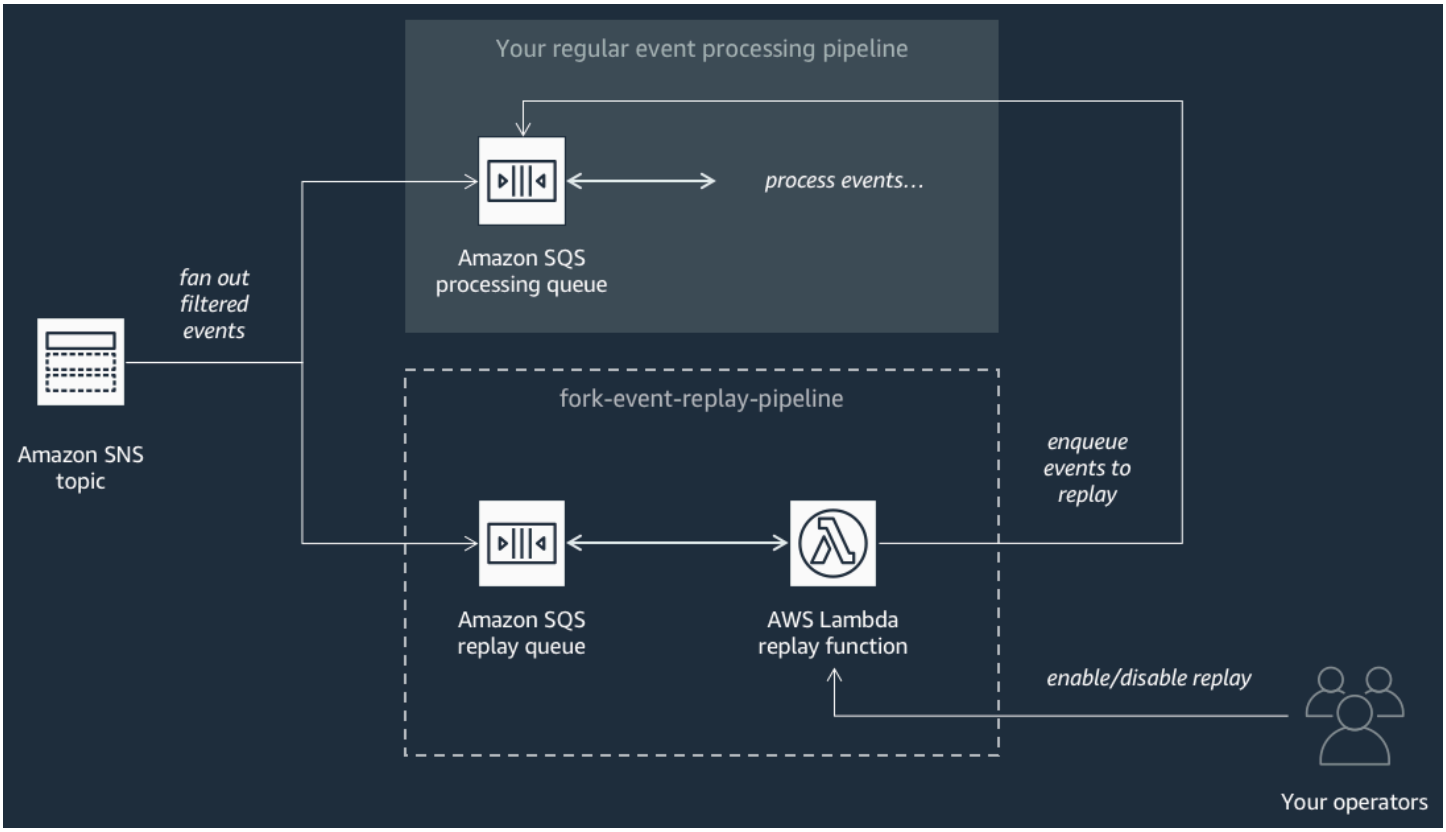
Um Ihren Firehose-Stream in Bezug auf die Pufferung, Transformierung und Komprimierung von Ereignissen zu optimieren, können Sie diese Pipeline konfigurieren.

Sie können auch konfigurieren, ob die Pipeline eine vorhandene OpenSearch Domain in Ihrem wiederverwenden AWS-Konto oder eine neue für Sie erstellen soll. Während der Indizierung von Ereignissen in der Suchdomäne können Sie mit Kibana Analysen für Ihre Ereignisse ausführen und visuelle Dashboards in Echtzeit aktualisieren.

Die Pipeline für das Replay von Events

Das folgende Diagramm zeigt die Pipeline für das Replay von Events. Um die Ereignisse aufzuzeichnen, die von Ihrem System in den letzten 14 Tagen verarbeitet wurden (beispielsweise für den Fall, dass Ihre Plattform nach einem Fehler wiederhergestellt werden muss), können Sie für diese Pipeline ein Abonnement Ihres Amazon SNS-Themas erstellen und anschließend die Ereignisse neu verarbeiten.

Diese Pipeline besteht aus einer Amazon SQS-Warteschlange, die die vom Amazon SNS-Thema bereitgestellten Ereignisse puffert, und einer AWS Lambda-Funktion, die Abrufe für diese Ereignisse aus der Warteschlange ausführt und sie erneut in die reguläre Pipeline für die Ereignisverarbeitung schiebt, die Ihr Thema abonniert hat.



> **Note**
>
> Standardmäßig ist die Funktion für das erneute Abspielen deaktiviert und schiebt Ihre Ereignisse nicht erneut in die reguläre Pipeline. Wenn Sie Ereignisse erneut verarbeiten müssen, müssen Sie die Amazon-SQS-Warteschlange für das erneute Abspielen als Ereignisquelle für die AWS Lambda-Funktion für das erneute Abspielen aktivieren.

AWS Event Fork Pipelines bereitstellen

Die AWSEvent Fork Pipelines Suite (verfügbar unter Show apps that create custom IAM roles or resource policies (Apps anzeigen, die angepasste IAM-Rollen oder Ressourcenrichtlinien erstellen)) ist als Gruppe öffentlicher Anwendungen in AWS Serverless Application Repositoryverfügbar. Sie können sie von hier über die AWS Lambda-Konsole bereitstellen und manuell testen. Informationen zum Bereitstellen von Pipelines über die AWS Lambda-Konsole finden Sie unter Abonnieren eine AWS Event Fork Pipeline für ein Amazon SNS-Thema.

In einem Produktionsszenario empfehlen wir die Einbettung von AWS Event Fork Pipelines innerhalb der allgemeinen AWSSAM-Vorlage Ihrer Anwendung. Sie können dies mittels der Funktion

„nested-application“ ausführen, indem Sie die Ressource `AWS::Serverless::Application` Ihrer AWSSAM-Vorlage hinzufügen, wobei sie auf die AWS SAR-`ApplicationId` und die `SemanticVersion` der verschachtelten Anwendung verweisen.

Sie können beispielsweise die Event Storage and Backup Pipeline für die Speicherung und Sicherung von Ereignissen als eine verschachtelte Anwendung verwenden, indem Sie den folgenden YAML-Codeausschnitt zum Abschnitt `Resources` Ihrer AWS SAM-Vorlage hinzufügen.

```
Backup:
  Type: AWS::Serverless::Application
  Properties:
    Location:
      ApplicationId: arn:aws:serverlessrepo:us-east-2:123456789012:applications/fork-
event-storage-backup-pipeline
      SemanticVersion: 1.0.0
    Parameters:
      #The ARN of the Amazon SNS topic whose messages should be backed up to the Amazon
 S3 bucket.
      TopicArn: !Ref MySNSTopic
```

Wenn Sie Parameterwerte angeben, können Sie intrinsische AWS CloudFormation-Funktionen verwenden, um auf andere Ressourcen in Ihrer Vorlage zu verweisen. Im oben gezeigten YAML-Codeauszug verweist der Parameter `TopicArn`beispielsweise auf die `AWS::SNS::Topic`-Ressourcen-`MySNSTopic`, die an anderer Stelle in der AWS SAM-Vorlage definiert wird. Weitere Informationen finden Sie in der Referenz für intrinsische Funktion im AWS CloudFormation-Benutzerhandbuch.

> **Note**
>
> Die AWS Lambda-Konsolenseite für Ihre AWS-SAR-Anwendung enthält die Schaltfläche Copy as SAM Resource (Als SAM-Ressource kopieren), die den für die Verschachtelung einer AWS-SAR-App notwendigen YAML-Code in die Zwischenablage kopiert.

Bereitstellen und Testen der AWS Event Fork Pipelines – Beispielanwendung

Um die Entwicklung Ihrer ereignisgesteuerten Anwendungen zu beschleunigen, können Sie Pipelines für die Ereignisbehandlung abonnieren, die von AWS Event Fork Pipelines zu Amazon SNS-Themen

unterstützt werden. AWS Event Fork Pipelines ist eine Suite von verschachtelten Open-Source-Anwendungen, basierend auf einem AWS Serverless Application Model (AWS-SAM), die Sie direkt in der AWS Event Fork Pipelines Suite (wählen Sie Anzeigen von Apps, die benutzerdefinierte IAM-Rollen oder Ressourcenrichtlinien erstellen) in Ihrem AWS-Konto verfügbar machen können. Weitere Informationen finden Sie unter Wie AWS Event Fork Pipelines funktionieren.

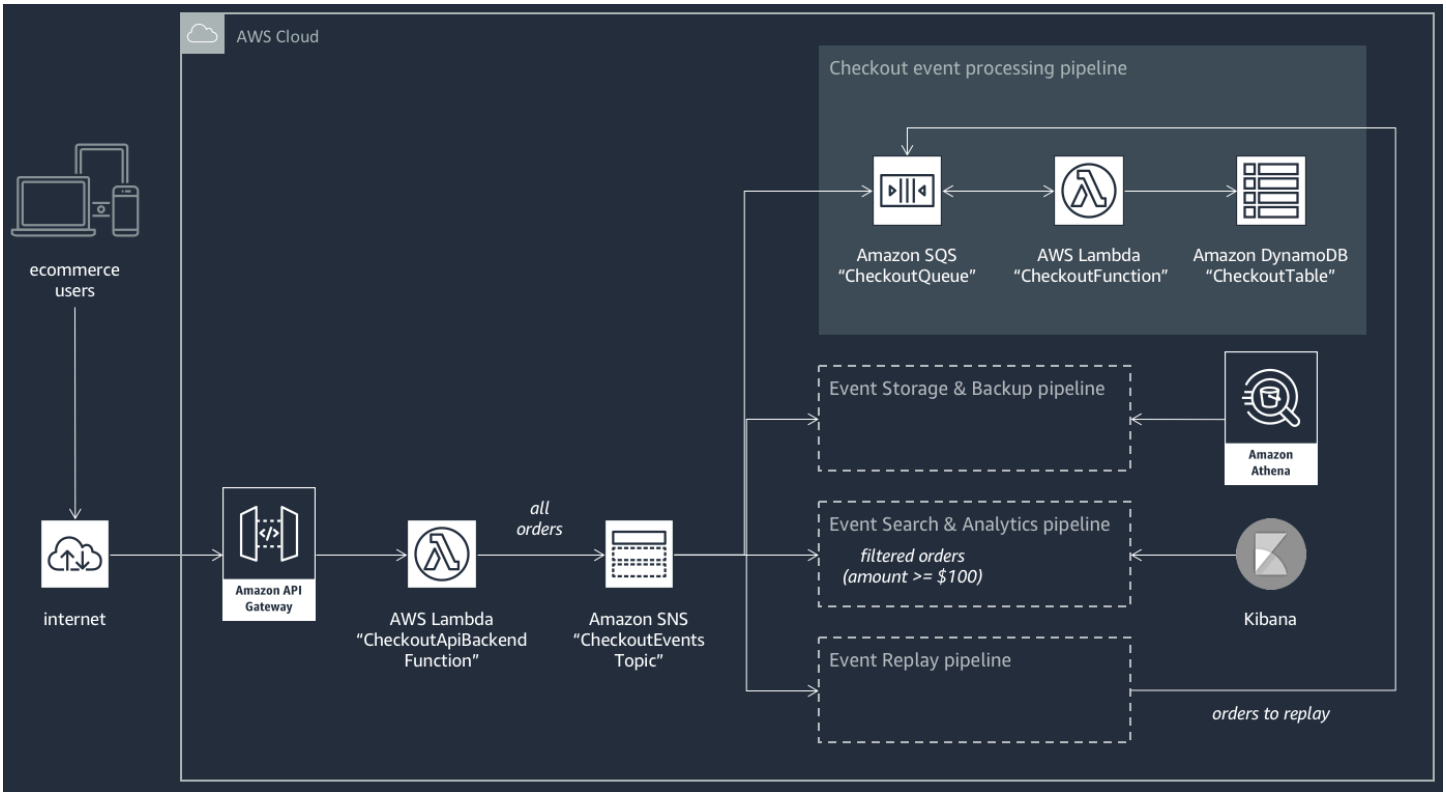
Diese Seite zeigt, wie Sie AWS Management Console zum Bereitstellen und Testen der AWSBeispielanwendung für Event Fork Pipelines nutzen können.

> **Important**
>
> Um unerwünschte Kosten zu vermeiden, nachdem Sie die AWSEvent Fork Pipelines-Beispielanwendung bereitgestellt haben, löschen Sie den zugehörigen AWS CloudFormation Stack. Weitere Informationen finden Sie unter Löschen eines Stacks in der AWS CloudFormation Konsole im AWS CloudFormationBenutzerhandbuch.

Themen

BeispielAWS Event Fork-Pipelines – Anwendungsfall

Das folgende Szenario beschreibt eine ereignisgesteuerte, Serverless-e-Commerce-Anwendung, die AWS Event Fork Pipeline verwendet. Sie können diese Beispiel-E-Commerce-Anwendung in der verwenden AWS Serverless Application Repository und sie dann AWS-Konto mithilfe der AWS LambdaKonsole in Ihrem bereitstellen, wo Sie sie testen und ihren Quellcode in untersuchen können GitHub.



Diese e-Commerce-Anwendung nimmt Aufträge von Käufern über eine von API Gateway gehostete RESTful-API und mit Unterstützung der AWS Lambda-Funktion `CheckoutApiBackendFunction` entgegen. Diese Funktion veröffentlicht alle empfangenen Aufträge zu einem Amazon SNS-Thema mit dem Namen `CheckoutEventsTopic`, das seinerseits die Aufträge an vier unterschiedliche Pipelines weitergibt.

Die erste Pipeline ist die reguläre Checkout-Verarbeitungs-Pipeline, die vom Eigentümer der e-Commerce-Anwendung entworfen und implementiert wurde. Diese Pipeline besitzt die Amazon SQS WarteschlangeCheckoutQueue, die alle empfangenen Bestellungen puffert, eine AWS Lambda-Funktion mit dem NamenCheckoutFunction, welche die Warteschlange zur Verarbeitung dieser Bestellungen abfragt, und die DynamoDB-Tabelle `CheckoutTable`, die alle platzierten Bestellungen sicher speichert.

Anwendung vonAWS Event Fork Pipelines

Die Komponenten der e-Commerce-Anwendung kümmern sich um die zentrale Geschäftslogik. Der Eigentümer der e-Commerce-Anwendung muss sich aber auch um Folgendes kümmern:

- Compliance – sichere, komprimierte Backups, die im Ruhezustand verschlüsselt sind und die Bereinigung sensibler Informationen

- Ausfallsicherheit – Wiederholung aktueller Bestellungen im Falle einer Unterbrechung des Fulfillment-Prozesses
- Durchsuchbarkeit – Ausführen von Analytik und Generierung von Metriken für aufgegebene Aufträge

Anstatt diese Ereignisverarbeitungslogik zu implementieren kann der Eigentümer der Anwendung AWSdas Amazon SNS-Thema `CheckoutEventsTopic` abonnieren lassen

- Die Pipeline für die Speicherung und Sicherung von Ereignissen ist für die Transformation von Daten konfiguriert, sodass Kreditkartendaten entfernt, Daten 60 Sekunden gepuffert, Daten mit GZIP komprimiert und mit dem verwalteten Standardschlüssel für Amazon S3 verschlüsselt werden. Dieser Schlüssel wird von AWS verwaltet und von AWS Key Management Service (AWS KMS) unterstützt.

  Weitere Informationen finden Sie unter Wählen Sie Amazon S3 für Ihr Ziel , Amazon Data Firehose Data Transformation und Konfigurieren von Einstellungen im Amazon Data Firehose-Entwicklerhandbuch.
- Die Pipeline für die Suche und Analyse von Ereignissen ist mit einer Index-Wiederholungsdauer von 30 Sekunden konfiguriert, sowie mit einem Bucket für die Speicherung von Aufträgen, die in der Suchdomäne nicht indiziert werden können, und einer Filterrichtlinie zur Einschränkung der indizierten Aufträge.

  Weitere Informationen finden Sie unter Wählen Sie OpenSearch Service für Ihr Ziel im Amazon-Data-Firehose-Entwicklerhandbuch.
- Die Pipeline für das Replay von Events ist mit dem Amazon-SQS-Warteschlangenteil der normalen Auftragsverarbeitungspipeline konfiguriert, die von Eigentümer der e-Commerce-Anwendung entworfen und implementiert wurde.

  Weitere Informationen finden Sie unter Name der Warteschlange und URL im Amazon Simple Queue Service Leitfaden für Entwickler.

Die folgende JSON-Filterrichtlinie ist in der Konfiguration für die Ereignissuche- und Analysepipeline eingestellt. Sie gleicht nur eingehende Aufträge mit einem Gesamtbetrag von mindestens 100 USD ab. Weitere Informationen finden Sie unter Amazon SNS Nachrichtenfilterung.

```
{
   "amount": [{ "numeric": [ ">=", 100 ] }]
```

```
}
```

Durch die Verwendung des AWS Event-Fork-Pipelines-Musters kann der Besitzer der e-Commerce-Anwendung Entwicklungslasten vermeiden, die oft mit der Kodierung nicht-differenzierender Logik für den Umgang mit Ereignissen zusammenhängen. Stattdessen kann AWS Event Fork Pipelines direkt aus AWS Serverless Application Repository in AWS-Konto bereitgestellt werden.

Schritt 1: So stellen Sie die Beispielanwendung bereit

1. Melden Sie sich an der AWS Lambda-Konsole an.
2. Wählen Sie im Navigationsbereich Functions (Funktionen) und dann Create function (Funktion erstellen) aus.
3. Gehen Sie auf der Seite Create function (Funktion erstellen) wie folgt vor:

   a. Wählen Sie Browse serverless app repository (Repository mit Serverless-Apps durchsuchen), Public applications (Öffentliche Anwendungen), Show apps that create custom roles or resource policies (Apps anzeigen, die benutzerdefinierte IAM-Rollen oder Ressourcenrichtlinien erstellen).

   b. Suchen Sie nach `fork-example-ecommerce-checkout-api` und wählen Sie dann die Anwendung aus.
4. Gehen Sie auf der Seite fork-example-ecommerce-checkout-api wie folgt vor:

   a. Geben Sie im Abschnitt Application settings (Anwendungseinstellungen) einen Application name (Anwendungsname) ein (zum Beispiel `fork-example-ecommerce-my-app`).

   > **Note**
   >
   > - Um Ihre Ressourcen später leichter finden zu können, behalten Sie das Präfix `fork-example-ecommerce`.
   > - Für jede Bereitstellung muss der Anwendungsname eindeutig sein. Wenn Sie einen Anwendungsnamen wiederverwenden, aktualisiert die Bereitstellung nur den zuvor bereitgestellten AWS CloudFormation-Stack (anstatt einen neuen zu erstellen).

   b. (Optional) Geben Sie eine der folgenden LogLevel Einstellungen für die Ausführung der Lambda-Funktion Ihrer Anwendung ein:

- `DEBUG`
- `ERROR`
- `INFO` (Standard)
- `WARNING`

5. Wählen Sie I acknowledge that this app creates custom IAM roles, resource policies and deploys nested applications (Ich bestätige, dass diese App benutzerdefinierte IAM-Rollen und Ressourcenrichtlinien erstellt und eingebettete Anwendungen bereitstellt) und dann unten auf der Seite Deploy (Bereitstellen) aus.

Auf der Seite Bereitstellungsstatus für fork-example-ecommerce-*my-app* zeigt Lambda den Status Ihre Anwendung wird bereitgestellt an.

Im Abschnitt Resources (Ressourcen) beginnt AWS CloudFormation mit der Erstellung des Stacks und zeigt den Status CREATE_IN_PROGRESS für jede Ressource an. Wenn der Prozess abgeschlossen ist, zeigt AWS CloudFormation den Status CREATE_COMPLETE an.

> **Note**
>
> Es kann 20 bis 30 Minuten dauern, bis alle Ressourcen bereitgestellt sind.

Wenn die Bereitstellung abgeschlossen ist, zeigt Lambda den Status Your application has been deployed (Ihre Anwendung wurde bereitgestellt) an.

Schritt 2: So führen Sie die Beispielanwendung aus

1. Wählen Sie in der AWS Lambda-Konsole im Navigationsbereich Applications (Anwendungen).
2. Suchen Sie auf der Seite Applications (Anwendungen) im Suchfeld nach `serverlessrepo-fork-example-ecommerce-`*`my-app`* und wählen Sie dann die Anwendung aus.
3. Gehen Sie im Abschnitt Resources (Ressourcen) wie folgt vor:

   a. Um die Ressource zu finden, deren Typ istApiGatewayRestApi, sortieren Sie die Ressourcen nach Typ , z. B. `ServerlessRestApi`, und erweitern Sie dann die Ressource.

   b. Es werden zwei verschachtelte Ressourcen angezeigt, vom Typ ApiGateway Bereitstellung und ApiGateway Stufe .

c. Kopieren Sie den Link Prod API endpoint (Prod-API-Endpunkt) und fügen Sie diesem `/checkout` an, z. B.:

```
https://abcdefghij.execute-api.us-east-2.amazonaws.com/Prod/checkout
```

4. Kopieren Sie den folgenden JSON-Code in eine Datei mit dem Namen `test_event.json`.

```
{
   "id": 15311,
   "date": "2019-03-25T23:41:11-08:00",
   "status": "confirmed",
   "customer": {
      "id": 65144,
      "name": "John Doe",
      "email": "john.doe@example.com"
   },
   "payment": {
      "id": 2509,
      "amount": 450.00,
      "currency": "usd",
      "method": "credit",
      "card-network": "visa",
      "card-number": "1234 5678 9012 3456",
      "card-expiry": "10/2022",
      "card-owner": "John Doe",
      "card-cvv": "123"
   },
   "shipping": {
      "id": 7600,
      "time": 2,
      "unit": "days",
      "method": "courier"
   },
   "items": [{
      "id": 6512,
      "product": 8711,
      "name": "Hockey Jersey - Large",
      "quantity": 1,
      "price": 400.00,
      "subtotal": 400.00
   }, {
      "id": 9954,
      "product": 7600,
```

```
      "name": "Hockey Puck",
      "quantity": 2,
      "price": 25.00,
      "subtotal": 50.00
   }]
}
```

5. Um eine HTTPS-Anfrage an Ihren API-Endpunkt zu senden, übergeben Sie die Nutzlast des Beispielereignisses als Eingabe, indem Sie einen `curl`-Befehl ausführen, zum Beispiel:

   ```
   curl -d "$(cat test_event.json)" https://abcdefghij.execute-api.us-
   east-2.amazonaws.com/Prod/checkout
   ```

   Die API gibt die folgende leere Antwort unter Angabe einer erfolgreichen Ausführung zurück:

   ```
   { }
   ```

Schritt 3: So prüfen Sie die Ausführung der Beispielanwendung und ihrer Pipelines

Schritt 1: So prüfen Sie die Ausführung der Beispiel-Checkout-Pipeline

1. Melden Sie sich bei der Amazon DynamoDB-Konsole an.
2. Wählen Sie im Navigationsbereich Tables (Tabellen) aus.
3. Suchen Sie nach `serverlessrepo-fork-example` und wählen Sie `CheckoutTable` aus.
4. Wählen Sie auf der Tabellendetailseite Items (Elemente) und dann das erstellte Element aus.

   Die gespeicherten Attribute werden angezeigt.

Schritt 2: So prüfen Sie die Ausführung der Pipeline für die Speicherung und Sicherung von Ereignissen

1. Melden Sie sich bei der Amazon S3-Konsole an.
2. Wählen Sie im Navigationsbereich Buckets (Buckets) aus.
3. Suchen Sie nach `serverlessrepo-fork-example` und wählen Sie dann `CheckoutBucket`.
4. Navigieren Sie durch die Verzeichnishierarchie, bis Sie eine Datei mit der Erweiterung finden `.gz`.
5. Um die Datei herunterzuladen, wählen Sie Actions (Aktionen), Open (Öffnen).

6. Die Pipeline ist mit einer Lambda-Funktion konfiguriert, die Kreditkarteninformationen aus Gründen der Compliance schützt.

   Um zu überprüfen, ob die gespeicherte JSON-Nutzlast Kreditkarteninformationen enthält, dekomprimieren Sie die Datei.

Schritt 3: So prüfen Sie die Ausführung der Ereignissuche- und Analysepipeline

1. Melden Sie sich bei der OpenSearch Servicekonsole an.
2. Wählen Sie im Navigationsbereich in My domains (Meine Domains) die Domäne mit dem Präfix `serverl-analyt` aus.
3. Die Pipeline ist mit einer Amazon SNS-Abonnement-Filterrichtlinie konfiguriert, die eine numerische Abgleichbedingung einrichtet.

   Um zu prüfen, ob ein Ereignis indiziert ist, da es sich auf einen Auftrag mit einem Wert über 100 USD bezieht, wählen Sie auf der Seite serverl-analyt-***abcdefgh1ijk*** Indices, checkout_events.

Schritt 4: So prüfen Sie die Ausführung der Ereignis-Wiederholungspipeline

1. Melden Sie sich bei der Amazon SQS-Konsole an.
2. Suchen Sie in der Liste der Warteschlangen nach `serverlessrepo-fork-example` und wählen Sie `ReplayQueue` aus.
3. Wählen Sie Nachrichten senden und empfangen.
4. In der Datei Senden und Empfangen von Nachrichten in fork-example-ecommerce-***my-app*** ...ReplayP – Dialogfeld -ReplayQueue-***123ABCD4E5F6***, wählen Sie Abfrage für Nachrichten aus.
5. Um zu prüfen, ob sich das Ereignis in der Warteschlange befindet, wählen Sie More Details (Weitere Einzelheiten) neben der Nachricht, die in der Warteschlange angezeigt wird.

Schritt 4: So simulieren Sie ein Problem und wiederholen Ereignisse zur Wiederherstellung

Schritt 1: So aktivieren Sie das simulierte Problem und senden eine zweite API-Anforderung

1. Melden Sie sich an der AWS Lambda-Konsole an.
2. Wählen Sie im Navigationsbereich Functions (Funktionen) aus.
3. Suchen Sie nach `serverlessrepo-fork-example`und wählen Sie `CheckoutFunction` aus.

4. Setzen Sie auf der Seite fork-example-ecommerce-***my-app*** -CheckoutFunction-***ABCDEF*** ... im Abschnitt Umgebungsvariablen die Variable BUG_ENABLED auf true und wählen Sie dann Speichern aus.

5. Kopieren Sie den folgenden JSON-Code in eine Datei mit dem Namen `test_event_2.json`.

```
{
    "id": 9917,
    "date": "2019-03-26T21:11:10-08:00",
    "status": "confirmed",
    "customer": {
       "id": 56999,
       "name": "Marcia Oliveira",
       "email": "marcia.oliveira@example.com"
    },
    "payment": {
       "id": 3311,
       "amount": 75.00,
       "currency": "usd",
       "method": "credit",
       "card-network": "mastercard",
       "card-number": "1234 5678 9012 3456",
       "card-expiry": "12/2025",
       "card-owner": "Marcia Oliveira",
       "card-cvv": "321"
    },
    "shipping": {
       "id": 9900,
       "time": 20,
       "unit": "days",
       "method": "plane"
    },
    "items": [{
       "id": 9993,
       "product": 3120,
       "name": "Hockey Stick",
       "quantity": 1,
       "price": 75.00,
       "subtotal": 75.00
    }]
 }
```

6. Um eine HTTPS-Anfrage an Ihren API-Endpunkt zu senden, übergeben Sie die Nutzlast des Beispielereignisses als Eingabe, indem Sie einen `curl`-Befehl ausführen, zum Beispiel:

   ```
   curl -d "$(cat test_event_2.json)" https://abcdefghij.execute-api.us-
   east-2.amazonaws.com/Prod/checkout
   ```

   Die API gibt die folgende leere Antwort unter Angabe einer erfolgreichen Ausführung zurück:

   ```
   { }
   ```

Schritt 2: So verifizieren Sie eine simulierte Datenbeschädigung

1. Melden Sie sich bei der Amazon DynamoDB-Konsole an.
2. Wählen Sie im Navigationsbereich Tables (Tabellen) aus.
3. Suchen Sie nach `serverlessrepo-fork-example` und wählen Sie `CheckoutTable` aus.
4. Wählen Sie auf der Tabellendetailseite Items (Elemente) und dann das erstellte Element aus.

   Die gespeicherten Attribute werden angezeigt, einige markiert als CORRUPTED (BESCHÄDIGT!)

Schritt 3: So deaktivieren Sie das simulierte Problem

1. Melden Sie sich an der AWS Lambda-Konsole an.
2. Wählen Sie im Navigationsbereich Functions (Funktionen) aus.
3. Suchen Sie nach `serverlessrepo-fork-exampleund` wählen Sie `CheckoutFunction` aus.
4. Legen Sie auf der Seite fork-example-ecommerce-***my-app*** -CheckoutFunction-***ABCDEF*** ... im Abschnitt Umgebungsvariablen die Variable BUG_ENABLED auf false fest und wählen Sie dann Speichern aus.

Schritt 4: So aktivieren Sie die Wiederholung zur Wiederherstellung nach dem Problem

1. Wählen Sie in der AWS Lambda-Konsole im Navigationsbereich Functions (Funktionen).
2. Suchen Sie nach `serverlessrepo-fork-example` und wählen Sie `ReplayFunction` aus.
3. Erweitern Sie den Abschnitt Designer (Designer), wählen Sie das Feld SQS (SQS) und dann im Abschnitt SQS (SQS) Enabled (Aktiviert) aus.

> **Note**
>
> Es dauert ca. 1 Minute, bis der Amazon SQS-Ereignisquellenauslöser aktiviert wird.

4. Wählen Sie Speichern.
5. Zur Anzeige der wiederhergestellten Attribute kehren Sie zur Amazon–DynamoDB-Konsole zurück.
6. Kehren Sie zur Deaktivierung der Wiederholung zur AWS Lambda-Konsole zurück und deaktivieren Sie den Amazon SQS-Ereignisquellenauslöser für `ReplayFunction`.

Abonnieren eine AWS Event Fork Pipeline für ein Amazon SNS-Thema

Um die Entwicklung Ihrer ereignisgesteuerten Anwendungen zu beschleunigen, können Sie Pipelines für die Ereignisbehandlung abonnieren, die von AWS Event Fork Pipelines zu Amazon SNS-Themen unterstützt werden. AWS Event Fork Pipelines ist eine Suite von verschachtelten Open-Source-Anwendungen, basierend auf einem AWS Serverless Application Model (AWS-SAM), die Sie direkt in der AWS Event Fork Pipelines Suite (wählen Sie Anzeigen von Apps, die benutzerdefinierte IAM-Rollen oder Ressourcenrichtlinien erstellen) in Ihrem AWS-Konto verfügbar machen können. Weitere Informationen finden Sie unter Wie AWS Event Fork Pipelines funktionieren.

In diesem Abschnitt wird gezeigt, wie Sie das AWS Management Console nutzen, um eine Pipeline bereitzustellen und dann AWS verschachtelte Pipelines zu einem Amazon SNS -Thema abonnieren können. Bevor Sie beginnen, erstellen Sie ein Amazon SNS-Thema.

Um die Ressourcen zu löschen, aus denen eine Pipeline besteht, suchen Sie die Pipeline auf der Seite Anwendungen von in der AWS LambdaKonsole, erweitern Sie den Abschnitt SAM-Vorlage, wählen Sie CloudFormation Stack und dann Andere Aktionen, Stack löschen aus.

Themen

- So stellen Sie die Pipeline für die Speicherung und Sicherung von Ereignissen bereit und abonnieren sie
- So stellen Sie die Ereignissuche- und Analyse-Pipeline bereit und abonnieren sie
- So stellen Sie die Ereignis-Wiederholungspipeline bereit und abonnieren sie

So stellen Sie die Pipeline für die Speicherung und Sicherung von Ereignissen bereit und abonnieren sie

> Für die Ereignisarchivierung und -analyse empfiehlt Amazon SNS jetzt, seine native Integration mit Amazon Data Firehose zu verwenden. Sie können Firehose-Bereitstellungsdatenströme für SNS-Themen abonnieren, mit denen Sie Benachrichtigungen an Archivierungs- und Analyseen dpunkte wie Amazon Simple Storage Service (Amazon S3)-Buckets, Amazon-Redshift-Tabellen, Amazon OpenSearch Service (OpenSearch Service) und mehr senden können. Die Verwendun g von Amazon SNS mit Firehose-Bereitstellungsdatenströmen ist eine vollständig verwaltete und codelose Lösung, für die Sie keine -AWS LambdaFunktionen verwenden müssen. Weitere Informationen finden Sie unter Fanout zu Firehose-Bereitstellungsdatenströmen.

In diesem Tutorial wird gezeigt, wie Sie die Event Storage Backup Pipeline bereitstellen und ein Amazon-SNS-Thema abonnieren. Dieser Prozess macht die der Pipeline zugewiesenen AWS SAM-Vorlage automatisch zu einen AWS CloudFormation Stack. Anschließend wird der Stack in Ihrem AWS-Konto-Konto bereitgestellt. Dieser Prozess erstellt und konfiguriert außerdem eine Reihe von Ressourcen, den Ereignisspeicher und die Sicherungspipeline umfassen, darunter:

- Amazon-SQS-Warteschlange
- Lambda-Funktion
- Firehose-Bereitstellungsdat
- Amazon S3 Backup Bucket

Weitere Informationen zum Konfigurieren eines Streams mit einem S3-Bucket als Ziel finden Sie unter `S3DestinationConfiguration` in der API-Referenz zu Amazon Data Firehose.

Weitere Informationen zum Transformieren von Ereignissen und zum Konfigurieren von Ereignispufferung, Ereigniskomprimierung und Ereignisverschlüsselung finden Sie unter Erstellen eines Amazon-Data-Firehose-Bereitstellungsdatenstroms im Amazon-Data-Firehose-Entwicklerhandbuch.

Weitere Informationen zur Filterung von Ereignissen finden Sie unter Filterrichtlinien für Amazon-SNS-Abonnements in diesem Leitfaden.

1. Melden Sie sich an der AWS Lambda-Konsole an.

2. Wählen Sie im Navigationsbereich Functions (Funktionen) und dann Create function (Funktion erstellen) aus.
3. Gehen Sie auf der Seite Create function (Funktion erstellen) wie folgt vor:

   a. Wählen Sie Browse serverless app repository (Repository mit Serverless-Apps durchsuchen), Public applications (Öffentliche Anwendungen), Show apps that create custom roles or resource policies (Apps anzeigen, die benutzerdefinierte IAM-Rollen oder Ressourcenrichtlinien erstellen).

   b. Suchen Sie nach `fork-event-storage-backup-pipeline` und wählen Sie dann die Anwendung aus.
4. Gehen Sie auf der Seite fork-event-storage-backup-pipeline wie folgt vor:

   a. Geben Sie im Abschnitt Application settings (Anwendungseinstellungen) einen Application name (Anwendungsname) ein (zum Beispiel `my-app-backup`).

   > **Note**
   >
   > - Für jede Bereitstellung muss der Anwendungsname eindeutig sein. Wenn Sie einen Anwendungsnamen wiederverwenden, aktualisiert die Bereitstellung nur den zuvor bereitgestellten AWS CloudFormation-Stack (anstatt einen neuen zu erstellen).

   b. (Optional) BucketArnGeben Sie für den ARN des S3-Buckets ein, in den eingehende Ereignisse geladen werden. Wenn Sie keinen Wert eingeben, wird in Ihrem AWS-Konto ein neuer S3-Bucket erstellt.

   c. (Optional) DataTransformationFunctionArnGeben Sie für den ARN der Lambda-Funktion ein, durch die die eingehenden Ereignisse transformiert werden. Wenn Sie keinen Wert eingeben, wird die Datentransformation deaktiviert.

   d. (Optional) Geben Sie eine der folgenden LogLevel Einstellungen für die Ausführung der Lambda-Funktion Ihrer Anwendung ein:

      - `DEBUG`
      - `ERROR`
      - `INFO` (Standard)
      - `WARNING`

e. TopicArnGeben Sie für den ARN des Amazon SNS-Themas ein, das diese Instance der Fork-Pipeline abonnieren soll.

f. (Optional) Geben Sie für StreamBufferingIntervalInSeconds und StreamBufferingSizeInMBs die Werte für die Konfiguration der Pufferung eingehender Ereignisse ein. Wenn Sie keine Werte eingeben, 300 Sekunden und 5 MB verwendet.

g. (Optional) Geben Sie eine der folgenden StreamCompressionFormat Einstellungen für die Komprimierung eingehender Ereignisse ein:

   - `GZIP`
   - `SNAPPY`
   - `UNCOMPRESSED` (Standard)
   - `ZIP`

h. (Optional) Geben Sie für das Zeichenfolgenpräfix einStreamPrefix, um Dateien zu benennen, die im S3-Sicherungs-Bucket gespeichert sind. Wenn Sie keinen Wert eingeben, wird kein Präfix verwendet.

i. (Optional) SubscriptionFilterPolicyGeben Sie für die Filterrichtlinie für Amazon SNS-Abonnements im JSON-Format ein, die zum Filtern eingehender Ereignisse verwendet werden soll. Die Filterrichtlinie bestimmt, welche Ereignisse im OpenSearch Service-Index indiziert werden. Wenn Sie keinen Wert eingeben, wird keine Filterung verwendet wird (alle Ereignisse werden indiziert).

j. (Optional) Geben Sie für die Zeichenfolge `MessageBody` oder einSubscriptionFilterPolicyScope, `MessageAttributes` um die nutzlastbasierte oder attributbasierte Nachrichtenfilterung zu aktivieren.

k. Wählen Sie I acknowledge that this app creates custom IAM roles, resource policies and deploys nested applications (Ich bestätige, dass diese App benutzerdefinierte IAM-Rollen und Ressourcenrichtlinien erstellt und eingebettete Anwendungen bereitstellt) und dann Deploy (Bereitstellen).

Auf der Seite Deployment status for ***my-app*** (Bereitstellungsstatus für my-app) zeigt Lambda den Status Your application is being deployed (Ihre Anwendung wird bereitgestellt) an.

Im Abschnitt Resources (Ressourcen) beginnt AWS CloudFormation mit der Erstellung des Stacks und zeigt den Status CREATE_IN_PROGRESS für jede Ressource an. Wenn der Prozess abgeschlossen ist, zeigt AWS CloudFormation den Status CREATE_COMPLETE an.

Wenn die Bereitstellung abgeschlossen ist, zeigt Lambda den Status Your application has been deployed (Ihre Anwendung wurde bereitgestellt) an.

Zu Ihrem Amazon SNS-Thema veröffentlichte Nachrichten werden in dem S3-Sicherungs-Bucket gespeichert, der von der Pipeline für die Speicherung und Sicherung von Ereignissen automatisch bereitgestellt wird.

So stellen Sie die Ereignissuche- und Analyse-Pipeline bereit und abonnieren sie

> Für die Ereignisarchivierung und -analyse empfiehlt Amazon SNS jetzt, seine native Integration mit Amazon Data Firehose zu verwenden. Sie können Firehose-Bereitstellungsdatenströme für SNS-Themen abonnieren, mit denen Sie Benachrichtigungen an Archivierungs- und Analyseen dpunkte wie Amazon Simple Storage Service (Amazon S3)-Buckets, Amazon-Redshift-Tabellen, Amazon OpenSearch Service (OpenSearch Service) und mehr senden können. Die Verwendun g von Amazon SNS mit Firehose-Bereitstellungsdatenströmen ist eine vollständig verwaltete und codelose Lösung, für die Sie keine -AWS LambdaFunktionen verwenden müssen. Weitere Informationen finden Sie unter Fanout zu Firehose-Bereitstellungsdatenströmen.

In diesem Tutorial wird gezeigt, wie Sie die Ereignissuche- und Analytik-Pipeline bereitstellen und von diesen ein Amazon SNS-Thema abonnieren lassen. Dieser Prozess macht die der Pipeline zugewiesenen AWS SAM-Vorlage automatisch zu einen AWS CloudFormation Stack. Anschließend wird der Stack in Ihrem AWS-Konto-Konto bereitgestellt. Dieser Prozess erstellt und konfiguriert außerdem eine Reihe von Ressourcen, die die Ereignissuche und die Analysepipeline umfassen, darunter:

- Amazon-SQS-Warteschlange
- Lambda-Funktion
- Firehose-Bereitstellungsdat
- Amazon OpenSearch -Service-Domain
- Amazon S3 Bucket für unzustellbare Nachrichten

Weitere Informationen zum Konfigurieren eines Streams mit einem Index als Ziel finden Sie unter `ElasticsearchDestinationConfiguration` in der API-Referenz zu Amazon Data Firehose.

Weitere Informationen zum Transformieren von Ereignissen und zum Konfigurieren von Ereignispufferung, Ereigniskomprimierung und Ereignisverschlüsselung finden Sie unter

Erstellen eines Amazon-Data-Firehose-Bereitstellungsdatenstroms im Amazon-Data-Firehose-Entwicklerhandbuch.

Weitere Informationen zur Filterung von Ereignissen finden Sie unter Filterrichtlinien für Amazon-SNS-Abonnements in diesem Leitfaden.

1. Melden Sie sich an der AWS Lambda-Konsole an.
2. Wählen Sie im Navigationsbereich Functions (Funktionen) und dann Create function (Funktion erstellen) aus.
3. Gehen Sie auf der Seite Create function (Funktion erstellen) wie folgt vor:

   a. Wählen Sie Browse serverless app repository (Repository mit Serverless-Apps durchsuchen), Public applications (Öffentliche Anwendungen), Show apps that create custom roles or resource policies (Apps anzeigen, die benutzerdefinierte IAM-Rollen oder Ressourcenrichtlinien erstellen).

   b. Suchen Sie nach `fork-event-search-analytics-pipeline` und wählen Sie dann die Anwendung aus.
4. Gehen Sie auf der Seite fork-event-search-analytics-pipeline wie folgt vor:

   a. Geben Sie im Abschnitt Application settings (Anwendungseinstellungen) einen Application name (Anwendungsname) ein (zum Beispiel `my-app-search`).

   > **Note**
   >
   > Für jede Bereitstellung muss der Anwendungsname eindeutig sein. Wenn Sie einen Anwendungsnamen wiederverwenden, aktualisiert die Bereitstellung nur den zuvor bereitgestellten AWS CloudFormation-Stack (anstatt einen neuen zu erstellen).

   b. (Optional) Geben Sie für den ARN der Lambda-Funktion einDataTransformationFunctionArn, die für die Transformation eingehender Ereignisse verwendet wird. Wenn Sie keinen Wert eingeben, wird die Datentransformation deaktiviert.

   c. (Optional) Geben Sie eine der folgenden LogLevel Einstellungen für die Ausführung der Lambda-Funktion Ihrer Anwendung ein:

      - `DEBUG`
      - `ERROR`
      - `INFO` (Standard)

- `WARNING`

d. (Optional) SearchDomainArnGeben Sie für den ARN der OpenSearch Service-Domain ein, einen Cluster, der die erforderliche Rechen- und Speicherfunktionalität konfiguriert. Wenn Sie keinen Wert eingeben, wird eine neue Domäne mit der Standard-Konfiguration erstellt.

e. TopicArnGeben Sie für den ARN des Amazon SNS-Themas ein, das diese Instance der Fork-Pipeline abonnieren soll.

f. Geben SearchIndexNameSie für den Namen des OpenSearch Service-Index für die Ereignissuche und -analyse ein.

> **Note**
>
> Die folgenden Kontingente gelten für Indexnamen:
>
> - Sie dürfen keine Großbuchstaben enthalten.
> - Es dürfen keine der folgenden Zeichen enthalten sein: `\ / * ? " < > | ` , #`
> - Sie dürfen nicht mit den folgenden Zeichen beginnen: `- + _`
> - Sie dürfen nicht folgende Zeichen sein: `. ..`
> - Sie dürfen nicht mehr als 80 Zeichen enthalten.
> - Sie dürfen nicht mehr als 255 Bytes enthalten.
> - Darf keinen Doppelpunkt enthalten (aus OpenSearch Service 7.0)

g. (Optional) Geben Sie eine der folgenden SearchIndexRotationPeriod Einstellungen für den Drehungszeitraum des OpenSearch Service-Index ein:

- `NoRotation` (Standard)
- `OneDay`
- `OneHour`
- `OneMonth`
- `OneWeek`

Die Indexrotation hängt dem Indexnamen einen Zeitstempel an, was den Ablauf veralteter Daten ermöglicht.

h. Geben Sie für den Namen des OpenSearch Servicetyps einSearchTypeName, um die Ereignisse in einem Index zu organisieren.

> **Note**
>
> - OpenSearch Servicetypnamen können jedes Zeichen (außer Nullbyte) enthalten, dürfen aber nicht mit beginnen_.
> - Für OpenSearch Service 6.x kann es nur einen Typ pro Index geben. Wenn Sie einen neuen Typ für einen vorhandenen Index angeben, der bereits einen anderen Typ hat, gibt Firehose einen Laufzeitfehler zurück.

i. (Optional) Geben Sie für StreamBufferingIntervalInSeconds und StreamBufferingSizeInMBs die Werte für die Konfiguration der Pufferung eingehender Ereignisse ein. Wenn Sie keine Werte eingeben, 300 Sekunden und 5 MB verwendet.

j. (Optional) Geben Sie eine der folgenden StreamCompressionFormat Einstellungen für die Komprimierung eingehender Ereignisse ein:

- `GZIP`
- `SNAPPY`
- `UNCOMPRESSED` (Standard)
- `ZIP`

k. (Optional) Geben Sie für das Zeichenfolgenpräfix einStreamPrefix, um Dateien zu benennen, die im S3-Bucket für unzustellbare Nachrichten gespeichert sind. Wenn Sie keinen Wert eingeben, wird kein Präfix verwendet.

l. (Optional) Geben Sie für die Wiederholungsdauer für Fälle einStreamRetryDurationInSecons, in denen Firehose keine Ereignisse im OpenSearch Service-Index indizieren kann. Wenn Sie keinen Wert eingeben, wird 300 Sekunden verwendet.

m. (Optional) SubscriptionFilterPolicyGeben Sie für die Filterrichtlinie für Amazon SNS-Abonnements im JSON-Format ein, die zum Filtern eingehender Ereignisse verwendet werden soll. Die Filterrichtlinie bestimmt, welche Ereignisse im OpenSearch Service-Index indiziert werden. Wenn Sie keinen Wert eingeben, wird keine Filterung verwendet wird (alle Ereignisse werden indiziert).

n. Wählen Sie I acknowledge that this app creates custom IAM roles, resource policies and deploys nested applications (Ich bestätige, dass diese App benutzerdefinierte IAM-Rollen

und Ressourcenrichtlinien erstellt und eingebettete Anwendungen bereitstellt) und dann Deploy (Bereitstellen).

Auf der Seite Bereitstellungsstatus für ***my-app-search*** zeigt Lambda den Status Ihre Anwendung wird bereitgestellt an.

Im Abschnitt Resources (Ressourcen) beginnt AWS CloudFormation mit der Erstellung des Stacks und zeigt den Status CREATE_IN_PROGRESS für jede Ressource an. Wenn der Prozess abgeschlossen ist, zeigt AWS CloudFormation den Status CREATE_COMPLETE an.

Wenn die Bereitstellung abgeschlossen ist, zeigt Lambda den Status Your application has been deployed (Ihre Anwendung wurde bereitgestellt) an.

Nachrichten, die zu Ihrem Amazon SNS-Thema veröffentlicht werden, werden im OpenSearch Service-Index indiziert, der automatisch von der Pipeline für Ereignissuche und Analyse bereitgestellt wird. Wenn die Pipeline ein Ereignis nicht indizieren kann, speichert sie es in einem S3-Bucket für unzustellbare Nachrichten.

So stellen Sie die Ereignis-Wiederholungspipeline bereit und abonnieren sie

In diesem Tutorial wird gezeigt, wie Sie die Ereignis-Wiederholungspipeline bereitstellen und von dieser ein Amazon-SNS-Thema abonnieren lassen. Dieser Prozess macht die der Pipeline zugewiesenen AWS SAM-Vorlage automatisch zu einen AWS CloudFormation Stack. Anschließend wird der Stack in Ihrem AWS-Konto-Konto bereitgestellt. Dieser Prozess erstellt und konfiguriert außerdem eine Reihe von Ressourcen, die die Ereignis-Wiederholungs-Pipeline umfassen, darunter eine Amazon-SQS-Warteschlange und eine Lambda-Funktion.

Weitere Informationen zur Filterung von Ereignissen finden Sie unter Filterrichtlinien für Amazon-SNS-Abonnements in diesem Leitfaden.

1. Melden Sie sich an der AWS Lambda-Konsole an.
2. Wählen Sie im Navigationsbereich Functions (Funktionen) und dann Create function (Funktion erstellen) aus.
3. Gehen Sie auf der Seite Create function (Funktion erstellen) wie folgt vor:

    a. Wählen Sie Browse serverless app repository (Repository mit Serverless-Apps durchsuchen), Public applications (Öffentliche Anwendungen), Show apps that create custom roles or resource policies (Apps anzeigen, die benutzerdefinierte IAM-Rollen oder Ressourcenrichtlinien erstellen).

b. Suchen Sie nach `fork-event-replay-pipeline` und wählen Sie dann die Anwendung aus.

4. Auf derfork-event-replay-pipeline-Seite, machen Sie das Folgende:

   a. Geben Sie im Abschnitt Application settings (Anwendungseinstellungen) einen Application name (Anwendungsname) ein (zum Beispiel `my-app-replay`).

   > **Note**
   >
   > Für jede Bereitstellung muss der Anwendungsname eindeutig sein. Wenn Sie einen Anwendungsnamen wiederverwenden, aktualisiert die Bereitstellung nur den zuvor bereitgestellten AWS CloudFormation-Stack (anstatt einen neuen zu erstellen).

   b. (Optional) Geben Sie eine der folgenden LogLevel Einstellungen für die Ausführung der Lambda-Funktion Ihrer Anwendung ein:

      - `DEBUG`
      - `ERROR`
      - `INFO` (Standard)
      - `WARNING`

   c. (Optional) ReplayQueueRetentionPeriodInSecondsGeben Sie für die Zeit in Sekunden ein, für die die Amazon SQS-Wiedergabewarteschlange die Nachricht speichert. Wenn Sie keinen Wert eingeben, wird 1.209.600 Sekunden (14 Tage) verwendet.

   d. TopicArnGeben Sie für den ARN des Amazon SNS-Themas ein, das diese Instance der Fork-Pipeline abonnieren soll.

   e. DestinationQueueNameGeben Sie für den Namen der Amazon SQS-Warteschlange ein, an die die Lambda-Wiedergabefunktion Nachrichten weiterleitet.

   f. (Optional) SubscriptionFilterPolicyGeben Sie für die Filterrichtlinie für Amazon SNS-Abonnements im JSON-Format ein, die zum Filtern eingehender Ereignisse verwendet werden soll. Die Filterrichtlinie legt fest, welche Ereignisse für die Wiederholung gepuffert werden. Wenn Sie keinen Wert eingeben, wird keine Filterung verwendet wird (alle Ereignisse werden für die Wiederholung gepuffert).

   g. Wählen Sie I acknowledge that this app creates custom IAM roles, resource policies and deploys nested applications (Ich bestätige, dass diese App benutzerdefinierte IAM-Rollen und Ressourcenrichtlinien erstellt und eingebettete Anwendungen bereitstellt) und dann Deploy (Bereitstellen).

Auf der Seite Bereitstellungsstatus für ***my-app-replay*** zeigt Lambda den Status Ihre Anwendung wird bereitgestellt an.

Im Abschnitt Resources (Ressourcen) beginnt AWS CloudFormation mit der Erstellung des Stacks und zeigt den Status CREATE_IN_PROGRESS für jede Ressource an. Wenn der Prozess abgeschlossen ist, zeigt AWS CloudFormation den Status CREATE_COMPLETE an.

Wenn die Bereitstellung abgeschlossen ist, zeigt Lambda den Status Your application has been deployed (Ihre Anwendung wurde bereitgestellt) an.

Für Ihr Amazon SNS-Thema veröffentlichte Nachrichten werden in der von der Ereignis-Wiederholungspipeline bereitgestellten Amazon SQS-Warteschlange automatisch gepuffert.

> **Note**
>
> Standardmäßig ist die Wiederholung deaktiviert. Um die Wiederholung zu aktivieren, navigieren Sie zur Seite der Funktion in der Lambda-Konsole, erweitern Sie den Abschnitt Designer, wählen Sie das Feld SQS und dann im Abschnitt SQS die Option Enabled (Aktiviert).

Verwenden von Amazon EventBridge Scheduler mit Amazon SNS

Amazon EventBridge Scheduler ist ein Serverless-Scheduler, mit dem Sie Aufgaben von einem zentralen, verwalteten Service aus erstellen, ausführen und verwalten können. Mit EventBridge Scheduler können Sie mithilfe von Cron- und Rate-Ausdrücken Zeitpläne für wiederkehrende Muster erstellen oder einmalige Aufrufe konfigurieren. Sie können flexible Zeitfenster für die Zustellung einrichten, Wiederholungslimits definieren und die maximale Aufbewahrungszeit für fehlgeschlagene API-Aufrufe festlegen.

Auf dieser Seite wird erläutert, wie Sie mit EventBridge Scheduler verwenden, um ein Amazon SNS-Thema nach einem Zeitplan zu veröffentlichen.

Themen

Einrichten der Ausführungsrolle

Wenn Sie einen neuen Zeitplan erstellen, muss EventBridge Scheduler über die Berechtigung verfügen, seinen Ziel-API-Vorgang in Ihrem Namen aufzurufen. Sie gewähren EventBridge Scheduler diese Berechtigungen mithilfe einer Ausführungsrolle. Die Berechtigungsrichtlinie, die Sie der Ausführungsrolle Ihres Zeitplans hinzufügen, definiert die erforderlichen Berechtigungen. Diese Berechtigungen hängen von der Ziel-API ab, die EventBridge Scheduler aufrufen soll.

Wenn Sie die EventBridge-Scheduler-Konsole zum Erstellen eines Zeitplans verwenden, wie im folgenden Verfahren, richtet EventBridge Scheduler automatisch eine Ausführungsrolle basierend auf Ihrem ausgewählten Ziel ein. Wenn Sie einen Zeitplan mit einem der EventBridge-Scheduler-SDKs (AWS CLI oder AWS CloudFormation) erstellen möchten, müssen Sie über eine vorhandene Ausführungsrolle verfügen, die die Berechtigungen gewährt, die EventBridge Scheduler zum Aufrufen eines Ziels benötigt. Weitere Informationen zum manuellen Einrichten einer Ausführungsrolle für Ihren Zeitplan finden Sie unter Einrichten einer Ausführungsrolle im Benutzerhandbuch für EventBridge Scheduler.

Erstellen eines Zeitplans

So erstellen Sie einen Zeitplan mithilfe der Konsole

1. Öffnen Sie die Amazon-EventBridge-Scheduler-Konsole unter https://console.aws.amazon.com/scheduler/home.
2. Wählen Sie auf der Seite Zeitpläne die Option Zeitplan erstellen aus.
3. Gehen Sie auf der Seite Zeitplandetails angeben im Abschnitt Zeitplanname und -beschreibung wie folgt vor:

   a. Geben Sie unter Zeitplanname einen Namen für Ihren Zeitplan ein. Zum Beispiel **`MyTestSchedule`**.

   b. (Optional) Geben Sie unter Beschreibung eine Beschreibung für Ihren Zeitplan ein. Zum Beispiel **`My first schedule`**.

   c. Wählen Sie für Zeitplangruppe eine Zeitplangruppe aus der Dropdown-Liste aus. Wenn Sie noch keine Gruppe haben, wählen Sie Standard. Um eine Zeitplangruppe zu erstellen, wählen Sie Eigenen Zeitplan erstellen.

   Sie verwenden Zeitplangruppen, um Tags zu Zeitplangruppen hinzuzufügen.

4. • Wählen Sie Ihre Zeitplanoptionen.

Vorkommen	Vorgehensweise	
Einmaliger Zeitplan Ein einmaliger Zeitplan ruft ein Ziel nur einmal zu dem von Ihnen angegebenen Datum und der angegeben en Uhrzeit auf.	Gehen Sie für Datum und Uhrzeit wie folgt vor: • Geben Sie ein gültiges Datum im `YYYY/MM/DD` -Format ein. • Geben Sie einen Zeitstempel im 24-Stunden-Format (`hh:mm`) ein. • Wählen Sie unter Zeitzone die Zeitzone aus.	

Vorkommen	Vorgehensweise	
Wiederkehrender Zeitplan Ein wiederkehrender Zeitplan ruft ein Ziel mit einer Rate auf, die Sie mit einem cron-Ausdruck oder einem Rate-Ausdruck angeben.	a. Gehen Sie bei Zeitplantyp wie folgt vor: • Um den Zeitplan mithilfe eines Cron-Ausdrucks zu definieren, wählen Sie Cron-basierter Zeitplan und geben Sie den Cron-Ausdruck ein. • Um den Zeitplan mithilfe eines Rate-Ausdrucks zu definieren, wählen Sie Rate-Ausdruck ein. Weitere Informationen zu Cron- und Rate-Ausdrücken finden Sie unter Zeitplantypen im EventBridge Scheduler im Benutzerhandbuch zu Amazon EventBridge Scheduler. b. Wählen Sie für Flexibles Zeitfenster die Option Aus, um die Option zu deaktivieren, oder wählen Sie eines der vordefinierten Zeitfenster aus. Wenn Sie beispielsweise 15 Minuten auswählen und einen wiederkehrenden	

Vorkommen	Vorgehensweise	
	Zeitplan festlegen, der sein Ziel einmal pro Stunde aufruft, wird der Zeitplan innerhalb von 15 Minuten nach Beginn jeder Stunde ausgeführt.	

5. (Optional) Wenn Sie im vorherigen Schritt Wiederkehrender Zeitplan ausgewählt haben, gehen Sie im Abschnitt Zeitrahmen wie folgt vor:

   a. Wählen Sie unter Zeitzone eine Zeitzone aus.

   b. Geben Sie für Startdatum und -uhrzeit ein gültiges Datum im `YYYY/MM/DD`-Format ein und geben Sie dann einen Zeitstempel im 24-Stunden-Format (`hh:mm`) an.

   c. Geben Sie für Enddatum und -uhrzeit ein gültiges Datum im `YYYY/MM/DD`-Format ein und geben Sie dann einen Zeitstempel im 24-Stunden-Format (`hh:mm`) an.

6. Wählen Sie Next (Weiter).

7. Wählen Sie auf der Seite Ziel auswählen den AWS-API-Vorgang aus, den EventBridge Scheduler aufruft:

   a. Wählen Sie Amazon SNS Publish (Amazon SNS Veröffentlichung).

   b. Wählen Sie im Bereich Publish (Veröffentlichen) ein SNS-Thema aus oder klicken Sie auf Create new SNS topic (Neues Thema erstellen).

   c. (Optional) Geben Sie eine JSON-Nutzlast ein. Wenn Sie keine Nutzlast eingeben, verwendet EventBridge Scheduler ein leeres Ereignis, um die Funktion aufzurufen.

8. Wählen Sie Next (Weiter).

9. Führen Sie auf der Seite Settings (Einstellungen) die folgenden Schritte aus:

   a. Um den Zeitplan zu aktivieren, schalten Sie unter Zeitplanstatus die Option Zeitplan aktivieren ein.

   b. Um eine Wiederholungsrichtlinie für Ihren Zeitplan zu konfigurieren, gehen Sie unter Wiederholungsrichtlinie und Warteschlange für unzustellbare Nachrichten (DLQ) wie folgt vor:

      - Aktivieren Sie die Option Wiederholen.

- Geben Sie unter Maximales Alter des Ereignisses die maximale(n) Stunde(n) und Minute(n) ein, die EventBridge Scheduler ein unverarbeitetes Ereignis aufbewahren muss.
- Die Höchstdauer beträgt 24 Stunden.
- Geben Sie unter Maximale Wiederholungsversuche an, wie oft EventBridge Scheduler den Zeitplan maximal wiederholen soll, wenn das Ziel einen Fehler zurückgibt.

  Der Maximalwert beträgt 185 Wiederholungen.

Bei Wiederholungsrichtlinien führt EventBridge Scheduler den Zeitplan erneut aus, wenn ein Zeitplan sein Ziel nicht aufrufen kann. Falls konfiguriert, müssen Sie die maximale Aufbewahrungszeit und Wiederholungsversuche für den Zeitplan festlegen.

c. Wählen Sie aus, wo EventBridge Scheduler nicht zugestellte Ereignisse speichert.

Option für Warteschlange für unzustellbare Nachrichten (DLQ)	Vorgehensweise	
Nicht speichern	Wählen Sie None.	
Speichert das Ereignis im selben AWS-Konto, in dem Sie den Zeitplan erstellen	a. Wählen Sie Eine Amazon-SQS-Warteschlange in meinem AWS-Konto als DLQ auswählen. b. Wählen Sie den Amazon-Ressourcennamen (ARN) der Amazon SQS-Warteschlange.	

Option für Warteschlange für unzustellbare Nachrichten (DLQ)	Vorgehensweise	
Speichert das Ereignis in einem anderen AWS-Konto als dem, in dem Sie den Zeitplan erstellen	a. Wählen Sie Angeben einer Amazon-SQS-Warteschlange in anderen AWS-Konten als DLQ. b. Geben Sie den Amazon-Ressourcennamen (ARN) der Amazon-SQS-Warteschlange ein.	

d. Um einen kundenverwalteten Schlüssel zur Verschlüsselung Ihrer Zieleingabe zu verwenden, wählen Sie unter Verschlüsselung die Option Verschlüsselungseinstellungen anpassen (erweitert).

   Wenn Sie diese Option wählen, geben Sie einen vorhandenen CMK-ARN ein oder wählen Sie Erstellen eines AWS KMS key, um zur AWS KMS-Konsole zu navigieren. Weitere Informationen darüber, wie EventBridge Scheduler Ihre Daten im Ruhezustand verschlüsselt, finden Sie unter Verschlüsselung im Ruhezustand im Benutzerhandbuch zu Amazon EventBridge Scheduler.

e. Damit EventBridge Scheduler eine neue Ausführungsrolle für Sie erstellt, wählen Sie Neue Rolle für diesen Zeitplan erstellen. Geben Sie dann einen Namen für Rollenname ein. Wenn Sie diese Option wählen, fügt EventBridge Scheduler der Rolle die erforderlichen Berechtigungen für Ihr Vorlagenziel hinzu.

10. Wählen Sie Next (Weiter).
11. Überprüfen Sie auf der Seite Zeitplan überprüfen und erstellen die Details Ihres Zeitplans. Wählen Sie in jedem Abschnitt Bearbeiten aus, um zu diesem Schritt zurückzukehren und seine Details zu bearbeiten.
12. Wählen Sie Zeitplan erstellen.

    Auf der Seite Zeitpläne können Sie eine Liste Ihrer neuen und vorhandenen Zeitpläne anzeigen. Überprüfen Sie in der Spalte Status, ob Ihr neuer Zeitplan aktiviert ist.

Zugehörige Ressourcen

Weitere Informationen zum EventBridge Scheduler finden Sie hier:

- EventBridge-Scheduler-Benutzerhandbuch
- EventBridge-Scheduler-API-Referenz
- EventBridge Scheduler – Preisgestaltung

Verwenden von Amazon SNS für Application-to-Person-Messaging (A2P)

Dieser Abschnitt enthält Informationen zur Verwendung von Amazon SNS für Benutzerbenachrichtigungen mit Abonnenten wie z. B. mobilen Apps, mobilen Telefonnummern und E-Mail-Adressen.

Themen

Text-Messaging (SMS)

Verwenden Sie Amazon SNS, um Textnachrichten oder SMS-Nachrichten an SMS-fähige Geräte zu senden. Sie können eine Nachricht direkt an eine Telefonnummer senden oder Sie können eine Nachricht an mehrere Telefonnummern gleichzeitig senden, indem Sie das Thema für diese Telefonnummern abonnieren und die Nachricht an das Thema senden.

Sie können SMS-Einstellungen für Ihr AWS-Konto einrichten, um Ihre SMS-Zustellungen Ihren Anwendungsfällen und Budgets anzupassen. Beispielsweise können Sie auswählen, ob die Nachrichten im Hinblick auf Kosten oder eine zuverlässige Bereitstellung optimiert werden. Sie können auch Ausgabenkontingenten für einzelne Nachrichtenzustellungen und monatliche Ausgabenkontingenten für Ihr AWS-Konto festlegen.

Sofern dies aufgrund von Gesetzen und Vorschriften erforderlich ist (z. B. in den USA und Kanada), haben die SMS-Empfänger die Möglichkeit, den Empfang zu deaktivieren. Das bedeutet, dass sie keine weiteren SMS-Nachrichten von Ihrem AWS-Konto erhalten werden. Nachdem die Benachrichtigungen durch den Empfänger deaktiviert wurden, können Sie, mit Einschränkungen, die Telefonnummer erneut aktivieren, sodass Sie mit dem Senden von Nachrichten fortfahren können.

Amazon SNS unterstützt SMS-Messaging in mehreren Regionen und Sie können Nachrichten in mehr als 200 Länder und Regionen senden. Weitere Informationen finden Sie unter Unterstützte Länder und Regionen.

Themen

SMS-Sandbox

Wenn Sie Amazon SNS zum Senden von SMS-Nachrichten verwenden, befindet sich Ihr AWS-Konto in der SMS-Sandbox. Die SMS-Sandbox bietet Ihnen eine sichere Umgebung, in der Sie Amazon SNS Funktionen ausprobieren können, ohne Ihren Ruf als SMS-Sender zu gefährden. Während sich Ihr Konto in der SMS-Sandbox befindet, können Sie alle Funktionen von Amazon SNS mit den folgenden Einschränkungen verwenden:

- Sie können SMS-Nachrichten nur an verifizierte Zieltelefonnummern senden.
- Sie können über bis zu 10 verifizierte Zielrufnummern verfügen.
- Sie können Zieltelefonnummern erst löschen, nachdem 24 oder mehr Stunden seit der Überprüfung oder dem letzten Verifizierungsversuch verstrichen sind.

Wenn Ihr Konto aus der Sandbox verschoben wird, werden diese Einschränkungen entfernt, und Sie können SMS-Nachrichten an jeden Empfänger senden.

Themen

Hinzufügen und Überprüfen von Telefonnummern in der SMS-Sandbox

Um mit dem Senden von SMS-Nachrichten zu beginnen, während sich Ihr AWS Konto in der SMS-Sandbox befindet, erstellen Sie eine Absenderidentität, fügen Sie die Zieltelefonnummern hinzu und überprüfen Sie sie dann.

> **Note**
>
> Wie bei Konten, die sich nicht in der SMS-Sandbox befinden, wird eine Ursprungsidentität erfordert, um SMS-Nachrichten in einigen Ländern oder Regionen an Empfänger in einigen Ländern oder Regionen senden zu können. Weitere Informationen finden Sie unter Unterstützte Länder und Regionen.
> Ursprungs-IDs enthalten Sender-IDs und verschiedene Arten von Ursprungsnummern. Um die vorhandenen Ursprungsnummern anzuzeigen, wählen Sie im Navigationsbereich der Amazon-SNS-Konsole Ursprungsnummern aus. Derzeit werden Sender-IDs in dieser Liste nicht angezeigt.

So fügen Sie Ziel-Telefonnummern hinzu und überprüfen diese

1. Melden Sie sich bei der Amazon SNS-Konsole an.
2. Erstellen Sie eine Ursprungsidentität für die Telefonnummer.
3. Wählen Sie im Konsolenmenü eine AWS Region aus, die SMS-Messaging unterstützt.
4. Wählen Sie im Navigationsbereich Text-Messaging (SMS) aus.
5. Wählen Sie auf der Seite Text-Messaging (SMS) unter Sandbox Zieltelefonnummern Telefonnummer hinzufügen.
6. Geben Sie unter Zieldetails den Ländercode und die Telefonnummer ein, geben Sie an, welche Sprache für die Verifizierungsnachricht verwendet werden soll und wählen Sie dann Telefonnummer hinzufügen.

   Amazon SNS sendet ein einmaliges Passwort (OTP) an die Zielrufnummer. Wenn die Zieltelefonnummer das OTP nicht innerhalb von 15 Minuten erhält, wählen Sie Bestätigungscode erneut senden aus. Sie können das OTP bis zu fünf Mal alle 24 Stunden an dieselbe Zieltelefonnummer senden.
7. Geben Sie in das Feld Verifizierungscode das OTP ein, das an die Zieltelefonnummer geschickt wurde und wählen Sie dann Telefonnummer verifizieren.

Die Zieltelefonnummer und ihr Überprüfungsstatus werden im Abschnitt Sandbox Zieltelefonnummern angezeigt. Lautet der Verifizierungsstatus ausstehend, war die Verifizierung nicht erfolgreich. Dies kann beispielsweise passieren, wenn Sie die Landesvorwahl nicht mit der Telefonnummer eingegeben haben. Sie können ausstehende oder verifizierte Zieltelefonnummern erst nach Ablauf von 24 Stunden oder mehr seit der Überprüfung oder dem letzten Verifizierungsversuch löschen.

8. Wiederholen Sie diese Schritte in jeder Region, in der Sie diese Zielrufnummer verwenden möchten.

Behebung von Problemen beim Nichterhalt eines OTP-Textes

Beheben Sie häufig auftretende Probleme, die verhindern können, dass eine Telefonnummer OTP-Texte empfängt.

- Amazon SNS SMS-Ausgabenlimit: Wenn Sie das Ausgabenlimit für den Versand von SMS-Nachrichten AWS-Konto überschritten haben, werden weitere Nachrichten, einschließlich OTP-Texte, möglicherweise erst zugestellt, wenn das Limit erhöht oder das Abrechnungsproblem gelöst ist.
- Telefonnummer, die nicht für SMS-Benachrichtigungen aktiviert wurde: In einigen Ländern oder Regionen müssen sich Empfänger dafür entscheiden, SMS-Nachrichten über Kurzcodes zu erhalten, die üblicherweise für OTP-Texte verwendet werden. Wenn die Telefonnummer des Empfängers nicht aktiviert ist, erhält er den OTP-Text nicht.
- Einschränkungen oder Filterung durch Mobilfunkanbieter: Einige Mobilfunkanbieter verfügen möglicherweise über Einschränkungen oder Filtermechanismen, die die Zustellung bestimmter Arten von SMS-Nachrichten, einschließlich OTP-Texten, verhindern. Dies könnte auf Sicherheitsrichtlinien oder Anti-Spam-Maßnahmen des Mobilfunkanbieters zurückzuführen sein.
- Ungültige oder falsche Telefonnummer: Wenn die vom Empfänger angegebene Telefonnummer falsch oder ungültig ist, wird der OTP-Text nicht zugestellt.
- Netzwerkprobleme: Vorübergehende Netzwerkprobleme oder Ausfälle können die Zustellung von SMS-Nachrichten, einschließlich OTP-Texten, an das Telefon des Empfängers verhindern.
- Verspätete Zustellung: In einigen Fällen kann es bei SMS-Nachrichten aufgrund von Netzwerküberlastungen oder anderen Faktoren zu Verzögerungen bei der Zustellung kommen. Der OTP-Text kann irgendwann zugestellt werden, er kann sich jedoch über den erwarteten Zeitrahmen hinaus verzögern.

- Sperrung oder Kündigung Ihres Kontos: Wenn es Probleme mit Ihrem Konto gibt AWS-Konto, wie z. B. Nichtzahlung oder Verstoß gegen die AWS Nutzungsbedingungen, können die Nachrichtenfunktionen von Amazon SNS, einschließlich OTP-Texten, gesperrt oder beendet werden.

Löschen von Telefonnummern aus der SMS-Sandbox

Sie können ausstehende oder verifizierte Zieltelefonnummern aus der SMS-Sandbox entfernen.

So löschen Sie Zieltelefonnummern aus der SMS-Sandbox

1. Warten Sie 24 Stunden nach der Verifizierung der Telefonnummer oder 24 Stunden nach dem letzten Verifizierungsversuch.
2. Melden Sie sich bei der Amazon SNS-Konsole an.
3. Wählen Sie im Konsolenmenü, wo Sie die Zieltelefonnummer eingegeben haben, eine AWSRegion, die SMS-Messaging unterstützt aus.
4. Wählen Sie im Navigationsbereich Text-Messaging (SMS) aus.
5. Wählen Sie auf der Seite Text-Messaging (SMS) unter Sandbox Zieltelefonnummern die Telefonnummer, die Sie löschen möchten und wählen Sie dann Telefonnummer löschen.
6. Geben Sie **`delete me`** ein, und wählen Sie dann Delete (Löschen), um zu bestätigen, dass Sie die Telefonnummer löschen möchten.

   Wenn 24 Stunden oder mehr vergangen sind, seit Sie die Zieltelefonnummer verifiziert haben oder versucht haben, eine zu verifizieren, wird diese gelöscht und Amazon SNS aktualisiert die Liste Ihrer Zieltelefonnummern.
7. Wiederholen Sie diese Schritte in jeder Region, in der Sie die Zieltelefonnummer hinzugefügt haben und nicht mehr verwenden möchten.

Verlassen der SMS-Sandbox

Um Ihr Konto AWS-Konto aus der SMS-Sandbox zu entfernen, müssen Sie zunächst Zielrufnummern hinzufügen, verifizieren und testen. Anschließend müssen Sie einen Fall mit AWS Support erstellen.

Um zu beantragen, dass Ihr AWS Konto aus der SMS-Sandbox entfernt wird

1. Überprüfen Sie die Telefonnummern

a. Öffnen Sie AWS-Konto die Amazon SNS-Konsole, während Sie sich in der SMS-Sandbox befinden.

b. Wählen Sie im Navigationsbereich unter Mobil die Option Textnachrichten (SMS) aus.

c. Fügen Sie im Abschnitt Sandbox-Zielrufnummern eine oder mehrere Zielrufnummern hinzu und überprüfen Sie sie. Diese Überprüfung stellt sicher, dass Sie erfolgreich Nachrichten senden und empfangen können.

2. Testen Sie die SMS-Veröffentlichung

   - Vergewissern Sie sich, dass Sie Nachrichten an mindestens eine verifizierte Telefonnummer senden und empfangen können. Ausführlichere Anweisungen zum Veröffentlichen von SMS-Nachrichten finden Sie unterVeröffentlichen auf einem Mobiltelefon.

3. Initiieren Sie die Sandbox-Bearbeitung

   - Wählen Sie auf der Seite Amazon SNS-Konsole Text-Messaging (SMS) unter Kontoinformationen SMS-Sandbox verlassen. Diese Aktion leitet Sie zum Amazon Support Center weiter und erstellt automatisch einen Support-Fall, bei dem die Option zur Erhöhung des Servicekontingents ausgewählt ist.

4. Füllen Sie das Formular aus

   - Gehen Sie im Support-Formular unter Erhöhung der Servicequote wie folgt vor:

     i. Wählen Sie SNS-Textnachrichten als Dienst aus.

     ii. Geben Sie die URL der Website oder den Namen der App an, von der aus Sie SMS-Nachrichten senden möchten.

     iii. Geben Sie die Art der Nachrichten an, die Sie senden möchten: Einmalpasswort, Werbe - oder Transaktionsnachricht.

     iv. Wählen Sie aus, AWS-Regionvon wo aus Sie SMS-Nachrichten senden möchten.

     v. Listen Sie die Länder oder Regionen auf, in denen Sie SMS-Nachrichten senden möchten.

     vi. Beschreiben Sie, wie sich Ihre Kunden für den Empfang von Nachrichten anmelden.

     vii. Fügen Sie alle Nachrichtenvorlagen hinzu, die Sie verwenden möchten.

5. Geben Sie Kontingent und Region an

   - Gehen Sie unter Requests (Anfragen) wie folgt vor:

i. Wählen Sie AWS-Region, wohin Sie Ihre verschieben möchten AWS-Konto.

ii. Wählen Sie Allgemeine Grenzwerte für den Ressourcentyp.

iii. Wählen Sie Exit SMS Sandbox für Quota.

iv. (Optional) Um zusätzliche Erhöhungen oder andere Anpassungen anzufordern, wählen Sie „Weitere Anfrage hinzufügen“ und geben Sie die erforderlichen Details an.

v. Geben Sie unter Neuer Kontingentwert das Limit in USD ein, das Sie beantragen.

6. Zusätzliche Details

   a. Geben Sie in der Fallbeschreibung alle zusätzlichen Details an, die für Ihre Anfrage relevant sind.

   b. Wählen Sie unter Kontaktoptionen Ihre bevorzugte Kontaktsprache aus.

7. Reichen Sie die Anfrage ein

   - Wählen Sie Senden, um Ihre Anfrage an zu senden AWS Support.

Das AWS Support Team gibt innerhalb von 24 Stunden eine erste Antwort auf Ihre Anfrage.

Da wir verhindern möchten, dass unerwünschte oder schädliche Inhalte in unseren Systemen eingehen, müssen wir jede Anfrage sorgfältig prüfen. Nach einer Prüfung kommen wir Ihrer Anfrage innerhalb dieses 24-Stunden-Zeitraums nach. Für den Fall, dass wir weitere Informationen von Ihnen benötigen, kann die Bearbeitung Ihrer Anfrage länger dauern.

Wenn Ihr Anwendungsfall gegen unsere Richtlinien verstößt, können wir Ihrer Anfrage möglicherweise nicht nachkommen.

Ursprungsidentitäten für SMS-Nachrichten

Wenn Sie SMS-Nachrichten über Amazon SNS senden, können Sie sich selbst gegenüber Ihren Empfängern mit den folgenden Arten an Ursprungsidentitäten identifizieren:

- Sender-IDs
- Ursprungsnummern

> **Note**
>
> Amazon SNS SMS-Messaging ist in Regionen verfügbar, in denen Amazon Pinpoint derzeit nicht unterstützt wird. Wenn Sie in Europa (Stockholm), Naher Osten (Bahrain), Europa (Paris), Südamerika (São Paulo) oder USA West (Nordkalifornien) arbeiten, öffnen Sie die Amazon Pinpoint Konsole in der Region USA Ost (Nord-Virginia), um Ihre 10DLC-Unternehmen und -Kampagne zu registrieren. Fordern Sie keine 10DLC-Nummer an. Verwenden Sie stattdessen die AWS-Service-Quotas-Konsole, um einen Fall für die Erhöhung des Service-Limits zu erstellen, während Sie die 10DLC-Nummer für diese Region anfordern. Weitere Informationen zum Anfordern von Ursprungsidentitäten finden Sie unter Anfordern von Support für SMS-Messaging mit Amazon SNS.

Sender-IDs

Eine Sender-ID ist ein alphabetischer Name zur Identifizierung des Senders einer SMS-Nachricht. Wenn Sie eine SMS-Nachricht mit einer Sender-ID senden und der Empfänger sich in einem Bereich befindet, in dem Sender-ID-Authentifizierung unterstützt wird, erscheint Ihre Sender-ID auf dem Gerät des Empfängers anstelle einer Telefonnummer. Eine Sender-ID stellt SMS-Empfängern mehr Informationen über den Sender zur Verfügung als eine Telefonnummer, Langwahl- oder Kurzwahlnummer.

Sender-IDs werden in verschiedenen Ländern und Regionen auf der ganzen Welt unterstützt. In einigen Regionen müssen Sie als Unternehmen, das SMS-Nachrichten an einzelne Kunden sendet, eine Sender-ID verwenden, die vorab bei einer Regulierungsbehörde oder einem Branchengremium registriert wurde. Eine vollständige Liste der Länder und Regionen, die Sender-IDs unterstützen oder erfordern, finden Sie unter Unterstützte Länder und Regionen.

Für die Nutzung einer Sender-ID fallen keine zusätzlichen Gebühren an. Die Unterstützung und Anforderungen für die Sender-ID-Authentifizierung sind jedoch unterschiedlich. Mehrere wichtige Märkten (einschließlich Kanada, China und USA) unterstützen keine Sender-ID. In einigen Regionen müssen Sie als Unternehmen, das SMS-Nachrichten an einzelne Kunden sendet, eine Sender-ID verwenden, die vorab bei einer Regulierungsbehörde oder einem Branchengremium registriert wurde.

> **Important**
>
> AWS verbietet SMS-Spoofing, bei dem die Absender-ID verwendet wird, um sich für eine andere Person, ein Unternehmen oder Produkt auszugeben. Verwenden Sie nur eine Sender-ID, die eine Marke oder ein Handelszeichen darstellt, das Sie besitzen.

Vorteile

Sender-IDs stellen dem Empfänger mehr Informationen über den Sender der Nachricht zur Verfügung. Es ist einfacher, Ihre Marke mit einer Sender-ID als mit einer Kurz- oder Langwahlnummer zu etablieren. Für die Nutzung einer Sender-ID fallen keine zusätzlichen Gebühren an.

Nachteile

Support und Anforderungen für die Sender-ID-Authentifizierung sind nicht in allen Ländern oder Regionen konsistent. Mehrere wichtige Märkten (einschließlich Kanada, China und USA) unterstützen keine Sender-ID. In einigen Regionen müssen Sie Ihre Sender-IDs durch eine Regulierungsbehörde vorab genehmigen lassen, damit Sie sie verwenden können.

Registrierung der Absender-ID nach Ländern

Sie müssen einen Fall beim AWS-Support eröffnen, um Absender-IDs für SMS-Nachrichten zu registrieren. Nachdem Sie die Supportanfrage gestellt haben, teilt AWS Ihnen mit, welche weiteren Dokumente erforderlich sind. Außerdem müssen Sie die folgenden Informationen für das entsprechende Land angeben, in dem Sie die Absender-ID registrieren.

Ländername	Nachrichtentyp	Formatbeschränkungen und Anforderungen	Voraussetzungen für die Registrierung
Australien (AU)	Transaktions- und werbebezogen	• Alphanumerisch • Maximal 11 Zeichen • Keine Leerzeichen • Keine Sonderzeichen • Die Absender-ID muss der	• Die Absender-ID, die Sie registrieren möchten • Aus welcherAWS-Region der Benutzer die API/den Service aufruft

Ländername	Nachrichtentyp	Formatbeschränkungen und Anforderungen	Voraussetzungen für die Registrierung
		Markenname des Unternehmens sein, das die SMS sendet	• Unternehmensname • Adresse des Unternehmens (einschließlich Stadt, Bundesstaat/Provinz, Postleitzahl) • Land des Unternehmens • Unternehmens-URL (Link zu Ihrer App oder Unternehmenswebsite) • Geschätztes monatliches Volumen • Erläuterung des Anwendungsfalls, Zweck der Nachrichten • Falls nicht offensichtlich, erläutern Sie den Zusammenhang zwischen dem Firmennamen und der Absender-ID • Offizielle Geschäfts-/Gewerbescheinnummer oder Umsatzsteuer-Ident

Ländername	Nachrichtentyp	Formatbeschränkungen und Anforderungen	Voraussetzungen für die Registrierung

			ifikationsnummer des Unternehmens
			• Nachricht envorlagen, die Sie versenden möchten
			• Eine Gewerbean meldelizenz. Zu den Beispielen gehören u. a.:
			  • Australian Business Number (ABN)
			  • Australian Company Number (ACN)
			  • Australian Registered Body Number (ARBN)
			  • Indigenous Corporation Number (ICN)
			• Ein Letter of Authorization (LOA)

Ländername	Nachrichtentyp	Formatbeschränkungen und Anforderungen	Voraussetzungen für die Registrierung
Weißrussland (BY)	Nur Senden von Transaktionsnachrichten	• Alphanumerisch • Maximal 11 Zeichen • Keine Leerzeichen • Keine Sonderzeichen • Die Absender-ID muss der Markenname des Unternehmens sein, das die SMS sendet	• Die Absender-ID, die Sie registrieren möchten • Aus welcherAWS-Region der Benutzer die API/den Service aufruft • Unternehmensname • Adresse des Unternehmens (einschließlich Stadt, Bundesstaat/Provinz, Postleitzahl) • Land des Unternehmens • Kontakt-Telefonnummer des Unternehmens • Unternehmens-URL (Link zu Ihrer App oder Unternehmenswebsite) • Geschätztes monatliches Volumen • Erläuterung des Anwendungsfalls, Zweck der Nachrichten

Ländername	Nachrichtentyp	Formatbeschränkungen und Anforderungen	Voraussetzungen für die Registrierung
			• Falls nicht offensichtlich, erläutern Sie den Zusammenhang zwischen dem Firmennamen und der Absender-ID • Offizielle Geschäfts-/Gewerbescheinnummer oder Umsatzsteuer-Identifikationsnummer des Unternehmens • Nachrichtenvorlagen, die Sie versenden möchten

Ländername	Nachrichtentyp	Formatbeschränkungen und Anforderungen	Voraussetzungen für die Registrierung
China (CN) In China ist keine Registrierung der Absender-ID erforderlich, jedoch eine Registrierung von Inhalten/Vorlagen mit vorangestellter Textsignatur (z. B. [Amazon]).	Die folgenden Schlüsselwörter sind nicht zulässig: • Falung Gong • SB • Tiananmen-Platz Nachrichten aus den folgenden Kategorien: • Kreditkarten • Digitale Zahlungen (einschließlich Kryptowährung) • Betrügerische Traffic-URLs (Phishing oder Spam) • Glücksspiel • Unangemessene Inhalte (Sexualität, Gewalt, Drogen, Alkohol) • Kredite • Plastische Chirurgie • Politik • Religion • Aktienhandel • Virtuelle Währungen	• Maximal 11 Zeichen • Keine Leerzeichen • Keine Sonderzeichen	• Unternehmensname • Adresse des Unternehmens • Bundesstaat oder Provinz • Land • Telefonnummer des Unternehmens • Unternehmenswebseite • Geschätztes monatliches Volumen • Nachrichtentyp: Werbung/transaktional • Erläuterung des Anwendungsfalls • Vorlagen, die Sie registrieren möchten (die gesendete SMS darf nicht von den bereitgestellten Vorlagen abweichen, um die Einhaltung der Vorschriften und den erfolgreichen Versand zu gewährleisten).

Ländername	Nachrichtentyp	Formatbeschränkungen und Anforderungen	Voraussetzungen für die Registrierung
	Signaturen in eckigen Klammern müssen dem SMS-Nachrichtentext vorangestellt werden. Für eine erfolgreiche Zustellung nach China müssen Sie eine Nachrichtensignatur in Ihrer Nachrichteninhaltsvorlage registrieren. Diese Nachrichtensignatur muss dem Nachrichteninhalt jeder gesendeten SMS vorangestellt werden. Wenn dem SMS-Nachrichtentext keine registrierte Signatur vorangestellt wird, kann dies zu blockierten oder gefilterten SMS führen. Signaturen müssen dem SMS-Text in eckigen Klammern vorangestellt werden.		Zum Beispiel [CompanyName] Ihr Einmalpasswort lautet {OTP}. Dieser Code läuft in 10 Minuten ab. • Bestätigung, dass Sie keine Werbeinhalte als Transaktionsnachrichtentypen versenden • Nachrichten-Signatur. Siehe Einschränkungen und Anforderungen im Signaturformat.

Ländername	Nachrichtentyp	Formatbeschränkungen und Anforderungen	Voraussetzungen für die Registrierung
Ägypten (EG)	Nur Senden von Transaktionsnachrichten	• Alphanumerisch • Maximal 11 Zeichen • Keine Leerzeichen • Keine Sonderzeichen	• Die Absender-ID, die Sie registrieren möchten • Aus welcherAWS-Region der Benutzer die API/den Service aufruft • Unternehmensname • Adresse des Unternehmens (einschließlich Stadt, Bundesstaat/Provinz, Postleitzahl) • Land des Unternehmens • Kontakt-Telefonnummer des Unternehmens • Unternehmens-URL (Link zu Ihrer App oder Unternehmenswebsite) • Geschätztes monatliches Volumen • Erläuterung des Anwendungsfalls, Zweck der Nachrichten

Ländername	Nachrichtentyp	Formatbeschränkungen und Anforderungen	Voraussetzungen für die Registrierung
			• Falls nicht offensichtlich, erläutern Sie den Zusammenhang zwischen dem Firmennamen und der Absender-ID • Hat Ihr Unternehmen eine Geschäftseinrichtung/ein Büro in Ägypten? Oder ist es ein nicht in Ägypten ansässiges Unternehmen, das nach Ägypten liefert? • Schriftliche Bestätigung, dass der Anwendungsfall alle Nachrichten umfasst, die mit dieser Absender-ID gesendet werden sollen • Falls nicht offensichtlich, geben Sie die Verbindung zwischen dem Firmennamen und der Absender-ID an • Nachrichtenvorlagen, die Sie versenden möchten

Ländername	Nachrichtentyp	Formatbeschränkungen und Anforderungen	Voraussetzungen für die Registrierung
Indien (IN)	Nur Senden von Transaktionsnachrichten	• Alphanumerisch • Maximal 6 Zeichen • Keine Leerzeichen • Keine Sonderzeichen	Siehe Anforderungen für die Registrierung der Absender-ID für Indien.

Ländername	Nachrichtentyp	Formatbeschränkungen und Anforderungen	Voraussetzungen für die Registrierung
Jordanien (JO)	Nur Senden von Transaktionsnachrichten	• Alphanumerisch • Maximal 11 Zeichen • Keine Leerzeichen • Keine Sonderzeichen	• Die Absender-ID, die Sie registrieren möchten • Aus welcherAWS-Region der Benutzer die API/den Service aufruft • Unternehmensname • Adresse des Unternehmens (einschließlich Stadt, Bundesstaat/Provinz, Postleitzahl) • Land des Unternehmens • Kontakt-Telefonnummer des Unternehmens • Unternehmens-URL (Link zu Ihrer App oder Unternehmenswebsite) • Geschätztes monatliches Volumen • Erläuterung des Anwendungsfalls, Zweck der Nachrichten

Ländername	Nachrichtentyp	Formatbeschränkungen und Anforderungen	Voraussetzungen für die Registrierung
			• Falls nicht offensichtlich, erläutern Sie den Zusammenhang zwischen dem Firmennamen und der Absender-ID • Unternehmensregistrierungsbescheinigung • Die Art der Nachricht, die Sie senden möchten (z. B. OTP, Alerts) • Schriftliche Bestätigung, dass der Anwendungsfall alle Nachrichten beschreibt, die mit dieser Absender-ID gesendet werden sollen • Nachrichtenvorlagen, die Sie versenden möchten

Ländername	Nachrichtentyp	Formatbeschränkungen und Anforderungen	Voraussetzungen für die Registrierung
Kuwait (KW)	Nur Senden von Transaktionsnachrichten	• Alphanumerisch • Maximal 11 Zeichen • Keine Leerzeichen • Keine Sonderzeichen	• Die Absender-ID, die Sie registrieren möchten • Aus welcherAWS-Region der Benutzer die API/den Service aufruft • Unternehmensname • Adresse des Unternehmens (einschließlich Stadt, Bundesstaat/Provinz, Postleitzahl) • Land des Unternehmens • Kontakt-Telefonnummer des Unternehmens • Unternehmens-URL (Link zu Ihrer App oder Unternehmenswebsite) • Geschätztes monatliches Volumen • Erläuterung des Anwendungsfalls, Zweck der Nachrichten

Ländername	Nachrichtentyp	Formatbeschränkungen und Anforderungen	Voraussetzungen für die Registrierung
			• Falls nicht offensichtlich, erläutern Sie den Zusammenhang zwischen dem Firmennamen und der Absender-ID • Die Art der Nachricht, die Sie senden möchten (z. B. OTP, Alerts) • Schriftliche Bestätigung, dass der Anwendungsfall alle Nachrichten beschreibt, die mit dieser Absender-ID gesendet werden sollen • Nachricht envorlagen, die Sie versenden möchten

Ländername	Nachrichtentyp	Formatbeschränkungen und Anforderungen	Voraussetzungen für die Registrierung
Philippinen (PH)	Nur Senden von Transaktionsnachrichten	• Alphanumerisch • Maximal 11 Zeichen • Keine Leerzeichen • Keine Sonderzeichen	• Die Absender-ID, die Sie registrieren möchten • Aus welcherAWS-Region der Benutzer die API/den Service aufruft • Unternehmensname • Adresse des Unternehmens (einschließlich Stadt, Bundesstaat/Provinz, Postleitzahl) • Land des Unternehmens • Kontakt-Telefonnummer des Unternehmens • Unternehmens-URL (Link zu Ihrer App oder Unternehmenswebsite) • Geschätztes monatliches Volumen • Erläuterung des Anwendungsfalls, Zweck der Nachrichten

Ländername	Nachrichtentyp	Formatbeschränkungen und Anforderungen	Voraussetzungen für die Registrierung
			• Falls nicht offensichtlich, erläutern Sie den Zusammenhang zwischen dem Firmennamen und der Absender-ID • Die Art der Nachricht, die Sie senden möchten (z. B. OTP, Alerts) • Schriftliche Bestätigung, dass der Anwendungsfall alle Nachrichten beschreibt, die mit dieser Absender-ID gesendet werden sollen • Nachricht envorlagen, die Sie versenden möchten

Ländername	Nachrichtentyp	Formatbeschränkungen und Anforderungen	Voraussetzungen für die Registrierung
Katar (QA)	Nur Senden von Transaktionsnachrichten	• Alphanumerisch • Maximal 11 Zeichen • Keine Leerzeichen • Keine Sonderzeichen	• Die Absender-ID, die Sie registrieren möchten • Aus welcherAWS-Region der Benutzer die API/den Service aufruft • Unternehmensname • Adresse des Unternehmens (einschließlich Stadt, Bundesstaat/Provinz, Postleitzahl) • Land des Unternehmens • Kontakt-Telefonnummer des Unternehmens • Unternehmens-URL (Link zu Ihrer App oder Unternehmenswebsite) • Geschätztes monatliches Volumen • Erläuterung des Anwendungsfalls, Zweck der Nachrichten

Ländername	Nachrichtentyp	Formatbeschränkungen und Anforderungen	Voraussetzungen für die Registrierung
			• Falls nicht offensichtlich, erläutern Sie den Zusammenhang zwischen dem Firmennamen und der Absender-ID • Art der gesendeten SMS • (Transaktional/ Promotional/OTP) Beachten Sie, dass aus Gründen der Konformität nur Transaktions-/ OTP-Nachrichten mit Absender-IDs verwendet werden können, die für Katar bestimmt sind. • Schriftliche Bestätigung, dass der Anwendungsfall alle Nachrichten beschreibt, die mit dieser Absender-ID gesendet werden sollen • Nachricht envorlagen, die Sie versenden möchten

Ländername	Nachrichtentyp	Formatbeschränkungen und Anforderungen	Voraussetzungen für die Registrierung
Russland (RU)	Nur Senden von Transaktionsnachrichten	• Alphanumerisch • Maximal 11 Zeichen • Keine Leerzeichen • Keine Sonderzeichen	• Die Absender-ID, die Sie registrieren möchten • Aus welcherAWS-Region der Benutzer die API/den Service aufruft • Unternehmensname • Adresse des Unternehmens (einschließlich Stadt, Bundesstaat/Provinz, Postleitzahl) • Land des Unternehmens • Kontakt-Telefonnummer des Unternehmens • Unternehmens-URL (Link zu Ihrer App oder Unternehmenswebsite) • Steuer-ID oder Lizenznummer • Unternehmensregistrierungsbescheinigung • E-Mail-Adresse des Ansprechpartners

Ländername	Nachrichtentyp	Formatbeschränkungen und Anforderungen	Voraussetzungen für die Registrierung
			• Geschätztes monatliches Volumen • Erläuterung des Anwendungsfalls, Zweck der Nachrichten • Falls nicht offensichtlich, erläutern Sie den Zusammenhang zwischen dem Firmennamen und der Absender-ID • Bestätigung, dass dieser Anwendungsfall für alle Nachrichten gilt, die von diesem Konto nach Russland gesendet werden • Bestätigung, dass Nachrichten, die keine Transaktionen betreffen, mit einer separaten Absender-ID gesendet werden müssen • 272 USD/monatliche Gebühr Bitte bestätigen Sie, dass Sie dieser wiederkeh

Ländername	Nachrichtentyp	Formatbeschränkungen und Anforderungen	Voraussetzungen für die Registrierung
			renden monatlichen Gebühr zustimmen: Ja/Nein • Nachricht envorlagen, die Sie versenden möchten

Ländername	Nachrichtentyp	Formatbeschränkungen und Anforderungen	Voraussetzungen für die Registrierung
Saudi-Arabien (SA)	Jede Absender-ID muss entweder transaktional oder werblich sein. Eine Absender-ID kann nicht für beide Arten von Datenverkehr verwendet werden. Beim Senden von OTP- oder 2FA-Verkehr darf die Absender-ID nur für diesen Zweck verwendet werden. Die Absenderkennungen von Werbeabsendern unterliegen der Liste „Bitte nicht stören“ (DND) von Saudi-Arabien.	• Länge der Promo-Absender-ID: 2-8 Zeichen, Absender-ID mit vorangestelltem „-AD“ • Länge der Transaktions-Absender-ID: 2-11 Zeichen • Die Absender-ID muss die Markenidentität des Absenders repräsentieren • Mindestens ein Buchstabe • Keine ASCII-Sonderzeichen (z. B. #, @) • Sie können Groß- und Kleinbuchstaben sowie die Zahlen 0 bis 9 angeben	Die Unterstützung für die Registrierung von Absender-IDs in Saudi-Arabien steht nur internationalen Unternehmen zur Verfügung. Wir unterstützen derzeit keine lokalen Unternehmen in Saudi-Arabien bei der Registrierung von Absender-IDs. • Die Absender-ID, die Sie registrieren möchten • Aus welcherAWS-Region der Benutzer die API/den Service aufruft • Unternehmensname • Adresse des Unternehmens (einschließlich Stadt, Bundesstaat/Provinz, Postleitzahl) • Land des Unternehmens

Ländername	Nachrichtentyp	Formatbeschränkungen und Anforderungen	Voraussetzungen für die Registrierung
			• Kontakt-Telefonnummer des Unternehmens • Unternehmens-URL (Link zu Ihrer App oder Unternehmenswebsite) • Geschätztes monatliches Volumen • Erläuterung des Anwendungsfalls, Zweck der Nachrichten • Falls nicht offensichtlich, erläutern Sie den Zusammenhang zwischen dem Firmennamen und der Absender-ID • Nachricht envorlagen, die Sie versenden möchten. • Wird diese Absender-ID für Transaktions- oder Werbeinhalte verwendet? • Bestätigung, dass Nachrichten, die keine Transakti

Ländername	Nachrichtentyp	Formatbeschränkungen und Anforderungen	Voraussetzungen für die Registrierung
			onen betreffen, mit einer separaten Absender-ID gesendet werden müssen • Bestätigen Sie beim Senden von 2FA- oder OTP-Verkehr, dass diese Absender-ID NUR für diesen Zweck verwendet wird
Singapur (SG)	Nur Senden von Transaktionsnachrichten	• Maximal 11 Zeichen • Keine Leerzeichen • Keine Sonderzeichen	Siehe Anforderungen für die Registrierung der Absender-ID für Singapur.

Ländername	Nachrichtentyp	Formatbeschränkungen und Anforderungen	Voraussetzungen für die Registrierung
Sri Lanka (LK)	Keine Einschränkungen oder besonderen Anforderungen	• Alphanumerisch • Maximal 11 Zeichen • Keine Leerzeichen • Keine Sonderzeichen	• Die Absender-ID, die Sie registrieren möchten • Aus welcherAWS-Region der Benutzer die API/den Service aufruft • Unternehmensname • Unternehmenstyp (lokal/international) • Unternehmens-URL (Link zu Ihrer App oder Unternehmenswebsite) • Erläuterung des Anwendungsfalls, Zweck der Nachrichten • Nachrichtenvorlagen, die Sie versenden möchten • Wird diese Absender-ID für Transaktions- oder Werbeinhalte verwendet? • Bestätigung, dass Nachrichten, die keine Transaktionen betreffen, mit einer separaten

Ländername	Nachrichtentyp	Formatbeschränkungen und Anforderungen	Voraussetzungen für die Registrierung
			Absender-ID gesendet werden müssen • Bestätigen Sie beim Senden von 2FA- oder OTP-Verkehr, dass diese Absender-ID nur für diesen Zweck verwendet wird

Ländername	Nachrichtentyp	Formatbeschränkungen und Anforderungen	Voraussetzungen für die Registrierung
Thailand (TH)	Keine Einschränkungen oder besonderen Anforderungen	• Alphanumerisch • Maximal 11 Zeichen • Keine Leerzeichen • Keine Sonderzeichen	• Die Absender-ID, die Sie registrieren möchten • Aus welcherAWS-Region der Benutzer die API/den Service aufruft • Unternehmensname • Adresse des Unternehmens (einschließlich Stadt, Bundesstaat/Provinz, Postleitzahl) • Land des Unternehmens • Kontakt-Telefonnummer des Unternehmens • Unternehmens-URL (Link zu Ihrer App oder Unternehmenswebsite) • Geschätztes monatliches Volumen • Erläuterung des Anwendungsfalls, Zweck der Nachrichten

Ländername	Nachrichtentyp	Formatbeschränkungen und Anforderungen	Voraussetzungen für die Registrierung
			• Falls nicht offensichtlich, erläutern Sie den Zusammenhang zwischen dem Firmennamen und der Absender-ID • Art der gesendeten SMS (Transaktional/Werbe-/OTP) • Nachrichtenvorlagen, die Sie versenden möchten

Ländername	Nachrichtentyp	Formatbeschränkungen und Anforderungen	Voraussetzungen für die Registrierung
Türkei (TR)	Keine Einschränkungen oder besonderen Anforderungen	• Alphanumerisch • Maximal 11 Zeichen • Keine Leerzeichen • Keine Sonderzeichen	• Die Absender-ID, die Sie registrieren möchten • Aus welcherAWS-Region der Benutzer die API/den Service aufruft • Unternehmensname • Adresse des Unternehmens (einschließlich Stadt, Bundesstaat/Provinz, Postleitzahl) • Unternehmens-URL (Link zu Ihrer App oder Unternehmenswebsite) • Wenn Ihr Unternehmen lokal oder international in der Türkei ist • Geschätztes monatliches Volumen • Erläuterung des Anwendungsfalls, Zweck der Nachrichten • Falls nicht offensichtlich, erläutern Sie

Ländername	Nachrichtentyp	Formatbeschränkungen und Anforderungen	Voraussetzungen für die Registrierung
			den Zusammenhang zwischen dem Firmennamen und der Absender-ID • Art der gesendeten SMS (Transaktional/Werbe-/OTP) • Nachrichtenvorlagen, die Sie versenden möchten

Ländername	Nachrichtentyp	Formatbeschränkungen und Anforderungen	Voraussetzungen für die Registrierung
Ukraine (UA)	Keine Einschränkungen oder besonderen Anforderungen	• Alphanumerisch • Maximal 11 Zeichen • Keine Leerzeichen • Keine Sonderzeichen	• Die Absender-ID, die Sie registrieren möchten • Aus welcherAWS-Region der Benutzer die API/den Service aufruft • Unternehmensname • Adresse des Unternehmens (einschließlich Stadt, Bundesstaat/Provinz, Postleitzahl) • Unternehmens-URL (Link zu Ihrer App oder Unternehmenswebsite) • USt-IdNr • Lokales oder internationales Unternehmen • Geschätztes monatliches Volumen • Erläuterung des Anwendungsfalls, Zweck der Nachrichten • Falls nicht offensichtlich, erläutern Sie

Ländername	Nachrichtentyp	Formatbeschränkungen und Anforderungen	Voraussetzungen für die Registrierung
			den Zusammenhang zwischen dem Firmennamen und der Absender-ID • Art der gesendeten SMS (Transaktional/Werbe-/OTP) • Nachrichtenvorlagen, die Sie versenden möchten

Ländername	Nachrichtentyp	Formatbeschränkungen und Anforderungen	Voraussetzungen für die Registrierung
Vereinigte Arabische Emirate (VAE)	Nur Senden von Transaktionsnachrichten	• Alphanumerisch • Generische Absender-IDs sind nicht zulässig und müssen die Marke identifizieren • Maximal 11 Zeichen • Keine Leerzeichen • Keine Sonderzeichen	• Die Absender-ID, die Sie registrieren möchten • Aus welcherAWS-Region der Benutzer die API/den Service aufruft • Unternehmensname • Adresse des Unternehmens (einschließlich Stadt, Bundesstaat/Provinz, Postleitzahl) • Land des Unternehmens • Kontakt-Telefonnummer des Unternehmens • Unternehmens-URL (Link zu Ihrer App oder Unternehmenswebsite) • Geschätztes monatliches Volumen • Falls nicht offensichtlich, erläutern Sie den Zusammenhang zwischen dem

Ländername	Nachrichtentyp	Formatbeschränkungen und Anforderungen	Voraussetzungen für die Registrierung
			Firmennamen und der Absender-ID • Erläuterung des Anwendungsfalls, Zweck der Nachrichten • Offizielle Geschäfts-/Gewerbeschein nummer oder Umsatzsteuer-Identifikationsnummer des Unternehmens • Schriftliche Bestätigung, dass der gesamte von dieser Absender-ID gesendete Datenverkehr anhand des angegebenen Anwendungsfalls beschrieben wird • Nachricht envorlagen, die Sie versenden möchten

Ländername	Nachrichtentyp	Formatbeschränkungen und Anforderungen	Voraussetzungen für die Registrierung
Vietnam (VN)	Nur Senden von Transaktionsnachrichten Marketing- und Werbebotschaften sind nicht erlaubt. Zu den verbotenen Inhalten gehören: • Inhalte für Erwachsene • Gemeinnützige Dienste • Kryptowährung • Lotterie • Mobiles Glücksspiel/Casinos • Sportwetten • Voting	• Maximal 11 Zeichen • Keine Leerzeichen • Keine Sonderzeichen	• Die Absender-ID, die Sie registrieren möchten • Aus welcher AWS-Region der Benutzer die API/den Service aufruft • Geschätztes monatliches Volumen • Falls nicht offensichtlich, erläutern Sie den Zusammenhang zwischen dem Firmennamen und der Absender-ID • Erläuterung des Anwendungsfalls, Zweck der Nachrichten • Schriftliche Bestätigung, dass der gesamte von dieser Absender-ID gesendete Datenverkehr anhand des angegebenen Anwendungsfalls beschrieben wird

Einschränkungen und Anforderungen im Signaturformat

Für eine erfolgreiche Zustellung nach China müssen Sie eine Nachrichtensignatur in Ihrer Nachrichteninhaltsvorlage registrieren. Diese Nachrichtensignatur muss dem Nachrichteninhalt jeder gesendeten SMS vorangestellt werden. Wenn dem SMS-Nachrichtentext keine registrierte Signatur vorangestellt wird, kann dies zu blockierten oder gefilterten SMS führen.

- Die Signatur muss dem SMS-Text in eckigen Klammern vorangestellt werden
- Standardtext muss in eckigen Klammern enthalten sein
- Unicode-Text muss eckige Klammern verwenden, um die Signatur zu enthalten — U+3010 LINKE ECKIGE KLAMMER und U+3011 RECHTE ECKIGE KLAMMER — Beispiel: [Hinweis]
- Muss zwischen 3 und 11 Zeichen lang sein.
- Chinesische/Lateinische Zeichen werden unterstützt

Anforderungen an die Absender-ID für Frankreich

Dieses Handbuch enthält die notwendigen Schritte und Richtlinien zur Erstellung einer speziellen Absender-ID, die französische Mobilfunkanbieter für den Versand von SMS-Textnachrichten nach Frankreich benötigen.

Themen

- Einrichtung einer dedizierten Absender-ID für Frankreich
- Richtlinien für Absender-ID-Namen

Einrichtung einer dedizierten Absender-ID für Frankreich

Sie können eine der folgenden Methoden verwenden, um eine dedizierte Sender-ID einzurichten. Amazon SNS verwendet die Absender-ID in Ihrem Namen für SMS-Nachrichten, die mithilfe der `Publish`-API veröffentlicht werden.

- Sie können über die Amazon-SNS-Konsole konfigurieren, welche Standard-SMS-ID für alle aktuellen SMS-Nachrichten verwendet wird. Weitere Informationen finden Sie unter Einstellung der SMS-Nachrichteneinstellungen mit dem AWS Management Console.
- Sie können die `Publish`-API verwenden, um die Absender-ID mithilfe des `AWS.SNS.SMS.SenderID`-Nachrichtenattributs festzulegen, wenn Sie Amazon SNS um die Veröffentlichung einer SMS-Nachricht bitten. Weitere Informationen finden Sie unter Senden einer Nachricht (Konsole).

Richtlinien für Absender-ID-Namen

- Der Name der Absender-ID muss alphanumerisch sein und darf maximal 11 Zeichen enthalten.
- Der Name einer Absender-ID darf keine Sonderzeichen oder Leerzeichen enthalten.
- Wir empfehlen, dass Sie für die Absender-ID und den Markennamen des Unternehmens, das die SMS-Textnachricht sendet, denselben Namen verwenden.

Anforderungen für die Registrierung der Absender-ID für Indien

Standardmäßig verwendet Amazon SNS beim Senden von Nachrichten an Empfänger in Indien Verbindungen mit einem Betreiber für internationale Fernverbindungen (International Long Distance Operator (ILDO)), um diese Nachrichten zu übertragen. Wenn Empfänger eine Nachricht sehen, die über eine ILDO-Verbindung gesendet wird, wird sie scheinbar von einer zufälligen numerischen ID gesendet (sofern Sie nicht einen dedizierten Kurzcode kaufen).

> **Note**
>
> Der Preis für das Senden von Nachrichten über lokale Routen wird auf der Seite Amazon SNS Weltweite SMS-Preise angezeigt. Der Preis für das Senden von Nachrichten über ILDO-Verbindungen ist höher als der Preis für das Senden von Nachrichten über lokale Routen.

Wenn Sie lieber eine alphabetische Sender-ID für Ihre SMS-Nachrichten verwenden möchten, müssen Sie diese Nachrichten über lokale Routen und nicht über ILDO-Routen senden. Um Nachrichten über lokale Routen zu senden, müssen Sie zunächst Ihre Anwendungsfall- und Nachrichtenvorlagen bei der Telecom Regulatory Authority of India (TRAI) durch die Distributed Ledger Technology (DLT)-Portale registrieren. Diese Registrierungspflichten wurden entwickelt, um die Anzahl der unerwünschten Nachrichten, die indische Verbraucher erhalten, zu reduzieren und Verbraucher vor potenziell schädlichen Nachrichten zu schützen. Dieser Registrierungsprozess wird von Vodafone India über den Service Vilpower verwaltet.

Themen

- Schritt 1: Registrierung bei TRAI
- Schritt 2: Sender-ID anfordern
- Schritt 3: Senden von SMS-Nachrichten
- Fehlerbehebung SMS-Nachricht an Empfänger in Indien

Schritt 1: Registrierung bei TRAI

Bevor Sie SMS-Nachrichten an Empfänger in Indien senden können, müssen Sie Ihre Organisation bei der Telecom Regulatory Authority of India (TRAI) registrieren. Während des Registrierungsvorgangs müssen Sie folgende Informationen angeben:

- Permanente Kontonummer (Permanent Account Number, PAN) Ihrer Organisation.
- Die Steuerabzugskontonummer (Tax Deduction Account Number, TAN) Ihrer Organisation.
- Die Steueridentifikationsnummer für Waren und Dienstleistungen (Goods and Services Tax Identification Number, GSTIN) Ihrer Organisation.
- Die Handelsregisternummer (Corporate Identity Number, CIN) Ihrer Organisation.
- Ein Genehmigungsschreiben für die Registrierung Ihrer Organisation.

Im Folgenden finden Sie eine Beispielliste einiger Registrierungswebsites für Distributed Ledger Technology (DLT), mit denen Sie Ihre Organisation bei der TRAI registrieren können (möglicherweise fallen Gebühren an). Der Registrierungsprozess variiert je nach Standort. Wenden Sie sich an die jeweiligen Support-Teams, um Unterstützung zu erhalten.

- BSNL DLT: Kostenlose Registrierung
- Jio Trueconnect: Erhebt eine Gebühr für den Abschluss des Registrierungsprozesses
- Smart Enterprise Solutions: Erhebt eine Gebühr für den Abschluss des Registrierungsprozesses
- Vilpower: Enthält eine Vorlage, die Sie herunterladen und nach Bedarf ändern können. Vilpower erhebt eine Gebühr für den Abschluss des Registrierungsprozesses.

So registrieren Sie Ihre Organisation bei der TRAI

Im Folgenden erfahren Sie, wie Sie Ihre Organisation mit Vilpower bei der TRAI registrieren.

1. Rufen Sie die Vilpower-Website unter https://www.vilpower.in in einem Webbrowser auf.
2. Wählen Sie Signup (Registrieren), um ein weiteres Konto zu erstellen. Führen Sie während des Registrierungsprozesses die folgenden Schritte aus:
   - Für den Typ der Entität, als die registriert werden soll, wählen Sie Als Unternehmen.
   - Verwenden Sie für den Telemarketer-Namen Infobip Private Limited - ALL. Wenn Sie dazu aufgefordert werden, tippen Sie **`Infobip`** und wählen Sie dann Infobip Private Limited – ALL aus der Dropdown-Liste.

- Geben Sie unter Enter Telemarketer ID (Telemarketer-ID eingeben) die Zeichenfolge **`110200001152`** ein.
- Wenn Sie aufgefordert werden, Ihre Header-IDs anzugeben, geben Sie die Sender-IDs ein, die Sie registrieren möchten.

> **Note**
>
> In Indien müssen Sender-IDs genau sechs Zeichen lang sein.

- Wenn Sie aufgefordert werden, Ihre Inhaltsvorlagen anzugeben, geben Sie den Nachrichteninhalt ein, den Sie an Ihre Empfänger senden möchten. Fügen Sie für jede Nachricht, die Sie senden möchten, eine Vorlage hinzu.

> **Note**
>
> Websites des DLT-Registrierungsanbieters werden nicht von Amazon Web Services verwaltet. Die Schritte auf den Websites können geändert werden.

Schritt 2: Sender-ID anfordern

Um eine Sender-ID in Indien anzufordern, müssen Sie eine AWS Support-Anfrage abschicken. Führen Sie die Schritte unter Anfordern von Sender-IDs auf. Machen Sie in Ihrer Anfrage die folgenden Angaben:

- Die AWS-Region, aus der der Sender plant, SMS-Nachrichten zu senden.
- Der Firmenname, der bei der DLT-Registrierung verwendet wird.
- Die Principal Entity ID (PEID), die Sie nach erfolgreicher Registrierung der DLT-Entität erhalten haben.
- Geschätzte monatliche Mengen.
- Eine Erklärung Ihres Anwendungsfalls.
- Eine Beschreibung des Opt-In-Flows des Endbenutzers.
- Bestätigung, dass Endbenutzer-Opt-Ins gesammelt und registriert werden.

Schritt 3: Senden von SMS-Nachrichten

Nach Registrierung Ihrer Organisation mit TRAI können Sie SMS-Nachrichten an Empfänger in Indien senden.

1. Melden Sie sich bei der Amazon SNS-Konsole an.
2. Legen Sie im Konsolenmenü unter der Regionsauswahl eine -Region fest, die SMS-Messaging unterstützt.
3. Wählen Sie im Navigationsbereich Text messaging (SMS) (Textnachrichten (SMS)) aus.
4. Wählen Sie auf der Seite Text-Messaging (SMS) Textnachricht veröffentlichen. Das Fenster SMS-Nachricht veröffentlichen öffnet sich.
5. Wählen Sie für Message type (Nachrichtentyp) eine der folgenden Optionen aus:

   - Werbenachrichten – Unkritische Nachrichten, wie beispielsweise Marketing-Nachrichten.

     Wählen Sie diese Option, wenn Sie numerische Sender-IDs verwenden.
   - Transaktionsnachrichten: Sensible Nachrichten, die Kundentransaktionen unterstützen, wie beispielsweise One-Time-Passcodes für eine Multi-Faktor-Authentifizierung (MFA).

     Wählen Sie diese Option, wenn Sie alphabetische oder alphanumerische Sender-IDs verwenden.

   Diese Einstellung auf Nachrichtenebene ersetzt den Standardnachrichtentyp, den Sie auf der Seite Text messaging preferences konfiguriert haben.

   Informationen zu den Gebühren für Aktions- und Transaktionsnachrichten finden Sie unter Global SMS Pricing.
6. Geben Sie im Feld Nummer die Telefonnummer ein, an die die Nachricht gesendet werden soll.
7. Geben Sie im Feld Message (Nachricht) die Art der zu versendenden Nachricht ein.

   Stellen Sie beim Hinzufügen von Inhalten zu SMS-Nachrichten sicher, dass sie genau mit dem Inhalt in der registrierten DLT-Vorlage übereinstimmen. Carrier blockieren SMS-Nachrichten, wenn ihr Nachrichteninhalt zusätzliche Zeichen, Leerzeichen, Satzzeichen oder nicht übereinstimmende Groß-/Kleinschreibung enthält. Variablen in einer Vorlage können 30 oder weniger Zeichen enthalten.
8. Geben Sie im Abschnitt Ursprungsidentitäten für die Sender-ID eine benutzerdefinierte ID ein, die 3 bis 11 Zeichen enthält.

Sender-IDs können für Werbenachrichten numerisch oder alphabetisch oder alphanumerisch für Transaktionsnachrichten sein. Die Sender-ID wird auf dem Empfänger-Gerät als Sender der Nachricht angezeigt.

Geben Sie für numerische Werbe-Absender-IDs, die für Indien registriert sind, die Absender-ID als Parameter für die Ursprungsnummer in der SMS-Sendeanfrage an.

9. Erweitern Sie den Abschnitt länderspezifischen Attribute und legen Sie die folgenden erforderlichen Attribute zum Senden von SMS-Nachrichten an Empfänger in Indien fest:

   - Entity-ID – Die Entity ID oder Principal Entity (PE) -ID, die Sie von der Aufsichtsbehörde für das Senden von SMS-Nachrichten an Empfänger in Indien erhalten haben.

     Dies ist eine benutzerdefinierte, von TRAI bereitgestellte Zeichenfolge mit 1 bis 50 Zeichen, die die Entität, die Sie bei TRAI registriert haben, eindeutig identifiziert.

   - ID der Vorlage – Die Vorlagen-ID, die Sie von der Aufsichtsbehörde für das Senden von SMS-Nachrichten an Empfänger in Indien erhalten haben.

     Dies ist eine benutzerdefinierte, von TRAI bereitgestellte Zeichenfolge mit 1 bis 50 Zeichen, die die Vorlage eindeutig identifiziert, die Sie bei TRAI registriert haben. Die Vorlagen-ID muss der Sender-ID, die Sie im vorherigen Schritt angegeben haben, und dem Nachrichteninhalt zugeordnet sein.

10. Wählen Sie Publish message (Nachricht veröffentlichen) aus.

Weitere Informationen über das Senden von SMS-Nachrichten an Empfänger in anderen Ländern siehe Veröffentlichen auf einem Mobiltelefon.

Fehlerbehebung SMS-Nachricht an Empfänger in Indien

Im Folgenden werden einige Gründe aufgeführt, warum Mobilfunkanbieter SMS-Nachrichten blockieren können:

- Es wurde keine Vorlage gefunden, die dem gesendeten Inhalt entspricht.

  Gesendete Inhalte: **`<#> 12345 is your OTP to verify mobile number. Your OTP is valid for 15 minutes -- ABC Pvt. Ltd.`**

  Zugeordnete Vorlage: Keine

Problem: Es gibt keine DLT-Vorlagen, die <#> oder {#var#} am Anfang der DLT-registrierten Vorlage haben.

- Der Wert einer Variablen überschreitet 30 Zeichen.

  Gesendete Inhalte: **12345 is your OTP code for ABC (ABC Company - India Private Limited) - (ABC 123456789). Share with your agent only. - ABC Pvt. Ltd.**

  Zugeordnete Vorlage: **{#var#} is your OTP code for {#var#} ({#var#}) - ({#var#} {#var#}). Share with your agent only. - ABC Pvt. Ltd.**

  Problem: Der Wert von „ABC Company - India Private Limited” in den gesendeten Inhalten übersteigt ein einzelnes {#var#}-Zeichenlimit von 30.
- Die Groß-/Kleinschreibung der Nachricht stimmt nicht mit der Groß-/Kleinschreibung in der Vorlage überein.

  Gesendete Inhalte: **12345 is your OTP code for ABC (ABC Company - India Private Limited) - (ABC 123456789). Share with your agent only. - ABC Pvt. Ltd.**

  Zugeordnete Vorlage: **{#var#} is your OTP code for {#var#} ({#var#}) - ({#var#} {#var#}). Share with your agent only. - ABC PVT. LTD.**

  Problem: Der Firmenname, der an die DLT-Match-Vorlage angehängt wird, wird großgeschrieben, während der gesendete Inhalt Teile des Namens in Kleinbuchstaben hat – „ABC Pvt. Ltd.”. gegenüber „ABC PVT. Ltd.”

Anforderungen für die Registrierung der Absender-ID für Singapur

Amazon-SNS-Kunden können SMS in Singapur mit einer Sender-ID senden, die über die Singapore SMS Sender ID Registry (SSIR) registriert wurde. Die SSIR wurde im März 2022 vom Singapore Network Information Centre (SGNIC) eingeführt, das der Info-communications Media Development Authority (IMDA) in Singapur unterstellt ist. Die SSIR ermöglicht es Organisationen, ihre Sender-ID zu registrieren, wenn sie SMS an Mobiltelefone in Singapur senden.

Damit Sie eine registrierte Sender-ID für Singapur verwenden können, müssen Sie eine Unique Entity Number (UEN) beantragen, damit Ihr Konto anschließend über SSIR für die Verwendung Ihrer Sender-ID zugelassen wird.

Wenn Sie Ihre ID nicht bis zum 30.01.2023 registrieren, wird die ID jeder Nachricht, die mit einer Sender-ID gesendet wird, gemäß den Regeln der Aufsichtsbehörde in LIKELY-SCAM geändert. Ab diesem Datum wird die Aufsichtsbehörde nicht registrierten Datenverkehr weiterhin nach eigenem Ermessen filtern oder blockieren.

Important

Verwenden Sie bei Anforderung der Absender-ID in den Amazon-Pinpoint-Regionen die Amazon-Pinpoint-Konsole, um die Sender-ID zu registrieren. Verwenden Sie die Registrierung einer Absender-ID für Singapur, um den Registrierungsprozess für andere Regionen als Amazon-Pinpoint-Regionen manuell abzuschließen.
Um sicherzustellen, dass Sie in Singapur weiterhin Nachrichten senden können, muss Ihre Registrierung bis zum 30.01.2023 abgeschlossen sein.
Es ist sehr wichtig, dass Sie die Registrierungsschritte in der folgenden Reihenfolge abschließen. Wenn Sie die Reihenfolge dieser Schritte nicht einhalten, wird möglicherweise Ihre Sender-ID vom Service blockiert oder die Sender-ID nicht auf dem Mobilgerät gespeichert.
Schritt 1. Registrieren für eine Unique Entity Number (UEN) in Singapur
Schritt 2. Verwenden Sie bei Anforderung der Absender-ID in den Amazon-Pinpoint-Regionen die Anweisung zur Registrierung der Amazon-Pinpoint-Absender-ID, um die Absender-ID zu registrieren.

- Um eine Absender-ID zu registrieren, wenn sich das Konto nicht in einer Amazon-Pinpoint-Region befindet, verwenden Sie die Anweisungen zur Registrierung der Absender-ID in Singapur, um die Absender-ID manuell zu registrieren.
- Wenn Sie SMS-Textnachrichten im Namen eines anderen Unternehmens senden, ist ein Autorisierungsschreiben (LOA) des Unternehmens erforderlich.
- Warten Sie nicht auf eine Genehmigung oder eine Statusänderung, nachdem Sie Ihre AWS-Absender-ID-Registrierung eingereicht haben. Fahren Sie sofort mit Schritt 3 fort.

Schritt 3. Registrierung einer Sender-ID beim Singapore Network Information Centre (SGNIC)

Themen

- Manueller Registrierungsvorgang zum Abschluss der Registrierung für die Sender-ID für Singapur
- Registrierung einer Sender-ID beim Singapore Network Information Centre (SGNIC)
- Status der Registrierung einer Sender-ID für Singapur
- Bearbeiten einer Sender-ID-Registrierung für Singapur
- Löschen einer Sender-ID-Registrierung für Singapur
- Probleme bei der Registrierung für Singapur
- Häufig gestellte Fragen zur Registrierung der Sender-ID für Singapur

Registrieren für eine Unique Entity Number (UEN) in Singapur

Um eine Registrierung bei der SSIR zu starten, müssen Sie zunächst eine Unique Entity Number (UEN) in Singapur beantragen. Eine UEN ist eine eindeutige Entitätsnummer, die Sie erhalten, wenn Sie Ihr Unternehmen bei der Account and Corporate Registry Authority (ACRA) registrieren. Weitere Informationen finden Sie unter Wer muss sich bei der ACRA registrieren?. Die Bearbeitungszeit kann variieren, je nachdem, wie einfach die ACRA Ihre Anfrage validieren kann.

Registrieren Ihrer Sender-ID für Singapur bei Amazon Pinpoint

Sobald Sie Ihre Unique Entity Number (UEN) in Singapur registriert haben, können Sie den Vorgang zur Registrierung der Sender-ID in der Amazon-Pinpoint-Konsole abschließen (nur für Amazon-Pinpoint-Regionen). Stellen Sie bei der Registrierung Ihrer Sender-ID sicher, dass die Informationen vollständig und korrekt sind, da Ihre Registrierung sonst ggf. abgelehnt wird.

> **Important**
>
> Die Informationen, die Sie über die Amazon-Pinpoint-Konsole einreichen, werden an unsere Carrier-Partner weitergegeben, um die Registrierung abzuschließen.

So registrieren Sie eine Absender-ID für Singapur:

Verwenden Sie diese Schritte zur Registrierung einer Absender-ID, wenn sich das Konto in einer Amazon-Pinpoint-Region befindet. Falls sich Ihr Konto nicht in einer Amazon-Pinpoint-Region befindet, finden Sie weitere Informationen unter Manueller Registrierungsvorgang zum Abschluss der Registrierung für die Sender-ID für Singapur.

1. Melden Sie sich bei der AWS-Managementkonsole an und öffnen Sie die Amazon-Pinpoint-Konsole unter https://console.aws.amazon.com/pinpoint/.

2. Klicken Sie im Navigationsbereich unter SMS and voice (SMS und Sprache) auf Phone numbers (Telefonnummern).
3. Wählen Sie auf der Registerkarte Sender ID registrations (Sender-ID-Registrierungen) die Option Create registration (Registrierung erstellen) aus.
4. Wählen Sie Singapur als Zielland aus.
5. Geben Sie im Abschnitt Company Information (Unternehmensinformationen) Folgendes ein:
   - Geben Sie unter Company Name (Unternehmensname) den Namen Ihres Unternehmens genau so ein, wie er in Ihrer UEN-Registrierung steht.
   - Geben Sie als Tax ID (Steuernummer) die UEN-Nummer ein, die Sie von der ACRA erhalten haben.
   - Geben Sie unter Unternehmenswebsite die vollständige URL der Website Ihres Unternehmens ein.
   - Geben Sie unter Address 1 (Adresse 1) die Adresse des Hauptsitzes Ihres Unternehmens ein.
   - Unter Address 2 (Adresse 2) – optional geben Sie bei Bedarf die Suite-Nummer Ihres Unternehmenshauptsitzes ein.
   - Geben Sie unter City (Ort) die Stadt Ihres Unternehmenshauptsitzes ein.
   - Geben Sie unter State (Bundesland) das Bundesland an, in dem sich Ihr Unternehmenshauptsitz befindet.
   - Geben Sie unter Zip Code (Postleitzahl) die Postleitzahl Ihres Unternehmenshauptsitzes ein.
   - Geben Sie unter Country (Land) den zweistelligen IDO-Ländercode ein.
6. Geben Sie im Abschnitt Contact Information (Kontaktinformationen) Folgende Informationen ein:
   - Geben Sie unter Vorname den Vornamen der Person ein, die Ansprechpartner für Ihr Unternehmens sein wird.
   - Geben Sie unter Nachname den Nachnamen der Person ein, die Ansprechpartner für Ihr Unternehmens sein wird.
   - Im Feld Support-E-Mail geben Sie die E-Mail-Adresse der Person ein, die Ansprechpartner für Ihr Unternehmen sein wird.
   - Geben Sie im Feld Support-Telefonnummer die Telefonnummer der Person ein, die Ansprechpartner für Ihr Unternehmen sein wird.
7. Geben Sie unter Sender ID Information (Sender-ID-Informationen) die Folgendes ein:
   - Geben Sie als Sender ID (Sender-ID) die Sender-ID ein, die Sie für Ihre Nachrichten anzeigen möchten.

- Wenn Sie bei Registrierung im Namen einer anderen Marke/Entität? mit ja antworten können, wählen Sie „Wahr“ aus. Wenn Sie nicht der Endbenutzer sind, der die Nachrichten sendet, gelten Sie als „Vertreter“ der anderen Marke/Entität.
- Wenn Sie bei Bild des Autorisierungsschreibens – optional, das Kästchen „Registrierung im Namen einer anderen Marke/Entität?“ aktiviert haben, laden Sie ein Bild des vollständigen Autorisierungsschreibens (LOA) hoch. Der unterstützte Dateityp ist PNG und die maximale Dateigröße beträgt 400 KB. Eine Vorlage für das LOA kann der Einfachheit halber heruntergeladen werden.
- Für die Sender ID connection – optional (Sender-ID-Verbindung – optional) können Sie weitere Details zur Verbindung zwischen der angeforderten SenderID und dem Unternehmensnamen hinzufügen.

8. Führen Sie unter Messaging Use Case (Anwendungsfall für Messaging) die folgenden Schritte aus:
   - Wählen Sie unter Monthly SMS Volume (Monatliches SMS-Aufkommen) die Anzahl der SMS-Nachrichten aus, die jeden Monat gesendet werden.
   - Wählen Sie unter Use Case Category (Anwendungsfall-Kategorie) einen der folgenden Anwendungsfalltypen für die Nummer aus:
     - Two-factor authentication (Zwei-Faktor-Authentifizierung) – verwenden Sie diese Option, um Zwei-Faktor-Authentifizierungscodes zu senden.
     - One-time passwords (Einmalige Passwörter) – verwenden Sie diese Option, um einem Benutzer ein einmaliges Passwort zu senden.
     - Benachrichtigungen – Verwenden Sie diese Option, wenn Sie Ihren Benutzern nur wichtige Benachrichtigungen senden möchten.
     - Polling and surveys (Abfragen und Umfragen) – Verwenden Sie diese Option, um Benutzer zu ihren Präferenzen zu befragen.
     - Info on demand (Info auf Anfrage) – Mit dieser Option werden Nachrichten an Benutzer gesendet, nachdem diese eine entsprechende Anfrage gesendet haben.
     - Promotions and Marketing (Werbeaktionen und Marketing) – Verwenden Sie diese Option, wenn Sie nur Marketingbotschaften an Ihre Benutzer senden möchten.
     - Andere – Verwenden Sie diese Option, wenn Ihr Anwendungsfall in keine andere Kategorie fällt. Vergewissern Sie sich, dass Sie die Use Case Details (Anwendungsfalldetails) für diese Option eingeben.
   - Füllen Sie die Use Case Details (Anwendungsfalldetails) – optional aus, um zusätzlichen Kontext zur ausgewählten Use Case Category (Anwendungsfallkategorie) bereitzustellen.

9. Führen Sie im Abschnitt Messaging Samples (Nachrichtenbeispiele) die folgenden Schritte aus:
   - Geben Sie unter Message Sample 1 (Nachrichtenbeispiel 1) eine Beispielnachricht für einen SMS-Nachrichtentext ein, der an Ihre Endbenutzer gesendet wird.
   - Geben Sie in den Feldern Nachrichtenbeispiel 2 – optional und Nachrichtenbeispiel 3 – optional können Sie bei Bedarf weitere Beispielnachrichten des SNS-Nachrichtentexts ein, der gesendet wird.
   - Jedes Textfeld für ein Message Sample (Nachrichtenbeispiel) hat eine maximale Zeichenbeschränkung von 306 Zeichen.
10. Klicken Sie abschließend auf Submit registration (Registrierung einreichen).

> **Important**
>
> Sie können Ihren Registrierungsstatus überprüfen, indem Sie den Anweisungen unter Status der Registrierung einer Sender-ID für Singapur folgen.
> Warten Sie nicht auf eine Genehmigung oder eine Statusänderung, nachdem Sie Ihre Absender-ID-Registrierung eingereicht haben. Gehen Sie sofort zu Registrierung einer Sender-ID beim Singapore Network Information Centre (SGNIC).

Manueller Registrierungsvorgang zum Abschluss der Registrierung für die Sender-ID für Singapur

Verwenden Sie diese Schritte zur Registrierung einer Absender-ID, wenn sich das Konto nicht in einer Amazon-Pinpoint-Region befindet. Falls sich Ihr Konto in einer Amazon-Pinpoint-Region befindet, finden Sie weitere Informationen unter Registrieren Ihrer Sender-ID für Singapur bei Amazon Pinpoint.

1. Laden Sie die Singapore_Sender_ID_Registration_LOA_Template.zip herunter und füllen Sie die erforderlichen Informationen aus.
2. Erstellen Sie einen Fall beim AWS-Support.
3. Wählen Sie auf der Registerkarte Open support cases (Offene Support-Fälle) die Option Create case (Fall erstellen) aus.
4. Wählen Sie Nach Erhöhungen des Servicelimits suchen und als Limittyp SNS Text Messaging aus.
5. Wählen Sie für Resource Type (Ressourcentyp) die Option Sender ID Registration (Sender-ID-Registrierung) aus.
6. Fügen Sie das LOA-Dokument an, dann können Sie die Anfrage einreichen.

Registrierung einer Sender-ID beim Singapore Network Information Centre (SGNIC)

> **Warning**
>
> Wenn Sie die Reihenfolge dieser Schritte nicht einhalten, wird möglicherweise Ihre Sender-ID vom Service blockiert oder die Sender-ID wird nicht auf dem Mobilgerät gespeichert.

1. Sie müssen zuerst Ihre Sender-ID für Singapur (SG) für Ihr Konto bei AWS registrieren (Amazon-Pinpoint-Konsole oder manuelle Registrierung für Nicht-Amazon-Pinpoint-Regionen). Sobald dies abgeschlossen ist, können Sie mit dem nächsten Schritt fortfahren.
2. Arbeiten Sie mit dem SGNIC zusammen, um Ihre Sender-ID mithilfe des Verfahrens unter SGNIC SMS Sender ID Registry (Registrierung der SGNIC-SMS-Sender-ID) zu registrieren.
   - Stellen Sie beim Abschluss des Vorgangs sicher, dass Sie alle folgenden Aggregatoren als Ihre teilnehmenden Aggregatoren angeben:
     - AMCS SG Private Limited (Amazon Media Communications Services)
     - Nexmo PTE LTD
     - Sinch Singapur PTE LTD
     - Telesign Singapur PTE LTD
     - Twilio Singapur PTD LTD

> **Note**
>
> Sie müssen für jedes einzelne AWS-Konto, das Sie für die Nutzung der Sender-ID benötigen, eine Sender-ID-Registrierung einreichen.

Status der Registrierung einer Sender-ID für Singapur

Wenn Sie Ihre Sender-ID für Singapur bei Amazon SNS registrieren, wird Ihre Registrierung in einem von fünf verschiedenen Status angezeigt:

- Erstellt – Ihre Registrierung wurde erstellt, aber nicht abgesendet.
- Abgesendet – Ihre Registrierung wurde abgesendet und wird überprüft.
- Wird geprüft – Ihre Registrierung wurde angenommen und wird überprüft. Es kann zwischen 1–3 Wochen dauern und in einigen Fällen auch länger, bis die Überprüfung abgeschlossen ist.

- Abgeschlossen – Ihre Registrierung wurde genehmigt und Sie können die Sender-ID verwenden.
- Aktualisierungen erforderlich – Sie müssen Ihre Registrierung korrigieren und erneut einreichen. Weitere Informationen finden Sie unter Bearbeiten einer Sender-ID-Registrierung für Singapur. Felder, die aktualisiert werden müssen, zeigen ein Warnsymbol und eine kurze Beschreibung des Problems.

Für alle Regionen außer Amazon-Pinpoint-Regionen sendet der AWSSupport bei der Registrierung eine E-Mail-Bestätigung oder erstellt einen Fall beim AWS-Support.

- Wählen Sie auf der Registerkarte Open support cases (Offene Support-Fälle) die Option Create case (Fall erstellen) aus.
- Wählen Sie Service Limit increase (Erhöhung des Servicelimits).
- Wählen Sie als Ressourcentyp die Option Sender ID Registration (Sender-ID-Registration) und als Limit die Option General Inquiry (Allgemeine Anfrage) aus.

Überprüfen Ihres Registrierungsstatus

1. Melden Sie sich bei der AWS-Managementkonsole an und öffnen Sie die Amazon-Pinpoint-Konsole unter https://console.aws.amazon.com/pinpoint/.
2. Klicken Sie im Navigationsbereich unter SMS and voice (SMS und Sprache) auf Phone numbers (Telefonnummern).
3. Wählen Sie auf der Registerkarte Sender ID registrations (Sender-ID-Registrierungen) die SenderID aus.
4. Sie können daraufhin den Registrierungsstatus jeder SenderID anzeigen.

Bearbeiten einer Sender-ID-Registrierung für Singapur

Nachdem Sie Ihre Registrierung für Amazon Pinpoint abgesendet haben, wird der Registrierungsstatus als Updates erforderlich angezeigt, wenn bei der Registrierung ein Problem aufgetreten ist. In diesem Status kann das Anmeldeformular bearbeitet werden. Felder, die aktualisiert werden müssen, werden mit einem Warnsymbol und einer kurzen Beschreibung des Problems versehen.

So bearbeiten Sie eine Sender-ID

1. Öffnen Sie die Amazon Pinpoint-Konsole unter https://console.aws.amazon.com/pinpoint/.

2. Klicken Sie im Navigationsbereich unter SMS and voice (SMS und Sprache) auf Phone numbers (Telefonnummern).
3. Wählen Sie auf der Registerkarte SenderID Registration (SenderID-Registrierung) die Nummer aus, die Sie bearbeiten möchten, und wählen Sie die Registration ID (Registrierungs-ID) aus.
4. Wählen Sie Update registration (Registrierung aktualisieren) aus, um das Formular zu bearbeiten und Felder, die ein Warnsymbol haben, zu korrigieren.
5. Wenn Sie registering on behalf of another brand/entity (im Namen einer anderen Marke/Entität registrieren), müssen Sie die zuvor eingereichten Dateien für das Letter of authorization image – optional (Bild des Autorisierungsschreibens – optional) erneut hochladen.
6.
> **Important**
>
> Überprüfen Sie erneut alle Felder, um sicherzustellen, dass sie korrekt sind.

7. Wählen Sie zum erneuten Absenden Submit registration (Registrierung absenden) aus, wenn Sie fertig sind.

Löschen einer Sender-ID-Registrierung für Singapur

Wenn Sie mit der Registrierung Ihrer Sender-ID für Singapur nicht mehr fortfahren möchten, können Sie die Registrierung löschen. Registrierungen können nur gelöscht werden, wenn ihr Status Erstellt oder Aktualisierung erforderlich lautet.

So löschen Sie eine Registrierung

1. Öffnen Sie die Amazon Pinpoint-Konsole unter https://console.aws.amazon.com/pinpoint/.
2. Klicken Sie im Navigationsbereich unter SMS and voice (SMS und Sprache) auf Phone numbers (Telefonnummern).
3. Wählen Sie auf der Registerkarte Sender ID (Sender-ID) die Registrierungs-ID aus, die Sie bearbeiten möchten, und wählen Sie Delete Registration (Registrierung löschen) aus.

Probleme bei der Registrierung für Singapur

Wenn Ihre Sender-ID für Singapur von Amazon Pinpoint nicht akzeptiert wird, wird eine Meldung mit dem Grund für die Ablehnung angezeigt. Wenn Sie Fragen zu diesen Ablehnungen haben, die in unseren Bewährten Methoden nicht beantwortet werden, können Sie eine Anfrage an unsere Support-Teams einreichen.

So reichen Sie eine Anfrage nach Informationen zu einer abgelehnten Sender-ID für Singapur ein

1. Öffnen Sie die Amazon Pinpoint-Konsole unter https://console.aws.amazon.com/pinpoint/.
2. Klicken Sie auf Support und wählen Sie dann Supportcenter.
3. Wählen Sie auf der Seite Support Fall erstellen aus.
4. Wählen Sie für Case type (Falltyp) Service limit increase (Erhöhen von Service-Limits) aus.
5. Wählen Sie für Limit Type (Limittyp) die Option Pinpoint SMS aus.
6. Führen Sie im Abschnitt Requests (Anfragen) folgende Schritte aus:
   - Wählen Sie bei Resource Type (Ressourcentyp) die Option Sender ID Registation (Sender-ID-Registrierung) aus.
   - Wählen Sie als Limit die Option Registration Rejection Query (Anfrage zur Registrierungsablehnung) aus.
7. Geben Sie als Use case description (Beschreibung des Anwendungsfalls) die abgelehnte Sender-ID für Singapur und den angegebenen Ablehnungsgrund ein.
8. Wählen Sie unter Contact options (Kontaktoptionen) für Preferred contact language (Bevorzugte Kontaktsprache) die Sprache, die Sie bei der Kommunikation mit dem AWS Support-Team bevorzugen.
9. Wählen Sie für Contact method (Kontaktmethode) die bevorzugte Methode zur Kommunikation mit dem AWS Support-Team.
10. Wählen Sie Submit (Absenden) aus.

Das AWS-Support-Team wird Ihnen dann in Ihrem AWS-Support-Fall Informationen dazu bereitstellen, warum die Registrierung Ihrer Sender-ID abgelehnt wurde.

Häufig gestellte Fragen zur Registrierung der Sender-ID für Singapur

Häufig gestellte Fragen zum Registrierungsprozess für Sender-ID-Nummern für Singapur mit Amazon Pinpoint.

Habe ich derzeit eine Sender-ID für Singapur?

So überprüfen Sie, ob Sie eine Sender-ID für Singapur haben

1. Öffnen Sie die Amazon Pinpoint-Konsole unter https://console.aws.amazon.com/pinpoint/.
2. Klicken Sie im Navigationsbereich unter SMS and voice (SMS und Sprache) auf Phone numbers (Telefonnummern).

3. Wählen Sie auf der Registerkarte SenderID Registration (SenderID-Registrierung) die Sender-ID aus, die Sie anzeigen möchten, und wählen Sie die Registration ID (Registrierungs-ID) aus.

Wie lange dauert die Registrierung?

Während eine typische Überprüfung 1–3 Wochen dauert, kann es in einigen Fällen bis zu 5 Wochen oder länger dauern, bis Ihre Daten von den Regierungsbehörden verifiziert wurden.

Was ist eine Unique Entity Number (UEN) und wie erhalte ich sie?

Eine UEN ist eine Unterehmens-ID in Singapur, die von der Accounting and Corporate Regulatory Agency (ACRA) ausgestellt wird. Lokale Unternehmen und Unternehmen in Singapur können eine UEN erhalten, indem sie eine über die ACRA beantragen. Sobald Sie das Registrierungs- und das Standardgründungsverfahren bestanden haben, wird sie ausgestellt. Sie können über Bizfile eine UEN bei der ACRA beantragen.

Muss ich mich für eine Sender-ID für Singapur registrieren?

Ja. Wenn Sie Ihre Sender-ID für Singapur derzeit nicht bis zum 30.01.2023 registriert haben, wird die ID jeder Nachricht, die mit einer Sender-ID gesendet wird, in LIKELY-SCAM geändert.

Wie registriere ich meine Sender-ID für Singapur mit Amazon Pinpoint?

Folgen Sie den Anweisungen unter „Registrieren Ihrer Sender-ID für Singapur mit Amazon Pinpoint“, um eine Sender-ID zu registrieren.

Was ist der Registrierungsstatus meiner Sender-ID für Singapur und was bedeutet er?

Folgen Sie den Anweisungen unter „Registrierungsstatus der Sender-ID für Singapur“, um Ihre Registrierung und Ihren Status zu überprüfen.

Welche Informationen muss ich angeben?

Sie müssen die Adresse Ihres Unternehmens, einen Geschäftskontakt und einen Anwendungsfall angeben. Die erforderlichen Informationen finden Sie unter „Registrieren Ihrer Sender-ID für Singapur mit Amazon Pinpoint“.

Was passiert, wenn meine Registrierung für die Sender-ID für Singapur abgelehnt wird?

Wenn Ihre Registrierung abgelehnt wird, wird ihr Status in „Updates erforderlich“ geändert und Sie können Aktualisierungen gemäß den Anweisungen unter „Bearbeiten einer Sender-ID für Singapur“ vornehmen.

Welche Berechtigungen benötige ich?

Der IAM-Benutzerrolle, mit der Sie die Amazon-Pinpoint-Konsole besuchen, muss mit der Berechtigung *„sms-voice:*“* aktiviert werden.

Ursprungsnummern

Eine Ursprungsnummer ist ein numerische Zeichenfolge, die die Telefonnummer des Senders einer SMS-Nachricht identifiziert. Wenn Sie eine SMS-Nachricht mit einer Ursprungsnummer senden, zeigt das Gerät des Empfängers die Ursprungsnummer als Telefonnummer des Senders an. Sie können verschiedene Ursprungsnummern nach Anwendungsfall angeben.

> **Tip**
>
> Um eine Liste aller vorhandenen Ursprungsnummern in Ihrem AWS-Konto anzuzeigen, wählen Sie im Navigationsbereich der Amazon-SNS-Konsole Ursprungsnummern aus.

Die Support für Ursprungsnummern ist in Ländern nicht verfügbar, in denen lokale Gesetze die Verwendung von Sender-IDs anstelle von Ursprungsnummern erfordern.

Themen

- 10DLC
- Gebührenfreie Nummern
- Kurzwahlnummern
- Person-zu-Person (P2P)-Langwahlnummern
- Vergleich der US-Produktnummern

10DLC

US-Carrier unterstützen nicht mehr die Verwendung von Application-to-Person-SMS-Messaging (A2P) über lokale, nicht registrierte Langwahlnummern. Für hochvolumiges A2P-SMS-Messaging bieten US-Carrier stattdessen eine neue Art von Langwahlnummern, genannt „10-Digit Long Codes” (10DLC).

Important

Ab dem 26. Januar 2023 führten die SMS-Anbieter von Amazon SNS neue manuelle Überprüfungsprozesse für 10DLC-Kampagnen ein, um die von US-Anbietern geäußerten Bedenken bezüglich SMS-Spam auszuräumen. Alternativ zu 10DLC können Sie SMS mit Kurzcodes und gebührenfreien Nummern in den USA senden.
Derzeit haben unsere SMS-Anbieter keine Service-Level-Ziele zur Dauer der Überprüfung der 10-DLC-Kampagnen angegeben. Die Überprüfungen werden ausgelöst, sobald einer 10DLC-Kampagne eine Nummer zugeordnet ist. Die Überprüfungen dauern länger als die zuvor von Amazon SNS mitgeteilte Schätzung von 14 Tagen.
Amazon SNS arbeitet täglich mit SMS-Anbietern zusammen, um Folgendes sicherzustellen:

- Anbieter schließen alle ausstehenden Überprüfungen der 10-DLC-Kampagnen so schnell wie möglich ab
- Anbieter priorisieren AWS-Anfragen in ihren Backlogs

Sie können den Status von 10DLC-Kampagnen überprüfen, indem du den Anweisungen unter 10DLC-Kampagnen folgst. Wenn zusätzliche Informationen für die Genehmigung einer 10DLC-Kampagne erforderlich sind, wird das AWS-Support-Team Sie benachrichtigen. Möglicherweise erfolgt die Registrierung einer gebührenfreien Nummer in den USA schneller als der Erhalt einer 10DLC-Nummer. Weitere Informationen zu gebührenfreien Nummern in den USA sowie zum Registrierungsprozess finden Sie unter Anforderungen und Ablauf der Registrierung von gebührenfreien Nummern.

Was ist 10DLC?

10DLC ist eine Art Langwahlnummer, die bei Netzbetreibern registriert ist, um hochvolumiges A2P-SMS-Messaging mit dem 10-stelligen Telefonnummernformat zu unterstützen. Amazon SNS bietet keine lokalen Langwahlnummern mehr als SMS-Produkt an und bietet stattdessen 10DLC an. 10DLC wirkt sich nicht auf Sie aus, wenn Sie nur Kurzwahlnummern und gebührenfreie Nummern verwenden.

10DLC ist eine 10-stellige Telefonnummer, die nur in den Vereinigten Staaten verwendet wird. Nachrichten, die von einem 10DLC an Empfänger gesendet werden, zeigen als Sender eine 10-stellige Zahl an. Im Gegensatz zu gebührenfreien Nummern unterstützt 10DLC sowohl transaktionales als auch Werbe- Messaging und kann jede Vorwahl in den USA enthalten.

Wenn Sie lokale Langwahlnummern haben, können Sie anfordern, dass ihre lokalen Langwahlnummern für 10DLC aktiviert werden. Führen Sie dazu den 10DLC-Registrierungsprozess aus und senden Sie dann ein Support-Ticket ein. Für den Fall, dass ein Problem mit der Aktivierung Ihrer Langwahlnummern für 10DLC auftritt, werden Sie benachrichtigt und angewiesen, über die Amazon Pinpoint (nicht die Amazon SNS)-Konsole eine neue 10DLC anzufordern. Informationen zum Einreichen eines Support-Tickets zum Konvertieren einer Langwahlnummer finden Sie unter Zuordnen einer Langwahlnummer mit einer 10DLC-Kampagne.

Um eine 10DLC-Nummer zu verwenden, registrieren Sie zuerst Ihr Unternehmen und erstellen Sie eine 10DLC-Kampagne mit der Amazon Pinpoint (nicht der Amazon SNS)-Konsole. AWS teilt diese Informationen mit The Campaign Registry, einem Dritten, der Ihre Registrierung auf der Grundlage der Informationen genehmigt oder ablehnt. In einigen Fällen erfolgt die Registrierung sofort. Wenn Sie sich beispielsweise zuvor bei der Kampagnenregistrierung registriert haben, haben diese möglicherweise bereits Ihre Informationen. Einige Kampagnen können jedoch eine Woche oder länger bis zur Genehmigung in Anspruch nehmen. Nachdem Ihr Unternehmen und Ihre 10DLC-Kampagne genehmigt wurden, können Sie eine 10DLC-Nummer erwerben und diese Ihrer Kampagne zuordnen. Das Anfordern eines 10DLC kann auch bis zu einer Woche bis zur Genehmigung dauern. Obwohl Sie mehrere 10DLCs einer einzelnen Kampagne zuordnen können, können Sie denselben 10DLC nicht über mehrere Kampagnen hinweg verwenden. Für jede Kampagne, die Sie erstellen, benötigen Sie einen eindeutigen 10DLC.

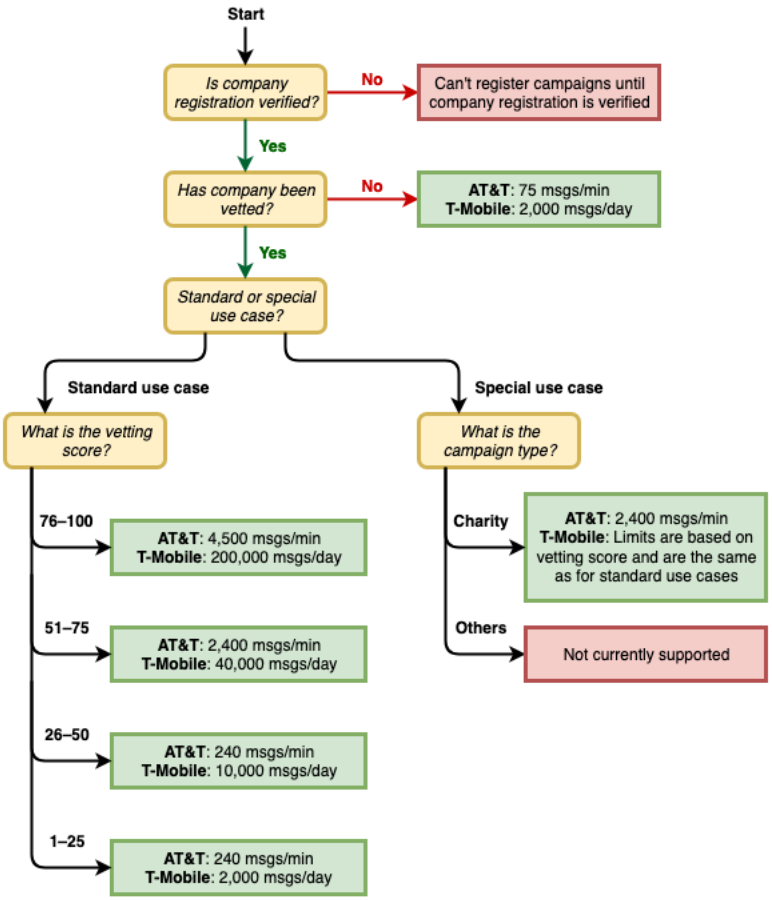
10DLC-Funktionen

Die Kapazitäten von 10DLC-Telefonnummern hängen von den Mobilfunkanbietern Ihrer Empfänger ab. AT&T bietet ein Limit basierend auf der Anzahl der Nachrichtenteile, die pro Minute für jede Kampagne gesendet werden können. T-Mobile bietet ein tägliches Limit für Nachrichten, die für jedes Unternehmen gesendet werden können, ohne Limit der Anzahl der Nachrichtenteile, die pro Minute gesendet werden können. Verizon hat keine Durchsatzlimits veröffentlicht, verwendet jedoch ein Filtersystem für 10DLC, das Spam, unerwünschte Nachrichten und missbräuchliche Inhalte entfernt. Dabei wird weniger Wert auf den tatsächlichen Nachrichtendurchsatz gelegt.

Neue 10DLC-Kampagnen, die mit nicht überprüften Unternehmen verbunden sind, können 75 Nachrichtenteile pro Minute an Empfänger, die AT&T verwenden, und 2.000 Nachrichten pro Tag an Empfänger senden, die T-Mobile verwenden. Das Unternehmenslimit gilt für alle Ihre 10DLC-Kampagnen zusammen. Wenn Sie beispielsweise ein Unternehmen und zwei Kampagnen registriert haben, gilt die tägliche Zuteilung von 2.000 Nachrichten an T-Mobile-Kunden für beide Kampagnen zusammen. Wenn Sie dasselbe Unternehmen in mehr als einem AWS-Konto registrieren, gilt die tägliche Zuteilung gleichermaßen für alle Konten zusammen.

Wenn Ihre Durchsatzanforderungen diese Limits überschreiten, können Sie eine Überprüfung Ihrer Unternehmensregistrierung beantragen. Wenn Sie Ihre Unternehmensregistrierung überprüfen lassen, analysiert ein externer Prüfanbieter Ihre Unternehmensdetails. Der Prüfanbieter stellt dann einen Überprüfungswert bereit, der die Kapazitäten Ihrer 10DLC-Kampagnen bestimmt. Für den Überprüfungsdienst wird eine einmalige Gebühr erhoben. Weitere Informationen finden Sie unter Überprüfen Ihrer 10DLC-Registrierung in Amazon SNS.

Ihre tatsächliche Durchsatzrate hängt von verschiedenen Faktoren ab, z. B. davon, ob Ihr Unternehmen überprüft wurde oder nicht, von Ihren Kampagnentypen und von Ihrem Überprüfungswert. Das folgende Flussdiagramm zeigt die Durchsatzraten für verschiedene Situationen.



Die Durchsatzraten für 10DLC werden von den US-Mobilfunkanbietern in Zusammenarbeit mit der Kampagnenregistrierung festgelegt. Weder Amazon SNS noch ein anderer SMS-Versanddienst können den 10DLC-Durchsatz über diese Tarife hinaus erhöhen. Wenn Sie hohe Durchsatzraten und hohe Zustellbarkeitsraten für alle US-Mobilfunkanbieter benötigen, empfehlen wir, eine Kurzwahlnummer zu verwenden. Weitere Informationen zum Erhalt einer Kurzwahlnummer finden Sie unter Anfordern dedizierter Kurzwahlnummern für SMS-Messaging mit Amazon SNS.

Erste Schritte mit 10DLC

Verwenden Sie die Amazon Pinpoint (nicht die Amazon SNS)-Konsole, um 10DLC anzufordern. Gehen Sie wie folgt vor, um 10DLC für die Verwendung mit Ihren 10DLC-Kampagnen einzurichten.

1. Registrieren Sie Ihr Unternehmen.

   Bevor Sie einen 10DLC anfordern können, muss Ihr Unternehmen bei der Kampagnenregistrierung registriert sein. Weitere Informationen finden Sie unter Registrieren eines Unternehmens. Die Registrierung erfolgt in der Regel sofort, es sei denn, die Kampagnenregistrierung erfordert weitere Informationen. Für die Registrierung Ihres Unternehmens wird eine einmalige Anmeldegebühr erhoben, die auf der Registrierungsseite angezeigt wird. Diese Einmalgebühr wird separat von Ihren monatlichen Gebühren für die Kampagne und 10DLC bezahlt.

   > **Note**
   >
   > Amazon SNS SMS-Messaging ist in Regionen verfügbar, in denen Amazon Pinpoint derzeit nicht unterstützt wird. Es gibt zwei verschiedene Fälle:
   >
   > a. Wenn Sie ein kommerzielles Cloud-Konto verwenden, müssen Sie die Amazon-Pinpoint-Konsole in der Region USA Ost (Nord-Virginia) öffnen, um Ihr 10DLC-Unternehmen und Ihre Kampagne zu registrieren. Fordern Sie keine 10DLC-Nummer an.
   >
   > b. Verwenden Sie die AWS-Service-Quotas-Konsole, um einen Fall für die Erhöhung des Service-Limits zu erstellen, während Sie die 10DLC-Nummer für diese Region anfordern. Weitere Informationen zu Regionen, in denen Amazon Pinpoint verfügbar ist, finden Sie unter Amazon Pinpoint Endpunkte und -Kontingente in der Allgemeine AWS-Referenz.
   >
   > c. Wenn Sie ein AWS GovCloud (US)-Konto verwenden, öffnen Sie die Amazon-Pinpoint-Konsole in der Region USA West, um Ihr 10DLC-Unternehmen und Ihre Kampagne zu registrieren. Fordern Sie keine 10DLC-Nummer an. Verwenden Sie stattdessen die AWS-Service-Quotas-Konsole, um einen Fall für die Erhöhung des Service-Limits zu erstellen, während Sie die 10DLC-Nummer für diese Region anfordern. Weitere Informationen zu Regionen, in denen Amazon Pinpoint verfügbar ist, finden Sie unter Amazon Pinpoint Endpunkte und -Kontingente in der Allgemeine AWS-Referenz.

2. (Optional, but recommended) Apply for vetting ((Optional, aber empfohlen) Überprüfung beantragen)

   Wenn Ihre Unternehmensregistrierung erfolgreich war, können Sie mit der Erstellung von 10DLC-Kampagnen für gemischte Anwendungsfälle mit geringem Volumen beginnen. Diese Kampagnen können bis zu 75 Nachrichten pro Minute an Empfänger senden, die AT&T verwenden, und Ihr registriertes Unternehmen kann 2.000 Nachrichten pro Tag an Empfänger senden, die T-Mobile verwenden. Wenn Ihr Anwendungsfall eine Durchsatzrate erfordert, die diese Werte überschreitet, können Sie eine Überprüfung Ihrer Unternehmensregistrierung beantragen. Die Überprüfung Ihrer Unternehmensregistrierung kann eine Erhöhung der Durchsatzraten für Ihre Unternehmen und Kampagnen zur Folge haben, dies ist jedoch nicht garantiert. Weitere Informationen zur Überprüfung finden Sie unter Überprüfen Ihrer 10DLC-Registrierung in Amazon SNS.
3. Registrieren Sie Ihre Kampagne.

   Nachdem Ihr Unternehmen registriert ist, erstellen Sie eine 10DLC-Kampagne und verknüpfen Sie sie mit einem Ihrer registrierten Unternehmen. Diese Kampagne wird zur Genehmigung an die Kampagnenregistrierung übermittelt. In den meisten Fällen erfolgt die 10DLC-Kampagnengenehmigung sofort, es sei denn, die Kampagnenregistrierung erfordert weitere Informationen. Weitere Informationen finden Sie unter Registrieren einer 10DLC-Kampagne.
4. Fordern Sie Ihre 10DLC-Nummer an.

   Nachdem Ihre 10DLC-Kampagne genehmigt wurde, können Sie einen 10DLC anfordern und diese Nummer der genehmigten Kampagne zuordnen. Ihre 10DLC-Kampagne kann nur eine dafür genehmigte Nummer verwenden. Siehe Anfordern von 10DLC-Nummern, gebührenfreien Nummern und P2P-Langwahlnummern für SMS-Messaging mit Amazon SNS.

10DLC-Registrierung und monatliche Gebühren

Es gibt Registrierungs- und monatliche Gebühren im Zusammenhang mit der Verwendung von 10DLC, wie die Registrierung Ihres Unternehmens und die 10DLC-Kampagne. Diese sind getrennt von allen anderen monatlichen Gebühren, die von AWS erhoben werden. Weitere Informationen finden Sie auf der Seite Amazon SNS Weltweite SMS-Preise.

Registrieren eines Unternehmens

Bevor Sie einen 10DLC anfordern können, muss Ihr Unternehmen bei der Kampagnenregistrierung (The Campaign Registry) registriert sein.

> **Note**
>
> Amazon SNS SMS-Messaging ist in Regionen verfügbar, in denen Amazon Pinpoint derzeit nicht unterstützt wird. Öffnen Sie in diesen Fällen die Amazon Pinpoint Konsole in der Region USA Ost (Nord-Virginia), um Ihr 10DLC-Unternehmen und Ihre Kampagne zu registrieren, aber fordern Sie keine 10DLC-Nummer an. Verwenden Sie stattdessen die AWS-Service-Quotas-Konsole, um einen Fall für die Erhöhung des Service-Limits zu erstellen, während Sie die 10DLC-Nummer für diese Region anfordern. Weitere Informationen zu Regionen, in denen Amazon Pinpoint verfügbar ist, finden Sie unter Amazon Pinpoint Endpunkte und -Kontingente in der Allgemeine AWS-Referenz.

Status der 10DLC-Unternehmensregistrierung

Wenn Sie Ihr Unternehmen oder Ihre Marke registrieren, wird einer von zwei Status zurückgegeben: entweder Unverified (Nicht verifiziert) oder Verified (Verifiziert). Wenn der Status für Ihre Unternehmensregistrierung Unverified (Nicht verifiziert) lautet, bedeutet dies, dass es ein Problem mit Ihrer Registrierung gab. Beispielsweise stimmt der von Ihnen angegebene registrierte Name des Unternehmens möglicherweise nicht genau mit dem registrierten Namen des Unternehmens überein, das mit der angegebenen Steuernummer verknüpft ist. Wenn Sie ein Problem mit den Details zu Ihrer Unternehmensregistrierung feststellen, können Sie diese korrigieren. Weitere Informationen zum Ändern der Details zu Ihrer Unternehmensregistrierung finden Sie unter Bearbeiten oder Löschen eines registrierten Unternehmens.

Wenn der Status für Ihre Unternehmensregistrierung „Verified“ (Verifiziert) lautet, waren die von Ihnen angegebenen Registrierungsdaten korrekt und Sie können mit der Erstellung von 10DLC-Kampagnen beginnen.

Registrieren Ihres Unternehmens oder Ihrer Marke

Sie müssen Ihr Unternehmen nur einmal registrieren. Nach der Registrierung können Sie Ihre Unternehmens- und Kontaktinformationen bearbeiten. Wenn Sie ein registriertes Unternehmen löschen möchten, erstellen Sie einen Fall beim AWS-Support. Weitere Informationen zum Bearbeiten oder Löschen von Unternehmensdetails finden Sie unter Bearbeiten oder Löschen eines registrierten Unternehmens.

So registrieren Sie ein Unternehmen

1. Melden Sie sich bei der AWS Management Console an und öffnen Sie die Amazon-Pinpoint-Konsole unter https://console.aws.amazon.com/pinpoint/.
2. Klicken Sie im Navigationsbereich unter SMS and voice (SMS und Sprache) auf Phone numbers (Telefonnummern).
3. Wählen Sie auf der Registerkarte 10DLC campaigns (10DLC-Kampagnen) die Option Register company (Unternehmen registrieren) aus.

> **Note**
>
> Auf der Seite Register your company (Unternehmen registrieren) wird die Registration fee(Registrierungsgebühr) angezeigt. Dies ist eine Einmalgebühr, die mit der Registrierung Ihres Unternehmens verbunden ist. Diese Kosten fallen separat zu allen anderen monatlichen Kosten oder Gebühren an. Sie werden Ihnen in Rechnung gestellt, wenn Sie Ihr Unternehmen registrieren oder die Details einer vorhandenen Unternehmensregistrierung ändern.

4. Führen Sie im Abschnitt Company info (Unternehmensinformationen) die folgenden Schritte aus:

   - Geben Sie unter Legal company name (Offizieller Unternehmensname) den Namen ein, unter dem das Unternehmen registriert ist. Der eingegebene Name muss exakt mit dem Unternehmensnamen übereinstimmen, der mit der angegebenen Steuernummer verknüpft ist.

   > **Important**
   >
   > Stellen Sie sicher, dass Sie den exakten offiziellen Namen Ihres Unternehmens verwenden. Nach der Übermittlung können Sie diese Informationen nicht mehr ändern. Falsche oder unvollständige Informationen können dazu führen, dass sich Ihre Registrierung verzögert oder dass sie abgelehnt wird.

   - Wählen Sie unter What type of legal form is this organization (Welche Art von Rechtsform ist diese Organisation) die Option aus, die Ihr Unternehmen am besten beschreibt.

> **Note**
>
> Die Optionen US government (US-Regierung) und Not-for-profit (Nicht gewinnorientiert) können nur verwendet werden, um in den USA ansässige Organisationen zu registrieren. Wenn Ihre Organisation in einem anderen Land als den USA ansässig ist, müssen Sie sie als Private for-profit (Privat – gewinnorientiert) registrieren, unabhängig von der tatsächlichen Rechtsform Ihrer Organisation.

- Wenn Sie im vorherigen Schritt Public for profit (Öffentlich – gewinnorientiert) ausgewählt haben, geben Sie das Tickersymbol Ihres Unternehmens sowie die Börse ein, an der das Unternehmen gelistet ist.
- Wählen Sie unter Country of registration (Registrierungsland) das Land aus, in dem das Unternehmen registriert ist.
- Geben Sie unter Doing Business As (DBA) or brand name (Geschäfte tätigend als oder Markenname) alle weiteren Namen ein, unter denen Ihr Unternehmen Geschäfte tätigt.
- Geben Sie unter Tax ID (Steuernummer) die Steuernummer Ihres Unternehmens ein. Die eingegebene ID hängt vom Land ab, in dem Ihr Unternehmen registriert ist.
  - Wenn Sie ein Unternehmen aus den USA oder von außerhalb der USA registrieren, das über eine IRS Employer Identification Number (EIN) verfügt, geben Sie Ihre neunstellige EIN ein. Der offizielle Unternehmensname, die EIN und die physische Adresse, die Sie angeben, müssen mit den Unternehmensinformationen übereinstimmen, die bei der Steuerbehörde registriert sind.
  - Wenn Sie ein kanadisches Unternehmen registrieren, geben Sie Ihre Unternehmensnummer des Bundesstaats oder der Provinz ein. Geben Sie nicht die von der CRA bereitgestellte Geschäftsnummer (Business Number, BN) ein. Der offizielle Unternehmensname, die Unternehmensnummer und die physische Adresse, die Sie angeben, müssen mit den Unternehmensinformationen übereinstimmen, die bei Corporations Canada registriert sind.
  - Wenn Sie ein Unternehmen registrieren, das in einem anderen Land ansässig ist, geben Sie die primäre Steuernummer für Ihr Land ein. In vielen Ländern ist dies der numerische Teil Ihrer Umsatzsteuer-Identifikationsnummer.
- Wählen Sie unter Vertical (Vertikalmarkt) die Kategorie aus, die das registrierte Unternehmen am besten beschreibt.

5. Führen Sie im Abschnitt Contact info (Kontaktinformationen) die folgenden Schritte aus:

- Geben Sie unter Address/Street (Adresse/Straße) die Straßenanschrift Ihres Unternehmens ein.
- Geben Sie unter City (Ort) den Ort ein, in der sich diese Straßenanschrift befindet.
- Geben Sie unter State or region (Bundesstaat/-land oder Region) den Bundesstaat/das Bundesland oder die Region ein, in dem/der sich die Adresse befindet.
- Geben Sie unter Zip Code/Postal Code (PLZ) die Postleitzahl für die Adresse ein.
- Geben Sie unter Company website (Unternehmenswebsite) die vollständige URL der Website Ihres Unternehmens ein. Fügen Sie „http://“ oder „https://“ am Anfang der Adresse ein.
- Geben Sie unter Support email (E-Mail-Adresse Support) eine E-Mail-Adresse ein.
- Geben Sie unter Support phone number (Telefonnummer Support) eine Telefonnummer mit Ländervorwahl ein.

> **Note**
>
> Die Kampagnenregistrierung benötigt die E-Mail-Adresse und Telefonnummer eines Ansprechpartners, falls die Registrierungsinformationen bei einem Vertreter Ihres Unternehmens überprüft werden müssen.

6. Wenn Sie fertig sind, klicken Sie auf Create. Ihre Unternehmensregistrierung wird an die Kampagnenregistrierung übermittelt. In den meisten Fällen wird Ihre Registrierung sofort akzeptiert und es wird ein Status angegeben.

Wenn der Status für Ihre Unternehmensregistrierung Verified (Verifiziert) lautet, können Sie mit der Erstellung von 10DLC-Kampagnen für gemischte Anwendungsfälle mit geringem Volumen beginnen. Sie können diese Art von Kampagne verwenden, um bis zu 75 Nachrichten pro Minute an Empfänger zu senden, die AT&T verwenden, und Ihr registriertes Unternehmen kann 2.000 Nachrichten pro Tag an Empfänger senden, die T-Mobile verwenden. Sie können auch Nachrichten an Empfänger senden, die andere US-amerikanische Netzbetreiber wie Verizon und US Cellular verwenden. Diese Anbieter setzen Durchsatzlimits nicht strikt durch, überwachen jedoch 10DLC-Nachrichten intensiv auf Anzeichen von Spam und Missbrauch.

Wenn Ihr Anwendungsfall eine Durchsatzrate erfordert, die diese Werte überschreitet, können Sie eine zusätzliche Überprüfung Ihrer Unternehmensregistrierung beantragen. Weitere Informationen zur Überprüfung Ihrer Markenregistrierung finden Sie unter Überprüfen Ihrer 10DLC-Registrierung in Amazon SNS.

Wenn der Status für Ihre Unternehmensregistrierung Unverified (Nicht verifiziert) lautet, gab es Probleme mit den bereitgestellten Informationen. Überprüfen Sie die bereitgestellten Informationen und bestätigen Sie, dass alle Felder die richtigen Informationen enthalten. Sie können Änderungen an einigen Teilen Ihrer Unternehmensregistrierung in der Amazon-Pinpoint-Konsole vornehmen. Weitere Informationen zum Ändern der Details zu Ihrer Unternehmensregistrierung finden Sie unter Bearbeiten einer 10DLC-Unternehmensregistrierung.

Überprüfen Ihrer 10DLC-Registrierung in Amazon SNS

Wenn Ihre Unternehmensregistrierung erfolgreich ist und Sie eine Kampagne mit höheren Durchsatzkapazitäten registrieren möchten, müssen Sie Ihre Unternehmensregistrierung überprüfen lassen.

Bei der Überprüfung Ihrer Registrierung analysiert eine Drittorganisation die von Ihnen bereitgestellten Unternehmensdaten und gibt einen Überprüfungswert zurück. Ein hoher Überprüfungswert kann zu höheren Durchsatzraten für Ihr 10DLC-Unternehmen und die damit verbundenen Kampagnen führen. Es ist jedoch nicht garantiert, dass Ihr Durchsatz durch die Überprüfung erhöht wird.

Überprüfungswerte werden nicht rückwirkend angewendet. Das heißt, wenn Sie bereits eine 10DLC-Kampagne erstellt haben und später Ihre Unternehmensregistrierung überprüfen lassen, wird der Überprüfungswert nicht automatisch auf Ihre bestehende Kampagne angewendet. Aus diesem Grund sollten Sie Ihr Unternehmen oder Ihre Marke überprüfen lassen, bevor Sie eine 10DLC-Kampagne erstellen.

> **Note**
>
> Für die Überprüfung Ihres Unternehmens oder Ihrer Marke wird eine nicht erstattungsfähige Gebühr in Höhe von 40 USD erhoben.

So lassen Sie Ihre Unternehmensregistrierung überprüfen

1. Melden Sie sich bei der AWS Management Console an und öffnen Sie die Amazon-Pinpoint-Konsole unter https://console.aws.amazon.com/pinpoint/.
2. Klicken Sie im Navigationsbereich unter SMS and voice (SMS und Sprache) auf Phone numbers (Telefonnummern).
3. Wählen Sie auf der Registerkarte 10DLC campaigns (10DLC-Kampagnen) das 10DLC company (10DLC-Unternehmen) aus, das Sie überprüfen lassen möchten.

4. Wählen Sie unten auf der Seite mit den Unternehmensdetails die Option Apply for vetting (Überprüfung beantragen) aus.
5. Wählen Sie im Fenster Apply for additional vetting (Zusätzliche Überprüfung beantragen) die Option Submit (Absenden) aus.

   Für Unternehmen mit Sitz in den USA dauert der Überprüfungsprozess in der Regel etwa eine Minute. Für nicht in den USA ansässige Unternehmen kann der Überprüfungsprozess erheblich länger dauern, je nachdem, wie leicht die Daten für dieses Land verfügbar sind.

Nachdem Sie einen Überprüfungantrag eingereicht haben, kehren Sie zur Seite mit den Unternehmensdetails zurück. Im Abschnitt Company vetting results (Ergebnisse der Unternehmensüberprüfung) werden der Status und die Ergebnisse Ihres Überprüfungsantrags angezeigt. Wenn der Überprüfungsprozess abgeschlossen ist, zeigt diese Tabelle einen Überprüfungswert in der Spalte Score (Ergebnis) an. Ihr Überprüfungswert bestimmt Ihre 10DLC-Durchsatzkapazitäten. Ihr Durchsatz variiert je nach Art der Kampagne, die Sie erstellen. Wenn Sie 10DLC-Kampagnen für gemischte Anwendungsfälle oder für Marketingzwecke erstellen, benötigen Sie einen höheren Überprüfungswert als für andere Kampagnentypen, um hohe Durchsatzraten zu erzielen. Weitere Informationen zu den Kapazitäten von 10DLC-Telefonnummern finden Sie unter 10DLC-Funktionen.

Wenn Sie die Details Ihrer Unternehmensregistrierung nach Abschluss des Überprüfungsprozesses ändern, können Sie die erneute Überprüfung Ihrer Registrierung beantragen. Wenn Sie nur den Vertikalmarkt für Ihre Unternehmensregistrierung ändern, ändert sich Ihr Überprüfungswert nicht. Wenn Sie andere Details als den Vertikalmarkt ändern, kann sich Ihr Überprüfungsergebnis ändern. In beiden Fällen wird Ihnen die einmalige Überprüfungsgebühr erneut in Rechnung gestellt.

Bearbeiten oder Löschen eines registrierten Unternehmens

Sie können einige der 10DLC-Registrierungsinformationen für Ihr Unternehmen direkt in der Amazon-Pinpoint-Konsole bearbeiten. Sie können eine 10DLC-Unternehmensregistrierung auch löschen, indem Sie einen Fall im AWS -Supportcenter erstellen.

Bearbeiten einer 10DLC-Unternehmensregistrierung

Nachdem Sie den 10DLC-Registrierungsprozess für ein Unternehmen abgeschlossen haben, können Sie die Details Ihrer Registrierung bearbeiten.

Wenn nach der Bearbeitung der Details zu Ihrer Unternehmensregistrierung eine Fehlermeldung angezeigt wird, gibt es möglicherweise andere Probleme bei Ihrer Registrierung. Sie können ein Ticket beim AWS Support öffnen, um weitere Informationen anzufordern.

So bearbeiten Sie eine Unternehmensregistrierung

1. Öffnen Sie die AWS SMS Konsole unter https://console.aws.amazon.com/sms-voice/.
2. Folgen Sie den Anweisungen zum Bearbeiten Ihrer Registrierung im Amazon Pinpoint SMS-Benutzerhandbuch.

Löschen einer 10DLC-Unternehmensregistrierung

Um eine Unternehmensregistrierung zu löschen

1. Öffnen Sie die AWS SMS Konsole unter https://console.aws.amazon.com/sms-voice/.
2. Folgen Sie den Anweisungen zum Löschen Ihrer Registrierung im Amazon Pinpoint SMS-Benutzerhandbuch.

Registrieren einer 10DLC-Kampagne

Wenn Sie eine 10DLC-Kampagne registrieren, geben Sie eine Beschreibung Ihres Anwendungsfalls sowie der Nachrichtenvorlagen an, die Sie verwenden möchten. Bevor Sie eine 10DLC-Kampagne erstellen und registrieren können, müssen Sie Ihr Unternehmen registrieren. Informationen zur Registrierung eines Unternehmens finden Sie unter Registrieren eines Unternehmens.

> **Note**
>
> Nachdem Sie Ihr Unternehmen registriert haben, zeigt Amazon Pinpoint einen von zwei Status für die Registrierung an: Verified (Verifiziert) oder Unverified (Nicht verifiziert). Sie können den Registrierungsprozess der 10DLC-Kampagne nur abschließen, wenn der Status Ihrer Unternehmensregistrierung Verified (Verifiziert) lautet. Sie können Kampagnen für gemischte Anwendungsfälle mit geringem Volumen erstellen.
> Wenn der Status Unverified (Nicht verifiziert) lautet, bedeutet dies in der Regel, dass einige der Daten inkorrekt waren, die Sie bei der Registrierung Ihres Unternehmens angegeben haben. Sie können keine 10DLC-Kampagnen erstellen, solange Ihr Unternehmen diesen Status aufweist. Sie können Ihre Unternehmensregistrierung ändern, um zu versuchen, die Probleme bei Ihrer Unternehmensregistrierung zu beheben. Weitere Informationen zum

Ändern von 10DLC-Unternehmensregistrierungen finden Sie unter Bearbeiten oder Löschen eines registrierten Unternehmens.

Auf dieser Seite geben Sie zunächst die Details zu dem Unternehmen an, für das Sie die 10DLC-Kampagne erstellen, und geben dann die Anwendungsfalldetails der Kampagne selbst an. Die Informationen auf dieser Seite werden dann der Kampagnenregistrierung zur Genehmigung zur Verfügung gestellt.

In diesem Abschnitt wählen Sie das Unternehmen aus, für das Sie die 10DLC-Kampagne erstellen möchten, und geben zusätzliche Details an.

So registrieren Sie eine 10DLC-Kampagne

1. Melden Sie sich bei der AWS Management Console an und öffnen Sie die Amazon-Pinpoint-Konsole unter https://console.aws.amazon.com/pinpoint/.
2. Wählen Sie unter SMS and voice (SMS und Sprache) die Option Phone numbers (Telefonnummern) aus.
3. Wählen Sie auf der Registerkarte 10DLC campaigns (10DLC-Kampagnen) die Option Create a 10DLC campaign (10DLC-Kampagne erstellen) aus.
4. Führen Sie auf der Seite Create a 10DLC campaign (10DLC-Kampagne erstellen) im Abschnitt Campaign info (Kampagnen-Informationen) die folgenden Schritte aus:

    a. Wählen Sie bei Unternehmensname das Unternehmen aus, für das Sie diese Kampagne erstellen. Wenn Sie das Unternehmen noch nicht registriert haben, müssen Sie das zuerst tun. Weitere Informationen zur Registrierung eines Unternehmens finden Sie unter Registrieren eines Unternehmens.

    b. Geben Sie unter 10DLC campaign name (Name der 10DLC-Kampagne) einen Namen für die Kampagne ein.

    c. Wählen Sie unter Vertical (Vertikalmarkt) die Option aus, die Ihr Unternehmen am besten darstellt.

    d. Geben Sie unter Help message (Hilfe-Nachricht) die Nachricht ein, die Ihre Kunden erhalten, wenn sie das Stichwort „HILFE“ an Ihre 10DLC-Telefonnummer senden.

    e. Geben Sie unter Stop message (Stopp-Nachricht) die Nachricht ein, die Ihre Kunden erhalten, wenn sie das Stichwort „STOPP“ an Ihre 10DLC-Telefonnummer senden.

> **Tip**
>
> Ihre Kunden können mit dem Wort „HILFE“ auf Ihre Nachrichten antworten, um mehr über die Nachrichten zu erfahren, die sie von Ihnen erhalten. Sie können auch mit „STOPP“ antworten, wenn sie nicht länger Nachrichten von Ihnen erhalten möchten. Die US-Mobilfunkanbieter verlangen, dass Sie Antworten auf diese beiden Stichwörter bereitstellen.
> Im Folgenden finden Sie ein Beispiel für eine HILFE-Antwort, die den Anforderungen der US-Mobilfunkanbieter entspricht:
> **`ExampleCorp Account Alerts: For help call 1-888-555-0142 or go to example.com. Msg&data rates may apply. Text STOP to cancel.`**
> Im Folgenden finden Sie ein Beispiel für eine konforme STOPP-Antwort.
> **`You are unsubscribed from ExampleCorp Account Alerts. No more messages will be sent. Reply HELP for help or call 1-888-555-0142.`**
> Ihre Antworten auf diese Stichwörter dürfen maximal 160 Zeichen enthalten.

5. Führen Sie im Abschnitt Campaign use case (Anwendungsfall für Kampagne) die folgenden Schritte aus:

   a. Wenn Sie einen Anwendungsfall im Zusammenhang mit wohltätigen Zwecken haben, wählen Sie unter Use case type (Anwendungsfalltyp) die Option Special (Speziell) aus. Wählen Sie andernfalls Standard aus.

   b. Wählen Sie für Anwendungsfall einen Anwendungsfall, der Ihrer Kampagne aus der vorhandenen Liste an Anwendungsfällen am meisten ähnelt. Die monatliche Gebühr für jeden Anwendungsfall wird neben dem Namen des Anwendungsfalls angezeigt.

> **Note**
>
> Die monatliche Gebühr für die Registrierung der 10DLC-Kampagne wird neben jedem Anwendungsfalltyp angezeigt. Für die meisten 10DLC-Kampagnentypen fällt die gleiche monatliche Gebühr an. Für die Registrierung von gemischten Anwendungsfällen mit geringem Volumen fällt eine geringere Gebühr als bei anderen Anwendungsfalltypen an. Kampagnen für gemischte Anwendungsfälle mit geringem Volumen unterstützen jedoch geringere Durchsatzraten als andere Kampagnentypen.

c. Geben Sie mindestens eine Sample SMS message (SMS-Beispielnachricht) ein. Dies ist die Beispielnachricht, die Sie an Ihre Kunden senden möchten. Wenn Sie mehrere Nachrichtenvorlagen für diese 10DLC-Kampagne verwenden möchten, fügen Sie diese ebenfalls ein.

> **Important**
>
> Verwenden Sie keinen Platzhaltertext für Ihre Beispielnachrichten. Die eingegebenen Beispielnachrichten sollten die tatsächlichen Nachrichten, die Sie senden möchten, so genau wie möglich widerspiegeln.

6. Der Abschnitt Campaign and content attributes (Kampagnen- und Inhaltsattribute) enthält eine Reihe von Ja oder Nein-Fragen im Zusammenhang mit den Besonderheiten der Kampagne. Einige Attribute sind obligatorisch, sodass Sie den Standardwert nicht ändern können.

   Stellen Sie sicher, dass die von Ihnen ausgewählten Attribute auf Ihre Kampagne zutreffen.

   Geben Sie an, ob die folgenden Punkte für die Kampagne gelten, die Sie registrieren:

   - Abonnentenanmeldung – Abonnenten können sich anmelden, um Nachrichten über diese Kampagne zu erhalten.
   - Abonnentenabmeldung – Abonnenten können sich abmelden, um keine Nachrichten mehr über diese Kampagne zu erhalten.
   - Abonnenten – Abonnenten können den Nachrichten-Sender kontaktieren, nachdem sie das HILFE-Schlüsselwort gesendet haben.
   - Zahlen-Pooling – Diese 10DLC-Kampagne verwendet mehr als 50 Telefonnummern.
   - Direkte Kreditvergabe oder Darlehensvereinbarung – Die Kampagne enthält Informationen über Direktkredite oder andere Darlehensvereinbarungen.
   - Eingebettete Links – Die 10DLC-Kampagne enthält einen eingebetteten Link. Links von gängigen URL-Kürzern wie beispielsweise TinyUrl oder Bit.ly sind nicht zulässig. Sie können jedoch URL-Kürzer verwenden, die benutzerdefinierte Domänen anbieten.
   - Embedded phone number (Eingebettete Telefonnummer) – Die Kampagne enthält eine eingebettete Telefonnummer, bei der es sich nicht um eine Kundenservicenummer handelt.
   - Affiliate-Marketing – Die 10DLC-Kampagne enthält Informationen aus Affiliate-Marketing.

- Altersabhängige Inhalte – Die 10DLC-Kampagne enthält altersgerechte Inhalte, wie sie in den Richtlinien des Mobilfunkbetreibers und CTIA (Cellular Telecommunications and Internet Association) definiert sind.

7. Wählen Sie Erstellen aus.

   Nachdem Sie die Registrierungsdetails für Ihre Kampagne übermittelt haben, wird die SMS- und Sprachseite geöffnet. Es wird eine Meldung angezeigt, die darauf hinweist, dass Ihre Kampagne übermittelt wurde und überprüft wird. Sie können den Status Ihrer Anfrage auf der Registerkarte 10DLC-Kampagnen sehen. Sie können den Status Ihrer Registrierung auf der Registerkarte 10DLC überprüfen. Der Status ist einer der folgenden:

   - Aktiv – Ihre 10DLC-Kampagne wurde genehmigt. Sie können eine 10DLC-Telefonnummer anfordern und diese Nummer mit Ihrer Kampagne verknüpfen. Weitere Informationen finden Sie unter Anfordern von 10DLC-Nummern, gebührenfreien Nummern und P2P-Langwahlnummern für SMS-Messaging mit Amazon SNS.
   - Pending (Ausstehend) – Ihre 10DLC-Kampagne wurde noch nicht genehmigt. In einigen Fällen kann die Genehmigung eine Woche oder länger dauern. Wenn sich der Status ändert, spiegelt die Amazon-Pinpoint-Konsole diese Änderung wider. Wir informieren Sie nicht über Statusänderungen.
   - Rejected (Abgelehnt) – Ihre 10DLC-Kampagne wurde abgelehnt. Um weitere Informationen zu erhalten, senden Sie eine Supportanfrage mit der Kampagnen-ID der abgelehnten Kampagne.
   - Ausgesetzt – Einer oder mehrere Carrier haben Ihre 10DLC-Kampagne ausgesetzt. Um weitere Informationen zu erhalten, senden Sie eine Supportanfrage mit der Kampagnen-ID der gesperrten Kampagne. Amazon Pinpoint enthält keine Gründe für die Sperrung auf der Konsole und wir informieren Sie nicht, wenn Ihre Kampagne ausgesetzt ist.

8. Wenn Ihr 10DLC genehmigt wurde, können Sie eine 10DLC-Nummer anfordern, die dieser Kampagne zugeordnet werden soll. Weitere Informationen zum Anfordern einer 10DLC-Nummer finden Sie unter Anfordern von 10DLC-Nummern, gebührenfreien Nummern und P2P-Langwahlnummern für SMS-Messaging mit Amazon SNS.

Verwenden von 10DLC-Kampagnen in mehreren AWS-Regionen

Wenn Sie ein Unternehmen registrieren, ist dieses Unternehmen in Ihrem AWS-Konto in allen AWS-Regionen verfügbar. Dies gilt jedoch nicht für 10DLC-Kampagnen. Eine 10DLC-Kampagne kann nur in der AWS-Region verwendet werden, in der sie registriert wurde.

Wenn Sie planen, 10DLC in mehr als einer AWS-Region zu verwenden, müssen Sie in jeder dieser Regionen separate 10DLC-Kampagnen registrieren. Dieser Schritt ist notwendig, um die Anforderungen der Anbieter zu erfüllen. Jede Kampagne, die Sie registrieren, wird Ihnen in Rechnung gestellt, auch wenn der Anwendungsfall genau der gleiche ist.

Die Registrierung mehrerer Kampagnen hat den zusätzlichen Vorteil, dass Sie Ihre Durchsatzraten für Nachrichten erhöhen, die Sie an Empfänger senden, die AT&T als Mobilfunkanbieter verwenden. Dies liegt daran, dass AT&T für jede Kampagne 10DLC-Durchsatzraten bietet. Vergleichen Sie dies mit der Art und Weise, wie T-Mobile den 10DLC-Durchsatz bewältigt, der auf einer täglichen Nachrichtenzuweisung für jedes Unternehmen basiert (unabhängig von der Anzahl der Kampagnen).

Bearbeiten oder Löschen einer 10DLC-Kampagne

Sie können die HILFE-Antwort, die STOPP-Antwort und die Beispielnachrichten für eine 10DLC-Kampagne mit der Amazon-Pinpoint-Konsole bearbeiten. Sie können 10DLC-Kampagnen ebenfalls mit der Konsole löschen.

Bearbeiten einer 10DLC-Kampagne

Nachdem Ihre Kampagne genehmigt wurde, können Sie die HILFE-, STOPP- und Beispielnachrichten ändern. Sie können auch zusätzliche Beispielmeldungen hinzufügen. Für Änderungen an diesen Feldern ist keine erneute Genehmigung durch die Kampagnenregistrierung oder Anbieter erforderlich. Sie können keine weiteren Felder ändern, nachdem die 10DLC-Kampagne genehmigt wurde.

Sie können über maximal fünf Beispielmeldungen verfügen. Sie können die Anzahl der Beispielnachrichten, die Sie ursprünglich registriert haben, nicht reduzieren. Wenn Sie Ihre Kampagne beispielsweise mit drei SMS-Beispielnachrichten registriert haben, können Sie die Anzahl der SMS-Beispielnachrichten nicht auf weniger als drei verringern.

> **Note**
>
> Wenn Sie andere Felder als HILFE, STOPP und Beispielnachrichten ändern möchten, müssen Sie zuerst die 10DLC-Kampagne löschen und dann die Kampagne neu erstellen, um die aktualisierten Informationen einzubeziehen.

So bearbeiten Sie eine 10DLC-Kampagne

1. Melden Sie sich bei der AWS Management Console an und öffnen Sie die Amazon-Pinpoint-Konsole unter https://console.aws.amazon.com/pinpoint/.
2. Klicken Sie im Navigationsbereich unter SMS and voice (SMS und Sprache) auf Phone numbers (Telefonnummern).
3. Wählen Sie auf der Registerkarte 10DLC campaigns (10DLC-Kampagnen) die Kampagne aus, die Sie bearbeiten möchten.
4. Wählen Sie im Abschnitt Campaign messages (Kampagnennachrichten) der Kampagnen-Details die Option Edit (Bearbeiten) aus.
5. Aktualisieren Sie eins der folgenden Felder:

   - Hilfe-Nachricht
   - Stopp-Nachricht
   - SMS-Beispielnachricht

   Sie können weder eine zuvor hinzugefügte Beispielnachricht noch den Inhalt einer Beispielnachricht löschen, damit das Feld leer ist. Wenn Sie den Inhalt einer Nachricht löschen, ohne diesen Inhalt zu ersetzen, wird die ursprüngliche Nachricht beim Aktualisieren verwendet.
6. Wählen Sie Update (Aktualisieren). Es wird das Bestätigungsbanner angezeigt, dass die Kampagnennachrichten aktualisiert wurden.

Löschen einer 10DLC-Kampagne

Sie können eine 10DLC-Kampagne mit der Amazon-Pinpoint-Konsole löschen. Bevor Sie eine 10DLC-Kampagne löschen, müssen Sie zunächst alle mit dieser Kampagne verknüpften Telefonnummern entfernen.

> **Important**
>
> Wenn Sie eine 10DLC-Nummer aus einer Kampagne entfernen, haben Sie keinen Zugriff mehr auf diese Nummer. Darüber hinaus können gelöschte 10DLC-Kampagnen nicht wiederhergestellt werden.

So löschen Sie eine 10DLC-Kampagne

1. Melden Sie sich bei der AWS Management Console an und öffnen Sie die Amazon-Pinpoint-Konsole unter https://console.aws.amazon.com/pinpoint/.
2. Klicken Sie im Navigationsbereich unter SMS and voice (SMS und Sprache) auf Phone numbers (Telefonnummern).
3. Wählen Sie auf der Registerkarte 10DLC campaigns (10DLC-Kampagnen) die Kampagne aus, die Sie bearbeiten möchten.
4. Notieren Sie sich die mit der Kampagne verknüpften Telefonnummern im Abschnitt Phone numbers (Telefonnummern).
5. Wählen Sie auf der Registerkarte Phone numbers (Telefonnummern) die 10DLC-Nummer aus, die Sie entfernen möchten, und wählen Sie dann Remove phone number (Telefonnummer entfernen) aus.

   > **Note**
   >
   > Dieser Schritt ist nur erforderlich, wenn mehrere 10DLC-Telefonnummern mit der Kampagne verknüpft sind. Wenn nur eine einzige Telefonnummer mit der 10DLC-Kampagne verknüpft ist, wird diese Nummer auf der Registerkarte 10DLC campaigns (10DLC-Kampagnen) angezeigt. Notieren Sie die Nummer, die auf der Registerkarte angezeigt wird.

6. Geben Sie im Bestätigungsfeld **`delete`** ein und wählen Sie dann Confirm (Bestätigen) aus. Oben auf der SMS- und Sprachseite wird eine Erfolgsmeldung angezeigt.
7. Wiederholen Sie die beiden vorherigen Schritte für jede mit der Kampagne verknüpfte 10DLC-Nummer.
8. Nachdem Sie alle mit der 10DLC-Kampagne verknüpften Nummern entfernt haben, wählen Sie die Registerkarte 10DLC campaigns (10DLC-Kampagnen) aus.
9. Wählen Sie die 10DLC-Kampagne aus, die Sie löschen möchten.
10. Wählen Sie in der oberen rechten Ecke der Seite 10DLC campaign details (Details der 10DLC-Kampagne) die Option Delete (Löschen) aus.
11. Geben Sie im Bestätigungsfeld **`delete`** ein und wählen Sie dann Confirm (Bestätigen) aus. Oben auf der SMS- und Sprachseite wird eine Erfolgsmeldung angezeigt.

Zuordnen einer Langwahlnummer mit einer 10DLC-Kampagne

Wenn Sie über eine vorhandene Langwahlnummer verfügen, können Sie diese Langwahlnummer einer Ihrer aktuellen 10DLC-Kampagnen zuordnen, indem Sie eine Supportanfrage einreichen. Die Langwahlnummer, die Sie der 10DLC-Kampagne zuordnen, kann nur mit dieser Kampagne verwendet werden und kann nicht für andere 10DLC-Kampagnen verwendet werden. Während Ihre Langwahlnummer auf 10DLC migriert wird, können Sie sie trotzdem verwenden. Sie können sie jedoch nicht für 10DLC-Kampagnen verwenden, bis sie genehmigt wurde.

Wenn Sie den Antrag stellen, benötigen Sie:

- Die Langwahlnummern, die mit einer 10DLC-Kampagne verknüpft werden sollen
- Die 10DLC-Kampagnen-ID, die mit der Langwahlnummer verknüpft werden soll

> **Note**
>
> Bevor Sie Langwahlnummern mit einer Kampagne verknüpfen können, müssen Sie diese 10DLC-Kampagne registrieren lassen. Wenn Sie noch keine 10DLC-Kampagne erstellt und registriert haben, siehe Registrieren einer 10DLC-Kampagne.

So weisen Sie 10DLC eine Langwahlnummer zu

1. Melden Sie sich bei der AWS Management Console an und öffnen Sie die Amazon-Pinpoint-Konsole unter https://console.aws.amazon.com/pinpoint/.
2. Wählen Sie unter Einstellungen und dann unter SMS und Sprache die Registerkarte Telefonnummern.
3. Wählen Sie die Langwahlnummer aus, die Sie in einen 10DLC konvertieren möchten.
4. Um das Supportcenter zu öffnen, wählen Sie 10DLC-Kampagne zuweisen aus.
5. Wählen Sie für den Falltyp Erhöhen von Service Limits aus.
6. Wählen Sie für den Limittyp Pinpoint.
7. Wählen Sie im Abschnitt Requests (Anfragen) die Region und dann für das Limit die Option 10 DLC - Associate existing US long code to 10DLC campaign (10 DLC – Bestehende US-Langwahlnummer mit 10DLC-Kampagne verknüpfen) aus.
8. Stellen Sie unter Fallbeschreibung für eine Anwendungsfallbeschreibung sicher, dass Sie die 10DLC-Kampagnen-ID sowie die Langwahlnummern eingeben, die Sie mit dieser Kampagne

in Verbindung bringen möchten. Sie können mehrere Langwahlnummern in die Anforderung aufnehmen, aber Sie sollten nur eine Kampagnen-ID angeben.

9. Wählen Sie unter Contact options (Kontaktoptionen) für Preferred contact language (Bevorzugte Kontaktsprache) die Sprache, die Sie bei der Kommunikation mit dem AWS Support-Team bevorzugen.
10. Wählen Sie für Contact method (Kontaktmethode) die bevorzugte Methode zur Kommunikation mit dem AWS Support-Team.
11. Wählen Sie Submit (Absenden) aus.

10DLC Kontoübergreifender Zugriff

Jede 10DLC-Telefonnummer ist mit einem einzigen Konto in einer einzigen AWS-Region verknüpft. Wenn Sie zum Senden von Nachrichten dieselbe 10DLC-Telefonnummer in mehr als einem Konto oder einer Region verwenden möchten, haben Sie zwei Möglichkeiten:

1. Sie können dasselbe Unternehmen und dieselbe Kampagne in jedem Ihrer AWS-Konten registrieren. Diese Registrierungen werden separat verwaltet und abgerechnet. Wenn Sie dasselbe Unternehmen in mehreren AWS-Konten registrieren, gilt die Anzahl der Nachrichten, die pro Tag an T-Mobile-Kunden gesendet werden können, für alle diese Konten zusammen.
2. Sie können den 10DLC-Registrierungsprozess in einem AWS-Konto abschließen und AWS Identity and Access Management (IAM) verwenden, um anderen Konten die Berechtigung zum Senden über Ihre 10DLC-Nummer zu erteilen.

> **Note**
>
> Diese Option ermöglicht einen echten kontenübergreifenden Zugriff auf Ihre 10DLC-Telefonnummern. Beachten Sie jedoch, dass Nachrichten, die von Ihren sekundären Konten gesendet werden, so behandelt werden, als ob sie von Ihrem primären Konto gesendet würden. Kontingente und Abrechnung werden auf das primäre Konto und nicht auf ein sekundäres Konto angerechnet.

Einrichten von kontenübergreifendem Zugriff mithilfe von IAM-Richtlinien

Sie können IAM-Rollen verwenden, um andere Konten mit Ihrem Hauptkonto zu verknüpfen. Anschließend können Sie Zugriffsberechtigungen von Ihrem primären Konto an Ihre sekundären Konten delegieren, indem Sie ihnen Zugriff auf die 10DLC-Nummern im primären Konto gewähren.

So gewähren Sie Zugriff auf eine 10DLC-Nummer in Ihrem primären Konto

1. Führen Sie den 10DLC-Registrierungsprozess im primären Konto aus, wenn Sie dies noch nicht getan haben. Dieser Prozess umfasst drei Schritte:
   - Registrieren Ihres Unternehmens. Weitere Informationen finden Sie unter „Registrieren Ihres Unternehmens oder Ihrer Marke zur Verwendung mit 10DLC“.
   - Registrieren Sie Ihre 10DLC-Kampagne (Anwendungsfall). Weitere Informationen finden Sie unter Registrieren einer 10DLC-Kampagne.
   - Ordnen Sie Ihrer 10DLC-Kampagne eine Telefonnummer zu. Weitere Informationen finden Sie unter Zuordnen einer Langwahlnummer mit einer 10DLC-Kampagne.
2. Erstellen Sie eine IAM-Rolle in Ihrem primären Konto, mit der ein anderes Konto die `Publish`-API-Operation für Ihre 10DLC-Telefonnummer aufrufen kann. Weitere Informationen zum Erstellen von Rollen finden Sie unter Erstellen von IAM-Rollen im IAM Benutzerhandbuch.
3. Delegieren und testen Sie die Zugriffsberechtigung Ihres primären Kontos mithilfe von IAM-Rollen mit anderen Konten, die Ihre 10DLC-Nummern verwenden müssen. Sie können zum Beispiel die Zugriffsberechtigung von Ihrem Produktionskonto zu Ihrem Entwicklungskonto delegieren. Weitere Informationen zum Delegieren und Testen von Berechtigungen finden Sie unter Delegieren des Zugriffs in allen AWS-Konten mithilfe von IAM-Rollen im IAM Benutzerhandbuch.
4. Senden Sie mithilfe der neuen Rolle eine Nachricht mit einer 10DLC-Nummer aus dem Hauptkonto. Weitere Informationen zur Verwendung von Rollen finden Sie unter Verwenden von IAM-Rollen im IAM-Benutzerhandbuch.

Abrufen von Informationen über Probleme bei der 10DLC-Registrierung

Es kann vorkommen, dass Sie eine Fehlermeldung erhalten, wenn Sie versuchen, Ihr Unternehmen oder Ihre 10DLC-Kampagne zu registrieren.

Probleme bei der Unternehmensregistrierung

Wenn Sie Ihr Unternehmen registriert haben, wird einer von zwei Status für die Registrierung angezeigt: Verified (Verifiziert) oder Unverified (Nicht verifiziert). Wenn der Status für die Unternehmensregistrierung Verified (Verifiziert) lautet, war die Unternehmensregistrierung erfolgreich. Sie können mit dem Erstellen von 10DLC-Kampagnen beginnen.

Wenn der Status für Ihre Unternehmensregistrierung Unverified (Nicht verifiziert) lautet, gab es Probleme mit den bereitgestellten Informationen. Die Amazon-Pinpoint-Konsole stellt Informationen dazu bereit, warum Ihre Unternehmensregistrierung diesen Status erhalten hat.

So zeigen Sie Registrierungsprobleme für Ihre 10DLC-Unternehmensregistrierung an

1. Melden Sie sich bei der AWS Management Console an und öffnen Sie die Amazon-Pinpoint-Konsole unter https://console.aws.amazon.com/pinpoint/.
2. Wählen Sie im Navigationsbereich unter SMS die Option Telefonnummern.
3. Wählen Sie in der Liste der Kampagnen auf der Registerkarte 10DLC campaigns (10DLC-Kampagnen) den Namen des Unternehmens aus, zu dem Sie weitere Informationen anzeigen möchten.
4. Die Seite mit den Unternehmensdetails enthält Informationen zu den Problemen, die bei der Registrierung identifiziert wurden. Wenn ein Feld im Abschnitt Company info (Unternehmensinformationen) ein Warnsymbol enthält, steht das Registrierungsproblem mit den Informationen in diesem Feld im Zusammenhang.

   Überprüfen Sie die bereitgestellten Informationen und bestätigen Sie, dass alle Felder die richtigen Informationen enthalten. Sie können Ihre Unternehmensregistrierung in der Amazon-Pinpoint-Konsole bearbeiten. Weitere Informationen zum Ändern der Details zu Ihrer Unternehmensregistrierung finden Sie unter Bearbeiten oder Löschen eines registrierten Unternehmens.

Probleme bei der Kampagnenregistrierung

Wenn Sie Ihre 10DLC-Kampagne registrieren, kann es vorkommen, dass eine Fehlermeldung angezeigt wird.

Wenn Sie das Problem bei der Registrierung nicht identifizieren können, erstellen Sie einen Fall im AWS Support Center, um weitere Informationen anzufordern. Gehen Sie wie folgt vor, um einen AWS-Support-Fall zu erstellen. Das AWS-Support-Team wird Ihnen dann Informationen dazu bereitstellen, warum die Registrierung Ihrer 10DLC-Kampagne abgelehnt wurde.

So reichen Sie eine Anfrage nach Informationen zu einer abgelehnten 10DLC-Kampagne ein

1. Melden Sie sich an der AWS Management Console unter https://console.aws.amazon.com/ an.
2. Klicken Sie im Menü Support (Support) auf Support Center (Support-Center).
3. Wählen Sie auf der Registerkarte Offene Support-Fälle die Option Fall erstellen aus.

4. Wählen Sie den Link Erhöhung des Servicelimits? und gehen Sie dann wie folgt vor:
   - Wählen Sie für Grenzwerttyp die Option Pinpoint-SMS aus.
5. Führen Sie unter Requests (Anforderungen) die folgenden Abschnitte aus:
   - Wählen Sie unter Region die AWS-Region aus, in der Sie versucht haben, die Kampagne zu registrieren.

     > **Note**
     >
     > Die Region ist im Abschnitt Anforderungen erforderlich. Auch wenn Sie diese Informationen im Abschnitt Falldetails angegeben haben, müssen Sie sie auch hier angeben.

   - Wählen Sie für Resource Type (Ressourcentyp) 10DLC Registration (10DLC-Registrierung) aus.
   - Wählen Sie für Grenzwert die Option Ablehnung der Registrierung des Unternehmens oder der 10DLC-Kampagne.
6. Wählen Sie unter Neuer Grenzwert die Grenzwerterhöhung für den Grenzwerttyp aus. Dieser Wert ist in der Regel **1**.
7. Geben Sie unter Fallbeschreibung die ID der abgelehnten 10DLC-Kampagne ein.
8. (Optional) Wenn Sie weitere Anforderungen einreichen möchten, wählen Sie Weitere Anforderung hinzufügen. Wenn Sie mehrere Anfragen durchführen, geben Sie für jede davon die erforderlichen Informationen an. Die erforderlichen Informationen finden Sie in den anderen Abschnitten von Anfordern von Support für SMS-Messaging mit Amazon SNS.
9. Wählen Sie unter Kontaktoptionen für Bevorzugte Kontaktsprache die Sprache aus, in der Sie Mitteilungen zu diesem Fall erhalten möchten.
10. Wenn Sie fertig sind, klicken Sie auf Submit (Absenden).

Gebührenfreie Nummern

Eine gebührenfreie Nummer (TFN) ist eine 10-stellige Zahl, die mit einer der folgenden Vorwahlnummern beginnt: 800, 888, 877, 866, 855, 844 oder 833. Sie können gebührenfreie Nummern (TFN) verwenden, um nur Transaktionsnachrichten zu senden.

Important

US-Mobilfunkanbieter haben kürzlich ihre Vorschriften geändert und verlangen, dass alle gebührenfreien Nummern (TFNs) vor dem 30. September 2022 bei einer Aufsichtsbehörde registriert werden müssen. Überprüfen Sie den Status Ihrer TFN unter the section called "Registrierungsstatus für gebührenfreie Nummern". Weitere Informationen zur Registrierung Ihres Unternehmens finden Sie unter the section called "Registrieren Ihrer gebührenfreien Nummer".
Es kann bis zu 15 Werktage dauern, bis die angeforderte Registrierung bearbeitet ist.
Update 3. März 2023: Ab dem 1. April 2023 wenden Mobilfunkanbieter die folgenden branchenweiten Schwellenwerte für Nachrichten an, die über eine nicht registrierte gebührenfreie Nummer gesendet werden:

- Tageslimit: 500 Nachrichten pro Tag, wird um 12:00 Uhr Pazifische Zeit zurückgesetzt
- Wöchentliches Limit: 1 000 Nachrichten pro Woche, wird am Sonntag um 12:00 Uhr Pazifische Zeit zurückgesetzt
- Monatliches Limit: 2 000 Nachrichten, wird am Ende des Kalendermonats um 12:00 Uhr Pazifische Zeit zurückgesetzt

Update 19. September 2022: Ab dem 1. Oktober 2022 wenden Mobilfunkanbieter die folgenden branchenweiten Schwellenwerte für Nachrichten an, die über eine nicht registrierte gebührenfreie Nummer gesendet werden:

- Tageslimit: 2 000 Nachrichten
- Wochenlimit: 12 000 Nachrichten
- Monatslimit: 25 000 Nachrichten

Wir empfehlen Ihnen dringend, Ihre Registrierung schnellstmöglich abzuschließen. Über nicht registrierte TFNs gesendete Nachrichten werden nach bestem Bemühen versendet. Die Nachrichten werden im Laufe der Zeit zunehmend gefiltert und blockiert, da die Mobilfunkanbieter nicht registrierten Datenverkehr weiterhin einschränken.

Themen

- Eine gebührenfreie Nummer kaufen
- Anforderungen und Ablauf der Registrierung von gebührenfreien Nummern
- Registrierungsstatus für gebührenfreie Nummern
- Bearbeiten, Verwerfen und Löschen Ihrer Registrierung
- Probleme bei der Registrierung
- Gebührenfreie Nummern – Häufig gestellte Fragen
- Vor- und Nachteile von gebührenfreien Nummern

Richtlinien für die Verwendung von gebührenfreien Nummern

TFNs werden in der Regel nur in den USA für Transaktionsnachrichten wie die Bestätigung der Registrierung oder für das Versenden von einmaligen Passwörtern verwendet. Sie können sowohl für Sprachnachrichten als auch für SMS verwendet werden. Der durchschnittliche Durchsatz beträgt drei Nachrichtenteile pro Sekunde (MPS). Dieser Durchsatz wird jedoch durch die Zeichenkodierung beeinflusst. Weitere Informationen zu den Auswirkungen der Zeichenkodierung auf Nachrichtenteile finden Sie unter SMS-Zeichenbeschränkungen in Amazon SNS. Weitere Informationen zur Registrierung einer TFN finden Sie unter Anforderungen und Ablauf der Registrierung von gebührenfreien Nummern.

Jedes Kundenkonto kann bis zu fünf TFNs enthalten. Wenn Sie mehr als 15, aber weniger als 100 Textnachrichten pro Sekunde senden, empfehlen wir Ihnen, eine oder mehrere 10DLC-Ursprungs-IDs zu registrieren. Wenn Ihre Anwendungsfälle das Senden von mehr als 100 Textnachrichten pro Sekunde erfordern, empfehlen wir Ihnen, eine oder mehrere Kurzwahlnummern zu kaufen und zu registrieren.

Wenn Sie eine TFN als Ursprungsnummer verwenden, folgen Sie diesen Richtlinien:

- Verwenden Sie keine verkürzten URLs, die von URL-Kürzern von Drittanbietern erstellt wurden, da diese Nachrichten eher als Spam gefiltert werden.

  Wenn Sie eine verkürzte URL verwenden müssen, ziehen Sie in Betracht, eine 10DLC-Nummer oder eine Kurzwahlnummer zu verwenden. Bei Verwendung von Kurzwahlnummern und 10DLCs müssen Sie Ihre Nachrichtenvorlage registrieren, wo Sie eine verkürzte URL angeben können.
- Beachten Sie, dass Antworten auf Schlüsselwort-Opt-Out (STOP) und Opt-In (UNSTOP) auf Carrier-Ebene festgelegt werden. Sie können diese Schlüsselwörter oder andere Schlüsselwörter nicht ändern. Sie können auch keine Nachrichten ändern, die gesendet werden, wenn Benutzer mit STOP und UNSTOP antworten.

- Senden Sie nicht denselben oder ähnlichen Nachrichteninhalt mit mehreren TFNs. Carrier nennen diese Praxis Snowshoeing oder Zahlen-Pooling und wählen Sie die Nachrichten zum Filtern.
- Alle Nachrichten, die sich auf die folgenden Branchen beziehen, können als eingeschränkt angesehen werden und unterliegen starken Filtern oder vollständigen Blockierungen. Dies kann Einmalpasswörter (OTP) und die Multi-Faktor-Authentifizierung (MFA) für Services umfassen, die sich auf eingeschränkte Kategorien beziehen.

  Wenn eine Registrierung mit der Begründung verweigert wurde, dass es sich um einen nicht konformen Anwendungsfall handelt, dies Ihrer Meinung nach jedoch nicht zutrifft, können Sie eine Anfrage über den Support einreichen. Weitere detaillierte Informationen hierzu finden Sie unter Probleme bei der Registrierung.

  In der folgenden Tabelle sind die Arten von eingeschränktem Inhalt angegeben:

Kategorie	Beispiele
Glücksspiel	• Apps/Websites • Casinos • Gewinnspiele
Finanzdienstleistungen mit hoher Risikostufe	• Autodarlehen • Kryptowährung • Inkasso • Zahltagdarlehen • Kurzfristige Zinsdarlehen • Hypothekendarlehen • Studentendarlehen • Aktienwarnungen
Schuldenerlass	• Schuldenkonsolidierung • Schuldenabbau • Bonitätsverbesserung
G-et-rich-quick Schemata	• W-ork-from-home Programme • Risiko-Investitionschancen

Kategorie	Beispiele
	• Pyramiden- oder mehrstufige Vermarktungssysteme
Verbotene/kontrollierte Stoffe	• Cannabis/CBD
Phishing	• Versuche, Benutzer dazu zu bringen, personenbezogene Informationen oder Website-Login-Informationen offenzulegen
S.H.A.F.T.	• Sex • Hass • Alkohol • Schusswaffen • Tabak/E-Zigaretten

Eine gebührenfreie Nummer kaufen

Um TFNs zu kaufen, verwenden Sie die Amazon Pinpoint-Konsole unter https://console.aws.amazon.com/sms-voice/. Weitere Informationen finden Sie unter Anforderungen und Ablauf der Registrierung von gebührenfreien Nummern.

Derzeit unterstützt Amazon Pinpoint SMS gebührenfreie Nummern sowohl für Sprach- als auch für SMS-Nachrichten. Amazon SNS unterstützt nur SMS-Messaging.

Anforderungen und Ablauf der Registrierung von gebührenfreien Nummern

> **Important**
>
> Eine TFN kann widerrufen werden, wenn sie für einen anderen Zweck als die angegebene Anwendungsfall genutzt wird.

Unzulässige Anwendungsfälle für gebührenfreie Nummern

Die Nachrichtenzustellung mit Amazon SNS ist in Fällen eingeschränkt, in denen die Nachrichten blockiert sind (z. B. Anwendungsfälle im Zusammenhang mit einer kontrollierten Substanz oder

bei Phishing) oder wenn ein hohes Maß an Filterung erwartet wird (z. B. bei Finanznachrichten mit hohem Risiko). Möglicherweise können Sie keine TFNs registrieren, die mit Anwendungsfällen für eingeschränkte Inhalte verbunden sind, die in Richtlinien für die Verwendung von gebührenfreien Nummern definiert werden.

Registrieren Ihrer gebührenfreien Nummer

Nach dem Kauf einer TFN müssen Sie die Nummer registrieren. Anweisungen dazu finden Sie unter Registrierungsprozess für gebührenfreie Nummern im Amazon Pinpoint-SMS-Benutzerhandbuch.

Self-Service-Registrierung für gebührenfreie Nummern in Amazon Pinpoint-SMS-Regionen

Wenn Sie die TFN in den Amazon Pinpoint-SMS-Regionen angefordert haben, führen Sie den Vorgang der Unternehmensregistrierung direkt in der Amazon Pinpoint-SMS-Konsole durch. Befolgen Sie dazu die Anweisungen im Formular zur Registrierung gebührenfreier Nummern in den USA im Amazon Pinpoint-SMS-Benutzerhandbuch.

Stellen Sie bei der Registrierung Ihrer TFN sicher, dass die Informationen vollständig und korrekt sind, da Ihre Registrierung sonst abgelehnt werden kann. Die eingegebenen Informationen sollten exakt mit dem Hauptsitz Ihres Unternehmens übereinstimmen.

Manueller formularbasierter Registrierungsprozess für gebührenfreie Nummern in anderen Regionen als Amazon Pinpoint-SMS-Regionen

1. Laden Sie diese Datei US_TFN_Registration.zip herunter und verwenden Sie das Beispiel-Registrierungsformular (AWS USA gebührenfreie Registrierungsformular-Business – Final.docx), um die erforderlichen Informationen in der CSV-Datei für die TFN-Registrierung (bulkUStfn – Final.csv) auszufüllen.

   Jede Registrierungsanfrage oder Anwendungsfall kann nur bis zu fünf TFNs enthalten. Wenn Sie der Meinung sind, dass Sie für eine Ausnahme von dieser Regel berechtigt sind, geben Sie eine detaillierte Erklärung zur Prüfung an. Führen Sie alle mit der Registrierung oder dem Anwendungsfall verknüpften Telefonnummern auf.
2. Erstellen Sie einen Fall beim AWS -Support. Fügen Sie dem Fall Ihre ausgefüllte CSV-Datei an und senden Sie die TFN-Anmeldeanfrage.
3. Wählen Sie Fall erstellen und anschließend Erhöhung des Servicegrenzwerts?
4. Wählen Sie für den Grenzwerttyp die Option SNS-SMS.
5. Wählen Sie für Resource Type (Ressourcentyp) 10DLC Registration or TFN Registration (10DLC- oder TFN-Registrierung) aus.

6. Fügen Sie das Dokument US_TFN_registration bei und senden Sie die Anfrage ab.

Wichtiger zu beachtender Punkt

1. Die Bearbeitung von Registrierungen kann nach Einreichen aller erforderlichen Informationen bis zu zwei Wochen dauern. Wenn Informationen fehlen oder unvollständig sind, verzögert sich der Registrierungsvorgang. Wenn Ihre Registrierung abgelehnt wird, helfen wir Ihnen, den Grund dafür zu finden, und schlagen Methoden vor, um Ihre Kampagne zu verbessern, damit sie registriert werden kann.
2. TFNs eignen sich gut für transaktionale Anwendungsfälle wie die Multi-Faktor-Authentifizierung (MFA), bei denen ein begrenzter Durchsatz erforderlich ist. Jede TFN kann bis zu drei Textnachrichtenteile pro Sekunde senden, und jedes Kundenkonto kann über bis zu fünf TFNs verfügen. Wenn Sie mehr als 15, aber weniger als 100 Textnachrichten pro Sekunde senden, empfehlen wir Ihnen, mindestens eine 10DLC-Urprungs-ID zu registrieren. Wenn Ihre Anwendungsfälle das Senden von mehr als 100 Textnachrichten pro Sekunde erfordern, empfehlen wir Ihnen, eine oder mehrere Kurzwahlnummern zu kaufen und zu registrieren. Weitere Details finden Sie unter Richtlinien für die Verwendung von gebührenfreien Nummern.

Registrierungsstatus für gebührenfreie Nummern

Informationen zum Überprüfen Ihres Registrierungsstatus finden Sie unter Überprüfen Ihres Registrierungsstatus im Amazon Pinpoint-SMS-Benutzerhandbuch.

Bearbeiten, Verwerfen und Löschen Ihrer Registrierung

Verwenden Sie das Amazon Pinpoint-SMS-Benutzerhandbuch, um die folgenden Aufgaben auszuführen:

- Bearbeiten Ihrer Registrierung
- Verwerfen Ihrer Registrierung
- Löschen Ihrer Registrierung
- Anzeigen Ihrer Registrierungsressourcen

Probleme bei der Registrierung

Wenn die Registrierung Ihrer gebührenfreien Nummer nicht akzeptiert wird, wird eine Meldung mit dem Grund für die Ablehnung angezeigt.

So reichen Sie eine Anfrage nach Informationen zu einer abgelehnten gebührenfreien Nummer ein

1. Melden Sie sich bei der AWS Management Console unter https://console.aws.amazon.com/ an.
2. Klicken Sie im Menü Support (Support) auf Support Center (Support-Center).
3. Wählen Sie auf der Registerkarte Offene Support-Fälle die Option Fall erstellen aus.
4. Wählen Sie den Link Erhöhung des Servicelimits? und gehen Sie dann wie folgt vor:
   - Wählen Sie für Limit Type (Limittyp) die Option Pinpoint SMS aus.
5. Führen Sie unter Requests (Anforderungen) die folgenden Abschnitte aus:
   - Wählen Sie unter Region die Region aus, in der Sie versucht haben, die Kampagne zu registrieren.

     > **Note**
     >
     > Die Region ist im Abschnitt Anforderungen erforderlich. Auch wenn Sie diese Informationen im Abschnitt Falldetails angegeben haben, müssen Sie sie auch hier angeben.

   - Wählen Sie für Resource Type (Ressourcentyp) 10DLC Registration or TFN Registration (10DLC- oder TFN-Registrierung) aus.
   - Wählen Sie für Grenzwert die Option Ablehnung der Registrierung des Unternehmens oder der Kampagne aus.
6. Wählen Sie unter Neuer Grenzwert die Grenzwerterhöhung für den Grenzwerttyp aus. Dieser Wert ist in der Regel **1**.
7. (Optional) Wenn Sie weitere Anforderungen einreichen möchten, wählen Sie Weitere Anforderung hinzufügen. Die erforderlichen Informationen finden Sie in den anderen Abschnitten von Anfordern von Support für SMS-Messaging mit Amazon SNS.
8. Geben Sie unter Fallbeschreibung die abgelehnte gebührenfreie Nummer ein.
9. Wählen Sie unter Kontaktoptionen für Bevorzugte Kontaktsprache die Sprache aus, in der Sie Mitteilungen zu diesem Fall erhalten möchten.
10. Wenn Sie fertig sind, klicken Sie auf Submit (Absenden).

Gebührenfreie Nummern – Häufig gestellte Fragen

Häufig gestellte Fragen zur Registrierung von TFN.

Besitze ich derzeit eine gebührenfreie Nummer?

So überprüfen Sie, ob Sie eine gebührenfreie Nummer besitzen

- Öffnen Sie die Amazon Pinpoint-SMS-Konsole unter https://console.aws.amazon.com/sms-voice/.
- Wählen Sie im Navigationsbereich unter SMS die Option Telefonnummern.
- Die Art der TFN wird als gebührenfrei gelistet.

Muss ich meine gebührenfreie Nummer registrieren?

Ja. Um eine TFN, die Sie derzeit besitzen, weiterhin nutzen zu können, müssen Sie sie vor dem 30. September 2022 registrieren. Wenn Sie nach dem 30. September 2022 eine neue TFN kaufen, müssen Sie sie registrieren, bevor Sie Nachrichten senden können.

Wie kaufe ich eine gebührenfreie Nummer?

Folgen Sie den Anweisungen unter Anfordern einer Telefonnummer mit der Amazon Pinpoint-SMS-Konsole, um eine TFN zu kaufen.

Wie registriere ich meine gebührenfreie Nummer?

Folgen Sie den Anweisungen unter the section called “Registrieren Ihrer gebührenfreien Nummer”, um eine TFN zu registrieren.

Was ist der Registrierungsstatus meiner gebührenfreien Nummer und was bedeutet er?

Folgen Sie den Anweisungen unter the section called “Registrierungsstatus für gebührenfreie Nummern”, um Ihre Registrierung und Ihren Status zu überprüfen.

Welche Informationen muss ich angeben?

Sie müssen die Adresse Ihres Unternehmens, einen Geschäftskontakt und einen Anwendungsfall für die TFN angeben. Die erforderlichen Informationen finden Sie unter the section called “Registrieren Ihrer gebührenfreien Nummer”.

Was passiert, wenn meine Registrierung abgelehnt wird?

Wenn Ihre Registrierung abgelehnt wird, wird der Status in Requires Updates (Updates erforderlich) geändert. Informationen zu Aktualisierungen finden Sie unter the section called “Bearbeiten, Verwerfen und Löschen Ihrer Registrierung”.

Welche Berechtigungen benötige ich?

Der IAM-Benutzer/die IAM-Rolle, den/die Sie zum Aufrufen der Amazon Pinpoint-SMS-Konsole verwenden, muss über die Berechtigung *„sms-voice:*“* verfügen. Andernfalls erhalten Sie die Fehlermeldung „Zugriff verweigert“.

Vor- und Nachteile von gebührenfreien Nummern

Vorteile

Gebührenfreie Ursprungsnummern haben höhere MPS-Werte als Langwahlnummern sowie eine gute Zustellbarkeit.

Nachteile

Es gibt keine Kontrolle über Opt-Outs und Opt-Ins, da diese auf Carrier-Ebene verwaltet werden.

Nehmen Sie keine verkürzten URLs in Ihre Nachricht auf und senden Sie keine Werbebotschaften mit der Nummer. Verwenden Sie stattdessen eine 10DLC-Nummer oder eine Kurzwahlnummer. Bei Verwendung einer Kurzwahl- oder 10DLC-Nummer müssen Sie Ihre Nachrichtenvorlagen registrieren. Diese Vorlagen können sowohl verkürzte URLs als auch Werbebotschaften enthalten. Weitere Informationen zu Kurzwahlnummern finden Sie unter Kurzwahlnummern. Weitere Informationen zu 10DLC-Nummern finden Sie unter 10DLC.

Kurzwahlnummern

Kurzwahlnummern sind numerische Sequenzen, die kürzer sind als eine reguläre Telefonnummer. Beispiel: In den Vereinigten Staaten und Kanada umfassen Standardtelefonnummern (Langwahlnummern) 11 Ziffern, während Kurzwahlnummern nur fünf- oder sechsstellig sind. Amazon SNS unterstützt dedizierte Kurzwahlnummern.

Dedizierte Kurzwahlnummern

Wenn Sie SMS-Nachrichten in großen Mengen an Empfänger in den USA oder Kanada senden, können Sie eine dedizierte Kurzwahlnummer kaufen. Im Gegensatz zu den Kurzwahlnummern im freigegebenen Pool sind dedizierte Kurzwahlnummern für Ihre ausschließliche Nutzung reserviert.

Vorteile

Die Verwendung einer einprägsamen Kurzwahlnummer kann dazu beitragen, Vertrauen aufzubauen. Wenn Sie sensible Daten übertragen müssen, wie z. B. einmalige Passwörter, sollten Sie dazu eine Kurzwahlnummer verwenden, damit Ihre Kunden schnell feststellen können, ob eine Nachricht tatsächlich von Ihnen stammt.

Wenn Sie eine Kampagne zur Kundenakquise starten, können Sie potenzielle Kunden einladen, ein Schlüsselwort an Ihre Kurzwahlnummer zu senden (z. B. „Text FUSSBALL in 10987 für Fußballneuigkeiten und -informationen“). Kurzwahlnummern sind leichter zu merken als Langwahlnummern und können von den Kunden einfacher auf ihren Geräten eingegeben werden. Sie können die Effektivität Ihrer Kampagnen erhöhen, indem Sie den Prozess zum Anmelden bei Ihren Marketing-Programmen für die Kunden so einfach wie möglich gestalten.

Da Mobilfunkbetreiber neue Kurzwahlnummern genehmigen müssen, bevor sie aktiv werden, ist es weniger wahrscheinlich, dass Nachrichten, die von Kurzwahlnummern stammen, als unerwünscht markiert werden.

Wenn Sie Kurzwahlnummern zum Senden von SMS-Nachrichten verwenden, können Sie eine größere Anzahl von Nachrichten pro 24-Stunden-Zeitraum senden als mit anderen Ursprungsidentitäten. Mit anderen Worten: Sie verfügen über eine viel höhere Sendequote. Außerdem können Sie eine sehr viel höhere Anzahl von Nachrichten pro Sekunde senden. Das heißt, Sie verfügen über eine viel höhere Senderate.

Nachteile

Für den Erwerb von Kurzwahlnummern fallen zusätzliche Kosten an und es kann einige Zeit dauern, sie zu implementieren. In den USA fällt z. B. eine einmalige Einrichtungsgebühr von 650.00 USD für jede Kurzwahlnummer sowie eine zusätzliche regelmäßige Gebühr von 995.00 USD pro Monat für jede Kurzwahlnummer an. Es kann 8 bis 12 Wochen dauern, bis Kurzwahlnummern in allen Betreibernetzen aktiviert werden. Zum Ermitteln von Preis und Bereitstellungszeit für ein anderes Land oder eine andere Region führen Sie das unter Anfordern dedizierter Kurzwahlnummern für SMS-Messaging mit Amazon SNS beschriebene Verfahren aus.

Person-zu-Person (P2P)-Langwahlnummern

> **Important**
>
> Ab dem 31. August 2023 ist es eine spezielle Nummer wie eine 10DLC-Nummer oder eine gebührenfreie Nummer erforderlich, um in den Vereinigten Staaten und ihren Territorien

(Puerto Rico, Guam, Amerikanisch-Samoa-Inseln und den US Jungferninseln) SMS-Nachrichten zu versenden. Ihre Langwahlnummernanfrage wird abgelehnt, wenn Sie die Vereinigten Staaten als Standort für diese Regionen verwenden.

Important

Ab dem 1. Juni 2021 unterstützen US-Telekommunikationsanbieter nicht mehr die Verwendung von P2P-Langwahlnummern für die Application-to-Person-Kommunikation (A2P) zu US-Destinationen. Stattdessen müssen Sie eine andere Art von Ursprungs-ID für diese Nachrichten verwenden. Weitere Informationen finden Sie unter 10DLC.

P2P-Langwahlnummern sind Telefonnummern, die das Nummernformat des Landes oder der Region verwenden, in dem bzw. der sich Ihre Empfänger befinden. P2P-Langwahlnummern werden auch als Langnummern oder virtuelle Mobiltelefonnummern bezeichnet. Beispiel: In den USA und Kanada enthalten P2P-Langwahlnummern 11 Stellen: die Zahl 1 (der Ländercode), eine 3-stellige Vorwahl und eine 7-stellige Telefonnummer.

Weitere Informationen zum Anfordern von P2P-Langwahlnummern finden Sie unter Anfordern von 10DLC-Nummern, gebührenfreien Nummern und P2P-Langwahlnummern für SMS-Messaging mit Amazon SNS.

Vorteile

Dedizierte P2P-Langwahlnummern sind Ihrem Amazon SNS-Konto vorbehalten. Sie werden nicht an andere Benutzer weitergegeben. Wenn Sie dedizierte P2P-Langwahlnummern verwenden, können Sie beim Senden der jeweiligen Nachricht angeben, welche P2P-Langwahlnummer Sie nutzen möchten. Wenn Sie mehrere Nachrichten an denselben Kunden senden, können Sie sicherstellen, dass für jede Nachricht dieselbe Telefonnummer angezeigt wird. Aus diesem Grund können dedizierte P2P-Langwahlnummern hilfreich sein, Ihre Marke oder Identität zu etablieren.

Nachteile

P2P-Langwahlnummern werden für A2P-Kommunikation zu US-Zielen nicht unterstützt.

Wenn Sie mehrere hundert Nachrichten pro Tag von einer dedizierten P2P-Langwahlnummer senden, können Mobilfunkbetreiber Ihre Telefonnummer als Sender unerwünschter E-Mails

identifizieren. Wenn Ihre P2P-Langwahlnummer entsprechend markiert wird, werden Ihre Nachrichten den Empfängern möglicherweise nicht zugestellt.

P2P-Langwahlnummern verfügen außerdem über begrenzten Durchsatz. Die maximalen Senderate variieren je nach Land. Weitere Informationen erhalten Sie vom AWS-Support. Wenn Sie SMS-Nachrichten in großen Mengen senden möchten oder eine Rate von mehr als einer Nachricht pro Sekunde planen, sollten Sie eine dedizierte Kurzwahlnummer kaufen.

Einige Anbieter erlauben es Ihnen nicht, P2P-Langwahlnummern zu verwenden, um A2P-SMS-Nachrichten zu senden, auch in den USA. Eine A2P-SMS ist eine Nachricht, die an das mobile Gerät eines Kunden gesendet wird, wenn dieser Kunde seine Mobiltelefonnummer an eine Anwendung sendet. A2P-Nachrichten stellen unidirektionale Kommunikation dar, wie Marketing-Nachrichten, einmalige Passwörter und Terminerinnerungen. Wenn Sie vorhaben, A2P-Nachrichten zu senden, sollten Sie eine dedizierte Kurzwahlnummer erwerben (wenn sich Ihre Kunden in den USA und Kanada befinden) oder eine Sender-ID verwenden (wenn Ihre Empfänger sich in einem Land oder einer Region befinden, in denen Sender-IDs unterstützt werden).

Eine 10DLC-Nummer wird nur zum Senden von Nachrichten innerhalb der USA verwendet. Für die Verwendung einer 10DLC-Nummer müssen Sie die Marke Ihres Unternehmens und die Kampagne registrieren, mit der Sie die Nummer verknüpfen möchten. Nach der Genehmigung können Sie eine 10DLC-Telefonnummer auf der Seite SMS und Spracheder Amazon-Pinpoint-Konsole unter https://console.aws.amazon.com/pinpoint/ anfordern. Nach der Anforderung dauert es 7-10 Tage, bis die Nummer genehmigt wird. Die Nummer kann nicht mit anderen Kampagnen verwendet werden.

Vergleich der US-Produktnummern

Diese Tabelle zeigt den Supportvergleich für US-Telefonnummern.

Produkt-Funktion	Kurzwahlnummer	Gebührenfreie Nummer	10DLC	
Zahlen-Format	5-6 Ziffern	10-stellige Zahl	10-stellige Zahl	
Support-Kanal	SMS	SMS	SMS	
SMS-Traffic-Typ	Werbeinhalte und Transaktionales	Transaktional	Werbeinhalte und Transaktionales	

Produkt-Funktion	Kurzwahlnummer	Gebührenfreie Nummer	10DLC	
Sicherheitsüberprüfung notwendig	Ja	Nein	Ja	
Geschätzte Bereitstellungszeit	12 Wochen[1]	15 Werktage	1 Woche	
SMS-Durchsatz (Anzahl der SMS-Nachrichten pro Sekunde)[2]	100 Nachrichtenteile pro Sekunde; höherer Durchsatz gegen eine zusätzliche Gebühr verfügbar.	3 Nachrichten pro Sekunde	Variiert je nach Ihrer 10DLC-Registrierung. Unterstützt bis zu 100 Nachrichtenteile pro Sekunde.	
Schlüsselwörter erforderlich	Opt-In, Opt-Out und HILFE (HELP)	STOP, UNSTOP. Diese sind netzwerkgemanagt. Sie können das Opt-Out und Opt-Back in Nachrichten nicht ändern.	Opt-In, Opt-Out und HILFE (HELP)	

[1]Schätzwert für die Bereitstellung enthält keine Genehmigungszeit.

[2] Weitere Informationen zur maximalen Größe für SMS-Nachrichten finden Sie unter Veröffentlichen auf einem Mobiltelefon.

Anfordern von Support für SMS-Messaging mit Amazon SNS

Bestimmte SMS-Optionen mit Amazon SNS sind für Ihr AWS-Konto nicht verfügbar, bis Sie AWS Support kontaktieren. Erstellen Sie einen Fall im AWS Support Center, um eine der folgenden Optionen anzufordern:

- Eine Erhöhung Ihres monatlichen Ausgabenschwellenwerts für SMS

  Standardmäßig liegt der monatliche Ausgabenschwellenwert bei 1,00 USD. Ihr Ausgabenschwellenwert bestimmt die Anzahl der Nachrichten, die Sie mit Amazon SNS senden können. Sie können einen Ausgabenschwellenwert anfordern, der der erwarteten monatlichen Menge an Nachrichten für Ihren SMS-Anwendungsfall entspricht.
- Eine Verschiebung aus der SMS-Sandbox, damit Sie SMS-Nachrichten ohne Einschränkungen versenden können. Weitere Informationen finden Sie unter Verlassen der SMS-Sandbox.
- Eine engagierte Ursprungsnummer
- Eine dedizierte Sender-ID

  Eine Sender-ID ist eine benutzerdefinierte ID, die auf dem Gerät des Empfängers als Sender angezeigt wird. Sie können z. B. Ihre Unternehmensmarke verwenden, um die Nachrichtenquelle leichter erkennbar zu machen. Der Support für Sender-IDs variiert je nach Land oder Region. Weitere Informationen finden Sie unter Unterstützte Länder und Regionen.

Wenn Sie Ihren Fall im AWS Support-Center erstellen, stellen Sie sicher, dass Sie alle erforderlichen Informationen für die Art der Anfrage, die Sie einsenden, angeben. Andernfalls muss AWS Support Sie kontaktieren, um diese Informationen zu erhalten, bevor fortgefahren wird. Indem Sie einen detaillierten Fall übermitteln, tragen Sie dazu bei, dass Ihr Fall ohne Verzögerungen gelöst werden kann. Informationen zu den Details, die für bestimmte Arten von SMS-Anfragen benötigt werden, finden Sie in den folgenden Themen.

Themen

Anfordern dedizierter Kurzwahlnummern für SMS-Messaging mit Amazon SNS

Eine Kurzwahlnummer ist eine Nummer, die Sie für eine große Anzahl von SMS-Nachrichten verwenden können. Kurzwahlnummern werden häufig für A2P-(Anwendung-an-Person)-Messaging, die Zwei-Faktor-Authentifizierung (2FA) und das Marketing eingesetzt. Eine Kurzwahlnummer enthält in der Regel zwischen drei und sieben Ziffern, je nach Land oder Region, zu der sie gehört.

Sie können nur Kurzwahlnummern zum Senden von Nachrichten an Empfänger in demselben Land verwenden, für das die Kurzwahlnummer gilt. Wenn Ihr Anwendungsfall die Verwendung von Kurzwahlnummern in mehr als einem Land erfordert, müssen Sie für jedes Land, in dem sich Ihre Empfänger befinden, eine separate Kurzwahlnummer anfordern.

Informationen zu den Preisen für Kurzwahlnummern finden Sie unter Amazon SNS-Preise.

> **Important**
>
> Wenn SMS-Messaging mit Amazon SNS neu für Sie ist, fordern Sie einen monatlichen Kostenschwellenwert für SMS-Nachrichten an, der den erwarteten Anforderungen Ihres SMS-Anwendungsfalls entspricht. Standardmäßig liegt der monatliche Kostenschwellenwert bei 1,00 USD. Sie können in demselben Supportfall, in dem Sie eine Kurzwahlnummer anfordern, auch eine Erhöhung Ihres Kostenschwellenwerts anfordern. Sie können aber auch einen eigenen Fall verwenden. Weitere Informationen finden Sie unter Anfordern von Erhöhungen Ihres monatlichen SMS-Ausgabenkontingents für Amazon SNS.
> Wenn Sie einen dedizierten Kurzcode zum Senden von Nachrichten anfordern, die PHI (Protected Health Information) enthalten, sollten Sie diesen Zweck in Ihrer Fallbeschreibung angeben, wenn Sie einen Support-Fall öffnen, wie unten beschrieben.

Einen Support-Fall bezüglich einer Amazon-SNS-Kurzwahl öffnen

Öffnen Sie einen Fall mit AWS Support indem Sie die folgenden Schritte ausführen.

> **Note**
>
> Einige der Felder im Antragsformular sind als „optional“ markiert. Allerdings benötigt AWS Support alle Informationen, die in den folgenden Schritten aufgeführt sind, um Ihre Anfrage zu bearbeiten. Wenn Sie nicht alle erforderlichen Informationen bereitstellen, kann es bei der Bearbeitung Ihrer Anfrage zu Verzögerungen kommen.

So fordern Sie eine dedizierte Kurzwahlnummer an

1. Wechseln Sie zum AWS Supportcenter.
2. Melden Sie sich an der AWS Management Console unter https://console.aws.amazon.com/ an.
3. Klicken Sie im Menü Support (Support) auf Support Center (Support-Center).
4. Wählen Sie auf der Registerkarte Offene Support-Fälle die Option Fall erstellen aus.
5. Wählen Sie den Link Erhöhung des Servicelimits? und gehen Sie dann wie folgt vor:

   - Wählen Sie als Grenzwerttyp die Option SNSSMS aus und gehen Sie dann wie folgt vor:
   - (Optional) Geben Sie unter Link zur Website oder App angeben, die SMS senden wird einen Link zu der Website oder Anwendung an, auf/in der Ihre Mitglieder sich für den Empfang Ihrer SMS-Nachrichten registrieren können.
   - (Optional) Wählen Sie für Art der Nachrichten, die gesendet werden sollen die Art der Nachrichten aus, die Sie mit Ihren Langwahlnummern senden möchten:
     - One-time Password (Einmaliges Passwort) – Nachrichten, die für Ihre Kunden Passwörter zur Authentifizierung bei Ihrer Website oder Anwendung bereitstellen.
     - Promotional (Werbung) – Nicht kritische Nachrichten, die Ihr Unternehmen oder Ihren Service bewerben, wie beispielsweise Sonderangebote oder Ankündigungen.
     - Transactional (Transaktionsnachrichten) – Wichtige Informationsmeldung, die Kundentransaktionen unterstützen, wie beispielsweise Bestellbestätigungen oder Kontowarnungen. Transaktionsnachrichten dürfen keine Werbeaktionen oder Marketinginhalte enthalten.
   - (Optional) Wählen Sie für Aus welcher AWS-Region werden Sie Nachrichten senden die Region, aus der Sie Nachrichten senden werden.
   - (Optional) Geben Sie unter In welche Länder möchten Sie Nachrichten senden die Länder an, an die Sie SMS-Nachrichten senden möchten.
   - (Optional) Geben Sie unter Wie Ihre Kunden dem Erhalt von Nachrichten zustimmen, die von Ihnen gesendet werden eine Beschreibung dazu ein, wie Ihre Kunden dem Erhalt von Nachrichten von Ihnen zustimmen..
   - (Optional) Geben Sie im Feld Bitte geben Sie die Nachrichtenvorlage an, die Sie verwenden möchten, um Nachrichten an Ihre Kunden zu senden die Vorlage ein, die Sie verwenden werden.
6. Führen Sie unter Requests (Anforderungen) die folgenden Abschnitte aus:

   - Wählen Sie als Region die AWS-Region für Ihre Kurzwahlanforderung aus.

> **Note**
>
> Die Region ist im Abschnitt Anforderungen erforderlich. Auch wenn Sie diese Informationen im Abschnitt Falldetails angegeben haben, müssen Sie sie auch hier angeben.

- Wählen Sie für Resource Type (Ressourcentyp) die Option Dedicated SMS Short Codes (Dedizierte SMS-Kurzwahlnummern) aus.
- Wählen Sie für Limit die Option aus, die am ehesten Ihrem Anwendungsfall entspricht.

7. Geben Sie für Neuer Grenzwert die Anzahl von Sender-IDs ein, die Sie anfordern. Dieser Wert ist in der Regel **1**.
8. (Optional) Wenn Sie weitere Anforderungen einreichen möchten, wählen Sie Weitere Anforderung hinzufügen. Die erforderlichen Informationen finden Sie in den anderen Abschnitten von Anfordern von Support für SMS-Messaging mit Amazon SNS.
9. Fassen Sie Ihren Anwendungsfall unter Fallbeschreibung zusammen und legen Sie dar, wie sich Ihre Empfänger für Nachrichten mit Ihrer Kurzwahlnummer anmelden werden. Geben Sie außerdem die folgenden Informationen an:

   - Unternehmensinformationen:
     - Unternehmensname
     - Unternehmensanschrift
     - Name und Telefonnummer für den primären Ansprechpartner für Ihre Anfrage
     - E-Mail-Adresse und gebührenfreie Telefonnummer für den Support in Ihrem Unternehmen
     - Steuer-ID des Unternehmens
     - Name Ihres Produkts oder Service

   - Benutzeranmeldung:
     - Unternehmenswebsite bzw. die Website, auf der sich Ihre Kunden für den Empfang von Nachrichten von Ihrer Kurzwahlnummer registrieren werden
     - Die Art, wie sich die Benutzer für den Empfang von Nachrichten von Ihrer Kurzwahlnummer anmelden werden. Geben Sie eine oder mehrere der folgenden Optionen an:
       - **`Text messages`**

        - **`Website`**
        - **`Mobile app`**
        - **`Other`** Bei „Sonstiges” bitte nähere Informationen bereitstellen.
    - Der Text für die Option, sich auf Ihrer Website, in Ihrer App o. ä. für den Empfang von Nachrichten zu registrieren.
    - Die Reihenfolge der Nachrichten, die Sie für Double-Opt-in-Verfahren verwenden möchten. Machen Sie alle nachstehenden Angaben:
        1. Die SMS-Nachricht, die gesendet werden soll, wenn sich ein Benutzer anmeldet. In dieser Nachricht wird der Benutzer gebeten, seine Zustimmung zum Empfang weiterer Nachrichten zu geben. Beispiel: ExampleCorp: JA antworten, um Warnungen zu Kontotransaktionen zu empfangen. Es können Gebühren für Nachrichten und Daten anfallen.
        2. Die Opt-in-Antwort, die Sie vom Benutzer erwarten. Dies ist in der Regel ein Schlüsselwort, wie beispielsweise JA.
        3. Die Bestätigungsnachricht, die gesendet werden soll, wenn Kunden dieses Schlüsselwort an Ihre Kurzwahlnummer senden. Beispiel: Sie sind jetzt für Kontowarnungen von ExampleCorp registriert. Es können Gebühren für Nachrichten und Daten anfallen. Zum Abbestellen STOP senden oder HILFE senden, um weitere Informationen zu erhalten.

- Der Zweck Ihrer Nachrichten:
    - Der Zweck der Nachrichten, die Sie mit Ihrer Kurzwahlnummer senden möchten. Geben Sie eine der folgenden Optionen an:
        - **`Promotions and marketing`**
        - **`Location-based services`**
        - **`Notifications`**
        - **`Information on demand`**
        - **`Group chat`**
        - **`Two-factor authentication (2FA)`**
        - **`Polling and surveys`**
        - **`Sweepstakes or contests`**
        - **`Other`** Bei „Sonstiges” bitte nähere Informationen bereitstellen.

  - Ob Sie Ihre Kurzwahlnummer für Werbe- oder Marketingnachrichten für ein anderes als Ihr eigenes Unternehmen verwenden möchten.
  - Ob Sie Ihren Kurzcode verwenden möchten, um Nachrichten zu versenden, die Protected Health Information (PHI) gemäß dem Health Insurance Portability and Accountability Act (HIPAA) und den damit verbundenen Gesetzen und Vorschriften enthalten.

- Nachrichteninhalt:
  - Die Nachricht, die gesendet werden soll, wenn Kunden durch Senden eines bestimmten Schlüsselworts Ihre Nachrichten abonnieren. Gehen Sie mit Bedacht vor, wenn Sie Schlüsselwort und Nachricht festlegen – es kann Wochen dauern, diese Nachricht zu ändern. Wenn wir Ihre Kurzwahlnummer erstellen, registrieren wir Schlüsselwort und Nachricht für die Mobilfunkanbieter in dem Land, in dem Sie die Kurzwahlnummer verwenden. Ihre Nachricht kann wie folgt aussehen: Willkommen bei *ProductName* Warnungen! Es fallen Nachrichten- und Datengebühren an.*2* Nachrichten pro Monat. Senden Sie HILFE; wenn Sie Unterstützung benötigen, und STOPP, wenn Sie keine Benachrichtigungen mehr erhalten möchten.
  - Die Antwort, die gesendet werden soll, wenn Kunden auf Ihre Nachrichten mit dem Schlüsselwort HELP antworten. Diese Nachricht muss Kontaktinformationen für den Kundensupport beinhalten. Beispiel: *ProductName*-Warnungen: Hilfe unter *example.com/help* oder *(800) 555-0199*. Es fallen Nachrichten- und Datengebühren an.*2* Nachrichten pro Monat. Antworten Sie mit STOPP, wenn Sie keine Benachrichtigungen mehr erhalten möchten.
  - Die Antwort, die gesendet werden soll, wenn Kunden auf Ihre Nachrichten mit dem Schlüsselwort STOP antworten. Diese Nachricht muss bestätigen, dass der Benutzer keine Nachrichten mehr von Ihnen erhalten wird. Beispiel: Sie haben das Abonnement der *ProductName*-Warnungen gekündigt. Es werden keine weiteren Nachrichten mehr gesendet. Senden Sie HILFE für Unterstützung oder wählen Sie *(800) 555-0199*.
  - Der Text, den Sie für die regelmäßige Erinnerung verwenden möchten, dass der Benutzer Ihre Nachrichten abonniert hat. Beispiel: Erinnerung: Sie haben Kontowarnungen von ExampleCorp abonniert. Es können Gebühren für Nachrichten und Daten anfallen. Zum Abbestellen STOP senden oder HILFE senden, um weitere Informationen zu erhalten.
  - Ein Beispiel für jede Art von Nachricht, die Sie mit Ihrer Kurzwahlnummer senden möchten. Geben Sie mindestens drei Beispiele an. Wenn Sie mehr als drei Arten von Nachrichten senden möchten, geben Sie Beispiele für alle an.

> **Important**
>
> Mobilfunkanbieter erfordern von uns zur Bereitstellung von Kurzwahlnummern alle oben aufgeführten Angaben. Wir können Ihre Anfrage erst verarbeiten, nachdem Sie alle diese Angaben gemacht haben.

10. (Optional) Wenn Sie weitere Anforderungen einreichen möchten, wählen Sie Weitere Anforderung hinzufügen. Wenn Sie mehrere Anfragen durchführen, geben Sie für jede davon die erforderlichen Informationen an. Die erforderlichen Informationen finden Sie in den anderen Abschnitten von Anfordern von Support für SMS-Messaging mit Amazon SNS.
11. Wählen Sie unter Kontaktoptionen für Bevorzugte Kontaktsprache die Sprache aus, in der Sie Mitteilungen zu diesem Fall erhalten möchten.
12. Wenn Sie fertig sind, klicken Sie auf Submit (Absenden).

Nachdem wir Ihre Anfrage erhalten haben, geben wir innerhalb von 24 Stunden eine erste Antwort. Unter Umständen werden wir Sie kontaktieren, um weitere Informationen anzufordern. Wenn wir Ihnen eine Kurzwahlnummer bereitstellen können, senden wir Ihnen Informationen über die Kosten, die mit dem Bezug einer Kurzwahlnummer in dem Land oder der Region, das bzw. die Sie in Ihrer Anforderung angegeben haben, verbunden sind. Wir geben außerdem den geschätzten erforderlichen Zeitaufwand für die Bereitstellung einer Kurzwahlnummer in Ihrem Land oder Ihrer Region an. Die Bereitstellung einer Kurzwahlnummer dauert in der Regel mehrere Wochen. Diese Verzögerung kann allerdings sehr viel kürzer oder länger sein, abhängig von dem Land oder der Region, auf das bzw. die sich die Kurzwahlnummer bezieht.

> **Note**
>
> Die Gebühren für die Verwendung von Kurzwahlnummern werden unverzüglich berechnet, nachdem wir Ihre Anforderung zur Verwendung von Kurzwahlnummern beim Funknetzbetreiber initiiert haben. Sie sind für die Zahlung dieser Gebühren verantwortlich, auch wenn die Kurzwahl noch nicht vollständig bereitgestellt wurde.

Da wir verhindern möchten, dass unerwünschte oder schädliche Inhalte in unseren Systemen eingehen, müssen wir jede Anfrage sorgfältig prüfen. Wenn Ihr Anwendungsfall gegen unsere Richtlinien verstößt, können wir Ihrer Anfrage möglicherweise nicht nachkommen.

Nächste Schritte

Sie haben eine Kurzwahlnummer bei Mobilfunkanbietern registriert und Ihre Einstellungen in der Amazon SNS-Konsole überprüft. Jetzt können Sie mit Amazon SNS SMS-Nachrichten mit Ihrer Kurzwahlnummer als Ursprungsnummer senden.

Anfordern von 10DLC-Nummern, gebührenfreien Nummern und P2P-Langwahlnummern für SMS-Messaging mit Amazon SNS

> **Important**
>
> Ab dem 1. Juni 2021 unterstützen US-Telekommunikationsanbieter die Verwendung von person-to-person (P2P)-Langwahlnummern für die application-to-person (A2P)-Kommunikation zu US-Zielen nicht mehr. Stattdessen müssen Sie eine andere Art von Ursprungs-ID für diese Nachrichten verwenden. Weitere Informationen finden Sie unter 10DLC.

Um 10DLC-Nummern, gebührenfreie Nummern und P2P-Langwahlnummern anzufordern, verwenden Sie die Amazon Pinpoint-Konsole. Für detaillierte Anweisungen siehe Anfordern einer Nummer im Amazon Pinpoint Benutzerhandbuch.

> **Important**
>
> US-Mobilfunkanbieter haben kürzlich ihre Vorschriften geändert und verlangen, dass alle gebührenfreien Nummern (TFNs) vor dem 30. September 2022 bei einer Aufsichtsbehörde registriert werden müssen. Weitere Informationen zur Registrierung einer gebührenfreien Nummer finden Sie unter Registrieren Ihrer gebührenfreien Nummer.
> Wenn Sie Ihre gebührenfreie Nummer am oder vor dem 30. September 2022 gekauft haben, wird ihr Status bis zum 1. Oktober 2022 mit Active (Aktiv) angegeben, es sei denn, Sie haben Ihre Registrierung abgeschlossen und sie wurde mit dem Status Completed (Abgeschlossen) zurückgegeben. Andernfalls wird sie in den Status Pending (Ausstehend) versetzt und Sie können damit erst Nachrichten senden, wenn Sie die Nummer registriert haben, die Registrierung zurückgegeben wurde oder ihr Status auf Active (Aktiv) festgelegt wurde.
> Die Registrierung kann bis zu 15 Werktage dauern.

Um Ihr 10DLC-Unternehmen und Ihre Kampagne zu registrieren, öffnen Sie die Amazon Pinpoint-Konsole in der Region USA Ost (Nord-Virginia). Anstatt eine 10DLC-Nummer anzufordern, verwenden Sie die AWS Service Quotas-Konsole, um einen Fall zur Erhöhung des Servicelimits zu erstellen, während Sie die 10DLC-Nummer für diese Region anfordern. Weitere Informationen zu Regionen, in denen Amazon Pinpoint verfügbar ist, finden Sie unter Amazon Pinpoint Endpunkte und -Kontingente in der Allgemeine AWS-Referenz.

> **Note**
>
> Wenn SMS-Messaging mit Amazon SNS neu für Sie ist, sollten Sie einen monatlichen Kostenschwellenwert für SMS-Nachrichten anfordern, der den erwarteten Anforderungen Ihres SMS-Anwendungsfalls entspricht. Standardmäßig liegt der monatliche Kostenschwellenwert bei 1,00 USD. Weitere Informationen finden Sie unter Anfordern von Erhöhungen Ihres monatlichen SMS-Ausgabenkontingents für Amazon SNS.

Anfordern von Sender-IDs für SMS-Messaging mit Amazon SNS

> **Important**
>
> Wenn SMS-Messaging mit Amazon SNS neu für Sie ist, fordern Sie einen monatlichen Kostenschwellenwert für SMS-Nachrichten an, der den erwarteten Anforderungen Ihres SMS-Anwendungsfalls entspricht. Standardmäßig liegt der monatliche Kostenschwellenwert bei 1,00 USD. Sie können in demselben Supportfall, in dem Sie eine Sender-ID anfordern, auch eine Erhöhung Ihres Ausgabenschwellenwerts anfordern. Wenn Sie möchten, können Sie auch einen eigenen Fall eröffnen. Weitere Informationen finden Sie unter Anfordern von Erhöhungen Ihres monatlichen SMS-Ausgabenkontingents für Amazon SNS.

In SMS-Messaging ist eine Sender-ID ein Name, der auf den Geräten der Empfänger als Sender der Nachricht angezeigt wird. Sender-IDs sind eine nützliche Möglichkeit, sich gegenüber den Empfängern Ihrer Nachrichten zu identifizieren.

Der Support für Sender-IDs variiert je nach Land. Beispielsweise unterstützen Mobilfunkanbieter in den USA keine Sender-IDs, aber die Anbieter in Indien verlangen, dass Sender-IDs von Sendern verwendet werden. Eine vollständige Liste der Länder , die Sender-IDs unterstützen, finden Sie unter Unterstützte Länder und Regionen.

> **Important**
>
> In einigen Ländern müssen Sie Sender-IDs registrieren, bevor Sie sie zum Senden von Nachrichten verwenden. Je nach Land kann dieser Registrierungsprozess mehrere Wochen dauern. Die Länder, die vorab registrierte Sender-IDs erfordern, sind in der Tabelle auf der Seite Supported Countries (Unterstützte Länder) angegeben.
> Sie können dieselbe Sender-ID in mehreren AWS-Konten für SMS-Nachrichten verwenden und registrieren. Wenn Sie über Enterprise Support verfügen und mehrere Vorlagen registrieren, befolgen Sie die unten stehenden Schritte und arbeiten Sie mit Ihrem Technical Account Manager zusammen, um sicherzustellen, dass Ihre Onboarding-Erfahrung koordiniert wird.

Wenn Sie Nachrichten an Empfänger in einem Land senden, in der Sender-IDs unterstützt werden, und dieses Land keine Registrierung Ihrer Sender-ID erfordert, müssen Sie keine zusätzlichen Schritte ausführen. Sie können sofort mit dem Senden von Nachrichten beginnen, die Sender-ID-Werte enthalten.

Sie müssen die Verfahren auf dieser Seite nur abschließen, wenn Sie planen, Nachrichten in ein Land zu senden, in der die Registrierung von Sender-IDs erforderlich ist.

Schritt 1: Öffnen eines Amazon SNS SMS-Falls

Wenn Sie planen, Nachrichten an Empfänger in einem Land zu senden, in dem Sender-IDs erforderlich sind, können Sie eine Sender-ID anfordern, indem Sie einen neuen Fall im AWS-Support-Center erstellen.

> **Note**
>
> Wenn Sie planen, Nachrichten an Empfänger in einem Land zu senden, in dem Sender-IDs zulässig, aber nicht erforderlich sind, müssen Sie keinen Fall im Support-Center öffnen. Sie können sofort mit dem Senden von Nachrichten beginnen, die Sender-IDs verwenden.

So fordern Sie eine Sender-ID an

1. Melden Sie sich an der AWS Management Console unter https://console.aws.amazon.com/ an.
2. Klicken Sie im Menü Support (Support) auf Support Center (Support-Center).

3. Wählen Sie auf der Registerkarte Offene Support-Fälle die Option Fall erstellen aus.
4. Wählen Sie den Link Erhöhung des Servicelimits? und gehen Sie dann wie folgt vor:

   - Wählen Sie für Limit Type (Limittyp) die Option Pinpoint SMS aus.
   - (Optional) Geben Sie unter Link zur Website oder App angeben, die SMS senden wird die Website oder Anwendung an, auf/in der Ihre Mitglieder sich für den Empfang Ihrer SMS-Nachrichten registrieren können.
   - (Optional) Wählen Sie für Art der Nachrichten, die gesendet werden sollen die Art der Nachrichten aus, die Sie mit Ihren Langwahlnummern senden möchten:
     - One-time Password (Einmaliges Passwort) – Nachrichten, die für Ihre Kunden Passwörter zur Authentifizierung bei Ihrer Website oder Anwendung bereitstellen.
     - Promotional (Werbung) – Nicht kritische Nachrichten, die Ihr Unternehmen oder Ihren Service bewerben, wie beispielsweise Sonderangebote oder Ankündigungen.
     - Transactional (Transaktionsnachrichten) – Wichtige Informationsmeldung, die Kundentransaktionen unterstützen, wie beispielsweise Bestellbestätigungen oder Kontowarnungen. Transaktionsnachrichten dürfen keine Werbeaktionen oder Marketinginhalte enthalten.
   - (Optional) Wählen Sie für Aus welcher AWS-Region werden Sie Nachrichten senden die AWS-Region, aus der Sie Nachrichten senden werden.
   - (Optional) Geben Sie für In welche Länder möchten Sie Nachrichten senden die Länder an, in denen Sie eine Sender-ID registrieren möchten. Die Unterstützung von Sender-IDs und die Anforderungen hinsichtlich der Registrierung von Sender-IDs unterscheiden sich je nach Land. Weitere Informationen finden Sie unter Unterstützte Länder und Regionen.

     Wenn die Liste der Länder das Zeichenlimit für dieses Textfeld überschreitet, können Sie stattdessen die Länder im Abschnitt Fallbeschreibung auflisten.
   - (Optional) Geben Sie unter Wie Ihre Kunden dem Erhalt von Nachrichten zustimmen, die von Ihnen gesendet werden eine Beschreibung dazu ein, wie Ihre Kunden dem Erhalt von Nachrichten von Ihnen zustimmen..
   - (Optional) Geben Sie im Feld Bitte geben Sie die Nachrichtenvorlage an, die Sie verwenden möchten, um Nachrichten an Ihre Kunden zu senden die Vorlage ein, die Sie verwenden werden.
5. Führen Sie unter Requests (Anforderungen) die folgenden Abschnitte aus:

   - Wählen Sie als Region die AWS-Region für Ihre Sender-ID aus.

> **Note**
>
> Die Region ist im Abschnitt Anforderungen erforderlich. Auch wenn Sie diese Informationen im Abschnitt Falldetails angegeben haben, müssen Sie sie auch hier angeben.

- Wählen Sie für Resource Type (Ressourcentyp) die Option General Limits (Allgemeine Limits) aus.
- Wählen Sie als Limit die Option SMS-Produktionszugriff aus.

6. Geben Sie für Neuer Grenzwert die Anzahl von Sender-IDs ein, die Sie anfordern. Dieser Wert ist in der Regel **1**.
7. (Optional) Wenn Sie weitere Anforderungen einreichen möchten, wählen Sie Weitere Anforderung hinzufügen. Die erforderlichen Informationen finden Sie in den anderen Abschnitten von Anfordern von Support für SMS-Messaging mit Amazon SNS.
8. Machen Sie unter Case description (Fallbeschreibung) für Use case description (Anwendungsfall-Beschreibung) die folgenden Angaben:
   - Die Sender-ID, die Sie registrieren möchten.
   - Die Vorlage, die Sie für Ihre SMS-Nachrichten verwenden möchten.
   - Die Anzahl der Nachrichten, die Sie pro Monat an jeden Empfänger senden möchten.
   - Informationen darüber, wie Ihre Kunden sich für den Empfang von Nachrichten von Ihnen entscheiden.
   - Der Name Ihres Unternehmens oder Ihrer Organisation.
   - Die Adresse, die Ihrem Unternehmen oder Ihrer Organisation zugeordnet ist.
   - Das Land, in dem sich Ihr Unternehmen oder Ihre Organisation befindet.
   - Eine Telefonnummer für Ihr Unternehmen oder Ihre Organisation.
   - Die URL der Website für Ihr Unternehmen oder Ihre Organisation.
9. Wählen Sie unter Kontaktoptionen für Bevorzugte Kontaktsprache die Sprache aus, in der Sie Mitteilungen zu diesem Fall erhalten möchten.
10. Wenn Sie fertig sind, klicken Sie auf Submit (Absenden).

Nachdem wir Ihre Anfrage erhalten haben, geben wir innerhalb von 24 Stunden eine erste Antwort. Unter Umständen werden wir Sie kontaktieren, um weitere Informationen anzufordern. Wenn wir Ihnen eine Sender-ID bereitstellen können, senden wir eine Schätzung, wie lange es bis zur Bereitstellung dauern wird.

Da wir verhindern möchten, dass unerwünschte oder schädliche Inhalte in unseren Systemen eingehen, müssen wir jede Anfrage sorgfältig prüfen. Wenn Ihr Anwendungsfall gegen unsere Richtlinien verstößt, können wir Ihrer Anfrage möglicherweise nicht nachkommen.

Schritt 2: Aktualisieren Ihrer SMS-Einstellungen in der Amazon SNS-Konsole

Wenn wir den Vorgang abschließen, Ihre Sender-ID zu erhalten, reagieren wir auf Ihren Fall. Wenn Sie diese Benachrichtigung erhalten, führen Sie die Schritte in diesem Abschnitt aus, um Amazon SNS so zu konfigurieren, dass Ihre Sender-ID als Standard-Sender-ID für alle mit Ihrem Konto gesendeten Nachrichten verwendet wird. Als Alternative können Sie auch festlegen, welche Sender-ID beim Veröffentlichen der Nachricht verwendet werden soll.

1. Melden Sie sich bei der Amazon SNS-Konsole an.
2. Wählen Sie im Navigationsbereich Mobil und dann SMS aus.
3. Wählen Sie im Abschnitt SMS-Präferenzen die Option Bearbeiten aus.
4. Geben Sie im Abschnitt Details im Feld Standard-Sender-ID die angegebene Sender-ID ein, die als Standardwert für alle Nachrichten aus Ihrem Konto verwendet werden soll.
5. Wenn Sie die gewünschten Änderungen vorgenommen haben, wählen Sie Save changes (Änderungen speichern) aus.

Nächste Schritte

Sie haben eine Sender-ID registriert und Ihre Einstellungen in der Amazon SNS-Konsole aktualisiert. Jetzt können Sie mit Amazon SNS SMS-Nachrichten mit Ihrer Sender-ID senden. SMS-Empfänger in unterstützten Länder sehen auf ihren Geräten Ihre Sender-ID als Sender dieser Nachricht. Wenn beim Veröffentlichen von Nachrichten eine andere Sender-ID verwendet wird, wird die hier konfigurierte Standard-ID außer Kraft gesetzt.

Anfordern von Erhöhungen Ihres monatlichen SMS-Ausgabenkontingents für Amazon SNS

Amazon SNS stellt Ausgabenkontingente zur Verfügung, mit denen Sie die maximalen Kosten pro Monat verwalten können, die durch das Senden von SMS über Ihr Konto entstehen. Das

Ausgabenkontingent begrenzt Ihr Risiko im Falle eines böswilligen Angriffs und verhindert, dass Ihre Upstream-Anwendung mehr Nachrichten sendet als erwartet. Sie können Amazon SNS so konfigurieren, dass die Veröffentlichung von SMS-Nachrichten eingestellt wird, wenn festgestellt wird, dass beim Senden einer SMS-Nachricht Kosten anfallen, die Ihr Ausgabenkontingent für den aktuellen Monat übersteigen.

Um sicherzustellen, dass Ihr Betrieb nicht beeinträchtigt wird, empfehlen wir, eine Ausgabenquote anzufordern, die hoch genug ist, um Ihre Produktions-Workloads zu unterstützen. Weitere Informationen finden Sie unter Schritt 1: Öffnen eines Amazon-SNS-SMS-Falls. Sobald Sie das Kontingent erhalten haben, können Sie Ihr Risiko verwalten, indem Sie das volle Kontingent oder einen kleineren Wert anwenden, wie unter Schritt 2: Aktualisieren Ihrer SMS-Einstellungen beschrieben. Wenn Sie einen kleineren Wert anwenden, können Sie Ihre monatlichen Ausgaben mit der Option steuern, bei Bedarf hochzuskalieren.

> **Important**
>
> Da Amazon SNS ein verteiltes System ist, beendet es das Senden von SMS-Nachrichten innerhalb von einigen Minuten, falls das Ausgabenkontingent überschritten wurde. Wenn Sie in diesem Zeitraum weiter SMS-Nachrichten senden, können Kosten entstehen, die Ihr Kontingent überschreiten.

Wir setzen das Ausgabenkontingent für alle neuen Konten auf 1,00 USD pro Monat fest. Dieses Kontingent soll Ihnen erlauben, die Möglichkeiten des Nachrichtenversands von Amazon SNS zu testen. Zum Anfordern einer Erhöhung des SMS-Ausgabenkontingents für Ihr Konto öffnen Sie einen Kontingenterhöhungs-Fall im AWS-Support-Center.

Schritt 1: Öffnen eines Amazon SNS SMS-Falls

Sie können eine Erhöhung Ihres monatlichen Ausgabenkontingents anfordern, indem Sie einen Kontingenterhöhungs-Fall im AWS-Support-Center öffnen.

> **Note**
>
> Einige der Felder im Antragsformular sind als „optional“ markiert. Allerdings benötigt AWS Support alle Informationen, die in den folgenden Schritten aufgeführt sind, um Ihre Anfrage zu bearbeiten. Wenn Sie nicht alle erforderlichen Informationen bereitstellen, kann es bei der Bearbeitung Ihrer Anfrage zu Verzögerungen kommen.

1. Melden Sie sich an der AWS Management Console unter https://console.aws.amazon.com/ an.
2. Klicken Sie im Menü Support (Support) auf Support Center (Support-Center).
3. Wählen Sie auf der Registerkarte Offene Support-Fälle die Option Fall erstellen aus.
4. Wählen Sie den Link Erhöhung des Servicelimits? und gehen Sie dann wie folgt vor:
   - Wählen Sie für Grenzwerttyp die Option SNSSMS aus.
   - (Optional) Geben Sie unter Link zur Website oder App angeben, die die SMS-Nachrichten senden wird Informationen über die Website, die Anwendung oder den Service an, die bzw. der SMS-Nachrichten senden wird.
   - (Optional) Wählen Sie für Art der Nachrichten, die gesendet werden sollen die Art der Nachrichten aus, die Sie mit Ihren Langwahlnummern senden möchten:
     - One-time Password (Einmaliges Passwort) – Nachrichten, die für Ihre Kunden Passwörter zur Authentifizierung bei Ihrer Website oder Anwendung bereitstellen.
     - Promotional (Werbung) – Nicht kritische Nachrichten, die Ihr Unternehmen oder Ihren Service bewerben, wie beispielsweise Sonderangebote oder Ankündigungen.
     - Transactional (Transaktionsnachrichten) – Wichtige Informationsmeldung, die Kundentransaktionen unterstützen, wie beispielsweise Bestellbestätigungen oder Kontowarnungen. Transaktionsnachrichten dürfen keine Werbeaktionen oder Marketinginhalte enthalten.
   - (Optional) Wählen Sie für Aus welcher AWS-Region werden Sie Nachrichten senden die Region, aus der Sie Nachrichten senden werden.
   - (Optional) Geben Sie für In welche Länder möchten Sie Nachrichten senden das Land oder die Region ein, in dem bzw. der Sie Kurzwahlnummern erwerben möchten.
   - (Optional) Geben Sie unter Wie entscheiden sich Ihre Kunden dafür, Nachrichten von Ihnen zu erhalten Einzelheiten zu Ihrem Anmeldeverfahren an.
   - (Optional) Geben Sie im Feld Bitte geben Sie die Nachrichtenvorlage an, die Sie verwenden möchten, um Nachrichten an Ihre Kunden zu senden die Vorlage ein, die Sie verwenden werden.
5. Führen Sie unter Requests (Anforderungen) die folgenden Abschnitte aus:
   - Wählen Sie für die Region die Region aus, aus der Sie Nachrichten senden möchten.

> **Note**
>
> Die Region ist im Abschnitt Anforderungen erforderlich. Auch wenn Sie diese Informationen im Abschnitt Falldetails angegeben haben, müssen Sie sie auch hier angeben.

- Wählen Sie für Resource Type (Ressourcentyp) die Option General Limits (Allgemeine Limits) aus.
- Wählen Sie für Limit Account Spend Threshold Increase (Erhöhung des Ausgabenschwellenwerts) aus.

6. Geben Sie für den Neuen Grenzwert den Höchstbetrag in USD ein, den Sie pro Kalendermonat für SMS-Nachrichten ausgeben möchten.
7. Machen Sie unter Case description (Fallbeschreibung) für Use case description (Anwendungsfall-Beschreibung) die folgenden Angaben:

   - Die Website oder App des Unternehmens oder Services, das/der SMS-Nachrichten sendet.
   - Den Service, der von Ihrer Website oder App bereitgestellt wird, und wie Ihre SMS-Nachrichten zu diesem Service beitragen.
   - Wie sich Benutzer bei Ihrer Website, App oder an einem anderen Ort anmelden, um Ihre SMS-Nachrichten freiwillig zu erhalten.

   Wenn Ihr angefordertes Ausgabenkontingent (der Wert, den Sie unter New quota value (Neuer Kontingentwert) angegeben haben), 10.000 USD überschreitet, geben Sie für jedes Zielland Ihrer Nachrichten die folgenden zusätzlichen Einzelheiten an:

   - Ob Sie eine Sender-ID oder eine Kurzwahlnummer verwenden. Wenn Sie eine Sender-ID verwenden, geben Sie an:
     - Die Sender-ID.
     - Ob die Sender-ID bei Mobilnetzanbietern in dem betreffenden Land registriert ist.
   - Die erwartete Höchstzahl der Transaktionen pro Sekunde (TPS) für Ihr Messaging.
   - Die durchschnittliche Nachrichtengröße.
   - Die Vorlage für die in das betreffende Land gesendeten Nachrichten.
   - (Optional) Eventuelle Zeichenverschlüsselungsanforderungen.

8. (Optional) Wenn Sie weitere Anforderungen einreichen möchten, wählen Sie Weitere Anforderung hinzufügen. Wenn Sie mehrere Anfragen durchführen, geben Sie für jede davon die erforderlichen Informationen an. Die erforderlichen Informationen finden Sie in den anderen Abschnitten von Anfordern von Support für SMS-Messaging mit Amazon SNS.
9. Wählen Sie unter Kontaktoptionen für Bevorzugte Kontaktsprache die Sprache aus, in der Sie Mitteilungen zu diesem Fall erhalten möchten.
10. Wenn Sie fertig sind, klicken Sie auf Submit (Absenden).

Das AWS Support-Team wird innerhalb von 24 Stunden erstmals auf die Anforderung antworten.

Da wir verhindern möchten, dass unerwünschte oder schädliche Inhalte in unseren Systemen eingehen, müssen wir jede Anfrage sorgfältig prüfen. Nach einer Prüfung kommen wir Ihrer Anfrage innerhalb dieses 24-Stunden-Zeitraums nach. Für den Fall, dass wir weitere Informationen von Ihnen benötigen, kann die Bearbeitung Ihrer Anfrage länger dauern.

Wenn Ihr Anwendungsfall gegen unsere Richtlinien verstößt, können wir Ihrer Anfrage möglicherweise nicht nachkommen.

Schritt 2: Aktualisieren Ihrer SMS-Einstellungen in der Amazon SNS-Konsole

Nachdem wir Sie darüber informiert haben, dass Ihr monatliches Ausgabenkontingent erhöht wurde, müssen Sie das Ausgabenkontingent für Ihr Konto in der Amazon SNS-Konsole anpassen.

> **Important**
>
> Sie müssen die folgenden Schritte ausführen, sonst wird Ihr SMS-Ausgabenlimit nicht erhöht.

So passen Sie Ihr Ausgabenkontingent in der Konsole an

1. Melden Sie sich bei der Amazon SNS-Konsole an.
2. Öffnen Sie das linke Navigationsmenü, erweitern Sie Mobil und wählen Sie dann SMS.
3. Wählen Sie auf der Seite Mobile text messaging (SMS) (Mobile Texttnachrichten (SMS)) im Abschnitt Text messaging preferences (Textnachrichteneinstellungen) die Option Edit (Bearbeiten) aus.
4. Geben Sie auf der Seite SMS-Voreinstellungen bearbeiten unter Details Ihr neues SMS-Ausgabenlimit unter Kontoausgabenlimit ein.

> **Note**
>
> Möglicherweise erhalten Sie eine Warnung, dass der eingegebene Wert größer als das Standardausgabenlimit ist. Sie können dies ignorieren.

5. Wählen Sie Save Changes.

> **Note**
>
> Wenn der Fehler „Invalid Parameter (Ungültiger Parameter)" angezeigt wird, überprüfen Sie den Kontakt beim AWS Support, und prüfen Sie, ob Sie das richtige neue SMS-Ausgabenlimit eingegeben haben. Wenn weiterhin ein Problem auftritt, öffnen Sie einen Fall im AWS Supportcenter.

Festlegen von SMS-Messaging-Einstellungen

Verwenden Sie Amazon SNS, um Einstellungen für SMS-Messaging anzugeben. Sie können beispielsweise festlegen, ob Zustellung zu Kosten oder zur Zuverlässigkeit optimiert werden, Ihr monatliches Ausgabelimit, wie Zustellungen protokolliert werden und ob Sie tägliche SMS-Nutzungsberichte erhalten möchten.

Diese Einstellungen gelten für jede SMS-Nachricht, die Sie von Ihrem Konto senden. Sie können jedoch auch beim Senden einer einzelnen Nachricht überschrieben werden. Weitere Informationen finden Sie unter Veröffentlichen auf einem Mobiltelefon.

Themen

- Einstellung der SMS-Nachrichteneinstellungen mit dem AWS Management Console
- Einstellung von Einstellungen (AWS SDKs)

Einstellung der SMS-Nachrichteneinstellungen mit dem AWS Management Console

1. Melden Sie sich bei der Amazon SNS-Konsole an.
2. Wählen Sie eine Region aus, die SMS-Messaging unterstützt.
3. Wählen Sie im Navigationsbereich Mobile (Mobil) und dann Text messaging (SMS) (Textnachrichten (SMS)) aus.

4. Wählen Sie auf der Seite Mobile text messaging (SMS) (Mobile Texttnachrichten (SMS)) im Abschnitt Text messaging preferences (Textnachrichteneinstellungen) die Option Edit (Bearbeiten) aus.
5. Führen Sie auf der Seite Edit text messaging preferences (SMS-Einstellungen bearbeiten) im Abschnitt Details Folgendes durch:

   a. Wählen Sie für Default message type (Standardnachrichtentyp) eine der folgenden Optionen aus:

      - Werbenachrichten – Nicht kritische Nachrichten (zum Beispiel Marketing). Amazon SNS optimiert die Übertragung von Nachrichten im Hinblick auf möglichst niedrige Kosten.
      - Transaktionsnachrichten (Standard) – Kritische Nachrichten, die Kundentransaktionen unterstützen, wie beispielsweise One-Time-Passcodes für eine Multi-Faktor-Authentifizierung (MFA). Amazon SNS optimiert die Übertragung von Nachrichten im Hinblick auf höchste Zuverlässigkeit.

      Informationen zu den Gebühren für Aktions- und Transaktionsnachrichten finden Sie unter Global SMS Pricing.

   b. (Optional) Geben Sie für Account spend limit (Kontoausgabenlimit) den Betrag in US-Dollar ein, den Sie pro Kalendermonat für SMS-Nachrichten ausgeben möchten.

      > ⚠ Important
      >
      > - Standardmäßig ist das Ausgabenkontingent auf 1,00 US-Dollar festgelegt. Wenn Sie das Service-Kontingent erhöhen möchten, senden Sie eine Anfrage.
      > - Wenn die in der Konsole festgelegte Menge Ihr Service-Kontingent überschreitet, stoppt Amazon SNS die Veröffentlichung von SMS-Nachrichten.
      > - Da Amazon SNS ein verteiltes System ist, beendet es das Senden von SMS-Nachrichten innerhalb von einigen Minuten nach der Überschreitung des Ausgabenkontingents. Wenn Sie während dieses Zeitraums weiter SMS-Nachrichten senden, können Kosten entstehen, die Ihr Kontingent überschreiten.

6. (Optional) Geben Sie im Feld Default sender ID (Standard-Sender-ID) eine benutzerdefinierte ID ein, wie z. B. Ihr Unternehmen, die als Sender des Empfängergeräts angezeigt wird.

> **Note**
>
> Der Support für Sender-IDs variiert je nach Land.

7. (Optional) Geben Sie den Namen des Amazon S3 Bucket Namen für Nutzungsberichte ein.

> **Note**
>
> Die S3-Bucket-Richtlinie muss Schreibzugriff auf Amazon SNS erteilen.

8. Wählen Sie Änderungen speichern aus.

Einstellung von Einstellungen (AWS SDKs)

Um Ihre SMS-Einstellungen mithilfe eines der AWS SDKs festzulegen, verwenden Sie die Aktion in diesem SDK, die der `SetSMSAttributes` Anfrage in der Amazon SNS SNS-API entspricht. Mit dieser Anforderung können Sie Werte zu den verschiedenen SMS-Attributen zuweisen – beispielsweise das monatliche Ausgabenkontingent und Ihren Standard-SMS-Typ (Werbung oder Transaktionen). Für alle SMS-Attribute siehe SetSMSAttributes in der Amazon Simple Notification Service API-Referenz.

Die folgenden Codebeispiele zeigen, wie Sie es verwenden. `SetSMSAttributes`

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

So verwenden Sie Amazon SNS zum Festlegen des DefaultSMSType-Attributs.

```
//! Set the default settings for sending SMS messages.
/*!
  \param smsType: The type of SMS message that you will send by default.
  \param clientConfiguration: AWS client configuration.
```

```
  \return bool: Function succeeded.
 */
bool AwsDoc::SNS::setSMSType(const Aws::String &smsType,
                             const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::SNS::Model::SetSMSAttributesRequest request;
    request.AddAttributes("DefaultSMSType", smsType);

    const Aws::SNS::Model::SetSMSAttributesOutcome outcome =
 snsClient.SetSMSAttributes(
            request);

    if (outcome.IsSuccess()) {
        std::cout << "SMS Type set successfully " << std::endl;
    }
    else {
        std::cerr << "Error while setting SMS Type: '"
                  << outcome.GetError().GetMessage()
                  << "'" << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Details zu API finden Sie unter SetSMSAttributes in der AWS SDK for C++ -API-Referenz.

CLI

AWS CLI

So legen Sie SMS-Nachrichtenattribute fest

Im folgenden `set-sms-attributes`-Beispiel wird die standardmäßige Absender-ID für SMS-Nachrichten auf `MyName` festgelegt.

```
aws sns set-sms-attributes \
    --attributes DefaultSenderID=MyName
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

- API-Details finden Sie unter SetSMSAttributes in der AWS CLI -Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SetSmsAttributesRequest;
import software.amazon.awssdk.services.sns.model.SetSmsAttributesResponse;
import software.amazon.awssdk.services.sns.model.SnsException;
import java.util.HashMap;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class SetSMSAttributes {
    public static void main(String[] args) {
        HashMap<String, String> attributes = new HashMap<>(1);
        attributes.put("DefaultSMSType", "Transactional");
        attributes.put("UsageReportS3Bucket", "janbucket");

        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();
        setSNSAttributes(snsClient, attributes);
        snsClient.close();
    }
```

```
    public static void setSNSAttributes(SnsClient snsClient, HashMap<String,
 String> attributes) {
        try {
            SetSmsAttributesRequest request = SetSmsAttributesRequest.builder()
                    .attributes(attributes)
                    .build();

            SetSmsAttributesResponse result =
 snsClient.setSMSAttributes(request);
            System.out.println("Set default Attributes to " + attributes + ".
 Status was "
                    + result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Details zu API finden Sie unter SetSMSAttributes in der AWS SDK for Java 2.x -API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie den Client in einem separaten Modul und exportieren Sie ihn.

```
import { SNSClient } from "@aws-sdk/client-sns";

// The AWS Region can be provided here using the `region` property. If you leave
 it blank
// the SDK will default to the region set in your AWS config.
```

```
export const snsClient = new SNSClient({});
```

Importieren Sie das SDK- und Client-Module und rufen Sie die API auf.

```
import { SetSMSAttributesCommand } from "@aws-sdk/client-sns";
import { snsClient } from "../libs/snsClient.js";

/**
 * @param {"Transactional" | "Promotional"} defaultSmsType
 */
export const setSmsType = async (defaultSmsType = "Transactional") => {
  const response = await snsClient.send(
    new SetSMSAttributesCommand({
      attributes: {
        // Promotional – (Default) Noncritical messages, such as marketing
 messages.
        // Transactional – Critical messages that support customer transactions,
        // such as one-time passcodes for multi-factor authentication.
        DefaultSMSType: defaultSmsType,
      },
    }),
  );
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: '1885b977-2d7e-535e-8214-e44be727e265',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   }
  // }
  return response;
};
```

- Weitere Informationen finden Sie im AWS SDK for JavaScript -Entwicklerhandbuch.
- Details zu API finden Sie unter SetSMSAttributes in der AWS SDK for JavaScript -API-Referenz.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

```
$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

try {
    $result = $SnSclient->SetSMSAttributes([
        'attributes' => [
            'DefaultSMSType' => 'Transactional',
        ],
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- Weitere Informationen finden Sie im AWS SDK for PHP -Entwicklerhandbuch.
- Details zu API finden Sie unter SetSMSAttributes in der AWS SDK for PHP -API-Referenz.

Senden von SMS-Nachrichten

In diesem Abschnitt wird beschrieben, wie SMS-Nachrichten gesendet werden.

Themen

So veröffentlichen Sie in einem Thema

Sie können eine einzelne SMS-Nachricht gleichzeitig an viele Telefonnummern veröffentlichen, indem Sie das Amazon SNS-Thema für diese Telefonnummern abonnieren. Ein SNS-Thema ist ein Kommunikationskanal, dem Sie Abonnenten hinzufügen können. Anschließend können Sie an all diese Abonnenten Nachrichten veröffentlichen. Ein Abonnent erhält alle zum Thema veröffentlichten Nachrichten, bis Sie das Abonnement kündigen, oder der Abonnent den Empfang von SMS-Nachrichten von Ihrem AWS-Konto deaktiviert.

Themen

- Senden einer Nachricht an ein Thema (Konsole)
- Senden einer Nachricht an ein Thema (AWS SDKs)

Senden einer Nachricht an ein Thema (Konsole)

Erstellen Sie ein Thema wie folgt

Führen Sie die folgenden Schritte aus, wenn Sie noch kein Thema besitzen, an das Sie SMS-Nachrichten senden möchten.

1. Melden Sie sich bei der Amazon SNS-Konsole an.
2. Wählen Sie im Konsolenmenü die Option AWSRegion, die SMS-Messaging unterstützt aus.
3. Wählen Sie im Navigationsbereich Themen aus.
4. Klicken Sie auf der Seite Themen auf Thema erstellen.
5. Gehen Sie auf der Seite Thema erstellen unter Details wie folgt vor:
   a. Wählen Sie unter Type (Typ) die Option Standard aus.
   b. Geben Sie für Name einen Themennamen ein.
   c. (Optional) Geben Sie unter Display name (Anzeigename) einen benutzerdefinierten Präfix für Ihre SMS-Nachrichten ein. Wenn Sie eine Nachricht an ein Thema senden, stellt Amazon SNS den Anzeigenamen voran, gefolgt von einer spitzen Klammer rechts (**>**) und einer Leerstelle. Anzeigenamen beachteten die Groß- und Kleinschreibung nicht und Amazon SNS konvertiert Anzeigenamen zu Großbuchstaben. Beispiel: Wenn der Anzeigename eines Themas `MyTopic` ist und die Nachricht lautet `Hello World!`, wird die Nachricht folgendermaßen angezeigt:

```
MYTOPIC> Hello World!
```

6. Wählen Sie Thema erstellen aus. Der Name des Themas und der Amazon-Ressourcenname (ARN) werden auf der Registerkarte Themen angezeigt.

So erstellen Sie ein SMS-Abonnement:

Anhand von Abonnements können Sie eine SMS-Nachricht an mehrere Empfänger senden, indem Sie die Nachricht nur einmal zu dem Thema veröffentlichen.

> **Note**
>
> Wenn Sie Amazon SNS zum Senden von SMS-Nachrichten verwenden, befindet sich Ihr AWS-Konto in der SMS-Sandbox. Die SMS-Sandbox bietet Ihnen eine sichere Umgebung, in der Sie Amazon SNS Funktionen ausprobieren können, ohne Ihren Ruf als SMS-Sender zu gefährden. Während sich Ihr Konto in der SMS-Sandbox befindet, können Sie alle Funktionen von Amazon SNS verwenden. Sie können jedoch SMS-Nachrichten nur an verifizierte Zieltelefonnummern senden. Weitere Informationen finden Sie unter SMS-Sandbox.

1. Melden Sie sich bei der Amazon SNS-Konsole an.
2. Wählen Sie im Navigationsbereich Subscriptions aus.
3. Wählen Sie auf der Seite Subscriptions (Abonnements) die Option Create subscription (Abonnement erstellen)aus.
4. Gehen Sie auf der Seite Abonnement erstellen unter Details wie folgt vor:
   a. Für Thema-ARN geben Sie ein oder wählen Sie den Amazon-Ressourcennamen (ARN) des Themas, an das Sie SMS-Nachrichten senden möchten.
   b. Wählen Sie für Protokoll SMS aus.
   c. Geben Sie für Endpunkt die Telefonnummer ein, die mit Ihrem Thema abonniert werden soll.
5. Klicken Sie auf Create subscription (Abonnement erstellen). Die Abonnementinformationen erscheint auf der Seite Abonnements.

   Wiederholen Sie diese Schritte, um weitere Telefonnummern hinzuzufügen. Sie können auch andere Abonnementtypen, wie E-Mail, hinzufügen.

So senden Sie eine Nachricht

Wenn Sie eine Nachricht veröffentlichen, versucht Amazon SNS diese Nachricht an alle Telefonnummern zu schicken, die das Thema abonniert haben.

1. Wählen Sie in der Amazon SNS-Konsole auf der Seite Themen den Namen des Themas, an das Sie SMS-Nachrichten senden möchten.
2. Wählen Sie auf der Seite mit den Details des Themas Publish message (Nachricht veröffentlichen) aus.
3. Geben Sie auf der Seite Nachricht für Thema veröffentlichen unter Nachrichtendetails wie folgt vor:

   a. Lassen Sie unter Subject (Betreff) das Feld leer, es sei denn, Ihr Thema enthält E-Mail-Abonnements und Sie möchten sowohl an E-Mail- und SMS-Abonnements Nachrichten veröffentlichen. Amazon SNS verwendet den Betreff, den Sie eingeben, als E-Mail-Betreffzeile.

   b. (Optional) Geben Sie für Time to Live (TTL) eine Anzahl an Sekunden an, in denen Amazon SNS Ihre SMS-Nachricht an Abonnenten des Mobilanwendungs-Endpunkts senden muss.
4. Gehen Sie unter Nachrichtentext wie folgt vor:

   a. Wählen Sie für Nachrichtenstruktur Identische Nutzlast für alle Bereitstellungsprotokolle, um die gleiche Nachricht an alle Protokolltypen zu schicken, die Ihr Thema abonniert haben. Oder wählen Sie Benutzerdefinierte Nutzlast für die einzelnen Bereitstellungsprotokolle, um die Nachricht für Abonnenten verschiedener Protokolltypen anzupassen. Sie können beispielsweise eine Standardnachricht für Telefonnummernabonnenten und eine benutzerdefinierte Nachricht für E-Mail-Abonnenten eingeben.

   b. Geben Sie für Nachrichtentext zum Senden an den Endpunkt Ihre Nachricht oder Ihre benutzerdefinierten Nachrichten je Zustellungsprotokoll ein.

      Wenn Ihr Thema über einen Anzeigenamen verfügt, fügt Amazon SNS diesen der Nachricht hinzu, wodurch sich die Nachrichtenlänge verlängert. Die Länge des Anzeigenamens setzt sich aus der Anzahl der Zeichen des Namens sowie zwei Zeichen für die spitze Klammer (**>**) und von Amazon SNS hinzugefügte Leerzeichen zusammen.

      Weitere Informationen über das Größenkontingent für SMS-Nachrichten finden Sie unter Veröffentlichen auf einem Mobiltelefon.

5. (Optional) Fügen Sie bei Nachrichtenattribute Nachrichtmetadaten ein, wie z. B. Zeitstempel, Signaturen und IDs.
6. Wählen Sie Publish message (Nachricht veröffentlichen) aus. Amazon SNS sendet die SMS-Nachricht und zeigt eine Erfolgsmeldung an.

Senden einer Nachricht an ein Thema (AWS SDKs)

Zur Verwendung eines AWS-SDK müssen Sie es mit Ihren -Anmeldeinformationen konfigurieren. Weitere Informationen finden Sie unter Freigegebene Konfigurations- und Anmeldeinformationsdateien im AWS Referenzhandbuch zu SDKs und Tools.

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie ein Amazon-SNS-Thema.
- Verknüpfen Sie Telefonnummern mit dem Thema.
- Veröffentlichen Sie SMS-Nachrichten im Thema, damit alle abonnierten Telefonnummern die Nachricht gleichzeitig empfangen.

Java

SDK für Java 2.x

> **Note**
>
> Auf GitHub finden Sie noch mehr. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS-Code-Beispiel- einrichten und ausführen.

Erstellen Sie ein Thema und geben Sie seinen ARN zurück.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.CreateTopicRequest;
import software.amazon.awssdk.services.sns.model.CreateTopicResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
```

```
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class CreateTopic {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <topicName>

                Where:
                   topicName - The name of the topic to create (for example,
 mytopic).

                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String topicName = args[0];
        System.out.println("Creating a topic with name: " + topicName);
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        String arnVal = createSNSTopic(snsClient, topicName);
        System.out.println("The topic ARN is" + arnVal);
        snsClient.close();
    }

    public static String createSNSTopic(SnsClient snsClient, String topicName) {
        CreateTopicResponse result;
        try {
            CreateTopicRequest request = CreateTopicRequest.builder()
                    .name(topicName)
                    .build();

            result = snsClient.createTopic(request);
            return result.topicArn();
```

```
        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
}
```

Abonnieren eines Endpunkts für ein Thema.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SnsException;
import software.amazon.awssdk.services.sns.model.SubscribeRequest;
import software.amazon.awssdk.services.sns.model.SubscribeResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class SubscribeTextSMS {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <topicArn> <phoneNumber>

                Where:
                   topicArn - The ARN of the topic to subscribe.
                   phoneNumber - A mobile phone number that receives
 notifications (for example, +1XXX5550100).
                """;

        if (args.length < 2) {
            System.out.println(usage);
            System.exit(1);
        }
```

```
        String topicArn = args[0];
        String phoneNumber = args[1];
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        subTextSNS(snsClient, topicArn, phoneNumber);
        snsClient.close();
    }

    public static void subTextSNS(SnsClient snsClient, String topicArn, String
 phoneNumber) {
        try {
            SubscribeRequest request = SubscribeRequest.builder()
                    .protocol("sms")
                    .endpoint(phoneNumber)
                    .returnSubscriptionArn(true)
                    .topicArn(topicArn)
                    .build();

            SubscribeResponse result = snsClient.subscribe(request);
            System.out.println("Subscription ARN: " + result.subscriptionArn() +
 "\n\n Status is "
                    + result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

Legen Sie Attribute für die Nachricht fest, z. B. die ID des Senders, den Höchstpreis und seinen Typ. Nachrichtenattribute sind optional.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SetSmsAttributesRequest;
import software.amazon.awssdk.services.sns.model.SetSmsAttributesResponse;
import software.amazon.awssdk.services.sns.model.SnsException;
import java.util.HashMap;
```

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class SetSMSAttributes {
    public static void main(String[] args) {
        HashMap<String, String> attributes = new HashMap<>(1);
        attributes.put("DefaultSMSType", "Transactional");
        attributes.put("UsageReportS3Bucket", "janbucket");

        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();
        setSNSAttributes(snsClient, attributes);
        snsClient.close();
    }

    public static void setSNSAttributes(SnsClient snsClient, HashMap<String,
 String> attributes) {
        try {
            SetSmsAttributesRequest request = SetSmsAttributesRequest.builder()
                    .attributes(attributes)
                    .build();

            SetSmsAttributesResponse result =
 snsClient.setSMSAttributes(request);
            System.out.println("Set default Attributes to " + attributes + ".
 Status was "
                    + result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

Veröffentlichen einer Nachricht für ein Thema. Die Nachricht wird an jeden Teilnehmer gesendet.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.PublishRequest;
import software.amazon.awssdk.services.sns.model.PublishResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class PublishTextSMS {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <message> <phoneNumber>

                Where:
                   message - The message text to send.
                   phoneNumber - The mobile phone number to which a message is
 sent (for example, +1XXX5550100).\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String message = args[0];
        String phoneNumber = args[1];
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();
        pubTextSMS(snsClient, message, phoneNumber);
        snsClient.close();
    }
```

```
    public static void pubTextSMS(SnsClient snsClient, String message, String
 phoneNumber) {
        try {
            PublishRequest request = PublishRequest.builder()
                    .message(message)
                    .phoneNumber(phoneNumber)
                    .build();

            PublishResponse result = snsClient.publish(request);
            System.out
                    .println(result.messageId() + " Message sent. Status was " +
 result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

Veröffentlichen auf einem Mobiltelefon

Sie können Amazon SNS verwenden, um SMS-Nachrichten direkt an ein Mobiltelefon zu senden, ohne die Telefonnummer für ein Amazon SNS-Thema zu abonnieren.

> **Note**
>
> Das Abonnieren von Telefonnummern zu einem Thema kann nach wie vor nützlich sein, wenn Sie jede Nachricht in mehreren Telefonnummern zugleich senden möchten. Weitere Anleitungen zum Veröffentlichen einer SMS-Nachricht in einem Thema finden Sie unter So veröffentlichen Sie in einem Thema.

Wenn Sie eine Nachricht senden, können Sie überprüfen, ob der Versand in Bezug auf Kosten und Verlässlichkeit optimiert ist. Sie können auch eine Sender-ID oder Ursprungsnummer angeben. Wenn Sie die Nachricht programmgesteuert über die Amazon SNS SNS-API oder die AWS SDKs senden, können Sie einen Höchstpreis für die Nachrichtenzustellung angeben.

Eine SMS-Nachricht kann maximal 140 Byte groß sein, wobei das Zeichenkontingent vom Codierungsschema abhängig ist. Die Anzahl der Zeichen einer SMS-Nachricht ist entsprechend wie folgt begrenzt:

- 160 GSM-Zeichen
- 140 ASCII-Zeichen
- 70 UCS-2-Zeichen

Wenn Sie eine Nachricht veröffentlichen, die das Größenkontingent überschreitet, sendet Amazon SNS mehrere Teilnachrichten, die jeweils dem Größenkontingent entsprechen. Nachrichten werden niemals inmitten eines Wortes geteilt, sondern immer zwischen zwei Wörtern. Für die gesamte veröffentlichte und in mehreren Teilen versendete SMS-Nachricht gilt ein Größenkontingent von 1600 Byte.

Wenn Sie eine SMS-Nachricht senden, geben Sie die Telefonnummer im E.164-Format an – eine Standardnummernstruktur für die Telefonnummerierung, die für die internationale Telekommunikation verwendet wird. Telefonnummern in diesem Format bestehen aus maximal 15 Zeichen sowie einem vorangestellten Plus-Zeichen (+) und der Ländervorwahl. Eine US-Telefonnummer im E.164-Format sieht beispielsweise wie folgt aus: +1XXX5550100.

Themen

- Senden einer Nachricht (Konsole)
- Senden einer Nachricht (SDKs)AWS

Senden einer Nachricht (Konsole)

1. Melden Sie sich bei der Amazon SNS-Konsole an.
2. Wählen Sie im Konsolenmenü die Option AWS Region, die SMS-Messaging unterstützt aus.
3. Wählen Sie im Navigationsbereich Text-Messaging (SMS) aus.
4. Wählen Sie auf der Seite Text-Messaging (SMS) Textnachricht veröffentlichen.
5. Wählen Sie auf der Seite Textnachricht veröffentlichen für Nachrichtentyp eines der Folgenden:
   - Werbenachrichten – Unkritische Nachrichten, wie beispielsweise Marketing-Nachrichten.
   - Transaktionsnachrichten: Sensible Nachrichten, die Kundentransaktionen unterstützen, wie beispielsweise One-Time-Passcodes für eine Multi-Faktor-Authentifizierung (MFA).

> **Note**
>
> Diese Einstellung auf Nachrichtenebene setzt den Standardnachrichtentyp auf Kontoebene außer Kraft. Auf der Seite Textnachrichten (SMS) im Abschnitt Text-Messaging-Einstellungen können SIe einen Standardnachrichtentyp auf Kontoebene festlegen.

Informationen zu den Gebühren für Aktions- und Transaktionsnachrichten finden Sie unter Weltweite SMS-Preise.

6. Geben Sie unter Zieltelefonnummer die Telefonnummer ein, an die Sie die Nachricht senden möchten.
7. Geben Sie im Feld Message (Nachricht) die Art der zu versendenden Nachricht ein.
8. (Optional) Legen Sie unter Ursprungsidentitäten fest, wie Sie sich selbst gegenüber Ihren Empfängern identifizieren möchten:

   - Geben Sie im Feld Sender ID (Sender-ID) eine Kunden-ID ein, die 3 bis 11 alphanumerische Zeichen enthält, einschließlich mindestens einem Buchstaben und ohne Leerzeichen. Die Sender-ID wird auf dem Empfänger-Gerät als Sender der Nachricht angezeigt. Sie können z. B. Ihre Unternehmensmarke verwenden, um die Nachrichtenquelle leichter erkennbar zu machen.

     Der Support für Sender-IDs variiert je nach Land und/oder Region. Beispiel: Für Nachrichten, die an eine US-Telefonnummer gesendet werden, wird keine Sender-ID angezeigt. Informationen zu Ländern und Regionen, die Sender-IDs unterstützen, finden Sie unter Unterstützte Länder und Regionen.

     Wenn Sie keine Sender-ID angeben, wird eines der folgenden Elemente als Ursprungsidentität angezeigt:

     - In Ländern, die Langwahlnummern unterstützen, wird die Langwahlnummer angezeigt.
     - In Ländern, in denen nur Sender-IDs unterstützt werden, wird HINWEIS angezeigt.

     Diese Sender-ID auf Nachrichtenebene ersetzt die standardmäßige Sender-ID, die Sie auf der Seite Text messaging preferences (SMS-Präferenzen) konfigurieren.

- Um eine Ursprungsnummer festzulegen, geben Sie eine Zeichenfolge von 5-14 Nummern ein, die als Telefonnummer des Senders auf dem Gerät des Empfängers angezeigt werden soll. Diese Zeichenfolge muss mit einer Ursprungsnummer übereinstimmen, die in Ihrem AWS-Konto für das Zielland konfiguriert ist. Bei der Ausgangsnummer kann es sich um eine 10DLC-Nummer, eine gebührenfreie Nummer, einen Langcode oder eine Kurzvorwahl handeln. person-to-person Weitere Informationen finden Sie unter Ursprungsidentitäten für SMS-Nachrichten.

  Wenn Sie keine Absendernummer angeben, wählt Amazon SNS basierend auf Ihrer Konfiguration eine Absendernummer aus, die für die SMS-Textnachricht verwendet werden soll. AWS-Konto

9. Wenn Sie SMS-Nachrichten an Empfänger in Indien senden, erweitern Sie Länderspezifische Attribute und legen Sie die folgenden Attribute fest:

   - Entity-ID – Die Entity ID oder Principal Entity (PE) -ID für das Senden von SMS-Nachrichten an Empfänger in Indien. Diese ID ist eine eindeutige Zeichenfolge von 1 bis 50 Zeichen, die die Telecom Regulatory Authority of India (TRAI) zur Identifizierung der Entität bereitstellt, die Sie bei der TRAI registriert haben.
   - ID der Vorlage – Die Vorlagen-ID für das Senden von SMS-Nachrichten an Empfänger in Indien. Diese ID ist eine eindeutige, von TRAI bereitgestellte Zeichenfolge mit 1 bis 50 Zeichen, die die Vorlage identifiziert, die Sie bei TRAI registriert haben. Die Vorlagen-ID muss der Sender-ID zugeordnet sein, die Sie für die Nachricht angegeben haben.

   Weitere Informationen zum Senden von SMS-Nachrichten an Empfänger in Indien finden Sie unter Anforderungen für die Registrierung der Absender-ID für Indien.

10. Wählen Sie Publish message (Nachricht veröffentlichen) aus.

> **Tip**
>
> Um SMS-Nachrichten von einer Ursprungsnummer zu senden, können Sie auch Ursprungsnummern im Navigationsbereich in der Amazon SNS-Konsole eingeben. Wählen Sie eine Ursprungsnummer, die in der Spalte Funktionen SMS enthält und wählen Sie dann Textnachricht veröffentlichen.

Senden einer Nachricht (SDKs)AWS

Um eine SMS-Nachricht mit einem der AWS SDKs zu senden, verwenden Sie den API-Vorgang in diesem SDK, der der `Publish` Anfrage in der Amazon SNS SNS-API entspricht. Mit dieser Anfrage können Sie eine SMS-Nachricht direkt an eine Telefonnummer senden. Sie können auch den `MessageAttributes`-Parameter verwenden, um Werte für die folgenden Attributnamen zu definieren:

`AWS.SNS.SMS.SenderID`

Eine benutzerdefinierte ID, die 3 bis 11 alphanumerische Zeichen oder Bindestriche (-) enthält, einschließlich mindestens einem Buchstaben und ohne Leerzeichen. Die Sender-ID wird auf dem Empfänger-Gerät als Sender der Nachricht angezeigt. Sie können z. B. Ihre Unternehmensmarke verwenden, um die Nachrichtenquelle leichter erkennbar zu machen.

Der Support für Sender-IDs variiert je nach Land oder Region. Beispiel: Für Nachrichten, die an eine US-Telefonnummer gesendet werden, wird keine Sender-ID angezeigt. Eine Liste der Länder und Regionen, die Sender-IDs unterstützen, finden Sie unter Unterstützte Länder und Regionen.

Wenn Sie keine Sender-ID angeben, erscheint eine Langwahlnummer als Sender-ID in den unterstützten Ländern oder Regionen. Für Länder oder Regionen, die eine alphabetische Sender-ID erfordern, wird NOTICE als Sender-ID angezeigt.

Dieses Attribut auf Nachrichtenebene ersetzt das `DefaultSenderID`-Attribut auf Kontoebene, das Sie bei der Nutzung der `SetSMSAttributes`-Anfrage festlegen.

`AWS.MM.SMS.OriginationNumber`

Eine benutzerdefinierte Zeichenfolge von 5 bis 14 Zahlen, die ein optionales vorzeitiges Pluszeichen (+) enthalten. Diese Zahlenfolge wird als Telefonnummer des Senders auf dem empfangenden Gerät angezeigt. Die Zeichenfolge muss mit einer Absendernummer übereinstimmen, die in Ihrem AWS Konto für das Zielland konfiguriert ist. Bei der Ausgangsnummer kann es sich um eine 10DLC-Nummer, eine gebührenfreie Nummer, einen person-to-person (P2P) -Langcode oder einen Kurzcode handeln. Weitere Informationen finden Sie unter Ursprungsnummern.

Wenn Sie keine Ursprungsnummer angeben, wählt Amazon SNS auf der Grundlage Ihrer AWS Kontokonfiguration eine Ursprungsnummer aus.

`AWS.SNS.SMS.MaxPrice`

Höchstpreis in USD, den Sie zu zahlen bereit sind, um die SMS-Nachricht zu senden. Wenn Amazon SNS festlegt, dass das Senden der Nachricht zu Kosten führen würde, die Ihren Maximalpreis übersteigen, wird die Nachricht nicht gesendet.

Dieses Attribut hat keine Auswirkung, wenn Ihre month-to-date SMS-Kosten das für das Attribut festgelegte Kontingent bereits überschritten haben. `MonthlySpendLimit` Sie können das `MonthlySpendLimit`-Attribut mit der `SetSMSAttributes`-Anfrage festlegen.

Wenn Sie die Nachricht an ein Amazon SNS-Thema senden, gilt bei jeder Nachrichtenübertragung an alle Telefonnummern, die das Thema abonniert haben, der Höchstpreis.

`AWS.SNS.SMS.SMSType`

Der Typ der gesendeten Nachricht:

- `Promotional` (Standard): Unkritische Nachrichten, wie beispielsweise Marketing-Nachrichten.
- `Transactional` – Sensible Nachrichten, die Kundentransaktionen unterstützen, wie beispielsweise One-Time-Passcodes für eine Multi-Faktor-Authentifizierung.

Dieses Attribut auf Nachrichtenebene ersetzt das `DefaultSMSType`-Attribut auf Kontoebene, das Sie bei der Nutzung der `SetSMSAttributes`-Anfrage festlegen.

`AWS.MM.SMS.EntityId`

Dieses Attribut ist nur für das Senden von SMS-Nachrichten an Empfänger in Indien erforderlich.

Das ist Ihre Entity ID oder Principal Entity (PE) -ID für das Senden von SMS-Nachrichten an Empfänger in Indien. Diese ID ist eine eindeutige Zeichenfolge von 1 bis 50 Zeichen, die die Telecom Regulatory Authority of India (TRAI) zur Identifizierung der Entität bereitstellt, die Sie bei der TRAI registriert haben.

`AWS.MM.SMS.TemplateId`

Dieses Attribut ist nur für das Senden von SMS-Nachrichten an Empfänger in Indien erforderlich.

Dies ist Ihre Vorlage für das Senden von SMS-Nachrichten an Empfänger in Indien. Diese ID ist eine eindeutige, von TRAI bereitgestellte Zeichenfolge mit 1 bis 50 Zeichen, die die Vorlage identifiziert, die Sie bei TRAI registriert haben. Die Vorlagen-ID muss der Sender-ID zugeordnet sein, die Sie für die Nachricht angegeben haben.

Senden einer Nachricht

In den folgenden Codebeispielen wird veranschaulicht, wie Sie SMS-Nachrichten über Amazon SNS veröffentlichen.

.NET

AWS SDK for .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

```
namespace SNSMessageExample
{
    using System;
    using System.Threading.Tasks;
    using Amazon;
    using Amazon.SimpleNotificationService;
    using Amazon.SimpleNotificationService.Model;

    public class SNSMessage
    {
        private AmazonSimpleNotificationServiceClient snsClient;

        /// <summary>
        /// Initializes a new instance of the <see cref="SNSMessage"/> class.
        /// Constructs a new SNSMessage object initializing the Amazon Simple
        /// Notification Service (Amazon SNS) client using the supplied
        /// Region endpoint.
        /// </summary>
        /// <param name="regionEndpoint">The Amazon Region endpoint to use in
        /// sending test messages with this object.</param>
        public SNSMessage(RegionEndpoint regionEndpoint)
        {
            snsClient = new
 AmazonSimpleNotificationServiceClient(regionEndpoint);
        }

        /// <summary>
```

```
        /// Sends the SMS message passed in the text parameter to the phone
 number
        /// in phoneNum.
        /// </summary>
        /// <param name="phoneNum">The ten-digit phone number to which the text
        /// message will be sent.</param>
        /// <param name="text">The text of the message to send.</param>
        /// <returns>Async task.</returns>
        public async Task SendTextMessageAsync(string phoneNum, string text)
        {
            if (string.IsNullOrEmpty(phoneNum) || string.IsNullOrEmpty(text))
            {
                return;
            }

            // Now actually send the message.
            var request = new PublishRequest
            {
                Message = text,
                PhoneNumber = phoneNum,
            };

            try
            {
                var response = await snsClient.PublishAsync(request);
            }
            catch (Exception ex)
            {
                Console.WriteLine($"Error sending message: {ex}");
            }
        }
    }
}
```

- Details zu API finden Sie unter Veröffentlichen in der AWS SDK for .NET -API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

```
/**
 * Publish SMS: use Amazon Simple Notification Service (Amazon SNS) to send an
 SMS text message to a phone number.
 * Note: This requires additional AWS configuration prior to running example.
 *
 *  NOTE: When you start using Amazon SNS to send SMS messages, your AWS account
 is in the SMS sandbox and you can only
 *  use verified destination phone numbers. See https://docs.aws.amazon.com/sns/
latest/dg/sns-sms-sandbox.html.
 *  NOTE: If destination is in the US, you also have an additional restriction
 that you have use a dedicated
 *  origination ID (phone number). You can request an origination number using
 Amazon Pinpoint for a fee.
 *  See https://aws.amazon.com/blogs/compute/provisioning-and-using-10dlc-
origination-numbers-with-amazon-sns/
 *  for more information.
 *
 *  <phone_number_value> input parameter uses E.164 format.
 *  For example, in United States, this input value should be of the form:
 +12223334444
 */

//! Send an SMS text message to a phone number.
/*!
  \param message: The message to publish.
  \param phoneNumber: The phone number of the recipient in E.164 format.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SNS::publishSms(const Aws::String &message,
                             const Aws::String &phoneNumber,
```

```
                                const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::SNS::Model::PublishRequest request;
    request.SetMessage(message);
    request.SetPhoneNumber(phoneNumber);

    const Aws::SNS::Model::PublishOutcome outcome = snsClient.Publish(request);

    if (outcome.IsSuccess()) {
        std::cout << "Message published successfully with message id, '"
                  << outcome.GetResult().GetMessageId() << "'."
                  << std::endl;
    }
    else {
        std::cerr << "Error while publishing message "
                  << outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Details zu API finden Sie unter Veröffentlichen in der AWS SDK for C++ -API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.PublishRequest;
import software.amazon.awssdk.services.sns.model.PublishResponse;
import software.amazon.awssdk.services.sns.model.SnsException;
```

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class PublishTextSMS {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <message> <phoneNumber>

                Where:
                   message - The message text to send.
                   phoneNumber - The mobile phone number to which a message is
 sent (for example, +1XXX5550100).\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String message = args[0];
        String phoneNumber = args[1];
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();
        pubTextSMS(snsClient, message, phoneNumber);
        snsClient.close();
    }

    public static void pubTextSMS(SnsClient snsClient, String message, String
 phoneNumber) {
        try {
            PublishRequest request = PublishRequest.builder()
                    .message(message)
                    .phoneNumber(phoneNumber)
                    .build();
```

```
            PublishResponse result = snsClient.publish(request);
            System.out
                    .println(result.messageId() + " Message sent. Status was " +
 result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Details zu API finden Sie unter Veröffentlichen in der AWS SDK for Java 2.x -API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

```
suspend fun pubTextSMS(messageVal: String?, phoneNumberVal: String?) {

    val request = PublishRequest {
        message = messageVal
        phoneNumber = phoneNumberVal
    }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.publish(request)
        println("${result.messageId} message sent.")
    }
}
```

- Details zu API finden Sie unter Publish (Veröffentlichen) in der AWS -SDK-für-Kotlin-API-Referenz.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Sends a text message (SMS message) directly to a phone number using Amazon
 SNS.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/
guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$message = 'This message is sent from a Amazon SNS code sample.';
$phone = '+1XXX5550100';

try {
    $result = $SnSclient->publish([
        'Message' => $message,
        'PhoneNumber' => $phone,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
```

```
    error_log($e->getMessage());
}
```

- Weitere Informationen finden Sie im AWS SDK for PHP -Entwicklerhandbuch.
- Details zu API finden Sie unter Veröffentlichen in der AWS SDK for PHP -API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

```
class SnsWrapper:
    """Encapsulates Amazon SNS topic and subscription functions."""

    def __init__(self, sns_resource):
        """
        :param sns_resource: A Boto3 Amazon SNS resource.
        """
        self.sns_resource = sns_resource


    def publish_text_message(self, phone_number, message):
        """
        Publishes a text message directly to a phone number without need for a
        subscription.

        :param phone_number: The phone number that receives the message. This
 must be
                             in E.164 format. For example, a United States phone
                             number might be +12065550101.
        :param message: The message to send.
        :return: The ID of the message.
        """
        try:
```

```
            response = self.sns_resource.meta.client.publish(
                PhoneNumber=phone_number, Message=message
            )
            message_id = response["MessageId"]
            logger.info("Published message to %s.", phone_number)
        except ClientError:
            logger.exception("Couldn't publish message to %s.", phone_number)
            raise
        else:
            return message_id
```

- Details zu API finden Sie unter Veröffentlichen in der AWS -API-Referenz zu SDK for Python (Boto3).

Überwachen der SMS-Aktivität

Durch die Überwachung der SMS-Aktivitäten behalten Sie Zieltelefonnummern, erfolgreiche oder fehlgeschlagene Zustellungen, Gründe für Ausfälle, Kosten und andere Informationen im Auge. Amazon SNS fasst die Statistiken in der Konsole zusammen und sendet Informationen an Amazon CloudWatch sowie tägliche SMS-Nutzungsberichte in einen Amazon S3-Bucket, den Sie angegeben haben.

Themen

- Anzeigen von SMS-Zustellungsstatistiken
- Anzeigen von Amazon CloudWatch-Metriken und Protokollen für SMS-Zustellungen
- Anzeigen von täglichen SMS-Nutzungsberichten

Anzeigen von SMS-Zustellungsstatistiken

Sie können Statistiken zu Ihren aktuellen SMS-Zustellungen über die Amazon SNS-Konsole anzeigen.

1. Melden Sie sich bei der Amazon SNS-Konsole an.
2. Legen Sie im Konsolenmenü unter der Regionsauswahl eine -Region fest, die SMS-Messaging unterstützt.
3. Wählen Sie im Navigationsbereich Text messaging (SMS) (Textnachrichten (SMS)) aus.

4. Zeigen Sie auf der Seite Text messaging (SMS) im Abschnitt Account stats (Kontostatistik) die Diagramme zu den Zustellungen Ihrer transaktions- und werbebezogenen SMS-Nachrichten an. Jedes Diagramm zeigt die folgenden Daten für die letzten 15 Tage:

   - Zustellungsrate (Prozentsatz der erfolgreichen Zustellungen)
   - Gesendet (Anzahl der Zustellversuche)
   - Fehlgeschlagen (Anzahl der fehlgeschlagenen Zustellungen)

Auf dieser Seite können Sie auch die Schaltfläche Usage (Nutzung) wählen, um zum Amazon S3-Bucket zu wechseln, in dem Ihre täglichen Nutzungsberichte gespeichert sind. Weitere Informationen finden Sie unter Anzeigen von täglichen SMS-Nutzungsberichten.

Anzeigen von Amazon CloudWatch-Metriken und Protokollen für SMS-Zustellungen

Sie können Amazon CloudWatch und Amazon CloudWatch Logs verwenden, um Ihre SMS-Nachrichtenzustellungen zu überwachen.

Themen

- Anzeigen von Amazon CloudWatch-Metriken
- Anzeigen von CloudWatch Logs
- Beispielprotokoll für eine erfolgreiche SMS-Zustellung
- Beispielprotokoll für eine fehlgeschlagene SMS-Zustellung
- Gründe für das Fehlschlagen der SMS-Zustellung

Anzeigen von Amazon CloudWatch-Metriken

Amazon SNS sammelt automatisch Metriken über Ihre SMS-Nachrichtenzustellungen und überträgt sie auf Amazon CloudWatch. Anhand von CloudWatch können Sie diese Metriken überwachen und Warnungen einrichten, die Sie warnen, wenn eine Metrik einen Schwellenwert überschreitet. Beispielsweise können Sie die CloudWatch-Metriken überwachen, um Ihre SMS-Zustellungsrate und Ihre SMS-Gebühren für den aktuellen Monat zu erfahren.

Weitere Informationen zur Überwachung von CloudWatch-Metriken, zum Einrichten von CloudWatch-Alarmen und über die verschiedenen Metrikarten finden Sie unter Überwachung von Amazon-SNS-Themen mit CloudWatch.

Anzeigen von CloudWatch Logs

Sie können Informationen zu erfolgreichen und fehlgeschlagenen SMS-Nachrichtenzustellungen sammeln, indem Sie Amazon SNS aktivieren, in Amazon CloudWatch Logs zu schreiben. Für jede SMS-Nachricht, die Sie senden, schreibt Amazon SNS ein Protokoll, das die Kosten für die Nachricht, den Erfolgs- oder Fehlerstatus, den Grund für das Fehlschlagen (wenn die Nachricht fehlgeschlagen ist), die Leerlaufzeit für die Nachricht und andere Informationen enthält.

So aktivieren und zeigen Sie CloudWatch Logs für Ihre SMS-Nachrichten an

1. Melden Sie sich bei der Amazon SNS-Konsole an.
2. Legen Sie im Konsolenmenü unter der Regionsauswahl eine -Region fest, die SMS-Messaging unterstützt.
3. Wählen Sie im Navigationsbereich Text messaging (SMS) (Textnachrichten (SMS)) aus.
4. Wählen Sie auf der Seite Mobile text messaging (SMS) (Mobile Texttnachrichten (SMS)) im Abschnitt Text messaging preferences (Textnachrichteneinstellungen) die Option Edit (Bearbeiten) aus.
5. Erweitern Sie auf der nächsten Seite den Abschnitt Protokollierung des Zustellungsstatus.
6. Geben Sie für Success sample rate (Erfolgsrate des Beispiels), den Prozentsatz der erfolgreichen SMS-Zustellungen an, für die Amazon SNS Protokolle in CloudWatch Logs schreibt. Beispiel:

   - Um nur Protokolle für fehlgeschlagene Zustellungen zu schreiben, setzen Sie diesen Wert auf 0.
   - Um Protokolle für 10 % der erfolgreichen Zustellungen zu schreiben, setzen Sie den Wert auf 10.

   Wenn Sie keinen Prozentsatz angeben, schreibt Amazon SNS Protokolle für alle erfolgreichen Zustellungen.
7. Gehen Sie wie folgt vor, um die angeforderten Genehmigungen bereitzustellen:

   - Zum Erstellen einer neuen Servicerolle wählen Sie Neue Servicerolle erstellen und dann Neue Rollen erstellen. Wählen Sie auf der nächsten Seite Erlauben, um Amazon SNS Schreibzugriff auf die Ressourcen Ihres Kontos zu geben.

- Um eine bestehende Servicerolle zu verwenden, wählen Sie Vorhandene Servicerolle verwenden und fügen Sie dann den ARN-Namen im Feld IAM-Rolle für erfolgreiche und fehlgeschlagene Übermittlungen.

  Die angegebene Servicerolle muss Schreibzugriff auf die Ressourcen Ihres Kontos ermöglichen. Weitere Informationen zum Erstellen von IAM-Rollen finden Sie unter Erstellen einer Rolle für einen AWS-Service im IAM Benutzerhandbuch.

8. Wählen Sie Save Changes.
9. Zurück auf der Seite Text-Messaging (SMS) gehen Sie zum Abschnitt Zustellungsstatusprotokolle, um verfügbare Protokolle anzuzeigen.

> **Note**
>
> Je nach Carrier der Zieltelefonnummer kann es bis zu 72 Stunden dauern, bis Zustellungsprotokolle in der Amazon-SNS-Konsole angezeigt werden.

Beispielprotokoll für eine erfolgreiche SMS-Zustellung

Das Zustellungsstatusprotokoll für eine erfolgreiche SMS-Zustellung ähnelt dem folgenden Beispiel:

```
{
    "notification": {
        "messageId": "34d9b400-c6dd-5444-820d-fbeb0f1f54cf",
        "timestamp": "2016-06-28 00:40:34.558"
    },
    "delivery": {
        "phoneCarrier": "My Phone Carrier",
        "mnc": 270,
        "numberOfMessageParts": 1,
        "destination": "+1XXX5550100",
        "priceInUSD": 0.00645,
        "smsType": "Transactional",
        "mcc": 310,
        "providerResponse": "Message has been accepted by phone carrier",
        "dwellTimeMs": 599,
        "dwellTimeMsUntilDeviceAck": 1344
    },
    "status": "SUCCESS"
}
```

Beispielprotokoll für eine fehlgeschlagene SMS-Zustellung

Das Zustellungsstatusprotokoll für eine fehlgeschlagene SMS-Zustellung ähnelt dem folgenden Beispiel:

```
{
    "notification": {
        "messageId": "1077257a-92f3-5ca3-bc97-6a915b310625",
        "timestamp": "2016-06-28 00:40:34.559"
    },
    "delivery": {
        "mnc": 0,
        "numberOfMessageParts": 1,
        "destination": "+1XXX5550100",
        "priceInUSD": 0.00645,
        "smsType": "Transactional",
        "mcc": 0,
        "providerResponse": "Unknown error attempting to reach phone",
        "dwellTimeMs": 1420,
        "dwellTimeMsUntilDeviceAck": 1692
    },
    "status": "FAILURE"
}
```

Gründe für das Fehlschlagen der SMS-Zustellung

Der Grund für das Fehlschlagen wird mit dem `providerResponse`-Attribut angegeben. SMS-Nachrichten können aus den folgenden Gründen fehlschlagen:

- Telefonnetzbetreiber hat die Nachricht als Spam blockiert
- Ziel befindet sich auf einer blockierten Liste
- Telefonnummer ist ungültig
- Nachrichtentext ist ungültig
- Telefonnentzbetreiber hat die Nachricht blockiert
- Telefonnetzbetreiber ist derzeit nicht erreichbar/nicht verfügbar
- Telefon blockiert SMS
- Telefon befindet sich auf einer blockierten Liste
- Telefon ist derzeit nicht erreichbar/nicht verfügbar
- Telefonnummer lehnt Empfang ab

- Zustellung würde den Maximalpreis überschreiten
- Unbekannter Fehler versucht das Telefon zu erreichen

Anzeigen von täglichen SMS-Nutzungsberichten

Sie können Ihre SMS-Zustellungen überwachen, indem Sie tägliche Nutzungsberichte von Amazon SNS abonnieren. Für jeden Tag, an dem Sie mindestens eine SMS-Nachricht senden, sendet Amazon SNS einen Nutzungsbericht als CSV-Datei an den angegebenen Amazon S3-Bucket. Es dauert 24 Stunden, bis der SMS-Nutzungsbericht im S3-Bucket verfügbar ist.

Themen

Informationen im täglichen Nutzungsbericht

Der Nutzungsbericht umfasst die folgenden Informationen für jede SMS-Nachricht, die Sie von Ihrem Konto senden.

Beachten Sie, dass der Bericht keine Nachrichten enthält, die an Empfänger gesendet werden, die sich vom Nachrichtenempfang abgemeldet haben.

- Zeitpunkt der Veröffentlichung für die Nachricht (in UTC)
- Nachrichten-ID
- Zieltelefonnummer
- Nachrichtentyp
- Zustellungsstatus
- Kosten für die Nachricht (in US-Dollar)
- Segmentnummer (eine Nachricht wird in mehrere Segmente aufgeteilt, wenn sie für eine einzelne Nachricht zu lang ist)
- Gesamtanzahl der Segmente

> **Note**
>
> Wenn Amazon SNS die Segmentnummer nicht erhalten hat, setzen wir deren Wert auf Null.

Abonnieren der täglichen Nutzungsberichte

Um tägliche Nutzungsberichte zu abonnieren, müssen Sie einen Amazon S3-Bucket mit den entsprechenden Berechtigungen erstellen.

So erstellen Sie einen Amazon-S3-Bucket für Ihre täglichen Nutzungsberichte

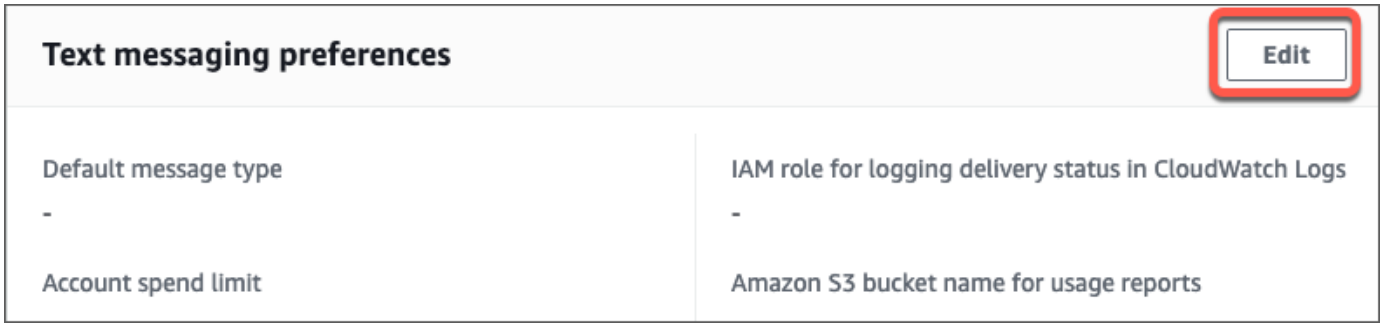
1. Melden Sie sich vom AWS-Konto, der SMS-Nachrichten sendet, in der Amazon S3-Konsole an.
2. Wählen Sie Create Bucket (Bucket erstellen) aus.
3. Unter Bucket Name (Bucket-Name) empfiehlt es sich, einen für Ihr Konto und Ihre Organisation eindeutigen Namen einzugeben. Verwenden Sie zum Beispiel das Muster `<my-bucket-prefix>-<account_id>-<org-id>`.

   Weitere Informationen über Konventionen und Einschränkungen für Bucket-Namen finden Sie unter Regeln für die Bucket-Benennung im Benutzerhandbuch zu Amazon Simple Storage Service.
4. Wählen Sie Create (Erstellen) aus.
5. Wählen Sie in der Tabelle Alle Buckets den Bucket aus.
6. Wählen Sie in der Registerkarte Genehmigungen Bucket-Richtlinie.
7. Geben Sie im Fenster Bucket Policy Editor (Bucket-Richtlinieneditor) eine Richtlinie ein, die den Amazon SNS-Serviceprinzipal berechtigt, in Ihren Bucket zu schreiben. Ein Beispiel finden Sie unter Beispiel einer Bucket-Richtlinie.

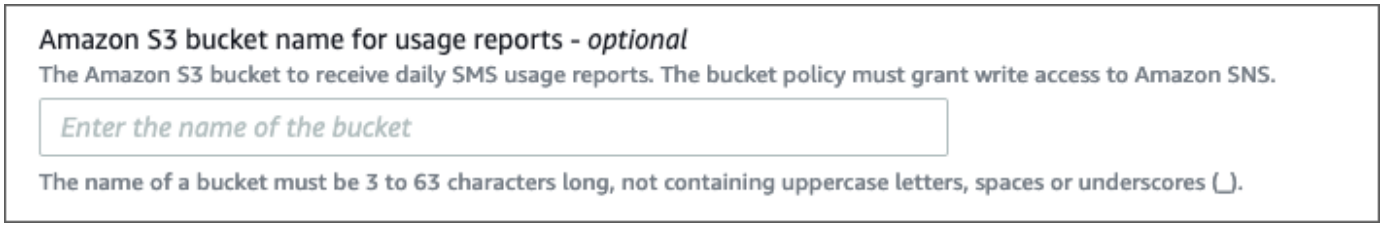
   Wenn Sie die Beispielrichtlinie verwenden, denken Sie daran, *`my-s3-bucket`* durch den Bucket-Namen zu ersetzen, den Sie in Schritt 3 gewählt haben.
8. Wählen Sie Save (Speichern) aus.

So abonnieren Sie tägliche Nutzungsberichte

1. Melden Sie sich bei der Amazon SNS-Konsole an.
2. Wählen Sie im Navigationsbereich Text messaging (SMS) (Textnachrichten (SMS)) aus.
3. Wählen Sie auf der Seite Text messaging (SMS) (Textnachrichten (SMS)) im Abschnitt Text messaging preferences (Textnachrichteneinstellungen) die Option Edit (Bearbeiten) aus.



4. Geben Sie auf der Seite Edit text messaging preferences (Textnachrichteneinstellungen bearbeiten) im Abschnitt Details Amazon S3 bucket name for usage reports (Amazon S3-Bucket-Name für Nutzungsberichte) an.



5. Wählen Sie Save Changes (Änderungen speichern) aus.

Beispiel einer Bucket-Richtlinie

Die folgende Richtlinie berechtigt den Amazon SNS-Service, die Aktionen `s3:PutObject`, `s3:GetBucketLocation` und `s3:ListBucket` durchzuführen.

AWS bietet Tools für alle Services mit Serviceprinzipalen, die Zugriff auf Ressourcen in Ihrem Konto erhalten haben. Wenn der Prinzipal in einer Amazon-S3-Bucket-Richtlinie ein AWS-Service-Prinzipal ist, können Sie `aws:SourceArn` oder den globalen Zustandsschlüssel `aws:SourceAccount` zum Schutz vor dem Problem des verwirrten Stellvertreters verwenden. Um zu begrenzen, welche Region und welches Konto der Bucket tägliche Nutzungsberichte erhalten kann, verwenden Sie `aws:SourceArn`, wie im folgenden Beispiel gezeigt. Wenn Sie die Regionen, die diese Berichte genieren können, nicht einschränken wollen, können Sie mit `aws:SourceAccount` einschränken, welche Konten die Berichte generieren. Wenn Sie den ARN der Ressource nicht kennen, verwenden Sie `aws:SourceAccount`.

Verwenden Sie das folgende Beispiel, das einen Schutz vor dem Problem des verwirrten Stellvertreters enthält, um eine Amazon-S3-Bucket für den Empfang von täglichen SMS-Nutzungsberichten von Amazon SNS zu erstellen.

```
{
```

```
 "Version": "2008-10-17",
 "Statement": [{
   "Sid": "AllowPutObject",
   "Effect": "Allow",
   "Principal": {
    "Service": "sns.amazonaws.com"
   },
   "Action": "s3:PutObject",
   "Resource": "arn:aws:s3:::my-s3-bucket/*",
   "Condition": {
    "StringEquals": {
     "aws:SourceAccount": "account_id"
    },
    "ArnLike": {
     "aws:SourceArn": "arn:aws:sns:region:account_id:*"
    }
   }
  },
  {
   "Sid": "AllowGetBucketLocation",
   "Effect": "Allow",
   "Principal": {
    "Service": "sns.amazonaws.com"
   },
   "Action": "s3:GetBucketLocation",
   "Resource": "arn:aws:s3:::my-s3-bucket",
   "Condition": {
    "StringEquals": {
     "aws:SourceAccount": "account_id"
    },
    "ArnLike": {
     "aws:SourceArn": "arn:aws:sns:region:account_id:*"
    }
   }
  },
  {
   "Sid": "AllowListBucket",
   "Effect": "Allow",
   "Principal": {
    "Service": "sns.amazonaws.com"
   },
   "Action": "s3:ListBucket",
   "Resource": "arn:aws:s3:::my-s3-bucket",
   "Condition": {
```

```
    "StringEquals": {
     "aws:SourceAccount": "account_id"
    },
    "ArnLike": {
     "aws:SourceArn": "arn:aws:sns:region:account_id:*"
    }
   }
  }
 ]
}
```

> **Note**
>
> Sie können Nutzungsberichte in Amazon-S3-Buckets veröffentlichen, die sich im Besitz von AWS-Konto befinden, die in der Amazon S3-Richtlinie im `Condition`-Element festgelegt sind. Für die Veröffentlichung von Nutzungsberichten an einen Amazon S3-Bucket, die einem anderen AWS-Konto gehören, siehe Wie kann ich S3-Objekte von einem anderen AWS-Konto kopieren?.

Beispiel eines täglichen Nutzungsberichts

Nachdem Sie die täglichen Nutzungsberichte abonniert haben, speichert Amazon SNS jeden Tag eine CSV-Datei mit Nutzungsdaten am folgenden Speicherort:

```
<my-s3-bucket>/SMSUsageReports/<region>/YYYY/MM/DD/00x.csv.gz
```

Jede Datei kann bis zu 50.000 Datensätze enthalten. Wenn die Berichte für einen Tag dieses Kontingent übersteigen, fügt Amazon SNS mehrere Dateien hinzu.

Nachstehend finden Sie einen Beispielbericht:

```
PublishTimeUTC,MessageId,DestinationPhoneNumber,MessageType,DeliveryStatus,PriceInUSD,PartNumbe
2016-05-10T03:00:29.476Z,96a298ac-1458-4825-
a7eb-7330e0720b72,1XXX5550100,Promotional,Message has been accepted by phone
 carrier,0.90084,0,1
2016-05-10T03:00:29.561Z,1e29d394-
d7f4-4dc9-996e-26412032c344,1XXX5550100,Promotional,Message has been accepted by phone
 carrier,0.34322,0,1
```

```
2016-05-10T03:00:30.769Z,98ba941c-afc7-4c51-
ba2c-56c6570a6c08,1XXX5550100,Transactional,Message has been accepted by phone
 carrier,0.27815,0,1
```

Verwalten von Telefonnummern und SMS-Abonnements

Amazon SNS bietet mehrere Optionen, mit denen Sie verwalten können, wer von Ihrem Konto SMS-Nachrichten erhält. Unter Berücksichtigung einer begrenzten Häufigkeit können Sie Telefonnummern, die den Empfang von SMS-Nachrichten von Ihrem Konto deaktiviert haben, hinzufügen. Um Nachrichten nicht länger an SMS-Abonnements zu senden, können Sie die Abonnements oder die Themen, die an diese Abonnements veröffentlichen, entfernen.

Themen

Deaktivieren des Empfangs von SMS-Nachrichten

Sofern dies aufgrund von Gesetzen und Vorschriften erforderlich ist (z. B. in den USA und Kanada), können SMS-Empfänger den Empfang der Nachrichten auf ihren Geräten deaktivieren, indem Sie auf die Nachricht folgendermaßen antworten:

- ARRET (Französisch)
- CANCEL
- END
- OPT-OUT
- OPTOUT
- QUIT
- REMOVE
- STOP
- TD
- UNSUBSCRIBE

Zum Deaktivieren muss der Empfänger auf die gleiche Ursprungsnummer antworten, die Amazon SNS verwendete, um die Nachricht zu versenden. Nach dem Abmelden erhält der Empfänger keine SMS-Nachrichten mehr von Ihnen, AWS-Konto es sei denn, Sie geben die Telefonnummer an.

Wenn die Telefonnummer ein Amazon SNS-Thema abonniert hat, wird das Abonnement selbst durch die Deaktivierung nicht entfernt, aber SMS-Nachrichten werden diesem Abonnement nicht länger zugestellt, es sei denn, Sie aktivieren die Telefonnummer erneut.

Verwalten von Telefonnummern und Abonnements (Konsole)

Sie können über die Amazon SNS-Konsole steuern, welche Telefonnummern SMS-Nachrichten von Ihrem Konto empfangen.

Aktivieren einer Telefonnummer, die den Empfang von SMS-Nachrichten deaktiviert hat

Sie können anzeigen, welche Telefonnummern den Empfang von SMS-Nachrichten von Ihrem Konto deaktiviert haben und diese Telefonnummern wieder aktivieren, um mit dem Senden von Nachrichten fortzufahren.

Sie können eine Telefonnummer nur einmal alle 30 Tage aktivieren.

1. Melden Sie sich bei der Amazon SNS-Konsole an.
2. Legen Sie im Konsolenmenü unter der Regionsauswahl eine -Region fest, die SMS-Messaging unterstützt.
3. Wählen Sie im Navigationsbereich Text messaging (SMS) (Textnachrichten (SMS)) aus.
4. Wählen Sie auf der Seite Text messaging (SMS) die Option View opted out phone numbers (Telefonnummern mit deaktiviertem Empfang anzeigen) aus. Die Seite Opted out phone numbers (Telefonnummern mit deaktiviertem Empfang) zeigt die Telefonnummern an, die den Empfang deaktiviert haben.
5. Markieren Sie das Kontrollkästchen für die Telefonnummer, die Sie aktivieren möchten, und wählen Sie Opt in (Anmelden). Die Telefonnummer ist nicht länger deaktiviert und empfängt erneut von Ihnen gesendete SMS-Nachrichten.

Löschen eines SMS-Abonnements

Löschen Sie ein SMS-Abonnement, um nicht länger SMS-Nachrichten über Veröffentlichungen zu Ihren Themen an diese Telefonnummer zu senden.

1. Wählen Sie im Navigationsbereich Subscriptions (Abonnements) aus.

2. Aktivieren Sie die Kontrollkästchen für die Abonnements, die Sie löschen möchten. Wählen Sie anschließend Actions (Aktionen) und Delete Subscriptions (Abonnements löschen) aus.
3. Wählen Sie im Fenster Löschen Löschen aus. Amazon SNS löscht das Abonnement und zeigt eine Erfolgsmeldung an.

Löschen eines Themas

Löschen Sie ein Thema, wenn Sie nicht länger Nachrichten an die Endpunkte veröffentlichen möchten, die dieses Thema abonniert haben.

1. Wählen Sie im Navigationsbereich Topics (Themen) aus.
2. Aktivieren Sie die Kontrollkästchen der Themen, die Sie löschen möchten. Wählen Sie anschließend Actions (Aktionen) und Delete Topics (Themen löschen) aus.
3. Wählen Sie im Fenster Löschen Löschen aus. Amazon SNS löscht das Thema und zeigt eine Erfolgsmeldung an.

Verwalten von Telefonnummern und Abonnements (AWS SDKs)

Sie können die AWS SDKs verwenden, um programmatische Anfragen an Amazon SNS zu stellen und zu verwalten, welche Telefonnummern SMS-Nachrichten von Ihrem Konto empfangen können.

Um ein AWS SDK verwenden zu können, müssen Sie es mit Ihren Anmeldeinformationen konfigurieren. Weitere Informationen finden Sie im Referenzhandbuch für AWS SDKs und Tools unter Dateien mit gemeinsam genutzten Konfigurationen und Anmeldeinformationen.

Anzeigen aller Telefonnummern, die den Empfang deaktiviert haben

Um alle Telefonnummern anzuzeigen, die den Empfang deaktiviert haben, senden Sie eine `ListPhoneNumbersOptedOut` Anfrage über die Amazon SNS-API.

Die folgenden Codebeispiele zeigen die Verwendung`ListPhoneNumbersOptedOut`.

CLI

AWS CLI

So führen Sie Abmeldungen für SMS-Nachrichten auf

Das folgende `list-phone-numbers-opted-out`-Beispiel listet die Telefonnummern auf, bei denen der Empfang von SMS-Nachrichten abbestellt wurde.

```
aws sns list-phone-numbers-opted-out
```

Ausgabe:

```
{
    "phoneNumbers": [
        "+15555550100"
    ]
}
```

- Einzelheiten zur API finden Sie ListPhoneNumbersOptedOutin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.ListPhoneNumbersOptedOutRequest;
import
 software.amazon.awssdk.services.sns.model.ListPhoneNumbersOptedOutResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class ListOptOut {
```

```
    public static void main(String[] args) {
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        listOpts(snsClient);
        snsClient.close();
    }

    public static void listOpts(SnsClient snsClient) {
        try {
            ListPhoneNumbersOptedOutRequest request =
 ListPhoneNumbersOptedOutRequest.builder().build();
            ListPhoneNumbersOptedOutResponse result =
 snsClient.listPhoneNumbersOptedOut(request);
            System.out.println("Status is " +
 result.sdkHttpResponse().statusCode() + "\n\nPhone Numbers: \n\n"
                    + result.phoneNumbers());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Einzelheiten zur API finden Sie ListPhoneNumbersOptedOutin der AWS SDK for Java 2.x API-Referenz.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

```
require 'vendor/autoload.php';
```

```
use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Returns a list of phone numbers that are opted out of receiving SMS messages
 from your AWS SNS account.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/
guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

try {
    $result = $SnSclient->listPhoneNumbersOptedOut();
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- Weitere Informationen finden Sie im AWS SDK for PHP -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie ListPhoneNumbersOptedOutin der AWS SDK for PHP API-Referenz.

Überprüfen, ob eine Telefonnummer den Empfang deaktiviert hat

Um zu erfahren, ob eine Telefonnummer den Empfang deaktiviert hat, senden Sie eine `CheckIfPhoneNumberIsOptedOut` Anfrage über die Amazon SNS-API.

Die folgenden Codebeispiele zeigen die Verwendung`CheckIfPhoneNumberIsOptedOut`.

.NET

AWS SDK for .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

```
    using System;
    using System.Threading.Tasks;
    using Amazon.SimpleNotificationService;
    using Amazon.SimpleNotificationService.Model;

    /// <summary>
    /// This example shows how to use the Amazon Simple Notification Service
    /// (Amazon SNS) to check whether a phone number has been opted out.
    /// </summary>
    public class IsPhoneNumOptedOut
    {
        public static async Task Main()
        {
            string phoneNumber = "+15551112222";

            IAmazonSimpleNotificationService client = new
AmazonSimpleNotificationServiceClient();

            await CheckIfOptedOutAsync(client, phoneNumber);
        }

        /// <summary>
        /// Checks to see if the supplied phone number has been opted out.
        /// </summary>
        /// <param name="client">The initialized Amazon SNS Client object used
        /// to check if the phone number has been opted out.</param>
        /// <param name="phoneNumber">A string representing the phone number
        /// to check.</param>
        public static async Task
CheckIfOptedOutAsync(IAmazonSimpleNotificationService client, string
phoneNumber)
        {
```

```
            var request = new CheckIfPhoneNumberIsOptedOutRequest
            {
                PhoneNumber = phoneNumber,
            };

            try
            {
                var response = await
client.CheckIfPhoneNumberIsOptedOutAsync(request);

                if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
                {
                    string optOutStatus = response.IsOptedOut ? "opted out" :
"not opted out.";
                    Console.WriteLine($"The phone number: {phoneNumber} is
{optOutStatus}");
                }
            }
            catch (AuthorizationErrorException ex)
            {
                Console.WriteLine($"{ex.Message}");
            }
        }
    }
```

- Einzelheiten zur API finden Sie CheckIfPhoneNumberIsOptedOutin der AWS SDK for .NET API-Referenz.

CLI

AWS CLI

So überprüfen Sie SMS-Nachrichten-Abmeldungen für eine Telefonnummer

Im folgenden `check-if-phone-number-is-opted-out` Beispiel wird geprüft, ob die angegebene Telefonnummer den Empfang von SMS-Nachrichten vom AWS Girokonto deaktiviert hat.

```
aws sns check-if-phone-number-is-opted-out \
    --phone-number +1555550100
```

Ausgabe:

```
{
    "isOptedOut": false
}
```

- Einzelheiten zur API finden Sie CheckIfPhoneNumberIsOptedOutunter AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import
 software.amazon.awssdk.services.sns.model.CheckIfPhoneNumberIsOptedOutRequest;
import
 software.amazon.awssdk.services.sns.model.CheckIfPhoneNumberIsOptedOutResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class CheckOptOut {
    public static void main(String[] args) {

        final String usage = """
```

```
                Usage:    <phoneNumber>

                Where:
                   phoneNumber - The mobile phone number to look up (for example,
 +1XXX5550100).

                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String phoneNumber = args[0];
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        checkPhone(snsClient, phoneNumber);
        snsClient.close();
    }

    public static void checkPhone(SnsClient snsClient, String phoneNumber) {
        try {
            CheckIfPhoneNumberIsOptedOutRequest request =
 CheckIfPhoneNumberIsOptedOutRequest.builder()
                    .phoneNumber(phoneNumber)
                    .build();

            CheckIfPhoneNumberIsOptedOutResponse result =
 snsClient.checkIfPhoneNumberIsOptedOut(request);
            System.out.println(
                    result.isOptedOut() + "Phone Number " + phoneNumber + " has
 Opted Out of receiving sns messages." +
                            "\n\nStatus was " +
 result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Einzelheiten zur API finden Sie CheckIfPhoneNumberIsOptedOutin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie den Client in einem separaten Modul und exportieren Sie ihn.

```
import { SNSClient } from "@aws-sdk/client-sns";

// The AWS Region can be provided here using the `region` property. If you leave
 it blank
// the SDK will default to the region set in your AWS config.
export const snsClient = new SNSClient({});
```

Importieren Sie das SDK- und Client-Module und rufen Sie die API auf.

```
import { CheckIfPhoneNumberIsOptedOutCommand } from "@aws-sdk/client-sns";

import { snsClient } from "../libs/snsClient.js";

export const checkIfPhoneNumberIsOptedOut = async (
  phoneNumber = "5555555555",
) => {
  const command = new CheckIfPhoneNumberIsOptedOutCommand({
    phoneNumber,
  });

  const response = await snsClient.send(command);
  console.log(response);
  // {
```

```
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: '3341c28a-cdc8-5b39-a3ee-9fb0ee125732',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   isOptedOut: false
  // }
  return response;
};
```

- Weitere Informationen finden Sie im AWS SDK for JavaScript -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie CheckIfPhoneNumberIsOptedOutin der AWS SDK for JavaScript API-Referenz.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Indicates whether the phone number owner has opted out of receiving SMS
 messages from your AWS SNS account.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/
guide_credentials.html
```

```
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$phone = '+1XXX5550100';

try {
    $result = $SnSclient->checkIfPhoneNumberIsOptedOut([
        'phoneNumber' => $phone,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- Weitere Informationen finden Sie im AWS SDK for PHP -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie CheckIfPhoneNumberIsOptedOutin der AWS SDK for PHP API-Referenz.

Aktivieren einer Telefonnummer, die den Empfang von SMS-Nachrichten deaktiviert hat

Um eine Telefonnummer zu aktivieren, senden Sie eine `OptInPhoneNumber` Anfrage über die Amazon SNS-API.

Sie können eine Telefonnummer nur einmal alle 30 Tage aktivieren.

Löschen eines SMS-Abonnements

Um ein SMS-Abonnement von einem Amazon SNS-Thema zu löschen, müssen Sie zunächst anhand einer `ListSubscriptions`-Anfrage über das Amazon SNS-API den ARN des Abonnements abrufen und anschließend den ARN einer `Unsubscribe`-Anfrage übergeben.

Die folgenden Codebeispiele zeigen die Verwendung`Unsubscribe`.

.NET

AWS SDK for .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

Melden Sie sich mit einem Abonnement-ARN von einem Thema ab.

```
/// <summary>
/// Unsubscribe from a topic by a subscription ARN.
/// </summary>
/// <param name="subscriptionArn">The ARN of the subscription.</param>
/// <returns>True if successful.</returns>
public async Task<bool> UnsubscribeByArn(string subscriptionArn)
{
    var unsubscribeResponse = await _amazonSNSClient.UnsubscribeAsync(
        new UnsubscribeRequest()
        {
            SubscriptionArn = subscriptionArn
        });
    return unsubscribeResponse.HttpStatusCode == HttpStatusCode.OK;
}
```

- Details zu API finden Sie unter Abmelden in der AWS SDK for .NET -API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

```
//! Delete a subscription to an Amazon Simple Notification Service (Amazon SNS)
 topic.
/*!
  \param subscriptionARN: The Amazon Resource Name (ARN) for an Amazon SNS topic
 subscription.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SNS::unsubscribe(const Aws::String &subscriptionARN,
                              const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::SNS::Model::UnsubscribeRequest request;
    request.SetSubscriptionArn(subscriptionARN);

    const Aws::SNS::Model::UnsubscribeOutcome outcome =
 snsClient.Unsubscribe(request);

    if (outcome.IsSuccess()) {
        std::cout << "Unsubscribed successfully " << std::endl;
    }
    else {
        std::cerr << "Error while unsubscribing " <<
 outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Details zu API finden Sie unter Abmelden in der AWS SDK for C++ -API-Referenz.

CLI

AWS CLI

So melden Sie sich von einem Thema ab

Im folgenden `unsubscribe`-Beispiel wird das angegebene Abonnement aus einem Thema gelöscht.

```
aws sns unsubscribe \
    --subscription-arn arn:aws:sns:us-west-2:0123456789012:my-
topic:8a21d249-4329-4871-acc6-7be709c6ea7f
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

- API-Details finden Sie unter Unsubscribe in der AWS CLI -Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SnsException;
import software.amazon.awssdk.services.sns.model.UnsubscribeRequest;
import software.amazon.awssdk.services.sns.model.UnsubscribeResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class Unsubscribe {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <subscriptionArn>

                Where:
                   subscriptionArn - The ARN of the subscription to delete.
```

```
                """;

        if (args.length < 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String subscriptionArn = args[0];
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        unSub(snsClient, subscriptionArn);
        snsClient.close();
    }

    public static void unSub(SnsClient snsClient, String subscriptionArn) {
        try {
            UnsubscribeRequest request = UnsubscribeRequest.builder()
                    .subscriptionArn(subscriptionArn)
                    .build();

            UnsubscribeResponse result = snsClient.unsubscribe(request);
            System.out.println("\n\nStatus was " +
 result.sdkHttpResponse().statusCode()
                    + "\n\nSubscription was removed for " +
 request.subscriptionArn());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Details zu API finden Sie unter Abmelden in der AWS SDK for Java 2.x -API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie den Client in einem separaten Modul und exportieren Sie ihn.

```
import { SNSClient } from "@aws-sdk/client-sns";

// The AWS Region can be provided here using the `region` property. If you leave
 it blank
// the SDK will default to the region set in your AWS config.
export const snsClient = new SNSClient({});
```

Importieren Sie das SDK- und Client-Module und rufen Sie die API auf.

```
import { UnsubscribeCommand } from "@aws-sdk/client-sns";
import { snsClient } from "../libs/snsClient.js";

/**
 * @param {string} subscriptionArn - The ARN of the subscription to cancel.
 */
const unsubscribe = async (
  subscriptionArn = "arn:aws:sns:us-east-1:xxxxxxxxxxxx:mytopic:xxxxxxxx-xxxx-
xxxx-xxxx-xxxxxxxxxxxx",
) => {
  const response = await snsClient.send(
    new UnsubscribeCommand({
      SubscriptionArn: subscriptionArn,
    }),
  );
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: '0178259a-9204-507c-b620-78a7570a44c6',
```

```
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   }
  // }
  return response;
};
```

- Weitere Informationen finden Sie im AWS SDK for JavaScript -Entwicklerhandbuch.
- Details zu API finden Sie unter Abmelden in der AWS SDK for JavaScript -API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

```
suspend fun unSub(subscriptionArnVal: String) {

    val request = UnsubscribeRequest {
        subscriptionArn = subscriptionArnVal
    }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        snsClient.unsubscribe(request)
        println("Subscription was removed for ${request.subscriptionArn}")
    }
}
```

- Details zu API finden Sie unter Unsubscribe (Abmelden) in der AWS -SDK-für-Kotlin-API-Referenz.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Deletes a subscription to an Amazon SNS topic.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/
guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$subscription = 'arn:aws:sns:us-east-1:111122223333:MySubscription';

try {
    $result = $SnSclient->unsubscribe([
        'SubscriptionArn' => $subscription,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- Weitere Informationen finden Sie im AWS SDK for PHP -Entwicklerhandbuch.
- Details zu API finden Sie unter Abmelden in der AWS SDK for PHP -API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

```
class SnsWrapper:
    """Encapsulates Amazon SNS topic and subscription functions."""

    def __init__(self, sns_resource):
        """
        :param sns_resource: A Boto3 Amazon SNS resource.
        """
        self.sns_resource = sns_resource


    @staticmethod
    def delete_subscription(subscription):
        """
        Unsubscribes and deletes a subscription.
        """
        try:
            subscription.delete()
            logger.info("Deleted subscription %s.", subscription.arn)
        except ClientError:
            logger.exception("Couldn't delete subscription %s.",
 subscription.arn)
            raise
```

- Details zu API finden Sie unter Abmelden in der AWS API-Referenz zu SDK for Python (Boto3).

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

```
    TRY.
        lo_sns->unsubscribe( iv_subscriptionarn = iv_subscription_arn ).
        MESSAGE 'Subscription deleted.' TYPE 'I'.
      CATCH /aws1/cx_snsnotfoundexception.
        MESSAGE 'Subscription does not exist.' TYPE 'E'.
      CATCH /aws1/cx_snsinvalidparameterex.
        MESSAGE 'Subscription with "PendingConfirmation" status cannot be
deleted/unsubscribed. Confirm subscription before performing unsubscribe
operation.' TYPE 'E'.
    ENDTRY.
```

- Details zu API finden Sie unter Abmelden in der API-Referenz für das AWS SDK für SAP ABAP.

Löschen eines Themas

Um ein Thema und alle seine Abonnements zu löschen, müssen Sie zunächst anhand einer `ListTopics`-Anfrage über die Amazon SNS-API den ARN des Abonnements einholen und anschließend den ARN einer `DeleteTopic`-Anfrage übergeben.

Die folgenden Codebeispiele zeigen, wie man es benutzt`DeleteTopic`.

.NET

AWS SDK for .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

Löschen Sie ein Thema mit seinem Themen-ARN.

```
/// <summary>
/// Delete a topic by its topic ARN.
/// </summary>
/// <param name="topicArn">The ARN of the topic.</param>
/// <returns>True if successful.</returns>
public async Task<bool> DeleteTopicByArn(string topicArn)
{
    var deleteResponse = await _amazonSNSClient.DeleteTopicAsync(
        new DeleteTopicRequest()
        {
            TopicArn = topicArn
        });
    return deleteResponse.HttpStatusCode == HttpStatusCode.OK;
}
```

- Einzelheiten zur API finden Sie DeleteTopicin der AWS SDK for .NET API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

```
//! Delete an Amazon Simple Notification Service (Amazon SNS) topic.
/*!
  \param topicARN: The Amazon Resource Name (ARN) for an Amazon SNS topic.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SNS::deleteTopic(const Aws::String &topicARN,
                              const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::SNS::Model::DeleteTopicRequest request;
    request.SetTopicArn(topicARN);

    const Aws::SNS::Model::DeleteTopicOutcome outcome =
 snsClient.DeleteTopic(request);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully deleted the Amazon SNS topic " << topicARN <<
 std::endl;
    }
    else {
        std::cerr << "Error deleting topic " << topicARN << ":" <<
                  outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Einzelheiten zur API finden Sie DeleteTopicin der AWS SDK for C++ API-Referenz.

CLI

AWS CLI

So löschen Sie das SNS-Thema

Das folgende `delete-topic`-Beispiel löscht die angegebene SNS-Thema.

```
aws sns delete-topic \
    --topic-arn "arn:aws:sns:us-west-2:123456789012:my-topic"
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

- Einzelheiten zur API finden Sie DeleteTopicin der AWS CLI Befehlsreferenz.

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

```
// SnsActions encapsulates the Amazon Simple Notification Service (Amazon SNS)
 actions
// used in the examples.
type SnsActions struct {
 SnsClient *sns.Client
}



// DeleteTopic delete an Amazon SNS topic.
func (actor SnsActions) DeleteTopic(topicArn string) error {
 _, err := actor.SnsClient.DeleteTopic(context.TODO(), &sns.DeleteTopicInput{
  TopicArn: aws.String(topicArn)})
 if err != nil {
  log.Printf("Couldn't delete topic %v. Here's why: %v\n", topicArn, err)
 }
 return err
}
```

- Einzelheiten zur API finden Sie DeleteTopicin der AWS SDK for Go API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.DeleteTopicRequest;
import software.amazon.awssdk.services.sns.model.DeleteTopicResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DeleteTopic {
    public static void main(String[] args) {
        final String usage = """

                Usage:     <topicArn>

                Where:
                   topicArn - The ARN of the topic to delete.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String topicArn = args[0];
        SnsClient snsClient = SnsClient.builder()
```

```
                .region(Region.US_EAST_1)
                .build();

        System.out.println("Deleting a topic with name: " + topicArn);
        deleteSNSTopic(snsClient, topicArn);
        snsClient.close();
    }

    public static void deleteSNSTopic(SnsClient snsClient, String topicArn) {
        try {
            DeleteTopicRequest request = DeleteTopicRequest.builder()
                    .topicArn(topicArn)
                    .build();

            DeleteTopicResponse result = snsClient.deleteTopic(request);
            System.out.println("\n\nStatus was " +
 result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Einzelheiten zur API finden Sie DeleteTopicin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie den Client in einem separaten Modul und exportieren Sie ihn.

```
import { SNSClient } from "@aws-sdk/client-sns";
```

```
// The AWS Region can be provided here using the `region` property. If you leave
 it blank
// the SDK will default to the region set in your AWS config.
export const snsClient = new SNSClient({});
```

Importieren Sie das SDK- und Client-Module und rufen Sie die API auf.

```
import { DeleteTopicCommand } from "@aws-sdk/client-sns";
import { snsClient } from "../libs/snsClient.js";

/**
 * @param {string} topicArn - The ARN of the topic to delete.
 */
export const deleteTopic = async (topicArn = "TOPIC_ARN") => {
  const response = await snsClient.send(
    new DeleteTopicCommand({ TopicArn: topicArn }),
  );
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: 'a10e2886-5a8f-5114-af36-75bd39498332',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   }
  // }
};
```

- Weitere Informationen finden Sie im AWS SDK for JavaScript -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie DeleteTopicin der AWS SDK for JavaScript API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

```
suspend fun deleteSNSTopic(topicArnVal: String) {

    val request = DeleteTopicRequest {
        topicArn = topicArnVal
    }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        snsClient.deleteTopic(request)
        println("$topicArnVal was successfully deleted.")
    }
}
```

- API-Details finden Sie DeleteTopicin der API-Referenz zum AWS SDK für Kotlin.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;
```

```
/**
 * Deletes an SNS topic and all its subscriptions.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/
guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$topic = 'arn:aws:sns:us-east-1:111122223333:MyTopic';

try {
    $result = $SnSclient->deleteTopic([
        'TopicArn' => $topic,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- Einzelheiten zur API finden Sie DeleteTopicin der AWS SDK for PHP API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

```
class SnsWrapper:
```

```
    """Encapsulates Amazon SNS topic and subscription functions."""

    def __init__(self, sns_resource):
        """
        :param sns_resource: A Boto3 Amazon SNS resource.
        """
        self.sns_resource = sns_resource


    @staticmethod
    def delete_topic(topic):
        """
        Deletes a topic. All subscriptions to the topic are also deleted.
        """
        try:
            topic.delete()
            logger.info("Deleted topic %s.", topic.arn)
        except ClientError:
            logger.exception("Couldn't delete topic %s.", topic.arn)
            raise
```

- Einzelheiten zur API finden Sie DeleteTopicin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

```
    TRY.
        lo_sns->deletetopic( iv_topicarn = iv_topic_arn ).
        MESSAGE 'SNS topic deleted.' TYPE 'I'.
      CATCH /aws1/cx_snsnotfoundexception.
        MESSAGE 'Topic does not exist.' TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie DeleteTopicin der API-Referenz zum AWS SDK für SAP ABAP.

Unterstützte Länder und Regionen

> **Important**
>
> Ab dem 31. August 2023 ist es eine spezielle Nummer wie eine 10DLC-Nummer oder eine gebührenfreie Nummer erforderlich, um in den Vereinigten Staaten und ihren Territorien (Guam, Puerto Rico, Amerikanisch-Samoa-Inseln und den US Jungferninseln) SMS-Nachrichten zu versenden.

Derzeit unterstützt Amazon SNS SMS-Nachrichten in den folgenden AWS Regionen:

Name der Region	Region	Endpunkt	Protokoll
USA Ost (Ohio)	us-east-2	sns.us-east-2.amazonaws.com	HTTP und HTTPS
USA Ost (Nord-Virginia)	us-east-1	sns.us-east-1.amazonaws.com	HTTP und HTTPS
USA West (Nordkalifornien)	us-west-1	sns.us-west-1.amazonaws.com	HTTP und HTTPS
USA West (Oregon)	us-west-2	sns.us-west-2.amazonaws.com	HTTP und HTTPS
Afrika (Kapstadt)	af-south-1	sns.af-south-1.amazonaws.com	HTTP und HTTPS
Asien-Pazifik (Hyderabad)	ap-south-2	sns.ap-south-2.amazonaws.com	HTTP und HTTPS

Name der Region	Region	Endpunkt	Protokoll
Asien-Pazifik (Jakarta)	ap-southeast-3	sns.ap-southeast-3.amazonaws.com	HTTP und HTTPS
Asien-Pazifik (Melbourne)	ap-southeast-4	sns.ap-southeast-4.amazonaws.com	HTTP und HTTPS
Asien-Pazifik (Mumbai)	ap-south-1	sns.ap-south-1.amazonaws.com	HTTP und HTTPS
Asien-Pazifik (Osaka)	ap-northeast-3	sns.ap-northeast-3.amazonaws.com	HTTP und HTTPS
Asien-Pazifik (Singapur)	ap-southeast-1	sns.ap-southeast-1.amazonaws.com	HTTP und HTTPS
Asien-Pazifik (Sydney)	ap-southeast-2	sns.ap-southeast-2.amazonaws.com	HTTP und HTTPS
Asien-Pazifik (Tokio)	ap-northeast-1	sns.ap-northeast-1.amazonaws.com	HTTP und HTTPS
Kanada (Zentral)	ca-central-1	sns.ca-central-1.amazonaws.com	HTTP und HTTPS
Europa (Frankfurt)	eu-central-1	sns.eu-central-1.amazonaws.com	HTTP und HTTPS
Europa (Irland)	eu-west-1	sns.eu-west-1.amazonaws.com	HTTP und HTTPS
Europa (London)	eu-west-2	sns.eu-west-2.amazonaws.com	HTTP und HTTPS
Europa (Mailand)	eu-south-1	sns.eu-south-1.amazonaws.com	HTTP und HTTPS
Europa (Paris)	eu-west-3	sns.eu-west-3.amazonaws.com	HTTP und HTTPS

Name der Region	Region	Endpunkt	Protokoll
Europa (Spanien)	eu-south-2	sns.eu-south-2.amazonaws.com	HTTP und HTTPS
Europa (Stockholm)	eu-north-1	sns.eu-north-1.amazonaws.com	HTTP und HTTPS
Europa (Zürich)	eu-central-2	sns.eu-central-2.amazonaws.com	HTTP und HTTPS
Israel (Tel Aviv)	il-central-1	sns.il-central-1.amazonaws.com	HTTP und HTTPS
Naher Osten (Bahrain)	me-south-1	sns.me-south-1.amazonaws.com	HTTP und HTTPS
Naher Osten (VAE)	me-central-1	sns.me-central-1.amazonaws.com	HTTP und HTTPS
Südamerika (São Paulo)	sa-east-1	sns.sa-east-1.amazonaws.com	HTTP und HTTPS
AWS GovCloud (USA-Ost)	us-gov-east-1	sns. us-gov-east-1.amazonaws.com	HTTP und HTTPS
AWS GovCloud (US-West)	us-gov-west-1	sns. us-gov-west-1.amazonaws.com	HTTP und HTTPS

Sie können Amazon SNS zum Senden von SMS-Nachrichten an die folgenden Länder und Regionen verwenden:

> **Note**
>
> Die Verwendung von Sender-IDs in unterstützten Ländern kann die SMS-Zustellung verbessern.

Land oder Region	ISO-Code	Ortsvorwahl	Unterstützt Kurzwahlnummern	Unterstützt Langwahlnummern	Unterstützt Sender-IDs	Unterstützt Zwei-Wege-SMS
A						
Afghanistan	AF	93	Nein	Nein	Ja	Nein
Albanien	AL	355	Nein	Nein	Ja	Nein
Algerien	DZ	213	Nein	Nein	Ja	Nein
Andorra	AD	376	Nein	Nein	Ja	Nein
Anguilla	AI	1-264	Nein	Nein	Ja	Nein
Antigua und Barbuda	AG	1-268	Nein	Nein	Ja	Nein
Argentinien	AR	54	Ja	Nein	Nein	Nein
Armenien	AM	374	Nein	Nein	Ja	Nein
Aruba	AW	297	Nein	Nein	Ja	Nein
Australien	AU	61	Nein	Ja	Registrierung erforderlich[1]	Ja
Österreich	AT	43	Ja	Ja	Ja	Ja
Aserbaidschan	AZ	994	Nein	Nein	Ja	Nein
B						
Bahamas	BS	1-242	Nein	Nein	Nein	Nein

Land oder Region	ISO-Code	Ortsvorwahl	Unterstützt Kurzwahlnummern	Unterstützt Langwahlnummern	Unterstützt Sender-IDs	Unterstützt Zwei-Wege-SMS
Bahrain	BH	973	Nein	Nein	Ja	Nein
Bangladesch	BD	880	Nein	Nein	Ja	Nein
Barbados	BB	1-246	Nein	Nein	Ja	Nein
Belarus	BY	375	Nein	Nein	Registrierung erforderlich[1]	Nein
Belgien	BE	32	Nein	Ja	Nein	Ja
Belize	BZ	501	Nein	Nein	Ja	Nein
Bermuda	BM	1-441	Nein	Nein	Ja	Nein
Bhutan	BT	975	Nein	Nein	Ja	Nein
Bolivien	BO	591	Nein	Nein	Ja	Nein
Bosnien und Herzegowina	BA	387	Nein	Nein	Ja	Nein
Botswana	BW	267	Nein	Nein	Ja	Nein
Brasilien	BR	55	Ja	Nein	Nein	Ja
Brunei	BN	673	Nein	Nein	Ja	Nein
Bulgarien	BG	359	Ja	Nein	Ja	Ja

Land oder Region	ISO-Code	Ortsvorwahl	Unterstützt Kurzwahlnummern	Unterstützt Langwahlnummern	Unterstützt Sender-IDs	Unterstützt Zwei-Wege-SMS
Burkina Faso	BF	226	Nein	Nein	Ja	Nein
Burundi	BL	257	Nein	Nein	Ja	Nein
C						
Kambodscha	KH	855	Nein	Nein	Ja	Nein
Kamerun	CM	237	Nein	Nein	Ja	Nein
Kanada	CA	1	Ja	Ja	Nein	Ja
Kap Verde	CV	238	Nein	Nein	Ja	Nein
Kaimaninseln	KY	1-345	Nein	Nein	Nein	Nein
Zentralafrikanische Republik	CF	236	Nein	Nein	Ja	Nein
Tschad	TD	235	Nein	Nein	Ja	Nein
Chile	CL	56	Ja	Ja	Nein	Ja
China	CN	86	Ja	Nein	Nein [2]	Ja
Kolumbien	CO	57	Ja	Ja	Nein	Ja
Komoren	KM	269	Nein	Nein	Ja	Nein
Cookinseln	CK	682	Nein	Nein	Ja	Ja
Costa Rica	CR	506	Nein	Nein	Nein	Nein

Land oder Region	ISO-Code	Ortsvorwahl	Unterstützt Kurzwahlnummern	Unterstützt Langwahlnummern	Unterstützt Sender-IDs	Unterstützt Zwei-Wege-SMS
Kroatien	HR	385	Nein	Nein	Ja	Nein
Zypern	CY	357	Nein	Nein	Ja	Nein
Tschechien (Tschechische Republik)	CZ	420	Nein	Ja	Ja	Ja
D						
Demokratische Republik Kongo	CD	243	Nein	Nein	Ja	Nein
Dänemark	DK	45	Ja	Ja	Ja	Ja
Dschibuti	DJ	253	Nein	Nein	Ja	Nein
Dominica	DM	1-767	Nein	Nein	Ja	Nein
Dominikanische Republik	DO	1-809, 1-829, 1-849	Ja	Nein	Nein	Ja
E						
Ecuador	EC	593	Ja	Nein	Nein	Ja
Ägypten	EG	20	Ja	Nein	Registrierung erforderlich[1]	Ja
El Salvador	SV	503	Nein	Nein	Nein	Nein

Land oder Region	ISO-Code	Ortsvorwahl	Unterstützt Kurzwahlnummern	Unterstützt Langwahlnummern	Unterstützt Sender-IDs	Unterstützt Zwei-Wege-SMS
Äquatorialguinea	GQ	240	Nein	Nein	Ja	Nein
Eritrea	ER	291	Nein	Nein	Ja	Nein
Estland	EE	372	Nein	Ja	Ja	Ja
Äthiopien	ET	251	Nein	Nein	Ja	Nein
F						
Färöer-Inseln	FO	298	Nein	Nein	Ja	Nein
Fidschi	FJ	679	Nein	Nein	Ja	Nein
Finnland	FI	358	Ja	Ja	Ja	Ja
Frankreich	FR	33	Ja	Nein	Ja	Ja
Französisch-Guayana	GF	594	Nein	Nein	Ja	Nein
Französisch-Polynesien	PF	689	Nein	Nein	Ja	Nein
G						
Gabun	GA	241	Nein	Nein	Ja	Nein
Gambia	GM	220	Nein	Nein	Ja	Nein
Georgien	GE	995	Nein	Nein	Ja	Nein

Land oder Region	ISO-Code	Ortsvorwahl	Unterstützt Kurzwahlnummern	Unterstützt Langwahlnummern	Unterstützt Sender-IDs	Unterstützt Zwei-Wege-SMS
Deutschland	DE	49	Ja	Ja	Ja	Ja
Ghana	GH	233	Nein	Nein	Ja	Nein
Gibraltar	GI	350	Nein	Nein	Ja	Nein
Griechenland	GR	30	Nein	Nein	Ja	Nein
Grönland	GL	299	Nein	Nein	Ja	Nein
Grenada	GD	1-473	Nein	Nein	Ja	Nein
Guadeloupe	GP	590	Nein	Nein	Ja	Nein
Guam	GU	1-671	Nein	Nein	Nein	Ja
Guatemala	GT	502	Nein	Nein	Nein	Nein
Guernsey	GG	44-1481	Nein	Nein	Ja	Nein
Guinea	GN	224	Nein	Nein	Ja	Nein
Guinea-Bissau	GW	245	Nein	Nein	Ja	N/A
Guyana	GY	592	Nein	Nein	Ja	Nein
H						
Haiti	H	509	Nein	Nein	Ja	Nein
Honduras	HN	504	Nein	Nein	Ja	Nein
Hong Kong	HK	852	Nein	Ja	Ja	Ja

Land oder Region	ISO-Code	Ortsvorwahl	Unterstützt Kurzwahlnummern	Unterstützt Langwahlnummern	Unterstützt Sender-IDs	Unterstützt Zwei-Wege-SMS
Ungarn	HU	36	Nein	Ja	Nein	Ja
I						
Island	IS	354	Nein	Nein	Ja	Nein
Indien	IN	91	Ja	Ja[4]	Registrierung erforderlich[3]	Ja
Indonesien	ID	62	Nein	Nein	Ja	Nein
Irak	IQ	964	Nein	Nein	Ja	Nein
Irland	IE	353	Nein	Ja	Ja	Ja
Isle of Man	IM	44-1624	Nein	Nein	Ja	Nein
Israel	IL	972	Nein	Ja	Ja	Ja
Italien	IT	39	Ja	Ja	Ja	Ja
Elfenbeinküste	CI	225	Nein	Nein	Ja	Nein
J						
Jamaika	JM	1-876	Nein	Nein	Ja	Nein
Japan	JP	81	Ja	Ja	Ja	Ja
Jersey	JE	44-1534	Nein	Ja	Ja	Ja

Land oder Region	ISO-Code	Ortsvorwahl	Unterstützt Kurzwahlnummern	Unterstützt Langwahlnummern	Unterstützt Sender-IDs	Unterstützt Zwei-Wege-SMS
Jordanien	JO	962	Nein	Nein	Registrierung erforderlich[1]	Nein
K						
Kasachstan	KZ	7	Nein	Nein	Ja	Nein
Kenia	KE	254	Nein	Nein	Ja	Nein
Kosovo	XK	383	Nein	Nein	Ja	Nein
Kuwait	KW	965	Nein	Nein	Registrierung erforderlich[1]	Nein
Kirgisistan	KG	996	Nein	Nein	Ja	Nein
L						
Laos	LA	856	Nein	Nein	Ja	Nein
Lettland	LV	371	Nein	Nein	Ja	Nein
Libanon	LB	961	Nein	Nein	Ja	Nein
Lesotho	LS	266	Nein	Nein	Ja	Nein
Liberia	LR	231	Nein	Ja	Nein	
Libyen	LY	218	Nein	Nein	Ja	Nein

Land oder Region	ISO-Code	Ortsvorwahl	Unterstützt Kurzwahlnummern	Unterstützt Langwahlnummern	Unterstützt Sender-IDs	Unterstützt Zwei-Wege-SMS
Liechtenstein	LI	423	Nein	Nein	Ja	Nein
Litauen	LT	370	Nein	Ja	Ja	Ja
Luxemburg	LU	352	Nein	Ja	Ja	Ja
M						
Macao	MO	853	Nein	Nein	Ja	Nein
Mazedonien	MK	389	Nein	Nein	Ja	Nein
Madagaskar	MG	261	Nein	Nein	Ja	Nein
Malawi	MW	265	Nein	Nein	Ja	Nein
Malaysia	MY	60	Ja	Nein	Nein	Ja
Malediven	MV	960	Nein	Nein	Ja	Nein
Mali	ML	223	Nein	Nein	Ja	Nein
Malta	MT	356	Nein	Nein	Ja	Nein
Marshallinseln, die	MH	692	Nein	Nein	Nein	Nein
Martinique	MQ	596	Nein	Nein	Ja	Nein
Mauretanien	MR	222	Nein	Nein	Ja	Nein
Mauritius	MU	230	Nein	Nein	Ja	Nein

Land oder Region	ISO-Code	Ortsvorwahl	Unterstützt Kurzwahlnummern	Unterstützt Langwahlnummern	Unterstützt Sender-IDs	Unterstützt Zwei-Wege-SMS
Mayotte	YT	262	Nein	Nein	Ja	Nein
Mexiko	MX	52	Ja	Nein	Nein	Ja
Mikronesien (Föderierte Staaten von)	FM	691	Nein	Nein	Nein	Nein
Moldau	MD	373	Nein	Nein	Ja	Nein
Monaco	MC	377	Nein	Nein	Nein	Nein
Mongolei	MN	976	Nein	Nein	Ja	Nein
Montenegro	ME	382	Nein	Nein	Ja	Nein
Montserrat	MS	1-664	Nein	Nein	Ja	Nein
Marokko	MA	212	Ja	Nein	Ja	Ja
Mosambik	MZ	258	Nein	Nein	Nein	Nein
Myanmar	MM	95	Nein	Ja	Ja	Ja
N						
Namibia	NA	264	Nein	Nein	Ja	Nein
Nepal	NP	977	Nein	Nein	Ja	Nein
Niederlande	NL	31	Ja	Ja	Ja	Ja

Land oder Region	ISO-Code	Ortsvorwahl	Unterstützt Kurzwahlnummern	Unterstützt Langwahlnummern	Unterstützt Sender-IDs	Unterstützt Zwei-Wege-SMS
Niederländische Antillen	AN	599	Nein	Nein	Ja	Nein
Neukaledonien	NC	687	Nein	Nein	Ja	Nein
Neuseeland[6]	NZ	64	Ja	Nein	Nein	Ja
Nicaragua	NI	505	Nein	Nein	Nein	Nein
Niger	NE	227	Nein	Nein	Ja	Nein
Nigeria	NG	234	Nein	Nein	Ja	Nein
Niue	NU	683	Nein	Nein	Ja	Nein
Norwegen	NO	47	Nein	Ja	Ja	Ja
O						
Oman	OM	968	Nein	Nein	Nein	N/A
P						
Pakistan	PK	92	Nein	Nein	Ja	N/A
Palästina	PS	970	Nein	Nein	Ja	Nein
Panama	PA	507	Nein	Nein	Ja	Nein
Papua-Neuguinea	PG	675	Nein	Nein	Ja	Nein
Paraguay	PY	595	Nein	Nein	Nein	Nein

Land oder Region	ISO-Code	Ortsvorwahl	Unterstützt Kurzwahlnummern	Unterstützt Langwahlnummern	Unterstützt Sender-IDs	Unterstützt Zwei-Wege-SMS
Peru	PE	51	Ja	Nein	Nein	Ja
Philippinen	PH	63	Nein	Ja	Registrierung erforderlich[1]	Ja
Polen	PL	48	Nein	Ja	Ja	Ja
Portugal	PT	351	Nein	Ja	Ja	Ja
Puerto Rico	PR	1-797, 1-939	Nein	Nein	Nein	Ja
Q						
Katar	QA	974	Nein	Nein	Registrierung erforderlich[1]	Nein
R						
Republik Kongo	CG	242	Nein	Nein	Nein	Nein
Réunion (Frankreich)	RE	262	Nein	Nein	Ja	Nein
Rumänien	RO	40	Nein	Ja	Ja	Ja

Land oder Region	ISO-Code	Ortsvorwahl	Unterstützt Kurzwahlnummern	Unterstützt Langwahlnummern	Unterstützt Sender-IDs	Unterstützt Zwei-Wege-SMS
Russland	RU	7	Ja	Nein	Registrierung erforderlich[1]	Ja
Ruanda	RW	250	Nein	Nein	Ja	Nein
S						
St. Kitts und Nevis	KN	1-869	Nein	Nein	Nein	Nein
St. Lucia	LC	1-758	Nein	Nein	Nein	Nein
Samoa	WS	685	Nein	Nein	Ja	Nein
San Marino	SM	378	Nein	Nein	Ja	Nein
São Tomé und Príncipe	ST	239	Nein	Nein	Ja	Nein
Saudi-Arabien	SA	966	Nein	Ja[4]	Registrierung erforderlich[1]	Nein
Senegal	SN	221	Nein	Nein	Ja	Nein
Serbien	RS	381	Nein	Nein	Ja	Nein
Seychellen	SC	248	Nein	Nein	Ja	Nein
Sierra Leone	SL	232	Nein	Nein	Ja	Nein

Land oder Region	ISO-Code	Ortsvorwahl	Unterstützt Kurzwahlnummern	Unterstützt Langwahlnummern	Unterstützt Sender-IDs	Unterstützt Zwei-Wege-SMS
Singapur	SG	65	Ja	Ja	Ja[5]	Ja
Slowakei	SK	421	Nein	Ja	Ja	Nein
Slowenien	SI	386	Nein	Nein	Ja	Nein
Salomoninseln	SB	677	Nein	Nein	Ja	Nein
Somalia	SO	252	Nein	Nein	Ja	Nein
Südafrika	ZA	27	Ja	Ja	Nein	Ja
Südkorea	KR	82	Nein	Nein	Nein	Nein
Südsudan	SS	211	Nein	Nein	Ja	Nein
Spanien	ES	34	Ja	Ja	Ja	Ja
Sri Lanka	LK	94	Nein	Nein	Registrierung erforderlich[1]	Nein
Surinam	SR	597	Nein	Nein	Ja	Nein
Swasiland	SZ	268	Nein	Nein	Ja	Nein
Schweden	SE	46	Ja	Ja	Ja	Ja
Schweiz	CH	41	Nein	Ja	Ja	Ja
T						
Taiwan	TW	886	Nein	Ja	Nein	Ja

Land oder Region	ISO-Code	Ortsvorwahl	Unterstützt Kurzwahlnummern	Unterstützt Langwahlnummern	Unterstützt Sender-IDs	Unterstützt Zwei-Wege-SMS
Tadschikistan	TJ	992	Nein	Nein	Ja	Nein
Tansania	TX	255	Nein	Nein	Ja	Nein
Thailand	TH	66	Nein	Ja	Registrierung erforderlich[1]	Ja
Timor-Leste	TL	670	Nein	Nein	Ja	Nein
Togo	TG	228	Nein	Nein	Ja	Nein
Tonga	TO	676	Nein	Nein	Ja	Nein
Trinidad und Tobago	TT	1-868	Nein	Nein	Ja	Nein
Tunesien	TN	216	Nein	Nein	Ja	Nein
Türkei	TR	90	Nein	Nein	Registrierung erforderlich[1]	Nein
Turkmenistan	TM	993	Nein	Nein	Nein	Nein
Turks- und Caicosinseln	TC	1-649	Nein	Nein	Ja	Nein

Land oder Region	ISO-Code	Ortsvorwahl	Unterstützt Kurzwahlnummern	Unterstützt Langwahlnummern	Unterstützt Sender-IDs	Unterstützt Zwei-Wege-SMS
Tuvalu	TC	688	Nein	Nein	Ja	Nein
U						
Uganda	UG	256	Nein	Nein	Ja	Nein
Ukraine	UA	380	Nein	Ja	Ja	Ja
Vereinigte Arabische Emirate (VAE)	AE	971	Ja	Ja	Registrierung erforderlich[1]	Ja
Großbritannien und Nordirland	GB	44	Ja	Ja	Ja	Ja
Vereinigte Staaten	US	1	Ja	Ja	Nein	Ja
Uruguay	UY	598	Ja	Nein	Nein	Ja
Usbekistan	UZ	998	Nein	Nein	Ja	Nein
V						
Vanuatu	VU	678	Nein	Nein	Ja	Nein
Venezuela	VE	58	Nein	Nein	Nein	Nein
Vietnam	VN	84	Nein	Nein	Registrierung erforderlich[1]	Nein

Land oder Region	ISO-Code	Ortsvorwahl	Unterstützt Kurzwahlnummern	Unterstützt Langwahlnummern	Unterstützt Sender-IDs	Unterstützt Zwei-Wege-SMS
Jungferninseln, Britische	VG	1-284	Nein	Nein	Ja	Nein
Jungferninseln, US	VI	1-340	Nein	Nein	Nein	Ja
W						
X						
Y						
Jemen	YE	967	Nein	Nein	Ja	Nein
Z						
Sambia	ZM	260	Nein	Nein	Ja	Nein
Simbabwe	ZW	263	Nein	Nein	Ja	Nein

Hinweise

1. Sender müssen eine vorab registrierte alphabetische Sender-ID verwenden. Informationen zum Anfordern einer Absender-ID von AWS Support finden Sie unterAnfordern von Sender-IDs für SMS-Messaging mit Amazon SNS. In einigen Ländern müssen Sender bestimmte Anforderungen erfüllen oder bestimmte Einschränkungen einhalten, um die Genehmigung zu erhalten. In diesen Fällen kontaktieren wir Sie AWS Support möglicherweise für weitere Informationen, nachdem Sie Ihre Sender-ID-Anfrage eingereicht haben.
2. Sender müssen für jeden Nachrichtentyp, den sie senden möchten, eine vorab registrierte Vorlage verwenden. Wenn ein Sender diese Anforderung nicht erfüllt, werden seine Nachrichten blockiert. Um eine Vorlage zu registrieren, öffnen Sie einen Amazon SNS SNS-SMS-Fall mit AWS Support. Wenn Sie den Fall erstellen, geben Sie dieselben Informationen an, die Sie für die

Anforderung einer Sender-ID angeben würden. Weitere Informationen finden Sie unter Anfordern von Sender-IDs für SMS-Messaging mit Amazon SNS. In einigen Ländern müssen Sender zusätzliche bestimmte Anforderungen erfüllen oder bestimmte Einschränkungen einhalten, um die Genehmigung zu erhalten. In diesen Fällen werden Sie AWS Support möglicherweise um zusätzliche Informationen gebeten.

> **Note**
>
> Um Nachrichten nach China zu senden, müssen Sie Ihre Vorlagen zunächst AWS Support zur Genehmigung registrieren.

3. Sender müssen eine vorab registrierte alphabetische Sender-ID verwenden. Weitere Registrierungsschritte sind erforderlich. Weitere Informationen finden Sie unter Anforderungen für die Registrierung der Absender-ID für Indien.

4. Langwahlnummern unterstützen in diesen Ländern nur eingehende Nachrichten. Mit anderen Worten: Sie können diese Langwahlnummern nicht für Nachrichten an Ihre Empfänger verwenden, aber Sie können sie verwenden, um Nachrichten von Ihren Empfängern zu erhalten. Diese Langwahlnummern sind nützlich, um Ihren Empfängern das Opt-Out zu ermöglichen, wenn Sie Nachrichten mit einer alphabetischen Sender-ID senden, da Sender-IDs nur ausgehende Nachrichten unterstützen.

5. Amazon-SNS kann SMS-Nachrichten nach Singapur mit einer Sender-ID senden, die über die Singapore SMS Sender ID Registry (SSIR) registriert wurde. Dieses Registry wurde von der Info-communications Media Development Authority (IMDA) in Singapur ins Leben gerufen. Weitere Informationen zu den Anforderungen für die Verwendung einer Sender-ID für Singapur finden Sie unter Anforderungen für die Registrierung der Absender-ID für Singapur.

   Sie können in Singapur auch SMS-Nachrichten mit nicht registrierten Sender-IDs oder alternativen Ursprungsidentitätstypen wie Kurz- oder Langwahlnummern senden.

6. Ohne eine spezielle Kurzwahlnummer versucht Amazon SNS immer noch, Nachrichten über einen gemeinsamen Pool von Kurzwahlnummern an neuseeländische Empfänger zu senden. Aufgrund der Beschränkungen der lokalen Mobilfunkanbieter in Bezug auf gemeinsam genutzte Nummern erfolgt die Zustellung über diese gemeinsamen Nummern nach bestem Wissen und Gewissen. Daher empfiehlt Amazon SNS dringend, eine spezielle Kurznummer für den gesamten Datenverkehr, der nach Neuseeland gesendet wird, zu erwerben. Nachrichten, die

URLs enthalten, müssen im Rahmen des speziellen Shortcode-Verfahrens auf die Zulassungsliste gesetzt werden. Weitere Informationen zum Kauf einer Kurzwahlnummer finden Sie unter Anfordern dedizierter Kurzwahlnummern für SMS-Messaging mit Amazon SNS.

Bewährte Methoden für SMS

Mobiltelefonbenutzer neigen zu einer sehr geringen Toleranz für unerwünschte SMS-Nachrichten. Die Antwortraten für unaufgeforderte SMS-Kampagnen sind fast immer sehr niedrig, was gleichzeitig auch eine schlechte Investitionsrendite bedeutet.

Außerdem überwachen Mobilfunkanbieter Massen-SMS-Sender kontinuierlich. Nummern, die offenbar unerwünschte Nachrichten senden, werden gedrosselt oder gesperrt.

Das Senden von unerwünschten Inhalten stellt auch einen Verstoß gegen die AWS Acceptable Use Policy dar. Das Amazon SNS-Team überprüft SMS-Kampagnen regelmäßig und kann Ihre Möglichkeiten zum Senden von Nachrichten drosseln oder sperren, wenn Sie offensichtlich unerwünschte Nachrichten senden.

Schließlich gelten in vielen Ländern, Regionen und Rechtsbereichen empfindliche Strafen für das Senden unerwünschter SMS-Nachrichten. So bestimmt beispielsweise in den USA der Telephone Consumer Protection Act (TCPA), dass Verbraucher einen Anspruch auf eine (vom Sender zu leistende) Zahlung von 500 bis 1.500 USD für jede unerwünschte Nachricht haben, die sie erhalten.

Dieser Abschnitt beschreibt verschiedene bewährte Methoden, die Ihnen dabei helfen können, die Einbeziehung Ihrer Kunden zu verbessern und hohe Strafzahlungen zu vermeiden. Beachten Sie jedoch, dass diesem Abschnitt keine Rechtsberatung darstellt. Wenden Sie sich immer an einen Rechtsanwalt, um juristischen Rat einzuholen.

Themen

- Anpassen Ihres Sendens basierend auf der Kundenbeteiligung
- Senden zu angemessenen Zeiten
- Vermeiden Sie Ermüdungen durch die Nutzung mehrerer Kanäle
- Verwenden dedizierter Kurzwahlnummern
- Bestätigen Sie Ihre Ziel-Telefonnummern
- Erstellen Sie Entwürfe unter Beachtung der Redundanz
- SMS-Limits und -Einschränkungen
- Verwalten von Schlüsselwörtern für die Abmeldung
- CreatePool
- PutKeyword
- Verwalten von Nummerneinstellungen
- SMS-Zeichenbeschränkungen in Amazon SNS

Einhaltung von Gesetzen, Vorschriften und Anforderungen der Anbieter

Für Verstöße gegen die Gesetze und Bestimmungen der Länder, in denen sich Ihre Kunden befinden, drohen erhebliche Geldbußen und Strafen. Daher müssen Sie die Gesetze kennen, die SMS-Messaging in den Ländern und Regionen regeln, in denen Sie geschäftlich tätig sind.

Die folgende Liste enthält Links auf die wichtigsten Gesetze zu SMS-Kommunikation in den größten Märkten der Welt.

- USA: Der Telephone Consumer Protection Act (TCPA) von 1991 gilt für bestimmte Arten von SMS-Nachrichten. Weitere Informationen finden Sie unter Rules and Regulations (Regeln und Vorschriften) auf der Website der Federal Communication Commission.
- Großbritannien und Nordirland: Die Privacy and Electronic Communications (EG-Richtlinie) (PECR) von 2003 gilt für bestimmte Arten von SMS-Nachrichten. Weitere Informationen finden Sie unter What are PECR? (Was sind PECR?) auf der Website des UK Information Commissioner's Office.
- Europäische Union: Die Datenschutzrichtlinie für elektronische Kommunikation von 2002 (auch ePrivacy-Richtlinie) gilt für einige Typen von SMS-Nachrichten. Weitere Informationen finden Sie im vollständigen Gesetzestext auf der Website Europa.eu.
- Kanada: Der Fighting Internet and Wireless Spam Act, auch als kanadisches Antispamgesetz (CASL) bekannt, gilt für bestimmte Arten von SMS-Nachrichten. Weitere Informationen finden Sie im vollständigen Gesetzestext auf der Website des kanadischen Parlaments.

- Japan: Das Gesetz zur Regulierung der Übermittlung von bestimmten elektronischen Mails kann für bestimmte Arten von SMS-Nachrichten gelten. Weitere Informationen finden Sie unter Japan's Countermeasures Against Spam auf der Website des japanischen Ministeriums für innere Angelegenheiten und Kommunikation.

Diese Gesetze können für Sie als Absender auch dann gelten, wenn Ihr Unternehmen oder Ihre Organisation nicht in einem dieser Länder ansässig ist. Einige der Gesetze in dieser Liste wurden ursprünglich für unerwünschte E-Mails oder Anrufe erlassen, wurden seither jedoch auch auf SMS-Nachrichten erweitert. In anderen Ländern und Regionen können andere Gesetze für die Übermittlung von SMS-Nachrichten gelten. Fragen Sie einen Anwalt in einem Land oder einer Region, in der sich Ihre Kunden befinden, um sich entsprechend beraten zu lassen.

In vielen Ländern haben die lokalen Netzbetreiber letztendlich die Befugnis, zu bestimmen, welche Art von Datenverkehr ihre Netze passiert. Dies bedeutet, dass die Netzbetreiber möglicherweise Beschränkungen für SMS-Inhalte auferlegen, die über die Mindestanforderungen der örtlichen Gesetzgebung hinausgehen.

Einholen von Berechtigungen

Senden Sie niemals Nachrichten an Empfänger, die nicht ausdrücklich darum gebeten haben, die Arten von Nachrichten zu erhalten, die Sie senden möchten. Teilen Sie keine Opt-in-Listen, auch nicht in Organisationen innerhalb desselben Unternehmens.

Wenn Empfänger über ein Onlineformular angeben können, dass sie Ihre Nachrichten erhalten möchten, fügen Sie dem Formular Systeme hinzu, die verhindern, dass automatisierte Scripts ohne Wissen der Benutzer Abonnements für sie abschließen. Sie sollten auch die Häufigkeit begrenzen, mit der ein Benutzer in einer einzelnen Sitzung eine Telefonnummer angeben kann.

Wenn Sie eine SMS-Opt-in-Anfrage erhalten, senden Sie dem Empfänger eine Nachricht mit der Bitte, zu bestätigen, dass er Nachrichten von Ihnen erhalten möchte. Senden Sie diesem Empfänger keine weiteren Nachrichten, bis er sein Abonnement bestätigt hat. Eine Bestätigungsnachricht für ein Abonnement kann wie folgt aussehen:

```
Text YES to join ExampleCorp alerts. 2 msgs/month. Msg & data rates may apply. Reply HELP for help, STOP to cancel.
```

Führen Sie Unterlagen mit Datum, Uhrzeit und Quelle der einzelnen Opt-in-Anfragen und Bestätigungen. Dies kann hilfreich sein, wenn ein Anbieter oder eine Regulierungsbehörde danach fragt, oder auch für Routineprüfungen Ihrer Kundenliste.

Opt-in-Workflow

In einigen Fällen (wie der gebührenfreien Registrierung in den USA oder der Kurzwahl-Registrierung) verlangen Mobilfunkanbieter, dass Sie Mockups oder Screenshots Ihres gesamten Opt-in-Workflows zur Verfügung stellen. Die Mockups oder Screenshots müssen dem Opt-in-Workflow, den Ihre Empfänger ausführen werden, sehr ähnlich sein.

Ihre Mockups oder Screenshots sollten alle unten aufgeführten erforderlichen Angaben enthalten, um ein Höchstmaß an Compliance zu gewährleisten.

Erforderliche Angaben

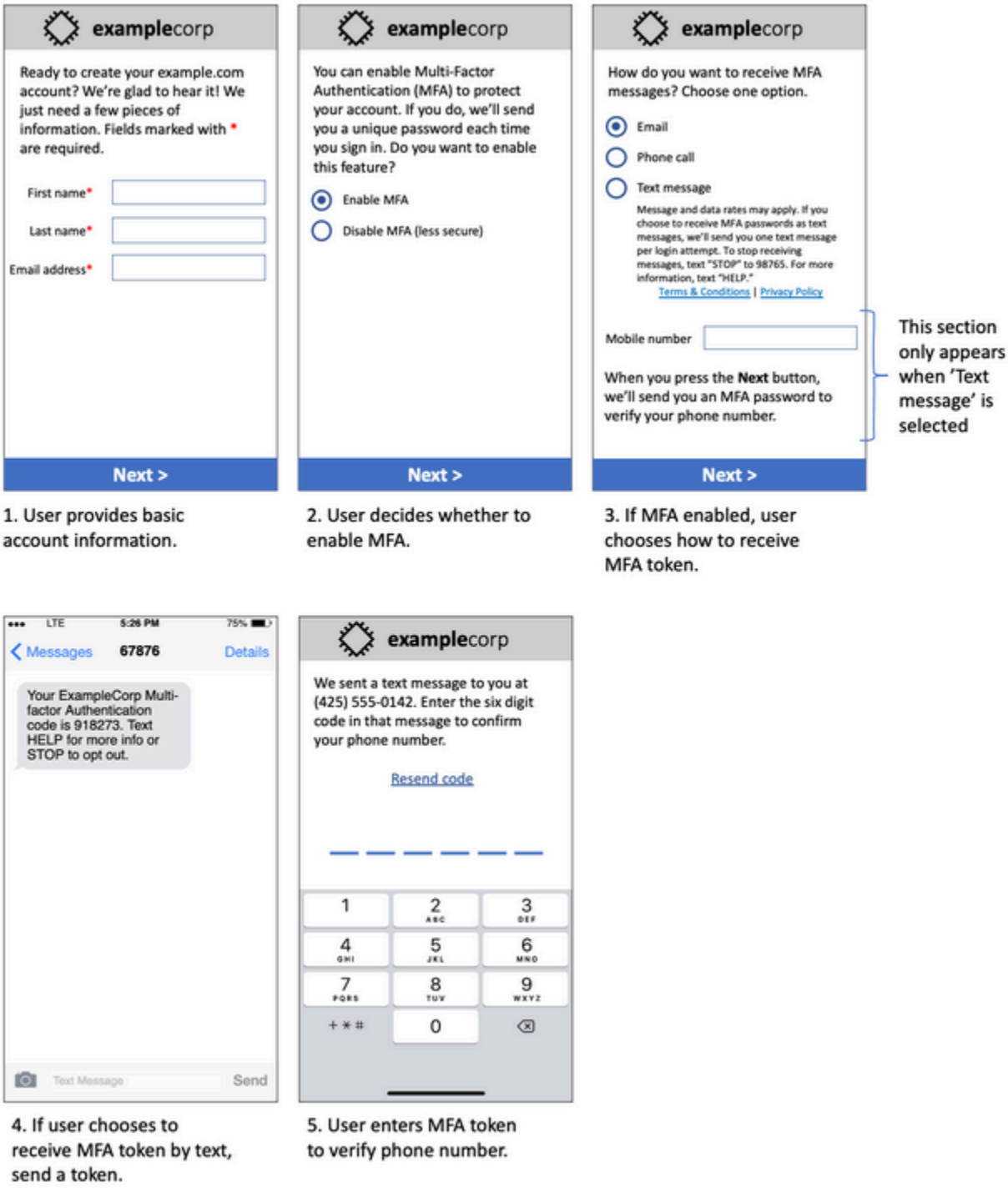
- Eine Beschreibung des Messaging-Anwendungsfalls, den Sie über Ihr Programm senden.
- Der Satz „Es können Gebühren für Nachrichten und Daten anfallen“.
- Ein Hinweis darauf, wie oft Empfänger Nachrichten von Ihnen erhalten. Beispielsweise könnte die Angabe für ein wiederkehrendes Messaging-Programm „eine Nachricht pro Woche“ lauten. Ein Anwendungsfall für ein Einmalpasswort oder eine Multi-Faktor-Authentifizierung könnte lauten: „Nachrichtenfrequenz variiert“ oder „eine Nachricht pro Anmeldeversuch“.
- Links zu Ihren Allgemeinen Geschäftsbedingungen und Datenschutzrichtlinien.

Häufige Ablehnungsgründe für nicht konforme Opt-ins

- Wenn der angegebene Firmenname nicht mit dem übereinstimmt, was im Mockup oder Screenshot angegeben ist. Alle nicht offensichtlichen Zusammenhänge sollten in der Beschreibung des Opt-in-Workflows erklärt werden.
- Wenn es den Anschein hat, dass eine Nachricht an den Empfänger gesendet wird, zuvor jedoch keine ausdrückliche Zustimmung eingeholt wurde. Die ausdrückliche Zustimmung ist eine Voraussetzung für alle Nachrichten.
- Wenn es den Anschein hat, dass der Empfang einer Textnachricht erforderlich ist, um sich für einen Service anzumelden. Dies ist nicht konform, wenn der Workflow keine Alternative zum Empfang einer Opt-in-Nachricht in anderer Form wie E-Mail oder Sprachanruf bietet.
- Wenn die Opt-in-Formulierungen vollständig in den Nutzungsbedingungen angegeben ist. Die Angaben sollten dem Empfänger immer zum Opt-in-Zeitpunkt vorgelegt werden und nicht in einem verknüpften Richtliniendokument enthalten sein.
- Wenn ein Kunde zugestimmt hat, eine bestimmte Art von Nachricht von Ihnen zu erhalten, und Sie ihm andere Arten von Textnachrichten senden. Der Kunde stimmt beispielsweise zu, Einmalpasswörter zu erhalten, erhält aber auch Umfragenachrichten.

- Wenn die erforderlichen Angaben (siehe oben) den Empfängern nicht vorgelegt werden.

Das folgende Beispiel erfüllt die Anforderungen der Mobilfunkanbieter für einen Anwendungsfall mit Multi-Faktor-Authentifizierung.



Mockup eines Anwendungsfalls mit Multi-Faktor-Authentifizierung

Es enthält finalisierten Text und Bilder und zeigt den gesamten Opt-in-Workflow mit Anmerkungen. Im Opt-in-Workflow muss der Kunde eindeutige, absichtliche Maßnahmen ergreifen, um seine Zustimmung zum Erhalt von Textnachrichten zu erteilen. Der Workflow enthält außerdem alle erforderlichen Angaben.

Andere Opt-in-Workflowtypen

Mobilfunkanbieter akzeptieren auch Opt-in-Workflows außerhalb von Anwendungen und Websites wie mündliches oder schriftliches Opt-in, wenn dies den oben genannten Anforderungen entspricht. Ein konformer Opt-in-Workflow und ein mündliches oder schriftliches Skript holen die ausdrückliche Zustimmung des Empfängers ein, einen bestimmten Nachrichtentyp zu erhalten. Beispiele hierfür sind das mündliche Skript, das ein Kundendienstmitarbeiter verwendet, um die Zustimmung einzuholen, bevor er die Daten in eine Servicedatenbank aufnimmt, oder eine Telefonnummer, die auf einem Werbeflyer aufgeführt ist. Zum Bereitstellen eines Mockups dieser Opt-in-Workflowtypen können Sie einen Screenshot Ihres Opt-in-Skripts, Marketingmaterial oder Ihre Datenbank bereitstellen, in der Nummern erfasst werden. Mobilfunkanbieter haben möglicherweise zusätzliche Fragen zu diesen Anwendungsfällen, wenn ein Opt-in nicht klar ist oder der Anwendungsfall bestimmte Volumina überschreitet.

Nicht an alte Listen senden

Empfänger ändern ihre Telefonnummer häufig. Eine Telefonnummer, für die Sie vor zwei Jahren eine Zustimmung zur Kontaktaufnahme eingeholt haben, könnte heute jemand anderem gehören. Verwenden Sie keine alte Liste von Telefonnummern für ein neues Messaging-Programm. Andernfalls werden wahrscheinlich einige Nachrichten ihren Empfänger nicht erreichen, weil die Nummer nicht mehr vergeben ist, und einige Personen werden sich abmelden, weil sie sich nicht erinnern können, Ihnen ihre Zustimmung gegeben zu haben.

Prüfung Ihrer Kundenlisten

Wenn Sie wiederkehrende SMS-Kampagnen versenden, sollten Sie Ihre Kundenlisten regelmäßig überprüfen. Dadurch stellen Sie sicher, dass nur Kunden, die an Ihren Nachrichten interessiert sind, diese auch erhalten.

Senden Sie bei der Prüfung Ihrer Liste jedem angemeldeten Kunden eine Nachricht, die diesen an sein Abonnement erinnert, begleitet von Anleitungen zum eventuellen Abbestellen der Nachrichten. Eine solche Erinnerungsnachricht kann wie folgt aussehen:

```
You're subscribed to ExampleCorp alerts. Msg & data rates may apply. Reply
HELP for help, STOP to unsubscribe.
```

Aufbewahren von Aufzeichnungen

Führen Sie Aufzeichnungen, aus denen hervorgeht, wann ein Kunde den Erhalt von SMS-Nachrichten von Ihnen angefordert hat und welche Nachrichten Sie an welche Kunden senden. Viele Ländern und Regionen auf der Welt fordern Sender von SMS-Nachrichten auf, dieses so aufzubewahren, dass Sie problemlos abgerufen werden können. Auch Mobilfunkanbieter können diese Informationen jederzeit von Ihnen anfordern. Die genauen Informationen, die Sie bereitstellen müssen, variieren je nach Land oder Region. Weitere Informationen über Aufbewahrungsanforderungen finden Sie in den Bestimmungen zu kommerziellem SMS Messaging in den Ländern oder Regionen, in denen Ihre Kunden ansässig sind.

Es kann vorkommen, dass ein Anbieter oder eine Regulierungsbehörde von uns verlangt, dass wir nachweisen, dass Kunden sich dafür entschieden haben, Nachrichten von Ihnen zu empfangen. In einem solchen Fall erhalten Sie vom AWS Support eine Liste mit Informationen, die der Anbieter oder die Behörde verlangen. Wenn Sie die notwendigen Informationen nicht bereitstellen können, schränken wir Ihre Möglichkeit, weitere SMS-Nachrichten zu senden, möglicherweise ein.

Formulieren Sie Ihre Nachrichten klar, ehrlich und präzise

SMS ist ein einzigartiges Medium. Das Limit von 160 Zeichen pro Nachricht bedeutet, dass Ihre Nachrichten präzise sein müssen. Techniken, die Sie möglicherweise in anderen Kommunikationskanälen wie E-Mail verwenden, gelten möglicherweise nicht für den SMS-Kanal und wirken möglicherweise sogar unehrlich oder irreführend, wenn sie mit SMS-Nachrichten verwendet werden. Wenn der Inhalt Ihrer Nachrichten nicht mit bewährten Methoden übereinstimmt, ignorieren die Empfänger Ihre Nachrichten möglicherweise. Im schlimmsten Fall identifizieren die Mobilfunkanbieter Ihre Nachrichten möglicherweise als Spam und blockieren künftige Nachrichten von Ihrer Telefonnummer.

Dieser Abschnitt enthält einige Tipps und Anregungen zum Erstellen eines effektiven SMS-Nachrichtentexts.

Identifizieren Sie sich als Absender

Ihre Empfänger sollten sofort erkennen können, dass eine Nachricht von Ihnen stammt. Absender, die diese bewährte Methode befolgen, geben am Anfang jeder Nachricht einen identifizierenden Namen („Programmname“) an.

Vermeiden Sie Folgendes:

```
Your account has been accessed from a new device. Reply Y to confirm.
```

Versuchen Sie stattdessen dieses Linkformat:

```
ExampleCorp Financial Alerts: You have logged in to your account from a
new device. Reply Y to confirm, or STOP to opt-out.
```

Versuchen Sie nicht, Ihre Nachricht wie eine persönliche Nachricht aussehen zu lassen.

Einige Vermarkter sind versucht, ihren SMS-Nachrichten eine persönliche Note zu verleihen, indem sie den Eindruck erwecken, dass ihre Nachrichten von einer Person stammen. Diese Strategie kann jedoch dazu führen, dass Ihre Nachricht wie ein Phishing-Versuch erscheint.

Vermeiden Sie Folgendes:

```
Hi, this is Jane. Did you know that you can save up to 50% at
Example.com? Click here for more info: https://www.example.com.
```

Versuchen Sie stattdessen dieses Linkformat:

```
ExampleCorp Offers: Save 25-50% on sale items at Example.com. Click here
to browse the sale: https://www.example.com. Text STOP to opt-out.
```

Seien Sie vorsichtig, wenn Sie über Geld sprechen.

Betrüger machen sich oft den Wunsch der Menschen zunutze, Geld zu sparen und zu erhalten. Lassen Sie Angebote nicht so erscheinen, als seien sie zu gut, um wahr zu sein. Nutzen Sie die Verlockung des Geldes nicht, um Menschen zu täuschen. Verwenden Sie keine Währungssymbole, um Geldbeträge anzugeben.

Vermeiden Sie Folgendes:

```
Save big $$$ on your next car repair by going to https://
www.example.com.
```

Versuchen Sie stattdessen dieses Linkformat:

```
ExampleCorp Offers: Your ExampleCorp insurance policy gets you discounts
at 2300+ repair shops nationwide. More info at https://www.example.com.
Text STOP to opt-out.
```

Verwenden Sie nur die notwendigen Zeichen

Unternehmen neigen oft dazu, ihre eingetragenen Marken zu schützen, indem sie Markensymbole wie ™ oder ® in ihre Nachrichten aufnehmen. Diese Symbole sind jedoch nicht Teil des Standardzeichensatzes (bekannt als GSM-Alphabet), der in einer SMS-Nachricht mit 160 Zeichen enthalten sein kann. Wenn Sie eine Nachricht senden, die eines dieser Zeichen enthält, wird Ihre Nachricht automatisch mit einem anderen Zeichencodierungssystem gesendet, das nur 70 Zeichen pro Nachricht unterstützt. Infolgedessen könnte Ihre Nachricht in mehrere Teile zerlegt werden. Da Ihnen jeder gesendete Nachrichtenteil in Rechnung gestellt wird, kann die Nachricht Sie mehr kosten als erwartet. Darüber hinaus erhalten Ihre Empfänger statt einer einzigen Nachricht möglicherweise mehrere aufeinanderfolgende Nachrichten von Ihnen. Weitere Informationen zur SMS-Zeichencodierung erhalten Sie unter SMS-Zeichenbeschränkungen in Amazon SNS.

Vermeiden Sie Folgendes:

```
ExampleCorp Alerts: Save 20% when you buy a new ExampleCorp Widget® at
example.com and use the promo code WIDGET.
```

Versuchen Sie stattdessen dieses Linkformat:

```
ExampleCorp Alerts: Save 20% when you buy a new ExampleCorp Widget(R) at
example.com and use the promo code WIDGET.
```

> **Note**
>
> Die beiden vorherigen Beispiele sind fast identisch. Das erste Beispiel enthält jedoch ein Symbol für eine eingetragene Marke (®), das nicht Bestandteil des GSM-Alphabets ist. Dadurch wird das erste Beispiel in zwei Nachrichtenteilen gesendet, während das zweite Beispiel als ein Nachrichtenteil seinen Empfänger erreicht.

Verwenden gültiger, sicherer Links

Wenn Ihre Nachricht Links enthält, überprüfen Sie die Links, um sicherzustellen, dass sie funktionieren. Testen Sie Ihre Links auf einem Gerät außerhalb Ihres Unternehmensnetzwerks, um sicherzustellen, dass sie ordnungsgemäß aufgelöst werden. Aufgrund des Limits von 160 Zeichen in SMS-Nachrichten werden sehr lange URLs u. U. auf mehrere Nachrichten aufgeteilt. Verwenden Sie Umleitungsdomänen, um verkürzte URLs bereitzustellen. Sie sollten jedoch keine kostenlosen

Linkverkürzungsdienste, wie tinyurl.com oder bitly.com, verwenden, da Netzbetreiber dazu neigen, Nachrichten zu filtern, die Links zu diesen Domänen enthalten. Sie können jedoch kostenpflichtige Linkverkürzungsdienste nutzen, solange Ihre Links auf eine Domäne verweisen, die ausschließlich der Nutzung Ihres Unternehmens oder Ihrer Organisation vorbehalten ist.

Vermeiden Sie Folgendes:

```
Go to https://tinyurl.com/4585y8mr today for a special offer!
```

Versuchen Sie stattdessen dieses Linkformat:

```
ExampleCorp Offers: Today only, get an exclusive deal on an ExampleCorp
Widget. See https://a.co/cFKmaRG for more info. Text STOP to opt-out.
```

Verwenden Sie nur eine begrenzte Anzahl von Abkürzungen

Das Limit von 160 Zeichen für den SMS-Kanal verleitet einige Absender dazu, in ihren Nachrichten verstärkt Abkürzungen zu verwenden. Die übermäßige Verwendung von Abkürzungen kann jedoch vielen Lesern unprofessionell erscheinen und dazu führen, dass einige Benutzer Ihre Nachricht als Spam melden. Es ist durchaus möglich, eine zusammenhängende Nachricht zu schreiben, ohne übermäßig viele Abkürzungen zu verwenden.

Vermeiden Sie Folgendes:

```
Get a gr8 deal on ExampleCorp widgets when u buy a 4-pack 2day.
```

Versuchen Sie stattdessen dieses Linkformat:

```
ExampleCorp Alerts: Today only—an exclusive deal on ExampleCorp Widgets
at example.com. Text STOP to opt-out.
```

Angemessene Antworten

Wenn ein Empfänger auf Ihre Nachrichten antwortet, stellen Sie sicher, dass Sie mit nützlichen Informationen reagieren. Zum Beispiel: Wenn ein Kunde auf eine Ihrer Nachrichten mit dem Schlüsselwort „HILFE“ antwortet, senden Sie diesem Informationen zu dem von ihm abonnierten Programm, zu der Anzahl der pro Monat gesendeten Nachrichten und Möglichkeiten zur

Kontaktaufnahme mit Ihnen für weitere Informationen zu. Eine „HILFE“-Antwort kann wie folgt aussehen:

```
HELP: ExampleCorp alerts: email help@example.com or call 425-555-0199. 2
msgs/month. Msg & data rates may apply. Reply STOP to cancel.
```

Wenn ein Kunde mit dem Schlüsselwort „STOPP“ antwortet, teilen Sie ihm mit, dass er fortan keine weiteren Nachrichten mehr erhalten wird. Eine „STOPP“-Antwort kann wie folgt aussehen:

```
You're unsubscribed from ExampleCorp alerts. No more messages will be sent.
Reply HELP, email help@example.com, or call 425-555-0199 for more info.
```

Anpassen Ihres Sendens basierend auf der Kundenbeteiligung

Die Prioritäten Ihrer Kunden können sich mit der Zeit ändern. Wenn Kunden Ihre Nachrichten nicht mehr nützlich finden, bestellen sie sie möglicherweise ganz ab oder melden sie sogar als unerwünscht. Daher ist es wichtig, dass Sie Ihr Sendeverhalten auf der Grundlage der Kundenbeteiligung anpassen.

Für Kunden, die nur selten mit Ihren Nachrichten interagieren, sollten Sie die Häufigkeit Ihrer Nachrichten entsprechend anpassen. Wenn Sie z. B. wöchentliche Nachrichten an interessierte Kunden senden, können Sie für weniger interessierte Kunden einen monatlichen Kurzbericht erstellen.

Entfernen Sie schließlich Kunden, die niemals mit Ihren Nachrichten interagieren, vollständig aus Ihren Kundenlisten. Dieser Schritt wird verhindert, dass die Kunden irgend wann verärgert auf Ihre Nachrichten reagieren. Außerdem sparen Sie dadurch Geld und schützen Ihren guten Ruf als Sender.

Senden zu angemessenen Zeiten

Senden Sie Nachrichten nur während üblicher Arbeitszeiten am Tage. Wenn Sie Nachrichten zur Mittagszeit oder mitten in der Nacht senden, ist es sehr wahrscheinlich, dass Kunden Ihre Nachrichten abbestellen, um davon nicht mehr gestört zu werden. Darüber hinaus ist es nicht sinnvoll, SMS-Nachrichten dann zu senden, wenn Ihre Kunden nicht sofort darauf reagieren können.

Wenn Sie Kampagnen oder Journeys an sehr große Zielgruppen senden, überprüfen Sie die Durchsatzraten für Ihre Ursprungsnummern. Teilen Sie die Anzahl der Empfänger durch Ihre Durchsatzrate, um zu bestimmen, wie lange es dauert, Nachrichten an alle Ihre Empfänger zu senden.

Vermeiden Sie Ermüdungen durch die Nutzung mehrerer Kanäle

Wenn Sie in Ihren Kampagnen mehrere Kommunikationskanäle (z. B. E-Mail, SMS und Push-Nachrichten) verwenden, senden Sie nicht die gleiche Nachricht über jeden Kanal. Wenn Sie dieselbe Nachricht gleichzeitig über mehrere Kanäle senden, fühlen sich Ihre Kunden sehr wahrscheinlich dadurch eher gestört.

Verwenden dedizierter Kurzwahlnummern

Wenn Sie Kurzwahlnummern verwenden, unterhalten Sie eine separate Kurzwahlnummer für jede Marke und jeden Nachrichtentyp. Wenn Ihr Unternehmen z. B. zwei Marken hat, verwenden Sie für jede davon eine eigene Kurzwahlnummer. Wenn Sie Transaktions- und Werbenachrichten versenden, verwenden Sie auch dafür jeweils separate Kurzwahlnummern. Weitere Informationen zum Anfordern von Kurzwahlnummern finden Sie unter Anfordern dedizierter Kurzwahlnummern für SMS-Messaging mit Amazon SNS.

Bestätigen Sie Ihre Ziel-Telefonnummern

Wenn Sie SMS-Nachrichten über Amazon SNS senden, wird Ihnen jeder Nachrichtenteil, den Sie senden, in Rechnung gestellt. Der Preis, den Sie pro Nachrichtenteil zahlen, hängt vom Land oder der Region des Empfängers ab. Weitere Informationen zu den SNS-Preisen finden Sie unter Amazon-SNS-Preise.

Wenn Amazon SNS eine Anfrage zum Senden einer SMS-Nachricht akzeptiert (aufgrund eines Aufrufs der SendMessages-API oder einer gestarteten Kampagne oder Journey), wird Ihnen das Senden dieser Nachricht in Rechnung gestellt. Diese Aussage gilt auch dann, wenn der beabsichtigte Empfänger die Nachricht nicht erhält. Wenn beispielsweise die Telefonnummer des Empfängers nicht mehr vergeben ist oder wenn die Nummer, an die Sie die Nachricht gesendet haben, keine gültige Mobiltelefonnummer war, wird Ihnen das Senden der Nachricht dennoch in Rechnung gestellt.

Amazon SNS akzeptiert gültige Anfragen zum Senden von SMS-Nachrichten und versucht, diese zuzustellen. Aus diesem Grund sollten Sie überprüfen, ob die Telefonnummern, an die Sie Nachrichten senden, gültige Mobiltelefonnummern sind. Sie können den Service zur Telefonnummernüberprüfung von Amazon SNS verwenden, um festzustellen, ob eine Telefonnummer gültig ist und um welche Art von Nummer es sich handelt (z. B. Mobil-, Festnetz- oder VoIP-Nummer). Weitere Informationen finden Sie unter Überprüfen von Telefonnummern in Amazon Pinpoint im Amazon-Pinpoint-Entwicklerhandbuch.

Erstellen Sie Entwürfe unter Beachtung der Redundanz

Für geschäftskritische Messaging-Programme empfehlen wir, Amazon SNS in mehr als einer AWS-Region zu konfigurieren. Amazon SNS ist in verschiedenen AWS-Regionen verfügbar. Eine Liste der Regionen, in denen Amazon SNS erhältlich ist, finden Sie in der Allgemeine AWS-Referenz.

Die Telefonnummern, die Sie für SMS-Nachrichten verwenden – einschließlich Kurzwahl-, Langwahl-, gebührenfreien Nummern und 10DLC-Nummern – können in AWS-Regionen nicht repliziert werden. Damit Sie Amazon SNS in mehreren Regionen verwenden können, müssen Sie daher in jeder Region, in der Sie Amazon SNS verwenden möchten, separate Telefonnummern anfordern. Wenn Sie beispielsweise eine Kurzwahlnummer verwenden, um Textnachrichten an Empfänger in den USA zu senden, müssen Sie in jeder AWS-Region, die Sie verwenden möchten, jeweils separate Kurzwahlnummern anfordern.

In einigen Ländern können Sie auch mehrere Arten von Telefonnummern verwenden, um die Redundanz zu erhöhen. In den USA können Sie beispielsweise Kurzwahlnummern, 10DLC-Nummern und gebührenfreie Nummern anfordern. Jeder dieser Telefonnummerntypen nimmt einen anderen Weg zum Empfänger. Mehrere Telefonnummerntypen verfügbar zu haben – entweder in derselben AWS-Region oder über mehrere AWS-Regionen verteilt – bietet eine zusätzliche Form der Redundanz, die zur Verbesserung der Ausfallsicherheit beitragen kann.

SMS-Limits und -Einschränkungen

Informationen zu SMS-Limits und -Einschränkungen finden Sie unter SMS-Limits und -Einschränkungen in Amazon Pinpoint im Benutzerhandbuch zu Amazon Pinpoint.

Verwalten von Schlüsselwörtern für die Abmeldung

SMS-Empfänger können ihre Geräte verwenden, um Nachrichten abzubestellen, indem sie mit einem Schlüsselwort antworten. Weitere Informationen finden Sie unter Deaktivieren des Empfangs von SMS-Nachrichten.

CreatePool

Verwenden Sie die API-Aktion `CreatePool`, um einen neuen Pool zu erstellen und diesem Pool eine festgelegte Ursprungsidentität zuzuordnen. Weitere Informationen finden Sie unter CreatePool in Amazon-Pinpoint-SMS- und Sprachnachrichten-API.

PutKeyword

Verwenden Sie die API-Aktion `PutKeyword`, um eine Schlüsselwortkonfiguration für eine Ursprungstelefonnummer oder einen Pool zu erstellen oder zu aktualisieren. Weitere Informationen finden Sie unter PutKeyword in Amazon-Pinpoint-SMS- und Sprachnachrichten-API.

Verwalten von Nummerneinstellungen

Sie können die Optionen im Abschnitt Number settings (Nummerneinstellungen) auf der Seite zu SMS- und Sprach-Einstellungen verwenden, um die Einstellungen für die dedizierten Kurzwahlnummern und langen Nummern zu verwalten, die Sie vom AWS-Support angefordert haben und die Ihrem Konto zugeordnet sind. Weitere Informationen finden Sie unter Verwalten von Nummerneinstellungen im Benutzerhandbuch zu Amazon Pinpoint.

SMS-Zeichenbeschränkungen in Amazon SNS

Eine einzelne SMS-Nachricht kann bis zu 140 Bytes enthalten. Die Anzahl der Zeichen für eine einzelne SMS-Nachricht hängt vom Typ der in der Nachricht verwendeten Zeichen ab.

Wenn Ihre Nachricht nur Zeichen aus dem Zeichensatz GSM 03.38, auch als GSM 7-Bit-Alphabet bekannt, enthält, sind bis zu 160 Zeichen zulässig. Enthält Ihre Nachricht nicht im Zeichensatz GSM 03.38 enthaltene Zeichen, sind 70 Zeichen zulässig. Wenn Sie eine SMS-Nachricht senden, bestimmt Amazon SNS automatisch die effizienteste Kodierung.

Enthält eine Nachricht mehr als die maximale Anzahl von Zeichen, wird die Nachricht in mehrere Teile aufgeteilt. Wenn Nachrichten in mehrere Teile aufgeteilt werden, enthält jedes Teil zusätzliche Informationen über den Nachrichtenteil, der ihm vorangestellt ist. Empfängt das Gerät des Empfängers dann die so getrennten Nachrichtenteile, verwendet es diese zusätzlichen Informationen, um alle Nachrichtenteile in der richtigen Reihenfolge anzuzeigen. Je nach Mobilfunkanbieter und Gerät des Empfängers können mehrere Nachrichten als eine Nachricht oder als Sequenz von separaten Nachrichten angezeigt werden. Daher verringert sich die Anzahl der Zeichen in den einzelnen Nachrichten auf 153 (für Nachrichten, die nur GSM 03.38 Zeichen enthalten) oder auf 67 (für Nachrichten, die andere Zeichen enthalten). Sie können vor dem Senden schätzen, wie viele Nachrichtenteile Ihre Nachricht enthält, indem Sie einen SMS-Längenrechner verwenden (es sind mehrere online verfügbar). Die maximal unterstützte Größe einer Nachricht beträgt 1600 GSM-Zeichen oder 630 Nicht-GSM-Zeichen. Weitere Informationen zum Durchsatz und zur Nachrichtengröße finden Sie unter Begrenzungen für SMS-Zeichen in Amazon Pinpoint im Amazon-Pinpoint-Benutzerhandbuch.

Wenn Sie die Anzahl der Nachrichtenteile für jede gesendete Nachricht anzeigen möchten, sollten Sie zuerst die Ereignisstreamingeinstellungen aktivieren. Dann erzeugt Amazon SNS ein `_SMS.SUCCESS`-Ereignis, sobald die Nachricht an den Mobilfunkanbieter des Empfängers zugestellt wurde. Der `_SMS.SUCCESS`-Ereignisdatensatz enthält ein Attribut namens `attributes.number_of_message_parts`. Dieses Attribut gibt die Anzahl der Nachrichtenteile an, die in der Nachricht enthalten sind.

> **Important**
>
> Wenn Sie eine Nachricht mit mehr als einem Nachrichtenteil senden, wird Ihnen die Anzahl der enthaltenen Nachrichtenteile in Rechnung gestellt.

Zeichensatz GSM 03.38

In der folgenden Tabelle sind alle Zeichen aus dem Zeichensatz GSM 03.38 aufgeführt. Wenn Sie eine Nachricht senden, die nur Zeichen aus der folgenden Tabelle enthält, kann die Nachricht bis zu 160 Zeichen enthalten.

GSM 03.38-Standardzeichen												
A	B	C	D	E	F	G	H	I	J	K	L	M
N	O	P	Q	R	S	T	U	V	W	X	Y	Z
a	b	c	d	e	f	g	h	i	j	k	l	m
n	o	p	q	r	S	t	u	V	w	x	y	z
à	Å	å	Ä	ä	Ç	É	é	è	ì	Ñ	ñ	ò
Ø	ø	Ö	ö	ù	Ü	ü	Æ	æ	ß	0	1	2
3	4	5	6	7	8	9	&	*	@	:	,	¤
$	=	!	>	#	-	¡	¿	(	<	%	.	+
£	?	"	)	§	;	'	/	_	¥	Δ	Φ	Γ
Λ	Ω	Π	Ψ	Σ	Θ	Ξ						

Der Zeichensatz GSM 03.38 enthält neben den in der vorherigen Tabelle gezeigten Symbole noch weitere. Diese Zeichen zählen jedoch als zwei Zeichen, da sie alle ein unsichtbares Escape-Zeichen enthalten:

- ^
- {
- }
- \
- [
-]
- ~
- |
- €

Darüber hinaus enthält der Zeichensatz GSM 03.38 die folgenden Zeichen, die nicht gedruckt werden:

- Ein Leerzeichen.
- Ein Zeilenvorschub-Steuerzeichen, das das Ende eine Textzeile und den Beginn einer neuen markiert.
- Ein Wagenrücklauf-Steuerzeichen, das zum Anfang einer Textzeile führt (folgt in der Regel einem Zeilenvorschubzeichen).
- Ein Escape-Steuerzeichen, das automatisch zu den Zeichen in der vorherigen Liste hinzugefügt wird.

Beispielnachrichten

Dieser Abschnitt enthält Beispiele für SMS-Nachrichten. In diesem Abschnitt werden für jedes Beispiel die Gesamtzahl der Zeichen sowie die Anzahl der Nachrichtenteile einer Nachricht angezeigt.

Beispiel 1: Lange Nachricht, die nur Zeichen im GSM 03.38-Alphabet enthält

Die folgende Nachricht enthält nur Zeichen, die im GSM 03.38-Alphabet enthalten sind.

```
Hello Carlos. Your Example Corp. bill of $100 is now available. Autopay is
scheduled for next Thursday, April 9. To view the details of your bill, go
to https://example.com/bill1.
```

Die obige Nachricht enthält 180 Zeichen und muss daher in mehrere Nachrichtenteile aufgeteilt werden. Wenn eine Nachricht in mehrere Nachrichtenteile aufgeteilt wird, kann jedes Teil 153 GSM 03.38-Zeichen enthalten. Daher wird diese Nachricht in zwei Teilen gesendet.

Beispiel 2: Nachricht, die Multibyte-Zeichen enthält

Die folgende Nachricht enthält chinesische Zeichen, die nicht dem GSM 03.38-Alphabet angehören.

```
##################################################·####1994#7#########
```

Die obige Nachricht enthält 71 Zeichen. Da jedoch fast alle Zeichen in der Nachricht nicht im GSM 03.38-Alphabet enthalten sind, werden zwei Nachrichtenteile gesendet. Jedes dieser Nachrichtenteile kann maximal 67 Zeichen enthalten.

Beispiel 3: Nachricht, die ein einzelnes Nicht-GSM-Zeichen enthält

Die folgende Nachricht enthält ein Zeichen, das nicht Teil des GSM 03.38-Alphabets ist. In diesem Beispiel ist das Zeichen ein schließendes einfaches Anführungszeichen (’), das kein regulärer Apostroph (') ist. Textverarbeitungsanwendungen wie Microsoft Word ersetzen häufig automatisch Apostrophe durch schließende einfache Anführungszeichen. Wenn Sie Ihre SMS-Nachrichten in Microsoft Word entwerfen und in Amazon SNS einfügen, sollten Sie diese Sonderzeichen entfernen und durch Apostrophe ersetzen.

```
John: Your appointment with Dr. Salazar’s office is scheduled for next
Thursday at 4:30pm. Reply YES to confirm, NO to reschedule.
```

Die obige Nachricht enthält 130 Zeichen. Da sie jedoch das schließende einfache Anführungszeichen enthält, das nicht Teil des GSM 03.38-Alphabets ist, werden zwei Nachrichtenteile gesendet.

Wenn Sie das schließende einfache Anführungszeichen in dieser Nachricht durch ein Apostroph (das im GSM 03.38-Alphabet enthalten ist) ersetzen, wird nur eine Nachricht gesendet.

Mobile Push-Benachrichtigungen

Mit Amazon SNS haben Sie die Möglichkeit, Push-Benachrichtigungen direkt an Apps auf mobilen Geräten zu senden. Push-Benachrichtigungen, die an einen mobilen Endpunkt gesendet wurden,

können in der mobilen App als Benachrichtigungen, Abzeichenaktualisierungen oder sogar als akustische Warnungen erfolgen.

Themen

Funktionsweise von Benutzerbenachrichtigungen

Senden Sie Push-Benachrichtigungen an mobile Geräte und Desktops mithilfe einer der folgenden unterstützten Push-Benachrichtigungsservices:

- Amazon Device Messaging (ADM)
- Apple Push Notification Service (APNs) für iOS und Mac OS X
- Baidu Cloud Push (Baidu)
- Firebase Cloud Messaging (FCM)
- Microsoft Push Notification Service für Windows Phone (MPNS)
- Windows Push Notification Services (WNS)

Push-Benachrichtigungs-Services wie APNs und FCM unterhalten mit jeder App und jedem zugehörigen Mobilgerät, das für die Nutzung der Services registriert ist, eine Verbindung. Wenn eine App oder ein Mobilgerät registriert werden, gibt der Push-Benachrichtigungsservice ein Gerätetoken zurück. Amazon SNS verwendet das Gerätetoken, um einen mobilen Endpunkt zu

erstellen, zu dem mobile Push-Benachrichtigungen direkt gesendet werden können. Damit Amazon SNS mit den verschiedenen Push-Benachrichtigungs-Services kommunizieren kann, müssen Sie Amazon SNS Ihre Anmeldeinformationen für die Push-Benachrichtigungs-Services mitteilen, um sie in Ihrem Namen zu verwenden. Weitere Informationen finden Sie unter Übersicht über den Benutzerbenachrichtigungsprozess.

Zusätzlich zum Senden von direkten Push-Benachrichtigungen, können Sie mit Amazon SNS auch Benachrichtigungen an mobile Endpunkte senden, die ein Thema abonniert haben. Das Konzept ist das gleiche, als wenn andere Endpunkttypen, wie Amazon SQS, HTTP/S, E-Mail und SMS ein Thema abonnieren, wie unter Was ist Amazon SNS? beschrieben. Der Unterschied besteht darin, dass Amazon SNS über Push-Benachrichtigungsservices kommuniziert, damit die abonnierten mobilen Endpunkte die an das Thema gesendeten Benachrichtigungen empfangen können.

Übersicht über den Benutzerbenachrichtigungsprozess

1. Rufen Sie die Anmeldeinformationen und das Geräte-Token für die mobilen Plattformen ab, die Sie unterstützen möchten.
2. Verwenden Sie die Anmeldeinformationen, um ein Plattformanwendungsobjekt (`PlatformApplicationArn`) mit Amazon SNS zu erstellen. Weitere Informationen finden Sie unter Erstellen einer Plattformanwendung.
3. Verwenden Sie die zurückgegebenen Anmeldeinformationen, um für Ihre mobile App und Ihr Gerät ein Geräte-Token von Push-Benachrichtigungs-Service anzufordern. Das erhaltene Token repräsentiert Ihre mobile App und Ihr Gerät.
4. Erstellen Sie unter Verwendung des Geräte-Tokens und des `PlatformApplicationArn` mit Amazon SNS ein Endpunkt-Objekt für die Plattform (`EndpointArn`). Weitere Informationen finden Sie unter Erstellen eines Plattformendpunkts.
5. Verwenden Sie den `EndpointArn`, um eine Nachricht an eine App auf einem Mobilgerät zu veröffentlichen. Weitere Informationen finden Sie unter Veröffentlichung auf einem mobilen Gerät und der API Veröffentlichen in der Amazon Simple Notification Service API-Referenz.

Einrichten einer mobilen App

In diesem Abschnitt wird beschrieben, wie Sie AWS Management Console mit den unter Voraussetzungen für Amazon SNS-Benutzerbenachrichtigungen beschriebenen Informationen mobile Anwendungen einrichten.

Themen

Voraussetzungen für Amazon SNS-Benutzerbenachrichtigungen

Um die mobilen Amazon SNS-Push-Benachrichtigungen zu verwenden, benötigen Sie Folgendes:

- Anmeldeinformationen, um mit einem der unterstützten Push-Benachrichtigungs-Services (ADM, APNs, Baidu, FCM, MPNS oder WNS) eine Verbindung herzustellen.
- Ein Geräte-Token oder eine Registrierungs-ID für die mobile App und das Gerät.
- Entsprechende Amazon SNS-Konfiguration, sodass Push-Benachrichtigungen an mobile Endpunkte gesendet werden.
- Eine mobile App, die registriert und so konfiguriert ist, dass sie einen der unterstützten Push-Benachrichtigungsservices verwenden kann.

Für die Registrierung Ihrer Anwendung mit einem Push-Benachrichtigungsservice sind mehrere Schritte erforderlich. Amazon SNS benötigt einige Informationen, die Sie dem Push-Benachrichtigungsservice mitgeteilt haben, um Push-Benachrichtigungen direkt an den mobilen Endpunkt zu senden. Im Allgemeinen benötigen Sie die erforderlichen Anmeldeinformationen zur Verbindungsherstellung mit dem Push-Benachrichtigungsservice, das vom Push-Benachrichtigungsservice bereitgestellte Geräte-Token oder eine Registrierungs-ID (die Ihr Mobilgerät und Ihre mobile App repräsentieren) sowie die mobile App, die mit dem Push-Benachrichtigungsservice registriert wurde.

Die genaue Art der Anmeldeinformationen kann zwischen den mobilen Plattformen variieren, jedoch müssen sie in jedem Fall beim Erstellen einer Verbindung mit der Plattform angegeben werden. An jede mobile App wird ein Satz an Anmeldeinformationen ausgegeben und dieser muss verwendet werden, um eine Nachricht an eine beliebige Instance dieser App zu senden.

Die spezifischen Namen variieren, je nachdem welcher Push-Benachrichtigungsservice verwendet wird. Um beispielsweise APNs als Push-Benachrichtigungsservice nutzen zu können, benötigen

Sie ein Gerätetoken. Alternativ wird das Geräte-Token bei Verwendung von FCM Registrierungs-ID genannt. Beim Geräte-Token oder bei der Registrierungs-ID handelt es sich um eine Zeichenfolge, die vom Betriebssystem des Mobilgeräts an die Anwendung gesendet wird. Auf diese Weise wird eine Instance einer mobilen App auf einem bestimmten Mobilgerät gekennzeichnet und fungiert so als eindeutige Bezeichnung dieser App/Geräte-Paarung.

Amazon SNS speichert die Anmeldeinformationen (sowie einige andere Einstellungen) als Plattformanwendungsressource. Die Geräte-Token (erneut mit einigen zusätzlichen Einstellungen) werden als Objekte dargestellt, die auch als Plattformendpunkte bezeichnet werden. Jeder Plattformendpunkt gehört zu einer bestimmten Plattformanwendung und mit jedem Plattformendpunkt kann mithilfe der Anmeldeinformationen, die in der entsprechenden Plattformanwendung gespeichert sind, kommuniziert werden.

Die folgenden Abschnitte beschreiben die Voraussetzungen für jeden der unterstützten Push-Benachrichtigungsservices. Sobald Sie die erforderlichen Informationen erworben haben, können Sie Push-Benachrichtigungen mithilfe der AWS Management Console oder den mobilen Amazon SNS-Push-APIs versenden. Weitere Informationen finden Sie unter Übersicht über den Benutzerbenachrichtigungsprozess.

Erstellen einer Plattformanwendung

Damit Amazon SNS Benachrichtigungen an mobile Endpunkte senden kann, unabhängig davon, ob dies direkt geschieht oder bei Endpunkten, die ein Thema abonniert haben, müssen Sie zunächst eine Plattformanwendung erstellen. Nachdem die Anwendung bei AWS registriert ist, besteht der nächste Schritt darin, einen Endpunkt für die App und das mobile Gerät zu erstellen. Der Endpunkt wird dann von Amazon SNS für das Senden von Benachrichtigungen an die App und an das Gerät verwendet.

So erstellen Sie eine Plattformanwendung

1. Melden Sie sich bei der Amazon SNS-Konsole an.
2. Wählen Sie im Navigationsbereich auf der linken Seite Mobile (Mobil) und danach Push notifications (Push-Benachrichtigungen) aus.
3. Wählen Sie im Abschnitt Platform applications (Plattformanwendungen) die Option Create platform applications (Plattformanwendung erstellen) aus.

   Eine Liste der AWS-Regionen, in denen Sie mobile Anwendungen erstellen können, finden Sie unter Unterstützte Regionen für mobile Anwendungen.

4. Geben Sie in das Feld Application name (Anwendungsname) einen Namen für die App ein.

   App-Namen dürfen nur große und kleine ASCII-Buchstaben, Zahlen, Unterstriche, Bindestriche und Punkte enthalten. Namen müssen auch 1–256 Zeichen lang sein.

5. Wählen Sie im Feld Push notification platform (Push-Benachrichtigungsplattform) die Plattform aus, bei der die App registriert ist, und geben Sie dann die entsprechenden Anmeldeinformationen ein.

   > **Note**
   >
   > Wenn Sie eine der Plattformen für Apple Push Notification Service (APNs) verwenden, können Sie zwischen Token- und Zertifikat-basierter Authentifizierung und dann Choose file (Datei wählen) auswählen, um die .p8- oder .p12-Datei (exportiert von Keychain Access) auf Amazon SNS hochzuladen.

6. Wählen Sie Create platform application (Plattformanwendung erstellen).

   Die App wird jetzt bei Amazon SNS registriert, ein Anwendungsobjekt wird für die ausgewählte Plattform erstellt und ein entsprechender PlatformApplicationArn zurückgegeben.

Erstellen eines Plattformendpunkts

Wenn eine App oder ein Mobilgerät bei einem Push-Benachrichtigungsservice registriert werden, gibt der Push-Benachrichtigungsservice ein Geräte-Token zurück. Amazon SNS verwendet das Gerätetoken, um einen mobilen Endpunkt zu erstellen, zu dem mobile Push-Benachrichtigungen direkt gesendet werden können. Weitere Informationen finden Sie unter Voraussetzungen für Amazon SNS-Benutzerbenachrichtigungen und Übersicht über den Benutzerbenachrichtigungsprozess.

In diesem Abschnitt wird der empfohlene Ansatz zum Erstellen eines Plattformendpunkts.

Themen

Erstellen eines Plattformendpunkts

Um Push-Benachrichtigungen zu einer App per Amazon SNS zu senden, muss zuerst das Geräte-Token der App bei Amazon SNS registriert werden, indem die "create platform endpoint"-Aktion aufgerufen wird. Diese Aktion verwendet den Amazon Resource Name (ARN) der Plattformanwendung und das Geräte-Token als Parameter und gibt den ARN des erstellten Plattformendpunkts zurück.

Die `CreatePlatformEndpoint`Aktion bewirkt Folgendes:

- Wenn der Plattformendpunkt bereits vorhanden ist, wird er nicht neu erstellt. Dem Aufrufer wird der ARN des vorhandenen Plattformendpunkts zurückgegeben.
- Wenn der Plattformendpunkt mit demselben Geräte-Token, aber anderen Einstellungen vorhanden ist, wird er nicht erneut erstellt. Es wird eine Ausnahme für den Aufrufer ausgelöst.
- Wenn der Plattformendpunkt nicht vorhanden ist, wird er erstellt. Dem Aufrufer wird der ARN des neu erstellten Plattformendpunkts zurückgegeben.

Sie sollten die Aktion nicht direkt bei jedem Start der Anwendung aufrufen. Sie sorgt nicht immer für die Bereitstellung eines funktionierenden Endpunkts. Dies kann zum Beispiel der Fall sein, wenn eine App auf einem Gerät deinstalliert und neu installiert wird und der Endpunkt für diese bereits vorhanden aber deaktiviert ist. Ein erfolgreicher Registrierungsprozess sollte folgendermaßen aussehen:

1. Stellen Sie sicher, dass ein Plattformendpunkt für diese App-Geräte-Kombination vorhanden ist.
2. Stellen Sie sicher, dass das Geräte-Token im Plattformendpunkt das neueste gültige Geräte-Token ist.
3. Stellen Sie sicher, dass der Plattformendpunkt aktiviert und bereit ist.

Pseudo-Code

Der folgende Pseudo-Code beschreibt ein empfohlenes Verfahren für die Erstellung eines funktionierenden, aktuellen, aktivierten Plattformendpunkts in einer Vielzahl von Startbedingungen. Dieser Ansatz funktioniert, wenn die App zum ersten Mal oder erneut registriert wird. Es ist egal, ob der Plattformendpunkt für diese App bereits vorhanden ist, ob der Plattformendpunkt aktiviert ist, ob er das richtige Geräte-Token hat und so weiter. Der Code kann mehrfach hintereinander aufgerufen werden. Er wird keine doppelten Plattformendpunkte erstellen oder einen vorhandenen Plattformendpunkt ändern, der bereits auf dem neuesten Stand und aktiviert ist.

```
retrieve the latest device token from the mobile operating system
if (the platform endpoint ARN is not stored)
  # this is a first-time registration
  call create platform endpoint
  store the returned platform endpoint ARN
endif

call get endpoint attributes on the platform endpoint ARN

if (while getting the attributes a not-found exception is thrown)
  # the platform endpoint was deleted
  call create platform endpoint with the latest device token
  store the returned platform endpoint ARN
else
  if (the device token in the endpoint does not match the latest one) or
      (get endpoint attributes shows the endpoint as disabled)
    call set endpoint attributes to set the latest device token and then enable the
 platform endpoint
  endif
endif
```

Dieser Ansatz kann jederzeit verwendet werden, wenn die App sich registrieren oder erneut registrieren möchte. Er kann auch beim Benachrichtigen von Amazon SNS über eine Änderung des Geräte-Tokens verwendet werden. In diesem Fall können Sie einfach eine Aktion mit dem neuesten Geräte-Token-Wert aufrufen. Es gibt einige wichtige Punkte zu diesem Ansatz:

- Es gibt zwei Fälle, in denen die „create platform endpoint“-Aktion aufgerufen werden kann. Sie kann direkt zu Beginn aufgerufen werden, wenn die App ihren eigenen Plattformendpunkt-ARN nicht kennt (z. B. während der Erstregistrierung). Sie kann auch aufgerufen werden, wenn der erste „get endpoint attributes“-Aktionsaufruf mit einer „Not-Found“-Ausnahme fehlschlägt (z. B. wenn die Anwendung ihren ARN kennt, dieser aber gelöscht wurde).
- Die „get endpoint attributes“-Aktion wird aufgerufen, um den Status des Plattformendpunkts zu überprüfen, obwohl der Plattformendpunkt gerade erstellt wurde. Dies geschieht, wenn der Plattformendpunkt bereits vorhanden, aber deaktiviert ist. In diesem Fall ist die „create platform endpoint“-Aktion erfolgreich, aber der Plattformendpunkt wird nicht aktiviert. Daher müssen Sie den Status des Plattformendpunkts vor der Rückgabe eines Erfolgs prüfen.

AWS SDK-Beispiel

Der folgende Code zeigt, wie der vorherige Pseudocode mithilfe der Amazon SNS SNS-Clients implementiert wird, die von den AWS SDKs bereitgestellt werden.

Um ein AWS SDK zu verwenden, müssen Sie es mit Ihren Anmeldeinformationen konfigurieren. Weitere Informationen finden Sie unter Freigegebene Konfigurations- und Anmeldeinformationsdateien im AWS -Referenzhandbuch zu SDKs und Tools.

CLI

AWS CLI

So erstellen Sie ein Plattformanwendungsendpunkt

Im folgenden `create-platform-endpoint`-Beispiel wird mithilfe des angegebenen Tokens ein Endpunkt für die angegebene Plattformanwendung erstellt.

```
aws sns create-platform-endpoint \
    --platform-application-arn arn:aws:sns:us-west-2:123456789012:app/GCM/
MyApplication \
    --token EXAMPLE12345...
```

Ausgabe:

```
{
      "EndpointArn": "arn:aws:sns:us-west-2:1234567890:endpoint/GCM/
MyApplication/12345678-abcd-9012-efgh-345678901234"
}
```

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.CreatePlatformEndpointRequest;
import software.amazon.awssdk.services.sns.model.CreatePlatformEndpointResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 *
 * In addition, create a platform application using the AWS Management Console.
 * See this doc topic:
 *
 * https://docs.aws.amazon.com/sns/latest/dg/mobile-push-send-register.html
 *
 * Without the values created by following the previous link, this code examples
 * does not work.
 */

public class RegistrationExample {
    public static void main(String[] args) {
        final String usage = """

                Usage:     <token> <platformApplicationArn>

                Where:
                   token - The name of the FIFO topic.\s
                   platformApplicationArn - The ARN value of platform
 application. You can get this value from the AWS Management Console.\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String token = args[0];
        String platformApplicationArn = args[1];
```

```
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        createEndpoint(snsClient, token, platformApplicationArn);
    }

    public static void createEndpoint(SnsClient snsClient, String token, String
 platformApplicationArn) {
        System.out.println("Creating platform endpoint with token " + token);
        try {
            CreatePlatformEndpointRequest endpointRequest =
 CreatePlatformEndpointRequest.builder()
                    .token(token)
                    .platformApplicationArn(platformApplicationArn)
                    .build();

            CreatePlatformEndpointResponse response =
 snsClient.createPlatformEndpoint(endpointRequest);
            System.out.println("The ARN of the endpoint is " +
 response.endpointArn());
        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

Weitere Informationen finden Sie unter Mobile Push-API-Aktionen.

Fehlerbehebung

Wiederholtes Aufrufen von „Create Platform Endpoint“ mit einem veralteten Geräte-Token

Insbesondere bei FCM-Endpunkten denken Sie vielleicht, dass es am besten ist, das erste Geräte-Token zu speichern, das für die Anwendung ausgestellt wurde, und dann bei jedem Start der Anwendung den Create Platform-Endpunkt mit diesem Geräte-Token aufzurufen. Dies scheint sinnvoll, da die App sich nicht mehr um den Status des Geräte-Tokens kümmern muss und Amazon SNS das Geräte-Token automatisch auf den aktuellen Wert aktualisiert. Diese Lösung hat jedoch mehrere ernsthafte Probleme:

- Amazon SNS stützt sich auf das Feedback von FCM, um abgelaufene Geräte-Tokens durch neue Geräte-Tokens zu aktualisieren. FCM erhält einige Zeit Informationen zu alten Geräte-Tokens. Dies gilt jedoch nicht unbegrenzt. Sobald FCM die Verbindung zwischen dem alten Geräte-Token und dem neuen Geräte-Token vergessen hat, ist Amazon SNS nicht mehr in der Lage, das im Plattformendpunkt gespeicherte Geräte-Token auf den korrekten Wert zu aktualisieren. Stattdessen wird der Plattformendpunkt einfach deaktiviert.
- Die Plattformanwendung enthält mehrere Plattformendpunkte mit demselben Geräte-Token.
- Amazon SNS arbeitet mit einem Kontingent an Plattformendpunkten, die gleichzeitig mit demselben Geräte-Token gestartet werden können. Irgendwann wird die Erstellung neuer Endpunkte mit einer „Invalid Parameter“-Ausnahme und der Fehlermeldung „This endpoint is already registered with a different token.“ fehlschlagen.

Weitere Informationen zur Verwaltung von FCM-Endpunkten finden Sie unter. Firebase Cloud Messaging (FCM) -Endpunktverwaltung

Erneutes Aktivieren eines Plattformendpunktes mit einem ungültigen Geräte-Token

Wenn eine mobile Plattform (z. B. APNs oder FCM) Amazon SNS darüber informiert, dass das in der Veröffentlichungsanforderung verwendete Geräte-Token ungültig ist, deaktiviert Amazon SNS den Plattformendpunkt für dieses Geräte-Token. Amazon SNS weist dann nachfolgende Veröffentlichungen für das Geräte-Token zurück. Möglicherweise glauben Sie, Sie könnten einfach den Plattformendpunkt neu aktivieren und so weiter veröffentlichen, doch wird dies in den meisten Fällen nicht funktionieren. Die veröffentlichten Nachrichten werden nicht zugestellt und der Plattformendpunkt wird bald wieder deaktiviert.

Der Grund hierfür ist, dass das Geräte-Token des Plattformendpunkts wirklich ungültig ist. Zustellungen können nicht abgeschlossen werden, da es nicht mehr mit einer installierten App übereinstimmt. Bei der nächsten Veröffentlichung informiert die mobile Plattform Amazon SNS erneut darüber, dass das Geräte-Token ungültig ist. Amazon SNS wird den Plattformendpunkt wieder deaktivieren.

Um einen deaktivierten Plattformendpunkt zu reaktivieren, muss dieser einem gültigen Geräte-Token zugeordnet (über einen „set endpoint attributes“-Aktionsaufruf) sein und dann aktiviert werden. Nur dann werden Zustellungen an den Plattformendpunkt erfolgreich sein. Das erneute Aktivieren eines Plattformendpunkts ohne Aktualisierung des Geräte-Tokens funktioniert nur, wenn ein dem Endpunkt zugeordnetes Geräte-Token ungültig ist und wieder gültig wird. Dies kann zum Beispiel der Fall sein, wenn eine App deinstalliert und dann auf demselben Mobilgerät neu installiert wurde und dasselbe Geräte-Token erhält. Der oben gezeigte Ansatz geht so vor. Er stellt sicher, dass der

Plattformendpunkt erst dann neu aktiviert wird, wenn überprüft wurde, dass das ihm zugeordnete Geräte-Token im Moment verfügbar ist.

Hinzufügen von Geräte-Token oder Registrierungs-IDs

Wenn Sie zum ersten Mal eine App und ein mobiles Gerät mit einem Benachrichtigungs-Service wie Apple Push Notification Service (APNs) und Firebase Cloud Messaging (FCM) registrieren, werden die Geräte-Token oder Registrierungs-IDs vom Benachrichtigungs-Service zurückgegeben. Wenn Sie den Gerät-Token oder die Registrierungs-IDs zu Amazon SNS hinzu; sie werden sie mit der API `PlatformApplicationArn` verwendet, um einen Endpunkt für die App und das Gerät zu erstellen. Wenn Amazon SNS einen Endpunkt erstellt, wird `EndpointArn` zurückgegeben. Mithilfe des `EndpointArn` weiß Amazon SNS an welche App und an welches mobile Gerät die Benachrichtigung gesendet werden soll.

Sie können mithilfe der folgenden Methoden Geräte-Token und Registrierungs-IDs zu Amazon SNS hinzufügen:

- Fügen Sie mithilfe der AWS manuell ein einzelnes Token zu AWS Management Console hinzu
- Laden Sie mehrere Token mithilfe der API `CreatePlatformEndpoint` hoch
- Registrieren Sie Token aus Geräten, die künftig Ihre Apps installieren werden

So fügen Sie manuell ein Geräte-Token oder eine Registrierungs-ID hinzu

1. Melden Sie sich bei der Amazon-SNS-Konsole an.
2. Wählen Sie Mobil und dann Push-Benachrichtigungen aus.
3. Wählen Sie im Abschnitt Plattformanwendungen Ihre Anwendung aus und klicken Sie dann auf Bearbeiten. Wenn Sie noch keine Plattformanwendung erstellt haben, erstellen Sie jetzt eine. Anweisungen dazu finden Sie unter Erstellen einer Plattformanwendung.
4. Wählen Sie Endpunkt hinzufügen aus.
5. Geben Sie in das Feld Endpoint Token (Endpunkt-Token) entweder die Token-ID oder die Registrierungs-ID ein, je nachdem mit welchem Benachrichtigungsservice Sie arbeiten. Geben Sie z. B. bei ADM und FCM die Registrierungs-ID ein.
6. (Optional) Geben Sie in das Feld User Data beliebige Informationen ein, die dem Endpunkt zugeordnet werden können. Amazon SNS verwendet diese Daten nicht. Die Daten müssen im UTF-8-Format sein und weniger als 2 KB haben.
7. Wählen Sie schließlich Add Endpoints (Endpunkte hinzufügen).

Mit dem erstellten Endpunkt können Sie Nachrichten direkt an ein mobiles Gerät senden oder an mobile Geräte, die ein Thema abonniert haben.

So laden Sie mehrere Token mithilfe der API **`CreatePlatformEndpoint`** hoch

In den folgenden Schritten wird beschrieben, wie Sie die Beispiel-Java-App (`bulkupload`-Paket) verwenden, die von AWS bereitgestellt wird, um mehrere Token (Geräte-Token oder Registrierungs-ID) zu Amazon SNS hochzuladen. Sie können diese Beispielanwendung verwenden, um die ersten Schritte zum Hochladen der vorhandenen Token zu unternehmen.

> **Note**
>
> Die folgenden Schritte verwenden die IDE Eclipse Java. Die Schritte setzen voraus, dass Sie AWS SDK for Java installiert haben und dass Sie über die AWS-Sicherheitsanmeldeinformationen für Ihr AWS-Konto verfügen. Weitere Informationen finden Sie unter AWS SDK for Java. Weitere Informationen zu den Anmeldedaten finden Sie unter Wie erhalte ich Sicherheitsanmeldeinformationen? in der Allgemeine AWS-Referenz.

1. Laden Sie die Datei snsmobilepush.zip herunter und entpacken Sie sie.
2. Erstellen Sie ein neues Java Project (-Java-Projekt) in Eclipse.
3. Importieren Sie den Ordner `SNSSamples` in das oberste Verzeichnis des neu erstellten Java-Projekts. Wählen Sie in Eclipse mit der rechten Maustaste den Namen des Java-Projekts und wählen Sie dann Import (Importieren). Erweitern Sie General (Allgemeines) und wählen Sie File System (Dateisystem) und danach Next (Weiter). Navigieren Sie zum Ordner `SNSSamples`, wählen Sie OK und schließlich Finish (Fertig stellen).
4. Laden Sie eine Kopie der Bibliothek OpenCSV library herunter und fügen Sie sie zum Build-Pfad des Pakets `bulkupload` hinzu.
5. Öffnen Sie die Datei `BulkUpload.properties`, die im Paket `bulkupload` enthalten ist.
6. Fügen Sie Folgendes zu `BulkUpload.properties` hinzu:
   - Den `ApplicationArn`, dem Sie Endpunkte hinzufügen möchten.
   - Den absoluten Pfad für den Speicherort der CSV-Datei, die die Token enthält.

- Die Namen der CSV-Dateien (wie z. B. `goodTokens.csv` und `badTokens.csv`), die erstellt werden sollen, um die Token zu protokollieren, die Amazon SNS korrekt parst und diejenigen, die fehlschlagen.
- (Optional) Die Zeichen, um die Trennzeichen und Anführungszeichen in der CSV-Datei, die das Token enthält, festzulegen.
- (Optional) Die Anzahl der zu verwendenden Threads, um gleichzeitig Endpunkte zu erstellen. Die Standardeinstellung ist 1 Thread.

Ihre vollständigen `BulkUpload.properties` sollten wie folgt aussehen:

```
applicationarn:arn:aws:sns:us-west-2:111122223333:app/FCM/fcmpushapp
csvfilename:C:\\mytokendirectory\\mytokens.csv
goodfilename:C:\\mylogfiles\\goodtokens.csv
badfilename:C:\\mylogfiles\\badtokens.csv
delimiterchar:'
quotechar:"
numofthreads:5
```

7. Führen Sie die Anwendung BatchCreatePlatformEndpointSample.java aus, um die Token zu Amazon SNS hochzuladen.

   In diesem Beispiel werden die Endpunkte, die für die erfolgreich zu Amazon SNS hochgeladenen Token erstellt wurden, in `goodTokens.csv` protokolliert, während die manipulierten Token in `badTokens.csv` protokolliert werden. Außerdem sollten Sie die STD-OUT-Protokolle in der Eclipse-Konsole finden und sie sollten in etwa folgendermaßen aussehen:

```
<1>[SUCCESS] The endpoint was created with Arn arn:aws:sns:us-
west-2:111122223333:app/FCM/fcmpushapp/165j2214-051z-3176-b586-138o3d420071
<2>[ERROR: MALFORMED CSV FILE] Null token found in /mytokendirectory/mytokens.csv
```

Registrieren von Token aus Geräten, die künftig Ihre Apps installieren werden

Verwenden Sie eine der folgenden beiden Optionen:

- Verwenden Sie den Amazon Cognito-Service: Ihre mobile App benötigt Anmeldeinformationen, um Endpunkte zu erstellen, die Ihrer Amazon SNS-Plattformanwendung zugeordnet sind. Wir empfehlen die Verwendung temporärer Anmeldeinformationen, die nach einem bestimmten

Zeitraum ablaufen. Für die meisten Szenarien empfehlen wir, dass Sie Amazon Cognito zum Erstellen temporärer Sicherheitsanmeldeinformationen verwenden. Weitere Informationen finden Sie im Amazon Cognito Entwicklerhandbuch. Wenn Sie benachrichtigt werden möchten, sobald eine App sich bei Amazon SNS registriert, können Sie sich für den Empfang eines Amazon SNS-Ereignisses anmelden, das den neuen Endpunkt-ARN meldet. Sie können auch die API `ListEndpointByPlatformApplication` verwenden, um die vollständige Liste der bei Amazon SNS registrierten Endpunkte zu erhalten.

- Verwenden Sie einen Proxy-Server: Wenn Ihre Anwendungsinfrastruktur bereits so eingerichtet ist, dass Ihre mobilen Apps bei jeder Installation aufgerufen werden und sich registrieren, können Sie diese Einstellungen weiterverwenden. Der Server fungiert als Proxy und leitet das Geräte-Token zusammen mit den zu speichernden Benutzerdaten an die mobilen Amazon SNS-Push-Benachrichtigungen weiter. Zu diesem Zweck verbindet sich der Proxy-Server mit Amazon SNS wobei er Ihre AWS-Anmeldeinformationen und den API-Aufruf `CreatePlatformEndpoint` zum Hochladen der Token-Informationen verwendet. Der neu erstellte Amazon-Ressourcenname (ARN) für den Endpunkt wird zurückgegeben und Ihr Server kann ihn für spätere Veröffentlichungsaufrufe an Amazon SNS-Aufrufe speichern.

Apple-Authentifizierungsmethoden

Sie können Amazon SNS ermächtigen, Push-Benachrichtigungen an Ihre iOS- oder macOS-App zu senden, indem Sie Informationen bereitstellen, die Sie als Entwickler der App identifizieren. Um sich zu authentifizieren, geben Sie entweder einen Schlüssel oder ein Zertifikat beim Erstellen einer Plattformanwendung an, die Sie beide von Ihrem Apple Developer-Konto aus erhalten können.

Signierschlüssel für Token

Ein privater Signierschlüssel, den Amazon SNS verwendet, um Authentifizierungstoken für Apple Push Notification Service (APNs) zu signieren.

Wenn Sie einen Signierschlüssel bereitstellen, verwendet Amazon SNS ein Token zur Authentifizierung mit APNs für jede Push-Benachrichtigung, die Sie senden. Mit Ihrem Signierschlüssel können Sie Push-Benachrichtigungen an Produktions- und Sandbox-Umgebungen von APNs senden.

Ihr Signierschlüssel läuft nicht ab und Sie können denselben Signierschlüssel für mehrere Apps verwenden. Weitere Informationen finden Sie unter Communicate with APNs using authentication tokens (Mithilfe von Authentifizierungstoken mit APNs kommunizieren) in der Apple Developer Account-Hilfe.

Zertifikat

Ein TLS-Zertifikat, das Amazon SNS für die Authentifizierung mit APNs verwendet, wenn Sie Push-Benachrichtigungen senden. Sie erhalten das Zertifikat aus Ihrer Apple-Entwicklerkonto.

Zertifikate laufen nach einem Jahr ab. Wenn dies geschieht, müssen Sie ein neues Zertifikat erstellen und es Amazon SNS bereitstellen. Weitere Informationen finden Sie unter Establishing a Certificate-Based Connection to APNs (Aufbau einer zertifikatbasierten Verbindung zu APNs) auf der Apple Developer-Website.

So verwalten Sie die APNs-Einstellungen mit der AWS-Managementkonsole

1. Melden Sie sich bei der Amazon-SNS-Konsole an.
2. Wählen Sie unter Mobil Push-Benachrichtigungen aus.
3. Wählen Sie die Anwendung aus, für die Sie die APNs-Einstellungen bearbeiten möchten, und wählen Sie dann Bearbeiten aus.
4. Auf der Seite Bearbeiten wählen Sie als Authentifiziertyp entweder Token oder Zertifikat aus.
5. Laden Sie die entsprechenden Anmeldeinformationen für das Zertifikat oder den Token-Signaturschlüssel. Sie finden diese Informationen in Ihrem Apple-Entwicklerkonto.
6. Abhängig von dem ausgewählten Authentifizierungstyp führen Sie einen der folgenden Schritte aus:
   - Wenn Sie Token auswählen, stellen Sie die folgenden Informationen aus Ihrem Apple Developer-Konto bereit. Amazon SNS benötigt diese Information, um Authentifizierungstoken zu generieren.
     - Signierschlüssel – Der Signierschlüssel für Authentifizierungstoken von Ihrem Apple Developer-Konto, den Sie als .p8-Datei herunterladen. Bei Apple können Sie Ihren Signierschlüssel nur einmal herunterladen.
     - Signing key ID (Signierschlüssel-ID) – Die Ihrem Signaierchlüssel zugeordnete ID. Amazon SNS benötigt diese Information, um Authentifizierungstoken zu generieren. Um diesen Wert in Ihrem Apple Developer-Konto zu finden, wählen Sie Certificates, IDs & Profiles (Zertifikate, IDs und Profile) und dann Ihren Schlüssel im Abschnitt Keys (Schlüssel) aus.
     - Team identifier (Team-Kennung) – Die Ihrem Apple Developer-Kontoteam zugewiesene ID. Sie finden diesen Wert auf der Seite Membership (Mitgliedschaft).

- Bundle identifier (Bundle-Kennung) – Die Ihrer App zugeordnete ID. Um diesen Wert zu finden, wählen Sie Certificates, IDs & Profiles (Zertifikate, IDs und Profile), dann App IDs (App-IDs) im Abschnitt Identifiers (Kennungen) und dann Ihre App aus.

- Wenn Sie die Option Certificate credentials (Zertifikatanmeldeinformationen) auswählen, geben Sie die folgenden Informationen an:

  - SSL certificate (SSL-Zertifikat) – Die .p12-Datei für Ihr TLS-Zertifikat. Sie können diese Datei von Keychain Access exportieren, nachdem Sie Ihr Zertifikat von Ihrem Apple Developer-Konto heruntergeladen und installiert haben.
  - Certificate password (Zertifikatpasswort) – Wenn Sie Ihrem Zertifikat ein Passwort zugewiesen haben, geben Sie es hier an.

7. Wenn Sie die gewünschten Änderungen vorgenommen haben, wählen Sie Save changes (Änderungen speichern) aus.

Firebase Cloud Messaging (FCM)-Authentifizierungsmethoden

In diesem Thema wird beschrieben, wie Sie die erforderlichen FCM-API-Anmeldeinformationen (HTTP v1) von Google abrufen, um sie mit der AWS API zu verwenden, AWS CLI und die AWS Management Console.

Themen

> **Important**
>
> 20. Juni 2023 — Google hat seine ältere HTTP-API für Firebase Cloud Messaging (FCM) als veraltet eingestuft. Amazon SNS unterstützt jetzt die Lieferung an alle Gerätetypen mithilfe der FCM-HTTP-v1-API. Wir empfehlen Ihnen, Ihre vorhandenen mobilen Push-Anwendungen am oder vor dem 1. Juni 2024 auf die neueste FCM-HTTP-v1-API zu migrieren, um Unterbrechungen zu vermeiden.
> 18. Januar 2024 — Amazon SNS hat die Unterstützung für die FCM-HTTP-v1-API für die mobile Übermittlung von Push-Benachrichtigungen an Android-Geräte eingeführt.

26. März 2024 — Amazon SNS unterstützt die FCM-HTTP-v1-API für Apple-Geräte und Webpush-Ziele. Wir empfehlen Ihnen, Ihre vorhandenen mobilen Push-Anwendungen am oder vor dem 1. Juni 2024 auf die neueste FCM-HTTP-v1-API zu migrieren, um Anwendungsunterbrechungen zu vermeiden.

Sie können Amazon SNS ermächtigen, Push-Benachrichtigungen an Ihre Anwendungen zu senden, indem Sie Informationen bereitstellen, die Sie als Entwickler der App identifizieren. Geben Sie zur Authentifizierung entweder einen API-Schlüssel oder ein Token an, wenn Sie eine Plattformanwendung erstellen. Sie können die folgenden Informationen von Ihrer Firebase-Anwendungskonsole abrufen:

API-Schlüssel

Der API-Schlüssel ist eine Anmeldeinformation, die beim Aufrufen der Legacy-API von Firebase verwendet wird. Die FCM-Legacy-APIs werden am 20. Juni 2024 von Google entfernt. Wenn Sie derzeit einen API-Schlüssel als Plattformanmeldeinformation verwenden, können Sie die Plattformanmeldeinformation aktualisieren, indem Sie Token als Option auswählen und die zugehörige JSON-Datei für Ihre Firebase-Anwendung hochladen.

Token

Beim Aufrufen der HTTP v1-API wird ein kurzlebiges Zugriffstoken verwendet. Dies ist die von Firebase vorgeschlagene API zum Senden von Push-Benachrichtigungen. Zum Generieren von Zugriffstoken stellt Firebase Entwicklern eine Reihe von Anmeldeinformationen in Form einer privaten Schlüsseldatei (auch als Datei service.json bezeichnet) zur Verfügung.

Voraussetzung

Sie benötigen Ihre FCM-service.json-Anmeldeinformationen, bevor Sie mit der Verwaltung von FCM-Einstellungen in Amazon SNS beginnen können. Informationen zum Abrufen Ihrer service.json-Anmeldeinformationen finden Sie unter Migrieren Sie von älteren FCM-APIs auf HTTP v1 in der Google-Firebase-Dokumentation.

Verwalten von FCM-Einstellungen (API)

Mithilfe der API können Sie FCM-Push-Benachrichtigungen erstellen. AWS Die Anzahl und Größe der Amazon SNS SNS-Ressourcen in einem AWS Konto sind begrenzt. Weitere Informationen finden Sie im Allgemeine AWS-Referenz Handbuch unter Endpunkte und Kontingente von Amazon Simple Notification Service.

Um eine FCM-Push-Benachrichtigung zusammen mit einem Amazon SNS SNS-Thema (AWS API) zu erstellen

Bei Verwendung von Schlüssel-Anmeldeinformationen lautet `PlatformCredential` `API key`. Bei Verwendung von Token-Anmeldeinformationen ist `PlatformCredential` eine private Schlüsseldatei im JSON-Format:

- `CreatePlatformApplication`

Um einen FCM-Anmeldeinformationstyp für ein vorhandenes Amazon SNS SNS-Thema (API) abzurufen AWS

Ruft den Anmeldeinformationstyp `"AuthenticationMethod": "Token"` oder `"AuthenticationMethod": "Key"` ab:

- GetPlatformApplicationAttributes

So legen Sie ein FCM-Attribut für ein vorhandenes Amazon-SNS-Thema fest (AWS -API)

Legt das FCM-Attribut fest:

- SetPlatformApplicationAttributes

Verwalten von FCM-Einstellungen (CLI)

Sie können FCM-Push-Benachrichtigungen mit der AWS Command Line Interface (CLI) erstellen. Die Anzahl und Größe der Amazon SNS SNS-Ressourcen in einem AWS Konto sind begrenzt. Weitere Informationen finden Sie unter Amazon Simple Notification Service-Endpunkte und -Kontingente.

So erstellen Sie eine FCM-Push-Benachrichtigung zusammen mit einem Amazon-SNS-Thema (AWS CLI)

Bei Verwendung von Schlüssel-Anmeldeinformationen lautet `PlatformCredential` `API key`. Bei Verwendung von Token-Anmeldeinformationen ist `PlatformCredential` eine private Schlüsseldatei im JSON-Format. Bei Verwendung der AWS CLI muss die Datei im Zeichenkettenformat vorliegen und Sonderzeichen müssen ignoriert werden. Um die Datei korrekt zu formatieren, empfiehlt Amazon SNS die Verwendung des folgenden Befehls: `SERVICE_JSON=`jq @json <<< cat service.json``

- create-platform-application

So rufen Sie einen FCM-Anmeldeinformationstyp für ein vorhandenes Amazon-SNS-Thema ab (AWS CLI)

Ruft den Anmeldeinformationstyp `"AuthenticationMethod": "Token"` oder `"AuthenticationMethod": "Key"` ab:

- get-platform-application-attributes

So legen Sie ein FCM-Attribut für ein vorhandenes Amazon-SNS-Thema fest (AWS CLI)

Legt das FCM-Attribut fest:

- set-platform-application-attributes

Verwalten von FCM-Einstellungen (Konsole)

Gehen Sie wie folgt vor, um die Anmeldeinformationen einzugeben, die Ihre Anwendung für die Verbindung mit FCM verwendet.

1. Melden Sie sich bei der Amazon-SNS-Konsole an.
2. Wählen Sie unter Mobil Push-Benachrichtigungen aus.
3. Wählen Sie eine vorhandene FCM-Anwendung und anschließend Bearbeiten aus. Wenn Sie noch keine Plattformanwendung erstellt haben, beachten Sie die Informationen unter Erstellen einer Plattformanwendung.
4. Wählen Sie auf der Seite Bearbeiten für Firebase-Cloud-Messaging-Anmeldeinformationen Token oder Schlüssel aus. Sie können die folgenden Informationen von Ihrer Firebase-Anwendungskonsole abrufen.
   - Wenn Sie Token auswählen, laden Sie eine gültige private Schlüsseldatei hoch. Der Inhalt dieser Datei wird verwendet, um beim Senden von Benachrichtigungen kurzlebige Zugriffstoken zu generieren.
   - Wenn Sie Schlüssel wählen, geben Sie den Google-API-Schlüssel ein.
5. Wenn Sie die gewünschten Änderungen vorgenommen haben, wählen Sie Save changes (Änderungen speichern) aus.

Verwandte Themen

- Verwenden von Google Firebase Cloud Messaging (FCM) v1-Nutzlasten in Amazon SNS

Firebase Cloud Messaging (FCM) -Endpunktverwaltung

Themen

Verwaltung und Pflege von Geräte-Token

Sie können die Zustellbarkeit der Push-Benachrichtigungen Ihrer mobilen Anwendung sicherstellen, indem Sie die folgenden Schritte ausführen:

1. Speichern Sie alle Geräte-Token, die entsprechenden Amazon SNS SNS-Endpunkt-ARNs und Zeitstempel auf Ihrem Anwendungsserver.
2. Entfernen Sie alle veralteten Token und löschen Sie die entsprechenden Amazon SNS SNS-Endpunkt-ARNs.

Beim ersten Start Ihrer App erhalten Sie ein Geräte-Token (auch als Registrierungstoken bezeichnet) für das Gerät. Dieses Geräte-Token wird vom Betriebssystem des Geräts geprägt und ist an Ihre FCM-Anwendung gebunden. Sobald Sie dieses Geräte-Token erhalten haben, können Sie es bei Amazon SNS als Plattformendpunkt registrieren. Wir empfehlen Ihnen, das Geräte-Token, den ARN des Amazon SNS SNS-Plattformendpunkts und den Zeitstempel zu speichern, indem Sie sie auf Ihrem Anwendungsserver oder einem anderen persistenten Speicher speichern. Informationen zum Einrichten Ihrer FCM-Anwendung zum Abrufen und Speichern von Gerätetokens finden Sie in der Firebase-Dokumentation von Google unter Registrierungstoken abrufen und speichern.

Es ist wichtig, dass Sie Tokens verwalten up-to-date . Die Geräte-Token Ihres Benutzers können sich unter den folgenden Bedingungen ändern:

1. Die mobile Anwendung wird auf einem neuen Gerät wiederhergestellt.
2. Der Benutzer deinstalliert oder aktualisiert die Anwendung.
3. Der Benutzer löscht die Anwendungsdaten.

Wenn sich Ihr Geräte-Token ändert, empfehlen wir Ihnen, den entsprechenden Amazon SNS SNS-Endpunkt mit dem neuen Token zu aktualisieren. Dadurch kann Amazon SNS die Kommunikation mit dem registrierten Gerät fortsetzen. Sie können dies tun, indem Sie den folgenden Pseudocode

in Ihrer mobilen Anwendung implementieren. Es beschreibt eine empfohlene Vorgehensweise für die Erstellung und Wartung aktivierter Plattformendpunkte. Dieser Ansatz kann bei jedem Start der mobilen Anwendungen oder als geplante Aufgabe im Hintergrund ausgeführt werden.

Pseudo-Code

Verwenden Sie den folgenden FCM-Pseudocode, um Gerätetokens zu verwalten und zu verwalten.

```
retrieve the latest token from the mobile OS
if (endpoint arn not stored)
    # first time registration
    call CreatePlatformEndpoint
    store returned endpoint arn
endif

call GetEndpointAttributes on the endpoint arn

if (getting attributes encountered NotFound exception)
    #endpoint was deleted
    call CreatePlatformEndpoint
    store returned endpoint arn
else
    if (token in endpoint does not match latest) or
        (GetEndpointAttributes shows endpoint as disabled)
        call SetEndpointAttributes to set the
                    latest token and enable the endpoint
    endif
endif
```

Weitere Informationen zu den Anforderungen für Token-Updates finden Sie unter Regelmäßiges Aktualisieren von Tokens in der Firebase-Dokumentation von Google.

Erkennung ungültiger Token

Wenn eine Nachricht mit einem ungültigen Geräte-Token an einen FCM v1-Endpunkt gesendet wird, erhält Amazon SNS eine der folgenden Ausnahmen:

- `UNREGISTERED`(HTTP 404) — Wenn Amazon SNS diese Ausnahme empfängt, erhalten Sie ein Ereignis mit einem Zustellungsfehler mit dem Wert `FailureType` von`InvalidPlatformToken`, und ein `FailureMessage` dem Endpunkt zugeordnetes Plattform-Token ist ungültig. Amazon SNS deaktiviert Ihren Plattformendpunkt, wenn eine Lieferung mit dieser Ausnahme fehlschlägt.

- `INVALID_ARGUMENT`(HTTP 400) — Wenn Amazon SNS diese Ausnahme empfängt, bedeutet dies, dass das Geräte-Token oder die Nachrichtennutzlast ungültig ist. Weitere Informationen finden Sie ErrorCodein der Firebase-Dokumentation von Google.

Da Amazon SNS in beiden Fällen zurückgesendet werden `INVALID_ARGUMENT` kann, gibt Amazon SNS ein of zurück`InvalidNotification`, und ein `FailureType` `FailureMessage` of der Benachrichtigungstext ist ungültig. Wenn Sie diesen Fehler erhalten, überprüfen Sie, ob Ihre Payload korrekt ist. Wenn sie korrekt ist, überprüfen Sie, ob das Geräte-Token korrekt ist up-to-date. Amazon SNS deaktiviert Ihren Plattformendpunkt nicht, wenn eine Lieferung fehlschlägt, mit dieser Ausnahme.

Ein weiterer Fall, in dem ein `InvalidPlatformToken` Zustellungsfehler auftritt, ist, wenn das registrierte Geräte-Token nicht zu der Anwendung gehört, die versucht, diese Nachricht zu senden. In diesem Fall gibt Google einen SENDER_ID_MISMATCH-Fehler zurück. Amazon SNS deaktiviert Ihren Plattformendpunkt, wenn eine Lieferung mit dieser Ausnahme fehlschlägt.

Alle beobachteten Fehlercodes, die von der FCM v1-API empfangen wurden, stehen Ihnen zur Verfügung, CloudWatch wenn Sie die Versandstatusprotokollierung für Ihre Anwendung einrichten.

Informationen zum Empfang von Zustellungsereignissen für Ihre Anwendung finden Sie unterVerfügbare Anwendungsereignisse.

Veraltete Token entfernen

Token gelten als veraltet, sobald die Nachrichtenzustellung an das Endgerät fehlschlägt. Amazon SNS legt diese veralteten Token als deaktivierte Endpunkte für Ihre Plattformanwendung fest. Wenn Sie auf einem deaktivierten Endpunkt veröffentlichen, gibt Amazon SNS ein `EventDeliveryFailure` Ereignis mit dem Wert `FailureType` of zurück`EndpointDisabled`, und `FailureMessage` der Endpunkt ist deaktiviert. Informationen zum Empfang von Zustellungsereignissen für Ihre Anwendung finden Sie unterVerfügbare Anwendungsereignisse.

Wenn Sie diesen Fehler von Amazon SNS erhalten, müssen Sie das veraltete Token in Ihrer Plattformanwendung entfernen oder aktualisieren.

Senden mobiler Push-Benachrichtigungen

In diesem Abschnitt wird beschrieben, wie Sie mobile Push-Benachrichtigungen senden.

Themen

- So veröffentlichen Sie in einem Thema

- Veröffentlichung auf einem mobilen Gerät
- Veröffentlichung mit plattformspezifischen Payloads

So veröffentlichen Sie in einem Thema

Sie können Amazon SNS auch verwenden, um Nachrichten an mobile Endpunkte zu senden, die ein Thema abonniert haben. Das Konzept ist das gleiche, als wenn andere Endpunkttypen, wie Amazon SQS, HTTP/S, E-Mail und SMS ein Thema abonnieren, wie unter Was ist Amazon SNS? beschrieben. Der Unterschied besteht darin, dass Amazon SNS über Benachrichtigungsdienste wie Apple Push Notification Service (APNS) und Google Firebase Cloud Messaging (FCM) kommuniziert. Durch den Benachrichtigungsservice erhalten die abonnierten mobilen Endpunkte Benachrichtigungen, die an das Thema gesendet wurden.

Veröffentlichung auf einem mobilen Gerät

Senden Sie Amazon SNS-Push-Benachrichtigungen direkt an einen Endpunkt, der eine App auf einem mobilen Gerät repräsentiert.

So senden Sie eine direkte Nachricht

1. Melden Sie sich bei der Amazon-SNS-Konsole an.
2. Wählen Sie im Navigationsbereich Push notifications (Push-Benachrichtigungen) aus.
3. Wählen Sie auf der Seite Mobile Push-Benachrichtigungen im Abschnitt Plattformanwendungen beispielsweise den Namen der Anwendung aus ***MyApp***.
4. Wählen Sie auf der ***MyApp***Seite im Abschnitt Endpoints einen Endpoint aus und klicken Sie dann auf Nachricht veröffentlichen.
5. Geben Sie auf der Seite Publish message to endpoint (Nachricht am Endpunkt veröffentlichen) die Nachricht ein, die in der Anwendung auf dem mobilen Gerät erscheinen wird, und wählen Sie dann Publish message (Nachricht veröffentlichen).

   Amazon SNS sendet die Benachrichtigung an den Plattformbenachrichtigungsservice, der sie dann wiederum an die Anwendung sendet.

Veröffentlichung mit plattformspezifischen Payloads

Sie können die AWS Management Console oder Amazon SNS SNS-APIs verwenden, um benutzerdefinierte Nachrichten mit plattformspezifischen Nutzlasten an mobile Geräte zu senden.

Weitere Informationen zur Verwendung der Amazon SNS APIs finden Sie unter Mobile Push-API-Aktionen und der Datei `SNSMobilePush.java` in `snsmobilepush.zip`.

Themen

- Senden von Nachrichten im JSON-Format
- Senden plattformspezifischer Nachrichten
- Senden von Nachrichten an eine Anwendung auf mehreren Plattformen
- Senden von Nachrichten an APNs als Warn- oder Hintergrundbenachrichtigungen
- Verwenden von Google Firebase Cloud Messaging (FCM) v1-Nutzlasten in Amazon SNS

Senden von Nachrichten im JSON-Format

Wenn Sie plattformspezifische Nutzlasten senden, müssen die Daten als Schlüssel-Wert-Paar-Zeichenfolgen im JSON-Format vorliegen und mit Anführungszeichen maskiert sein.

Die folgenden Beispiele zeigen eine benutzerdefinierte Nachricht für die FCM-Plattform.

```
{
"GCM": "{\"fcmV1Message\": {\"message\": {\"notification\": {\"title\": \"Hello\",
 \"body\": \"This is a test.\"}, \"data\": {\"dataKey\": \"example\"}}}}"
}
```

Senden plattformspezifischer Nachrichten

Sie können nicht nur benutzerdefinierte Daten als Schlüssel-Wert-Paare, sondern auch plattformspezifische Schlüssel-Wert-Paare senden.

Das folgende Beispiel zeigt den Einschluss der FCM-Parameter `time_to_live` und `collapse_key` nach den Schlüsselwertpaaren der benutzerdefinierten Daten im FCM `data`-Parameter .

```
{
"GCM": "{\"fcmV1Message\": {\"message\": {\"notification\": {\"title\": \"TitleTest\",
 \"body\": \"Sample message for Android or iOS endpoints.\"}, \"data\":{\"time_to_live
\": 3600,\"collapse_key\":\"deals\"}}}}"
}
```

Eine Liste der von jedem der Push-Benachrichtigungsservices in Amazon SNS unterstützten Schlüssel-Wert-Paare finden Sie unter:

> **Important**
>
> Amazon SNS unterstützt jetzt die Firebase Cloud Messaging (FCM) HTTP v1-API zum Senden von mobilen Push-Benachrichtigungen an Android-Geräte.
> 26. März 2024 — Amazon SNS unterstützt die FCM-HTTP-v1-API für Apple-Geräte und Webpush-Ziele. Wir empfehlen Ihnen, Ihre vorhandenen mobilen Push-Anwendungen am oder vor dem 1. Juni 2024 auf die neueste FCM-HTTP-v1-API zu migrieren, um Anwendungsunterbrechungen zu vermeiden.

- Nutzlastschlüsselreferenz in der APNs-Dokumentation
- Firebase Cloud Messaging-HTTP-Protokoll in der FCM-Dokumentation
- Senden einer Nachricht in der ADM-Dokumentation

Senden von Nachrichten an eine Anwendung auf mehreren Plattformen

Wenn Sie eine Nachricht an eine Anwendung senden möchten, die auf Geräten für mehrere Plattformen installiert ist, so wie z. B. FCM und APNs, müssen Sie zuerst mit den mobilen Endpunkte ein Thema in Amazon SNS abonnieren und dann die Nachricht für das Thema veröffentlichen.

Das folgende Beispiel zeigt eine Nachricht, die an mobile Endpunkte auf APNs, FCM und ADM gesendet werden soll, die ein Thema abonniert haben:

```
{
  "default": "This is the default message which must be present when publishing a
 message to a topic. The default message will only be used if a message is not present
 for
one of the notification platforms.",
  "APNS": "{\"aps\":{\"alert\": \"Check out these awesome deals!\",\"url\":
\"www.amazon.com\"} }",
  "GCM": "{\"data\":{\"message\":\"Check out these awesome deals!\",\"url\":
\"www.amazon.com\"}}",
  "ADM": "{\"data\":{\"message\":\"Check out these awesome deals!\",\"url\":
\"www.amazon.com\"}}"
}
```

Senden von Nachrichten an APNs als Warn- oder Hintergrundbenachrichtigungen

Amazon SNS kann Nachrichten als `alert`- oder `background`-Benachrichtigungen an APNs senden (weitere Informationen finden Sie unter Pushing Background Updates to Your App in der APN-Dokumentation).

- Eine `alert` APNs-Benachrichtigung informiert den Benutzer, indem eine Warnmeldung angezeigt wird, ein Audiosignal wiedergegeben oder ein Abzeichen zum Symbol Ihrer Anwendung hinzugefügt wird.
- Eine `background` APNs-Benachrichtigung wird aktiviert oder weist Ihre Anwendung an, auf den Inhalt der Benachrichtigung zu reagieren, ohne den Benutzer zu informieren.

Angeben benutzerdefinierter APNs-Header-Werte

Wir empfehlen, benutzerdefinierte Werte für das `AWS.SNS.MOBILE.APNS.PUSH_TYPE` reservierte Nachrichtenattribut mithilfe der Amazon SNS `Publish` SNS-API-Aktion, AWS SDKs oder der anzugeben. AWS CLI Im folgenden CLI-Beispiel wird für das angegebene Thema für `content-available` `1` und für `apns-push-type` `background` festgelegt.

```
aws sns publish \
--endpoint-url https://sns.us-east-1.amazonaws.com \
--target-arn arn:aws:sns:us-east-1:123456789012:endpoint/APNS_PLATFORM/MYAPP/1234a567-bc89-012d-3e45-6fg7h890123i \
--message '{"APNS_PLATFORM":"{\"aps\":{\"content-available\":1}}"}' \
--message-attributes '{ \
  "AWS.SNS.MOBILE.APNS.TOPIC":
{"DataType":"String","StringValue":"com.amazon.mobile.messaging.myapp"}, \
  "AWS.SNS.MOBILE.APNS.PUSH_TYPE":{"DataType":"String","StringValue":"background"} \
  "AWS.SNS.MOBILE.APNS.PRIORITY":{"DataType":"String","StringValue":"5"}}', \
--message-structure json
```

Ableiten des APNs-Push-Typ-Headers aus der Nutzlast

Wenn Sie den `apns-push-type` APNs-Header nicht festlegen, setzt Amazon SNS den Header auf `alert` oder `background`. Dies ist abhängig vom `content-available`-Schlüssel im `aps`-Verzeichnis Ihrer JSON-formatierten APNs-Nutzlastkonfiguration.

> **Note**
>
> Amazon SNS kann nur `alert`- oder `background`-Header ableiten, auch wenn der Header `apns-push-type` auf andere Werte gesetzt werden kann.

- `apns-push-type` wird auf `alert` gesetzt
  - Wenn das `aps`-Verzeichnis einen auf 1 gesetzten `content-available`-Schlüssel und einen oder mehrere Schlüssel enthält, die Benutzerinteraktionen auslösen
  - Wenn das `aps`-Verzeichnis einen auf `0` gesetzten `content-available`-Schlüssel enthält oder wenn der `content-available`-Schlüssel nicht vorhanden ist
  - Wenn der Wert des `content-available`-Schlüssels keine Ganzzahl und kein boolescher Wert ist
- `apns-push-type` wird auf `background` gesetzt
  - Wenn das `aps`-Verzeichnis nur auf 1 gesetzte `content-available` und keine anderen Schlüssel enthält, die Benutzerinteraktionen auslösen.

> **Important**
>
> Wenn Amazon SNS ein Rohkonfigurationsobjekt für APNs als reine Hintergrundbenachrichtigung sendet, muss das `aps`-Verzeichnis einen auf 1 gesetzten `content-available`-Schlüssel enthalten. Auch wenn benutzerdefinierte Schlüssel eingeschlossen werden können, darf das `aps`-Verzeichnis keine Schlüssel enthalten, die Benutzerinteraktionen (z. B. Warnmeldungen, Abzeichen oder Audiosignale) auslösen.

Es folgt ein Beispiel für ein unformatiertes Konfigurationsobjekt.

```
{
  "APNS": "{\"aps\":{\"content-available\":1},\"Foo1\":\"Bar\",\"Foo2\":123}"
}
```

In diesem Beispiel legt Amazon SNS den APNs-Header `apns-push-type` für die Nachricht auf `background` fest. Wenn Amazon SNS erkennt, dass das `apn`-Verzeichnis den `content-available` Schlüssel festgelegt auf 1 enthält, und keine anderen Schlüssel enthält, die Benutzerinteraktionen auslösen können, legt es den Header auf `background`.

Verwenden von Google Firebase Cloud Messaging (FCM) v1-Nutzlasten in Amazon SNS

Amazon SNS unterstützt die Verwendung der FCM-HTTP-v1-API zum Senden von Benachrichtigungen an Android-, iOS- und Webpush-Ziele. Dieses Thema enthält Beispiele für die Payload-Struktur beim Veröffentlichen von mobilen Push-Benachrichtigungen über die CLI oder die Amazon SNS SNS-API.

Sie können die folgenden Nachrichtentypen in Ihre Payload aufnehmen, wenn Sie eine FCM-Benachrichtigung senden:

- Datennachricht — Eine Datennachricht wird von Ihrer Client-App verarbeitet und enthält benutzerdefinierte Schlüssel-Wert-Paare. Wenn Sie eine Datennachricht erstellen, müssen Sie den `data` Schlüssel mit einem JSON-Objekt als Wert angeben und dann Ihre benutzerdefinierten Schlüssel-Wert-Paare eingeben.
- Benachrichtigungsnachricht oder Anzeigenachricht — Eine Benachrichtigung enthält einen vordefinierten Satz von Schlüsseln, die vom FCM SDK verarbeitet werden. Diese Schlüssel variieren je nach Gerätetyp, an den Sie liefern. Weitere Informationen zu plattformspezifischen Benachrichtigungsschlüsseln finden Sie im Folgenden:
  - Android-Benachrichtigungstasten
  - APNS-Benachrichtigungsschlüssel
  - Schlüssel für Webpush-Benachrichtigungen

Weitere Informationen zu FCM-Nachrichtentypen finden Sie unter Nachrichtentypen in der Firebase-Dokumentation von Google.

Inhalt

Verwenden der FCM v1-Payload-Struktur zum Senden von Nachrichten

Wenn Sie zum ersten Mal eine FCM-Anwendung erstellen oder die Funktionen von FCM v1 nutzen möchten, können Sie sich dafür entscheiden, eine FCM v1-formatierte Payload zu senden. Dazu müssen Sie den Schlüssel der obersten Ebene angeben. `fcmV1Message` Weitere Informationen zum

Erstellen von FCM v1-Payloads finden Sie unter Migration von älteren FCM-APIs zu HTTP v1 und Plattformübergreifendes Anpassen einer Nachricht in der Firebase-Dokumentation von Google.

FCM v1-Beispielnutzlast, die an Amazon SNS gesendet wurde:

> **Note**
>
> Der im folgenden Beispiel verwendete `GCM` Schlüsselwert muss als Zeichenfolge kodiert werden, wenn eine Benachrichtigung mit Amazon SNS veröffentlicht wird.

```
{
  "GCM": "{
     \"fcmV1Message\": {
        \"validate_only\" : false,
        \"message\" :
            {
            \"notification\": {
                \"title\": \"string\",
                \"body\": \"string\"
            },
            \"data\": {
                \"dataGen\": \"priority message\",
            },
            \"android\": {
                \"priority\": \"high\",
                \"notification\": {
                    \"body_loc_args\": [
                    \"string\"
                    ],
                    \"title_loc_args\": [
                    \"string\"
                    ],
                    \"sound\": \"string\",
                    \"title_loc_key\": \"string\",
                    \"title\": \"string\",
                    \"body\": \"string\",
                    \"click_action\": \"clicky_clacky\",
                    \"body_loc_key\": \"string\"
                },
                \"data\": {
                    \"dataAndroid\": \"priority message\",
```

```
            },
            \"ttl\": \"10023.32s\"
        },
        \"apns\": {
            \"payload\": {
                \"aps\": {
                    \"alert\": {
                        \"subtitle\": \"string\",
                        \"title-loc-args\": [
                        \"string\"
                        ],
                        \"title-loc-key\": \"string\",
                        \"loc-args\": [
                        \"string\"
                        ],
                        \"loc-key\": \"string\",
                        \"title\": \"string\",
                        \"body\": \"string\"
                    },
                    \"category\": \"Click\",
                    \"content-available\": 0,
                    \"sound\": \"string\",
                    \"badge\": 5
                }
            }
        },
        \"webpush\": {
            \"notification\": {
                \"badge\": \"5\",
                \"title\": \"string\",
                \"body\": \"string\"
            },
            \"data\": {
                \"dataWeb\": \"priority message\",
            }
        }
    }
  }"
}
```

Achten Sie beim Senden einer JSON-Nutzlast darauf, das `message-structure` Attribut in Ihre Anfrage aufzunehmen und es auf zu setzen. `json`

CLI-Beispiel:

```
aws sns publish --topic $TOPIC_ARN --message '{"GCM": "{\"fcmV1Message\": {\"message\":
{\"notification\":{\"title\":\"string\",\"body\":\"string\"},\"android\":{\"priority
\":\"high\",\"notification\":{\"title\":\"string\",\"body\":\"string\"},\"data\":
{\"customAndroidDataKey\":\"custom key value\"},\"ttl\":\"0s\"},\"apns\":{\"payload
\":{\"aps\":{\"alert\":{\"title\":\"string\", \"body\":\"string\"},\"content-
available\":1,\"badge\":5}}},\"webpush\":{\"notification\":{\"badge\":\"URL\",\"body
\":\"Test\"},\"data\":{\"customWebpushDataKey\":\"priority message\"}},\"data\":
{\"customGeneralDataKey\":\"priority message\"}}}}", "default": "{\"notification\":
 {\"title\": \"test\"}"}' --region $REGION --message-structure json
```

Weitere Informationen zum Senden von Payloads im FCM v1-Format finden Sie in der Firebase-Dokumentation von Google:

- Migrieren Sie von älteren FCM-APIs zu HTTP v1
- Über FCM-Nachrichten
- REST-Ressource: projects.messages

Verwenden der veralteten Payload-Struktur zum Senden von Nachrichten an die FCM v1-API

Bei der Migration zu FCM v1 müssen Sie die Payload-Struktur, die Sie für Ihre älteren Anmeldeinformationen verwendet haben, nicht ändern. Amazon SNS wandelt Ihre Payload in die neue FCM v1-Payload-Struktur um und sendet sie an Google.

Payload-Format der Eingabe-Nachricht:

```
{
  "GCM": "{\"notification\": {\"title\": \"string\", \"body\": \"string\",
 \"android_channel_id\": \"string\", \"body_loc_args\": [\"string\"], \"body_loc_key\":
 \"string\", \"click_action\": \"string\", \"color\": \"string\", \"icon\": \"string
\", \"sound\": \"string\", \"tag\": \"string\", \"title_loc_args\": [\"string\"],
 \"title_loc_key\": \"string\"}, \"data\": {\"message\": \"priority message\"}}"
}
```

Nachricht an Google gesendet:

```
{
  "message": {
    "token": "***",
    "notification": {
```

```
      "title": "string",
      "body": "string"
    },
    "android": {
      "priority": "high",
      "notification": {
        "body_loc_args": [
          "string"
        ],
        "title_loc_args": [
          "string"
        ],
        "color": "string",
        "sound": "string",
        "icon": "string",
        "tag": "string",
        "title_loc_key": "string",
        "title": "string",
        "body": "string",
        "click_action": "string",
        "channel_id": "string",
        "body_loc_key": "string"
      },
      "data": {
        "message": "priority message"
      }
    },
    "apns": {
      "payload": {
        "aps": {
          "alert": {
            "title-loc-args": [
              "string"
            ],
            "title-loc-key": "string",
            "loc-args": [
              "string"
            ],
            "loc-key": "string",
            "title": "string",
            "body": "string"
          },
          "category": "string",
          "sound": "string"
```

```
        }
      }
    },
    "webpush": {
      "notification": {
        "icon": "string",
        "tag": "string",
        "body": "string",
        "title": "string"
      },
      "data": {
        "message": "priority message"
      }
    },
    "data": {
      "message": "priority message"
    }
  }
}
```

Mögliche Risiken

- Die Zuordnung von Legacy zu Version 1 unterstützt weder den Apple Push Notification Service (APNS) `headers` noch die `fcm_options` Schlüssel. Wenn Sie diese Felder verwenden möchten, senden Sie eine FCM v1-Payload.
- In einigen Fällen benötigt FCM v1 Nachrichten-Header, um stille Benachrichtigungen an Ihre APNs-Geräte zu senden. Wenn Sie derzeit stille Benachrichtigungen an Ihre APNs-Geräte senden, funktionieren diese mit dem älteren Ansatz nicht. Stattdessen empfehlen wir, die FCM v1-Payload zu verwenden, um unerwartete Probleme zu vermeiden. Eine Liste der APNs-Header und wofür sie verwendet werden, findest du unter Kommunizieren mit APNs im Apple Developer Guide.
- Wenn Sie beim Senden Ihrer Benachrichtigung das TTL Amazon SNS SNS-Attribut verwenden, wird es nur im `android` Feld aktualisiert. Wenn Sie das TTL APNS-Attribut festlegen möchten, verwenden Sie die FCM v1-Payload.
- Die `webpush` Schlüssel `androidapns`, und werden zugeordnet und mit allen bereitgestellten relevanten Schlüsseln gefüllt. Wenn Sie beispielsweise angeben, was ein Schlüssel ist`title`, der von allen drei Plattformen gemeinsam genutzt wird, füllt die FCM-v1-Zuordnung alle drei Plattformen mit dem von Ihnen angegebenen Titel auf.
- Bei einigen von Plattformen gemeinsam genutzten Schlüsseln werden unterschiedliche Werttypen erwartet. Beispielsweise `apns` erwartet der übergebene `badge` Schlüssel einen Integer-Wert,

während der übergebene `badge` Schlüssel einen String-Wert `webpush` erwartet. In Fällen, in denen Sie den `badge` Schlüssel angeben, füllt die FCM v1-Zuordnung nur den Schlüssel auf, für den Sie einen gültigen Wert angegeben haben.

Ereignisse, die bei der FCM-Zustellung fehlschlagen

Die folgende Tabelle enthält den Amazon SNS SNS-Fehlertyp, der den Fehler-/Statuscodes entspricht, die von Google für FCM v1-Benachrichtigungsanfragen erhalten wurden. Alle beobachteten Fehlercodes, die von der FCM v1-API empfangen wurden, stehen Ihnen zur Verfügung, CloudWatch wenn Sie die Versandstatusprotokollierung für Ihre Anwendung einrichten.

FCM-Fehler-/Status code	Amazon SNS SNS-Fehlertyp	Fehlernachricht	Ursache und Abhilfemaßnahme
`UNREGISTERED`	`InvalidPlatformToken`	Das dem Endpunkt zugeordnete Plattform token ist nicht gültig.	Das an Ihren Endpunkt angehängte Geräte-Token ist veraltet oder ungültig. Amazon SNS hat Ihren Endpunkt deaktiviert. Aktualisieren Sie den Amazon SNS SNS-Endpunkt auf das neueste Geräte-Token.
`INVALID_ARGUMENT`	`InvalidNotification`	Der Text der Benachrichtigung ist ungültig.	Das Geräte-Token oder die Nachrichten-Payload sind möglicherweise ungültig. Stellen Sie sicher, dass Ihre Nachrichten-Payload gültig ist. Wenn die Nachrichten-Payload gültig ist, aktualisieren Sie den Amazon SNS

FCM-Fehler-/Status code	Amazon SNS SNS-Fehlertyp	Fehlernachricht	Ursache und Abhilfemaßnahme
			SNS-Endpunkt auf das neueste Geräte-Token.
`SENDER_ID_MISMATCH`	`InvalidPlatformToken`	Das dem Endpunkt zugeordnete Plattform-Token ist nicht gültig.	Die mit dem Geräte-Token verknüpfte Plattformanwendung ist nicht berechtigt, an das Geräte-Token zu senden. Stellen Sie sicher, dass Sie die richtigen FCM-Anmeldeinformationen in Ihrer Amazon SNS-Plattformanwendung verwenden.
`UNAVAILABLE`	`DependencyUnavailable`	Abhängigkeit ist nicht verfügbar.	FCM konnte die Anfrage nicht rechtzeitig bearbeiten. Alle von Amazon SNS ausgeführten Wiederholungsversuche sind fehlgeschlagen. Sie können diese Nachrichten in einer Warteschlange (Dead-Letter Queue, DLQ) speichern und sie zu einem späteren Zeitpunkt erneut versenden.

FCM-Fehler-/Status code	Amazon SNS SNS-Fehlertyp	Fehlernachricht	Ursache und Abhilfemaßnahme
`INTERNAL`	`UnexpectedFailure`	Unerwarteter Fehler. Bitte wenden Sie sich an Amazon. Fehlerphrase [Interner Fehler].	Beim Versuch, Ihre Anfrage zu bearbeiten, ist auf dem FCM-Server ein Fehler aufgetreten. Alle von Amazon SNS ausgeführten Wiederholungsversuche sind fehlgeschlagen. Sie können diese Nachrichten in einer Warteschlange (Dead-Letter Queue, DLQ) speichern und sie zu einem späteren Zeitpunkt erneut versenden.
`THIRD_PARTY_AUTH_ERROR`	`InvalidCredentials`	Die Anmeldeinformationen für die Plattformanwendung sind nicht gültig.	Eine Nachricht, die an ein iOS-Gerät oder ein Webpush-Gerät gerichtet war, konnte nicht gesendet werden. Stellen Sie sicher, dass Ihre Entwicklungs- und Produktionsanmeldedaten gültig sind.

FCM-Fehler-/Status code	Amazon SNS SNS-Fehlertyp	Fehlernachricht	Ursache und Abhilfemaßnahme
`QUOTA_EXCEEDED`	`Throttled`	Die Anfrage wurde von [gcm] gedrosselt.	Ein Nachricht enratenkontingent, ein Gerätenac hrichtenratenkonti ngent oder ein Themen-Nachrichten ratenkontingent wurde überschritten. Informationen zur Behebung dieses Problems finden Sie ErrorCodein der Firebase-Dokumenta tion von Google.
`PERMISSIO N_DENIED`	`InvalidNo tification`	Der Text der Benachrichtigung ist ungültig.	Im `PERMISSIO N_DENIED` Ausnahmefall ist der Aufrufer (Ihre FCM-Anwendung) nicht berechtigt, den angegeben en Vorgang in der Nutzlast auszuführ en. Navigieren Sie zu Ihrer FCM-Konso le und überprüfe n Sie, ob in Ihren Anmeldeinformation en die erforderlichen API-Aktionen aktiviert sind.

Mobile App-Attribute

Amazon Simple Notification Service (Amazon SNS) unterstützt die Protokollierung des Zustellungsstatus von Push-Benachrichtigungen. Nach der Konfiguration von Anwendungsattributen werden Protokolleinträge für Nachrichten, die von Amazon SNS an mobile Endpunkte gesendet wurden, an CloudWatch Logs übermittelt. Die Protokollierung des Zustellungsstatus von Nachrichten verhilft Ihnen zu besseren betrieblichen Einblicken, wie beispielsweise den Folgenden:

- Sie erfahren, ob eine Push-Benachrichtigungsmitteilung von Amazon SNS an den Push-Benachrichtigungsservice zugestellt wurde.
- Identifizieren Sie die Antwort, die vom Push-Benachrichtigungsservice an den Amazon SNS gesendet wurde.
- Ermitteln Sie die dwell-Zeit der Nachricht (das Zeitfenster zwischen dem Zeitstempel der Veröffentlichung bis direkt vor der Übergabe an einen Push-Benachrichtigungsservice).

Verwenden Sie für die Konfiguration der Anwendungsattribute für den Zustellstatus von Nachrichten die AWS Management Console, AWS Software Development Kits (SDKs) oder eine Abfrage-API.

Themen

Konfigurieren der Attribute für den Zustellungsstatus der Nachricht mithilfe der AWS Management Console

1. Melden Sie sich bei der Amazon-SNS-Konsole an.
2. Wählen Sie im Navigationsbereich die Option Mobile (Mobil) und dann Push notifications (Push-Benachrichtigungen) aus.
3. Wählen Sie aus dem Abschnitt Platform applications (Plattformapplikationen) die Anwendung, die die Endpunkte enthält, für die Sie CloudWatch Logs erhalten möchten.
4. Wählen Sie Application Actions (Anwendungsaktionen) und anschließend Delivery status (Zustellungsstatus).

5. Wählen Sie im Dialogfeld Delivery Status (Zustellungsstatus)die Option Create IAM Roles (IAM-Rollen erstellen).

   Sie werden anschließend zur IAM-Konsole umgeleitet.
6. Wählen Sie Allow (Zulassen), um Amazon SNS in Ihrem Namen Schreibzugriff auf CloudWatch Logs zu gewähren.
7. Kehren Sie nun zum Dialogfeld Delivery Status (Zustellungsstatus) zurück und geben Sie einen Wert in das Feld Percentage of Success to Sample (0-100) ((Erfolgsrate des Beispiels (0-100)) ein für den Prozentsatz der erfolgreich gesendeten Nachrichten, für die Sie CloudWatch Logs erhalten möchten.

   > **Note**
   >
   > Nachdem Sie für den Zustellungsstatus der Nachrichten Anwendungsattribute konfiguriert haben, generieren alle fehlgeschlagenen Nachrichtenzustellungen CloudWatch Logs.
8. Wählen Sie anschließend Save Configuration (Konfiguration speichern). Sie können nun CloudWatch Logs, das den Zustellungsstatus der Nachricht enthält, anzeigen und analysieren. Weitere Informationen zur Verwendung von CloudWatch finden Sie in der CloudWatch-Dokumentation.

Amazon SNS Nachrichtenzustellungsstatus CloudWatch Log Beispiele

Nachdem Sie für einen Anwendungsendpunkt Statusattribute für Nachrichtenübermittlungen konfiguriert haben, wird CloudWatch Logs generiert. Beispielprotokolle im JSON-Format werden wie folgt angezeigt:

ERFOLG

```
{
  "status": "SUCCESS",
  "notification": {
    "timestamp": "2015-01-26 23:07:39.54",
    "messageId": "9655abe4-6ed6-5734-89f7-e6a6a42de02a"
  },
  "delivery": {
    "statusCode": 200,
    "dwellTimeMs": 65,
```

```
    "token": "Examplei7fFachkJ1xjlqT64RaBkcGHochmf1VQAr9k-
IBJtKjp7fedYPzEwT_Pq3Tu0lroqro1cwWJUvgkcPPYcaXCpPWmG3Bqn-
wiqIEzp5zZ7y_jsM0PKPxKhddCzx6paEsyay9Zn3D4wNUJb8m6HXrBf9dqaEw",
    "attempts": 1,
    "providerResponse": "{\"multicast_id\":5138139752481671853,\"success
\":1,\"failure\":0,\"canonical_ids\":0,\"results\":[{\"message_id\":
\"0:1422313659698010%d6ba8edff9fd7ecd\"}]}",
    "destination": "arn:aws:sns:us-east-2:111122223333:endpoint/FCM/FCMPushApp/
c23e42de-3699-3639-84dd-65f84474629d"
  }
}
```

FEHLER

```
{
  "status": "FAILURE",
  "notification": {
    "timestamp": "2015-01-26 23:29:35.678",
    "messageId": "c3ad79b0-8996-550a-8bfa-24f05989898f"
  },
  "delivery": {
    "statusCode": 8,
    "dwellTimeMs": 1451,
    "token": "examp1e29z6j5c4df46f80189c4c83fjcgf7f6257e98542d2jt3395kj73",
    "attempts": 1,
    "providerResponse": "NotificationErrorResponse(command=8, status=InvalidToken,
 id=1, cause=null)",
    "destination": "arn:aws:sns:us-east-2:111122223333:endpoint/APNS_SANDBOX/
APNSPushApp/986cb8a1-4f6b-34b1-9a1b-d9e9cb553944"
  }
}
```

Eine Liste der Antwortcodes für den Push-Benachrichtigungsservice finden Sie unter Antwortcodes der Plattformen.

Konfigurieren der Statusattribute für Nachrichtenübermittlungen mit den AWS SDKs

Die AWS SDKs stellen APIs in verschiedenen Sprachen für die Verwendung von Statusattributen für Nachrichtenübermittlungen in Amazon SNS bereit.

Das folgende Java-Beispiel zeigt, wie Sie mit der `SetPlatformApplicationAttributes`-API Anwendungsattribute für den Status von Nachrichtenübermittlungen von Push-

Benachrichtigungsmitteilungen konfigurieren. Sie können die folgenden Attribute für den Status von Nachrichtenübermittlungen verwenden: `SuccessFeedbackRoleArnFailureFeedbackRoleArn` und `SuccessFeedbackSampleRate`. Die Attribute `SuccessFeedbackRoleArn` und `FailureFeedbackRoleArn` werden dazu verwendet, um Amazon SNS in Ihrem Namen Schreibzugriff auf CloudWatch Logs zu gewähren. Das Attribut `SuccessFeedbackSampleRate` dient zum Festlegen des Prozentsatzes der Samplerate (0-100) der erfolgreich zugestellten Nachrichten. Nachdem Sie das Attribut `FailureFeedbackRoleArn` konfiguriert haben, generieren alle fehlgeschlagenen Zustellungen CloudWatch Logs.

```
SetPlatformApplicationAttributesRequest setPlatformApplicationAttributesRequest = new
 SetPlatformApplicationAttributesRequest();
Map<String, String> attributes = new HashMap<>();
attributes.put("SuccessFeedbackRoleArn", "arn:aws:iam::111122223333:role/SNS_CWlogs");
attributes.put("FailureFeedbackRoleArn", "arn:aws:iam::111122223333:role/SNS_CWlogs");
attributes.put("SuccessFeedbackSampleRate", "5");
setPlatformApplicationAttributesRequest.withAttributes(attributes);
setPlatformApplicationAttributesRequest.setPlatformApplicationArn("arn:aws:sns:us-
west-2:111122223333:app/FCM/FCMPushApp");
sns.setPlatformApplicationAttributes(setPlatformApplicationAttributesRequest);
```

Weitere Informationen zu SDK für Java finden Sie unter Erste Schritte mit dem AWS SDK for Java.

Antwortcodes der Plattformen

Die folgende Liste enthält Links für die Antwortcodes des Push-Benachrichtigungsservice:

Push-Benachrichtigungsservice	Antwortcodes
Amazon Device Messaging (ADM)	Weitere Informationen finden Sie unter Antwortformat in der ADM Dokumentation.
Apple Push Notification Service (APNs)	Weitere Informationen finden Sie unter HTTP/2 Response from APNs in Communicating with APNs im Local and Remote Notification Programming Guide.
Firebase Cloud Messaging (FCM)	Weitere Informationen finden Sie unter Downstream Message Error Response Codes

Push-Benachrichtigungsservice	Antwortcodes
	in der Firebase Cloud Messaging Dokumentation.
Microsoft Push Notification Service für Windows Phone (MPNS)	Weitere Informationen finden Sie unter Push Notification Service Response Codes for Windows Phone 8 in der Windows 8 Development Dokumentation.
Windows Push Notification Services (WNS)	Weitere Informationen finden Sie unter "Response codes" in Push Notification Service Request and Response Headers (Windows Runtime Apps) in der Windows 8 Development-Dokumentation.

Mobile App-Ereignisse

Amazon SNS löst bei Auftreten bestimmter Anwendungsereignisse Benachrichtigungen aus. Im Anschluss daran können Sie eine programmgesteuerte Aktion für das Ereignis definieren. Ihre Anwendung muss einen Support für einen Push-Benachrichtigungsservice, wie Apple Push Notification Service (APNs), Firebase Cloud Messaging (FCM) und Windows Push Notification Services (WNS) einschließen. Sie richten Benachrichtigungen über Anwendungsereignisse mithilfe der Amazon SNS SNS-Konsole oder der AWS SDKs ein. AWS CLI

Themen

- Verfügbare Anwendungsereignisse
- Senden mobiler Push-Benachrichtigungen

Verfügbare Anwendungsereignisse

Ereignisbenachrichtigungen einer Anwendung können nachverfolgen, wann Endpunkte für individuelle Plattformen erstellt, gelöscht oder aktualisiert wurden. Sie können außerdem Versandfehler nachverfolgen. Im Folgenden sind die Attributnamen für die Anwendungsereignisse aufgeführt.

Attributname	Benachrichtigungsauslöser
`EventEndpointCreated`	Ein neuer Plattformendpunkt wird Ihrer Anwendung hinzugefügt.
`EventEndpointDeleted`	Alle Plattformendpunkte, die Ihrer Anwendung zugewiesen sind, werden gelöscht.
`EventEndpointUpdated`	Alle Attribute der Plattformendpunkte, die Ihrer Anwendung zugewiesen sind, werden geändert.
`EventDeliveryFailure`	Bei einer Übermittlung an einen der Plattformendpunkte, die Ihrer Anwendung zugewiesen wurden, ist ein dauerhafter Fehler aufgetreten. **Note** Um von der Plattform-Anwendung ausgehende Fehler bei der Übermittlung zu verfolgen, abonnieren Sie für die Anwendung Statusereignisse für die Nachrichtenübermittlung. Weitere Informationen finden Sie unter Verwendung von Amazon SNS-Anwendungsattributen für den Status von Nachrichtenübermittlungen.

Sie können jedes Attribut einer Anwendung zuordnen, die dann diese Ereignisbenachrichtigungen empfangen kann.

Senden mobiler Push-Benachrichtigungen

Um Ereignisbenachrichtigungen für Anwendungen zu versenden, geben Sie ein Thema an, das die Benachrichtigungen für den jeweiligen Ereignistyp erhalten soll. Sobald Amazon SNS die Benachrichtigungen sendet, kann das Themas diese an Endpunkte weiterleiten, die eine programmgesteuerte Aktion durchführen werden.

> **Important**
>
> Anwendungen mit großem Volumen rufen häufige Ereignisbenachrichtigungen für Anwendungen hervor (z. B. Zehntausende), durch die Endpunkte, die für die Verwendung durch Menschen vorgesehen sind, überflutet werden (z. B. E-Mail-Adressen, Telefonnummern und mobile Anwendungen). Beachten Sie folgende Richtlinien, wenn Sie Ereignisbenachrichtigungen für Anwendungen an ein Thema senden:
>
> - Jedes Thema, das Benachrichtigungen erhält, sollte nur Abonnements für programmatische Endpunkte wie HTTP- oder HTTPS-Endpunkte, Amazon SQS SQS-Warteschlangen oder Funktionen enthalten. AWS Lambda
> - Um den Bearbeitungsaufwand, der durch Benachrichtigungen ausgelöst wird, zu minimieren, reduzieren Sie die Anzahl der Abonnements eines jeden Themas (z. B. auf fünf oder weniger).

Sie können Benachrichtigungen über Anwendungsereignisse über die Amazon SNS SNS-Konsole, die AWS Command Line Interface (AWS CLI) oder die AWS SDKs senden.

AWS Management Console

1. Melden Sie sich bei der Amazon-SNS-Konsole an.
2. Wählen Sie im Navigationsbereich die Option Mobile (Mobil) und dann Push notifications (Push-Benachrichtigungen) aus.
3. Wählen Sie auf der Seite Push-Benachrichtigungen für Mobilgeräte im Abschnitt Plattformanwendungen eine Anwendung aus und klicken Sie dann auf Bearbeiten.
4. Erweitern Sie den Abschnitt Event notifications (Ereignisbenachrichtigungen).
5. Wählen Sie Actions (Aktionen) und Configure events (Ereignisse konfigurieren) aus.
6. Geben Sie die ARNs für Themen ein, die für die folgenden Ereignisse verwendet werden sollen:
   - Ersteller Endpunkt
   - Gelöschter Endpunkt
   - Aktualisierter Endpunkt
   - Übermittlungsfehler
7. Wählen Sie Änderungen speichern aus.

AWS CLI

Führen Sie den Befehl set-platform-application-attributes aus.

Im folgenden Beispiel wird das gleiche Amazon SNS-Thema für alle vier Anwendungsereignisse definiert:

```
aws sns set-platform-application-attributes
--platform-application-arn arn:aws:sns:us-east-1:12345EXAMPLE:app/FCM/
MyFCMPlatformApplication
--attributes EventEndpointCreated="arn:aws:sns:us-
east-1:12345EXAMPLE:MyFCMPlatformApplicationEvents",
EventEndpointDeleted="arn:aws:sns:us-
east-1:12345EXAMPLE:MyFCMPlatformApplicationEvents",
EventEndpointUpdated="arn:aws:sns:us-
east-1:12345EXAMPLE:MyFCMPlatformApplicationEvents",
EventDeliveryFailure="arn:aws:sns:us-
east-1:12345EXAMPLE:MyFCMPlatformApplicationEvents"
```

AWS SDKs

Richten Sie Benachrichtigungen über Anwendungsereignisse ein, indem Sie mithilfe eines AWS SDK eine `SetPlatformApplicationAttributes` Anfrage mit der Amazon SNS SNS-API einreichen.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele, einschließlich Hilfe bei den ersten Schritten und Informationen zu früheren Versionen, finden Sie unterAmazon SNS mit einem AWS SDK verwenden.

Mobile Push-API-Aktionen

Um die mobilen Amazon SNS-Push-APIs zu verwenden, müssen Sie zuerst die Voraussetzungen für den Push-Benachrichtigungs-Service, z. B. Apple Push Notification Service (APNs) und Firebase Cloud Messaging (FCM) erfüllen. Weitere Informationen zu diesen Voraussetzungen finden Sie unter Voraussetzungen für Amazon SNS-Benutzerbenachrichtigungen.

Um mit den APIs eine Push-Benachrichtigung an eine mobile App und ein Gerät zu senden, müssen Sie zunächst die `CreatePlatformApplication`-Aktion verwenden. Diese gibt ein `PlatformApplicationArn`-Attribut zurück. Das `PlatformApplicationArn`-Attribut wird dann von `CreatePlatformEndpoint` verwendet. So wird ein `EndpointArn`-Attribut zurückgeben. Anschließend können Sie mit dem `EndpointArn`-Attribut mit der `Publish`-Aktion eine Benachrichtigung an eine mobile App und ein Gerät senden oder Sie können das `EndpointArn`-

Attribut mit der `Subscribe`-Aktion zum Abonnieren eines Themas verwenden. Weitere Informationen finden Sie unter Übersicht über den Benutzerbenachrichtigungsprozess.

Die mobilen Amazon SNS-Push-APIs sind Folgende:

`CreatePlatformApplication`

Erstellt ein Plattformanwendungsobjekt für einen unterstützten Push-Benachrichtigungsservice (z. B. APNs und FCM), bei dem sich Geräte und Mobilgeräte-Apps registriert können. Gibt ein `PlatformApplicationArn`-Attribut zurück, das für die `CreatePlatformEndpoint`-Aktion verwendet wird.

`CreatePlatformEndpoint`

Erstellt einen Endpunkt für ein Gerät und eine mobile App für einen der unterstützten Push-Benachrichtigungsservices. `CreatePlatformEndpoint` verwendet das `PlatformApplicationArn`-Attribut, das der `CreatePlatformApplication`-Aktion zurückgegeben wurde. Das `EndpointArn`-Attribut, das bei der Verwendung von `CreatePlatformEndpoint` zurückgegeben wird, wird mit der `Publish`-Aktion verwendet, um eine Benachrichtigung an die mobile App und das Gerät zu senden.

`CreateTopic`

Erstellt ein Thema, über das die Nachrichten veröffentlicht werden können.

`DeleteEndpoint`

Löscht den Endpunkt für ein Gerät und eine mobile App für einen der unterstützten Push-Benachrichtigungsservices.

`DeletePlatformApplication`

Löscht ein Plattformanwendungsobjekt.

`DeleteTopic`

Löscht ein Thema und alle seine Abonnements.

`GetEndpointAttributes`

Ruft die Endpunktattribute für ein Gerät und eine mobile App ab.

`GetPlatformApplicationAttributes`

Ruft die Attribute des Plattformanwendungsobjekts ab.

`ListEndpointsByPlatformApplication`

Listet die Endpunkte und Endpunktattribute für Geräte und mobile Apps in einem unterstützten Push-Benachrichtigungsservice auf.

`ListPlatformApplications`

Listet die Plattformanwendungsobjekte für die unterstützten Push-Benachrichtigungsservices auf.

`Publish`

Sendet eine Benachrichtigung an alle abonnierten Endpunkte des Themas.

`SetEndpointAttributes`

Legt die Attribute für einen Endpunkt für ein Gerät und eine mobile App fest.

`SetPlatformApplicationAttributes`

Legt die Attribute des Plattformanwendungsobjekts fest.

`Subscribe`

Bereitet über das Senden einer Bestätigungsnachricht an den Endpunkt das Abonnieren eines Endpunkts vor. Um ein Abonnement zu erstellen, muss der Eigentümer des Endpunkts die ConfirmSubscription-Aktion mit dem Token aus der Bestätigungsnachricht aufrufen.

`Unsubscribe`

Löscht ein Abonnement.

Fehler Mobile-Push-API

Fehler, die von den Amazon SNS-APIs für mobile Push-Benachrichtigungen zurückgegeben werden, finden Sie in der folgenden Tabelle. Weitere Informationen über die Amazon SNS-APIs für mobile Push-Benachrichtigungen finden Sie unter Mobile Push-API-Aktionen.

Fehler	Beschreibung	HTTPS-Statuscode	API-Aktion
Anwendungsname ist eine Null-Zeichenfolge	Der erforderliche Anwendungsname ist auf Null festgelegt.	400	`CreatePlatformApplication`

Fehler	Beschreibung	HTTPS-Statuscode	API-Aktion
Plattformname ist eine Null-Zeichenfolge	Der erforderliche Plattformname ist auf Null festgelegt.	400	`CreatePlatformApplication`
Plattformname ist ungültig	Für den Plattformnamen wurde ein ungültiger Wert oder ein Wert angegeben, der sich außerhalb des gültigen Bereichs befindet.	400	`CreatePlatformApplication`
APNs – Prinzipal ist kein gültiges Zertifikat	Für den APNs-Prinzipal, bei dem es sich um das SSL-Zertifikat handelt, wurde ein ungültiges Zertifikat angegeben. Weitere Informationen finden Sie unter CreatePlatformApplication in der Amazon Simple Notification Service API-Referenz.	400	`CreatePlatformApplication`
APNs –Bei dem Prinzipal handelt es sich um eine gültige Zertifizierung, jedoch ist sie nicht in einem PEM-Format angegeben	Für den APNs-Prinzipal, bei dem es sich um das SSL-Zertifikat handelt, wurde zwar ein gültiges Zertifikat angegeben, jedoch nicht im PEM-Format.	400	`CreatePlatformApplication`

Fehler	Beschreibung	HTTPS-Statuscode	API-Aktion
Bei dem APNs-Prinzipal handelt es sich um ein abgelaufenes Zertifikat	Für den APNs-Prinzipal, bei dem es sich um das SSL-Zertifikat handelt, wurde ein abgelaufenes Zertifikat angegeben.	400	`CreatePlatformApplication`
APNs – Prinzipal ist kein von Apple ausgestelltes Zertifikat	Für den APNs-Prinzipal, bei dem es sich um das SSL-Zertifikat handelt, wurde ein nicht von Apple ausgestelltes Zertifikat angegeben.	400	`CreatePlatformApplication`
APNs – Prinzipal wurde nicht angegeben	Der APNs-Prinzipal, bei dem es sich um das SSL-Zertifikat handelt, wurde nicht angegeben.	400	`CreatePlatformApplication`
APNs – Anmeldeinformationen wurden nicht angegeben	Die APNs-Anmeldeinformationen, bei denen es sich um den privaten Schlüssel handelt, wurden nicht angegeben. Weitere Informationen finden Sie unter CreatePlatformApplication in der Amazon Simple Notification Service API-Referenz.	400	`CreatePlatformApplication`

Fehler	Beschreibung	HTTPS-Statuscode	API-Aktion
APNs – Anmeldeinformationen liegen nicht in einem gültigen PEM-Format vor	Die APNs-Anmeldeinformationen, bei denen es sich um den privaten Schlüssel handelt, liegen nicht in einem gültigen PEM-Format vor.	400	`CreatePlatformApplication`
FCM – serverAPIKey wurde nicht angegeben	Die FCM-Anmeldeinformationen, bei denen es sich um den API-Schlüssel handelt, wurden nicht angegeben. Weitere Informationen finden Sie unter CreatePlatformApplication in der Amazon Simple Notification Service API-Referenz.	400	`CreatePlatformApplication`
FCM – serverAPIKey ist leer	Die FCM-Anmeldeinformationen, bei denen es sich um den API-Schlüssel handelt, sind leer.	400	`CreatePlatformApplication`
FCM – serverAPIKey ist eine Null-Zeichenfolge	Die FCM-Anmeldeinformationen, bei denen es sich um den API-Schlüssel handelt, sind Null.	400	`CreatePlatformApplication`

Fehler	Beschreibung	HTTPS-Statuscode	API-Aktion
FCM – serverAPIKey ist ungültig	Die FCM-Anmeldeinformationen, bei denen es sich um den API-Schlüssel handelt, sind ungültig.	400	`CreatePlatformApplication`
ADM – clientsecret wurde nicht angegeben	Der erforderliche Clientschlüssel liegt nicht vor.	400	`CreatePlatformApplication`
ADM – clientsecret ist eine Null-Zeichenfolge	Die erforderliche Zeichenfolge für den Clientschlüssel ist Null.	400	`CreatePlatformApplication`
ADM – client_secret ist eine leere Zeichenfolge	Die erforderliche Zeichenfolge für den Clientschlüssel ist leer.	400	`CreatePlatformApplication`
ADM – client_secre ist nicht gültig	Die erforderliche Zeichenfolge für den Clientschlüssel ist nicht gültig.	400	`CreatePlatformApplication`
ADM – client_id ist eine leere Zeichenfolge	Die erforderliche Zeichenfolge für die Client-ID ist leer.	400	`CreatePlatformApplication`
ADM – clientId wurde nicht angegeben	Die erforderliche Zeichenfolge für die Client-ID liegt nicht vor.	400	`CreatePlatformApplication`

Fehler	Beschreibung	HTTPS-Statuscode	API-Aktion
ADM – clientid ist eine Null-Zeichenfolge	Die erforderliche Zeichenfolge für die Client-ID ist Null.	400	`CreatePlatformApplication`
ADM – client_id ist nicht gültig	Die erforderliche Zeichenfolge für die Client-ID ist nicht gültig.	400	`CreatePlatformApplication`
EventEndpointCreated zeigt ein ungültiges ARN-Format	EventEndpointCreated zeigt ein ungültiges ARN-Format.	400	`CreatePlatformApplication`
EventEndpointDeleted zeigt ein ungültiges ARN-Format	EventEndpointDeleted zeigt ein ungültiges ARN-Format.	400	`CreatePlatformApplication`
EventEndpointUpdated zeigt ein ungültiges ARN-Format	EventEndpointUpdated zeigt ein ungültiges ARN-Format.	400	`CreatePlatformApplication`
EventDeliveryAttemptFailure zeigt ein ungültiges ARN-Format	EventDeliveryAttemptFailure zeigt ein ungültiges ARN-Format.	400	`CreatePlatformApplication`
EventDeliveryFailure zeigt ein ungültiges ARN-Format	EventDeliveryFailure zeigt ein ungültiges ARN-Format.	400	`CreatePlatformApplication`
EventEndpointCreated ist kein vorhandenes Thema	EventEndpointCreated ist kein vorhandenes Thema.	400	`CreatePlatformApplication`

Fehler	Beschreibung	HTTPS-Statuscode	API-Aktion
EventEndpointDeleted ist kein vorhandenes Thema	EventEndpointDeleted ist kein vorhandenes Thema.	400	`CreatePlatformApplication`
EventEndpointUpdated ist kein vorhandenes Thema	EventEndpointUpdated ist kein vorhandenes Thema.	400	`CreatePlatformApplication`
EventDeliveryAttemptFailure ist kein vorhandenes Thema	EventDeliveryAttemptFailure ist kein vorhandenes Thema.	400	`CreatePlatformApplication`
EventDeliveryFailure ist kein vorhandenes Thema	EventDeliveryFailure ist kein vorhandenes Thema.	400	`CreatePlatformApplication`
ARN der Plattform ist ungültig	ARN der Plattform ist ungültig.	400	`SetPlatformAttributes`
ARN der Plattform ist gültig, gehört aber nicht dem Benutzer	ARN der Plattform ist gültig, gehört aber nicht dem Benutzer.	400	`SetPlatformAttributes`
APNs – Prinzipal ist kein gültiges Zertifikat	Für den APNs-Prinzipal, bei dem es sich um das SSL-Zertifikat handelt, wurde ein ungültiges Zertifikat angegeben. Weitere Informationen finden Sie unter CreatePlatformApplication in der Amazon Simple Notification Service API-Referenz.	400	`SetPlatformAttributes`

Fehler	Beschreibung	HTTPS-Statuscode	API-Aktion
APNs –Bei dem Prinzipal handelt es sich um eine gültige Zertifizierung, jedoch ist sie nicht in einem PEM-Format angegeben	Für den APNs-Prinzipal, bei dem es sich um das SSL-Zertifikat handelt, wurde zwar ein gültiges Zertifikat angegeben, jedoch nicht im PEM-Format.	400	`SetPlatformAttributes`
Bei dem APNs-Prinzipal handelt es sich um ein abgelaufenes Zertifikat	Für den APNs-Prinzipal, bei dem es sich um das SSL-Zertifikat handelt, wurde ein abgelaufenes Zertifikat angegeben.	400	`SetPlatformAttributes`
APNs – Prinzipal ist kein von Apple ausgestelltes Zertifikat	Für den APNs-Prinzipal, bei dem es sich um das SSL-Zertifikat handelt, wurde ein nicht von Apple ausgestelltes Zertifikat angegeben.	400	`SetPlatformAttributes`
APNs – Prinzipal wurde nicht angegeben	Der APNs-Prinzipal, bei dem es sich um das SSL-Zertifikat handelt, wurde nicht angegeben.	400	`SetPlatformAttributes`

Fehler	Beschreibung	HTTPS-Statuscode	API-Aktion
APNs – Anmeldeinformationen wurden nicht angegeben	Die APNs-Anmeldeinformationen, bei denen es sich um den privaten Schlüssel handelt, wurden nicht angegeben. Weitere Informationen finden Sie unter CreatePlatformApplication in der Amazon Simple Notification Service API-Referenz.	400	`SetPlatformAttributes`
APNs – Anmeldeinformationen liegen nicht in einem gültigen PEM-Format vor	Die APNs-Anmeldeinformationen, bei denen es sich um den privaten Schlüssel handelt, liegen nicht in einem gültigen PEM-Format vor.	400	`SetPlatformAttributes`
FCM – serverAPI Key wurde nicht angegeben	Die FCM-Anmeldeinformationen, bei denen es sich um den API-Schlüssel handelt, wurden nicht angegeben. Weitere Informationen finden Sie unter CreatePlatformApplication in der Amazon Simple Notification Service API-Referenz.	400	`SetPlatformAttributes`

Fehler	Beschreibung	HTTPS-Statuscode	API-Aktion
FCM – serverAPI Key ist eine Null-Zeichenfolge	Die FCM-Anmeldeinformationen, bei denen es sich um den API-Schlüssel handelt, sind Null.	400	`SetPlatformAttributes`
ADM – clientId wurde nicht angegeben	Die erforderliche Zeichenfolge für die Client-ID liegt nicht vor.	400	`SetPlatformAttributes`
ADM – clientid ist eine Null-Zeichenfolge	Die erforderliche Zeichenfolge für die Client-ID ist Null.	400	`SetPlatformAttributes`
ADM – clientsecret wurde nicht angegeben	Der erforderliche Clientschlüssel liegt nicht vor.	400	`SetPlatformAttributes`
ADM – clientsecret ist eine Null-Zeichenfolge	Die erforderliche Zeichenfolge für den Clientschlüssel ist Null.	400	`SetPlatformAttributes`
EventEndpointUpdated zeigt ein ungültiges ARN-Format	EventEndpointUpdated zeigt ein ungültiges ARN-Format.	400	`SetPlatformAttributes`
EventEndpointDeleted zeigt ein ungültiges ARN-Format	EventEndpointDeleted zeigt ein ungültiges ARN-Format.	400	`SetPlatformAttributes`
EventEndpointUpdated zeigt ein ungültiges ARN-Format	EventEndpointUpdated zeigt ein ungültiges ARN-Format.	400	`SetPlatformAttributes`

Fehler	Beschreibung	HTTPS-Statuscode	API-Aktion
EventDeliveryAttemptFailure zeigt ein ungültiges ARN-Format	EventDeliveryAttemptFailure zeigt ein ungültiges ARN-Format.	400	`SetPlatformAttributes`
EventDeliveryFailure zeigt ein ungültiges ARN-Format	EventDeliveryFailure zeigt ein ungültiges ARN-Format.	400	`SetPlatformAttributes`
EventEndpointCreated ist kein vorhandenes Thema	EventEndpointCreated ist kein vorhandenes Thema.	400	`SetPlatformAttributes`
EventEndpointDeleted ist kein vorhandenes Thema	EventEndpointDeleted ist kein vorhandenes Thema.	400	`SetPlatformAttributes`
EventEndpointUpdated ist kein vorhandenes Thema	EventEndpointUpdated ist kein vorhandenes Thema.	400	`SetPlatformAttributes`
EventDeliveryAttemptFailure ist kein vorhandenes Thema	EventDeliveryAttemptFailure ist kein vorhandenes Thema.	400	`SetPlatformAttributes`
EventDeliveryFailure ist kein vorhandenes Thema	EventDeliveryFailure ist kein vorhandenes Thema.	400	`SetPlatformAttributes`
ARN der Plattform ist ungültig	Der ARN der Plattform ist ungültig.	400	`GetPlatformApplicationAttributes`

Fehler	Beschreibung	HTTPS-Statuscode	API-Aktion
ARN der Plattform ist gültig, gehört aber nicht dem Benutzer	Der ARN der Plattform ist gültig, gehört aber nicht dem Benutzer.	403	`GetPlatformApplicationAttributes`
Der angegebene Token ist ungültig	Das angegebene Token ist ungültig.	400	`ListPlatformApplications`
ARN der Plattform ist ungültig	Der ARN der Plattform ist ungültig.	400	`ListEndpointsByPlatformApplication`
ARN der Plattform ist gültig, gehört aber nicht dem Benutzer	Der ARN der Plattform ist gültig, gehört aber nicht dem Benutzer.	404	`ListEndpointsByPlatformApplication`
Der angegebene Token ist ungültig	Das angegebene Token ist ungültig.	400	`ListEndpointsByPlatformApplication`
ARN der Plattform ist ungültig	Der ARN der Plattform ist ungültig.	400	`DeletePlatformApplication`
ARN der Plattform ist gültig, gehört aber nicht dem Benutzer	Der ARN der Plattform ist gültig, gehört aber nicht dem Benutzer.	403	`DeletePlatformApplication`
ARN der Plattform ist ungültig	Der ARN der Plattform ist ungültig.	400	`CreatePlatformEndpoint`

Fehler	Beschreibung	HTTPS-Statuscode	API-Aktion
ARN der Plattform ist gültig, gehört aber nicht dem Benutzer	Der ARN der Plattform ist gültig, gehört aber nicht dem Benutzer.	404	`CreatePlatformEndpoint`
Der Token wurde nicht angegeben	Das Token wurde nicht angegeben.	400	`CreatePlatformEndpoint`
Token hat nicht die richtige Länge	Das Token hat nicht die richtige Länge.	400	`CreatePlatformEndpoint`
Kunden-/Benutzerdaten sind zu groß	Die Kunden-/Benutzerdaten dürfen in der UTF-8-Codierung maximal 2048 Byte lang ein.	400	`CreatePlatformEndpoint`
Endpunkt-ARN ist ungültig	Der Endpunkt-ARN ist ungültig.	400	`DeleteEndpoint`
Der Endpunkt-ARN ist gültig, gehört aber nicht dem Benutzer	Der Endpunkt-ARN ist gültig, gehört aber nicht dem Benutzer.	403	`DeleteEndpoint`
Endpunkt-ARN ist ungültig	Der Endpunkt-ARN ist ungültig.	400	`SetEndpointAttributes`
Der Endpunkt-ARN ist gültig, gehört aber nicht dem Benutzer	Der Endpunkt-ARN ist gültig, gehört aber nicht dem Benutzer.	403	`SetEndpointAttributes`
Der Token wurde nicht angegeben	Das Token wurde nicht angegeben.	400	`SetEndpointAttributes`
Token hat nicht die richtige Länge	Das Token hat nicht die richtige Länge.	400	`SetEndpointAttributes`

Fehler	Beschreibung	HTTPS-Statuscode	API-Aktion
Kunden-/Benutzerdaten sind zu groß	Die Kunden-/Benutzerdaten dürfen in der UTF-8-Codierung maximal 2048 Byte lang ein.	400	`SetEndpointAttributes`
Endpunkt-ARN ist ungültig	Der Endpunkt-ARN ist ungültig.	400	`GetEndpointAttributes`
Der Endpunkt-ARN ist gültig, gehört aber nicht dem Benutzer	Der Endpunkt-ARN ist gültig, gehört aber nicht dem Benutzer.	403	`GetEndpointAttributes`
Ziel-ARN ist ungültig	Der Ziel-ARN ist ungültig.	400	`Publish`
Der Ziel-ARN ist gültig, gehört aber nicht dem Benutzer	Der Ziel-ARN ist gültig, gehört aber nicht dem Benutzer.	403	`Publish`
Nachrichtenformat ist ungültig	Das Nachrichtenformat ist ungültig.	400	`Publish`
Nachrichtengröße ist größer als die vom Protokoll/End-Service unterstützte Größe	Die Nachrichtengröße ist größer als die vom Protokoll/End-Service unterstützte Größe.	400	`Publish`

Verwenden des Nachrichtenattributs Time To Live (TTL) von Amazon SNS für Mobile-Push-Benachrichtigungen

Amazon Simple Notification Service (Amazon SNS) unterstützt das Festlegen eines Time To Live (TTL)-Nachrichtenattributs für Mobile Push-Benachrichtigungen. Dies gilt zusätzlich zu der bestehenden Möglichkeit, TTL innerhalb des Amazon SNS SNS-Nachrichtentexts für die mobilen

Push-Benachrichtigungsdienste festzulegen, die dies unterstützen, wie Amazon Device Messaging (ADM) und Firebase Cloud Messaging (FCM) beim Senden an Android.

Das TTL-Nachrichtenattribut wird verwendet, um Ablauf-Metadaten über eine Nachricht anzugeben. So können Sie die Zeit festlegen, die dem Push-Benachrichtigungs-Service wie z. B. Apple Push Notification Service (APNs) oder FCM zur Verfügung steht, um die Nachricht an den Endpunkt zu übermitteln. Wenn die Nachricht aus irgendwelchen Gründen (z. B. bei Ausschalten des Mobilgeräts) innerhalb des angegebenen TTL nicht zustellbar ist, wird die Nachricht gelöscht und es werden keine weiteren Zustellversuche unternommen. Um TTL innerhalb von Nachrichtenattributen anzugeben, können Sie die AWS Software Development Kits (SDKs) oder die AWS Management Console Abfrage-API verwenden.

Themen

TTL-Nachrichtenattribute für Push-Benachrichtigungsservices

Im Folgenden finden Sie eine Liste der TTL-Nachrichtenattribute für Push-Benachrichtigungsdienste, die Sie bei Verwendung der AWS SDKs oder der Abfrage-API festlegen können:

Push-Benachrichtigungsservice	TTL-Nachrichtenattribut
Amazon Device Messaging (ADM)	`AWS.SNS.MOBILE.ADM.TTL`
Apple Push Notification Service (APNs)	`AWS.SNS.MOBILE.APNS.TTL`
Apple Push Notification Service Sandbox (APNs_SANDBOX)	`AWS.SNS.MOBILE.APNS_SANDBOX.TTL`
Baidu Cloud Push (Baidu)	`AWS.SNS.MOBILE.BAIDU.TTL`
Firebase Cloud Messaging (FCM beim Senden an Android)	`AWS.SNS.MOBILE.FCM.TTL`
Windows Push Notification Services (WNS)	`AWS.SNS.MOBILE.WNS.TTL`

Jeder Push-Benachrichtigungsservice geht anders mit TTL um. Amazon SNS bietet eine abstrakte Ansicht des TTL über alle Push-Benachrichtigungsservices, was das Festlegen des TTL erleichtert. Wenn Sie die AWS Management Console TTL (in Sekunden) angeben, müssen Sie den TTL-Wert nur einmal eingeben. Amazon SNS berechnet dann die TTL für jeden der ausgewählten Push-Benachrichtigungsdienste, wenn die Nachricht veröffentlicht wird.

Der TTL ist relativ zum Veröffentlichungszeitpunkt. Bevor eine Push-Benachrichtigung an einen bestimmten Push-Benachrichtigungsservice übergeben wird, berechnet Amazon SNS die dwell-Zeit (die Zeit zwischen dem Veröffentlichungszeitstempel und unmittelbar vor der Übergabe an einen Push-Benachrichtigungsservice) für die Push-Benachrichtigung und leitet den verbleibenden TTL an den betreffenden Push-Benachrichtigungsservice weiter. Wenn der TTL kürzer ist als die dwell-Zeit, versucht Amazon SNS keine Veröffentlichung.

Wenn Sie eine TTL für eine Push-Benachrichtigung angeben, muss der TTL-Wert eine positive Ganzzahl sein, es sei denn, der Wert von `0` hat eine spezifische Bedeutung für den Push-Benachrichtigungsdienst, z. B. bei APNs und FCM (beim Senden an Android). Wenn der TTL-Wert auf `0` festgelegt ist und der Push-Benachrichtigungsservice für `0` keine spezifische Bedeutung vorsieht, wird die Nachricht von Amazon SNS verworfen. Weitere Informationen zum TTL-Parameter mit dem Wert `0` bei Verwendung von APNs finden Sie unter Table A-3 Item identifiers for remote notifications in der Dokumentation Binary Provider API.

Rangfolge bei der Bestimmung von TTL

Die Rangfolge, die Amazon SNS zur Bestimmung des TTL-Werts für eine Push-Benachrichtigung verwendet, basiert auf der folgenden Reihenfolge, wobei die niedrigste Zahl die höchste Priorität hat:

1. TTL-Nachrichtenattribut
2. TTL-Nachrichtentext
3. Standard-TTL des Push-Benachrichtigungsservice (variiert je nach Service)
4. Standard-TTL von Amazon SNS (4 Wochen)

Wenn Sie für die gleiche Nachricht verschiedene TTL-Werte festlegen (einen in den Nachrichtenattributen und einen anderen im Nachrichtentext), ändert Amazon SNS den TTL im Nachrichtentext so, dass er dem in dem Nachrichtenattribut angegebenen TTL-Wert entspricht.

Angeben von TTL mit dem AWS Management Console

1. Melden Sie sich bei der Amazon-SNS-Konsole an.

2. Wählen Sie im Navigationsbereich die Option Mobile (Mobil) und dann Push notifications (Push-Benachrichtigungen) aus.
3. Wählen Sie auf der Seite Mobile push notifications (Mobile Push-Benachrichtigungen) im Abschnitt Platform applications (Plattformanwendungen) eine Anwendung aus.
4. Wählen Sie auf der ***MyApplication***Seite im Abschnitt Endpoints einen Anwendungsendpunkt aus und klicken Sie dann auf Nachricht veröffentlichen.
5. Geben Sie im Abschnitt Message details (Nachrichtendetails) die TTL ein (die Anzahl an Sekunden, die dem Push-Benachrichtigungs-Service zur Verfügung stehen, um die Nachricht an den Endpunkt zu übermitteln).
6. Wählen Sie Publish message (Nachricht veröffentlichen) aus.

Unterstützte Regionen für mobile Anwendungen

Derzeit können Sie mobile Anwendungen in den folgenden Regionen erstellen:

- US East (Ohio)
- USA Ost (Nord-Virginia)
- USA West (Nordkalifornien)
- USA West (Oregon)
- Africa (Cape Town)
- Asia Pacific (Hongkong)
- Asien-Pazifik (Jakarta)
- Asien-Pazifik (Mumbai)
- Asia Pacific (Osaka)
- Asia Pacific (Seoul)
- Asien-Pazifik (Singapur)
- Asien-Pazifik (Sydney)
- Asien-Pazifik (Tokio)
- Canada (Central)
- Europe (Frankfurt)
- Europa (Irland)
- Europe (London)

- Europa (Mailand)
- Europe (Paris)
- Europe (Stockholm)
- Naher Osten (Bahrain)
- Naher Osten (VAE)
- Südamerika (São Paulo)
- AWS GovCloud (USA-West)

Bewährte Methoden für mobile Push-Benachrichtigungen

In diesem Abschnitt werden bewährte Methoden beschrieben, die Ihnen dabei helfen können, die Kundenbindung zu verbessern.

Endpunktverwaltung

Es können Probleme bei der Zustellung auftreten, wenn sich Geräte-Token aufgrund der Aktion eines Benutzers auf dem Gerät ändern (z. B. bei der Neuinstallation einer App auf dem Gerät) oder wenn sich Zertifikatsaktualisierungen auf Geräte auswirken, die eine bestimmte iOS-Version nutzen. Als bewährte Methode empfiehlt Apple eine Registrierung bei APNs bei jedem Start der App.

Da sich das Geräte-Token nicht jedes Mal ändert, wenn eine App von einem Benutzer geöffnet wird, kann die idempotente API `CreatePlatformEndpoint` verwendet werden. Dies kann jedoch zu Duplikaten für dasselbe Gerät führen, wenn das Token selbst ungültig ist oder wenn der Endpunkt gültig, aber deaktiviert ist (z. B. wenn Produktions- und Sandbox-Umgebungen nicht übereinstimmen).

Es kann ein Mechanismus zur Verwaltung von Geräte-Token wie der im Pseudo-Code verwendet werden.

Informationen zur Verwaltung und Pflege von FCM v1-Geräte-Token finden Sie unter. Firebase Cloud Messaging (FCM) -Endpunktverwaltung

Protokollieren des Zustellungsstatus

Für die Überwachung des Zustellungsstatus von Push-Benachrichtigungen empfehlen wir, die Protokollierung des Zustellungsstatus für Ihre Amazon-SNS-Plattformanwendung zu aktivieren. Dies hilft bei der Behebung von Zustellungsfehlern, da die Protokolle Anbieter-Antwortcodes enthalten,

die vom Push-Plattformservice zurückgegeben wurden. Weitere Informationen zum Aktivieren der Protokollierung des Zustellungsstatus finden Sie unter Wie greife ich auf Zustellungsprotokolle für Amazon SNS-Themen für Push-Benachrichtigungen zu?.

Ereignis-Benachrichtigungen

Um Endpunkte auf ereignisgesteuerte Weise zu verwalten, können Sie die Funktion Ereignisbenachrichtigungen verwenden. Dies ermöglicht es dem konfigurierten Amazon-SNS-Thema, Ereignisse an die Abonnenten zu verteilen, wie beispielsweise eine Lambda-Funktion für Plattformanwendungsereignisse zur Erstellung, Löschung, Aktualisierung und fehlgeschlagenen Zustellung von Endpunkten.

E-Mail-Benachrichtigungen

Auf dieser Seite wird beschrieben, wie Sie mithilfe von, oder AWS SDK for .NET eine E-Mail-Adresse für ein Amazon SNS SNS-Thema abonnieren. AWS Management Console AWS SDK for Java

Hinweise

- Sie können den Text der E-Mail-Nachricht nicht anpassen. Die E-Mail-Übermittlungsfunktion dient dazu, interne Systemwarnungen bereitzustellen, keine Marketingnachrichten.
- Das direkte Abonnieren von E-Mail-Endpunkten wird nur für Standardthemen unterstützt.
- Der E-Mail-Übermittlungsdurchsatz wird gemäß Amazon SNS- Kontingentefestgelegt.

Important

Informationen zum Verhindern, dass Empfänger von Mailinglisten alle Empfänger von E-Mails zu Amazon-SNS-Themen abmelden, finden Sie unter Einrichten eines E-Mail-Abonnements, das zum Abbestellen eine Authentifizierung voraussetzt vom AWS -Support.

Um eine E-Mail-Adresse für ein Amazon SNS SNS-Thema zu abonnieren, verwenden Sie AWS Management Console

1. Melden Sie sich bei der Amazon-SNS-Konsole an.

2. Wählen Sie im linken Navigationsbereich Subscriptions (Abonnements).
3. Wählen Sie auf der Seite Subscriptions (Abonnements) die Option Create subscription (Abonnement erstellen)aus.
4. Führen Sie auf der Seite Create subscription (Abonnement erstellen) im Abschnitt Details die folgenden Schritte aus:

   a. Für das Thema-ARN wählen Sie den Amazon-Ressourcennamen (ARN) eines Themas.

   b. Wählen Sie unter Protocol (Protokoll) die Option Email (E-Mail) aus.

   c. Geben Sie unter Endpunkt Ihre E-Mail-Adresse ein.

   d. (Optional) Um eine Filterrichtlinie zu konfigurieren, erweitern Sie den EintragAbonnement-FilterrichtlinieAbschnitts erstellt. Weitere Informationen finden Sie unter Filterrichtlinien für Amazon-SNS-Abonnements.

   e. (Optional) Zur Aktivierung der nutzlastbasierten Filterung konfigurieren Sie `Filter Policy Scope` auf `MessageBody`. Weitere Informationen finden Sie unter Filterrichtlinien für Amazon-SNS-Abonnements – Geltungsbereich.

   f. (Optional) Um eine Warteschlange für unzustellbare Nachrichten für das Abonnement zu konfigurieren, erweitern Sie den Abschnitt Redrive-Richtlinie (Warteschlange für unzustellbare Nachrichten). Weitere Informationen finden Sie unter Amazon SNS Queues für unzustellbare Nachrichten (DLQs).

   g. Wählen Sie Create subscription (Abonnement erstellen) aus.

      Die Konsole erstellt das Abonnement und öffnet die Abodetails.

Sie müssen das Abonnement bestätigen, bevor die E-Mail-Adresse Nachrichten empfangen kann.

Ein Abonnement bestätigen

1. Überprüfen Sie Ihren E-Mail-Posteingang und wählen Sie Abonnement bestätigen in der E-Mail von Amazon SNS.
2. Amazon SNS öffnet Ihren Webbrowser und zeigt eine Abonnementbestätigung mit Ihrer Abonnement-ID an.

So abonnieren Sie mithilfe eines SDK eine E-Mail-Adresse für ein Amazon SNS SNS-Thema AWS

Um ein AWS SDK verwenden zu können, müssen Sie es mit Ihren Anmeldeinformationen konfigurieren. Weitere Informationen finden Sie unter Freigegebene Konfigurations- und Anmeldeinformationsdateien im AWS -Referenzhandbuch zu SDKs und Tools.

Die folgenden Codebeispiele zeigen, wie Sie es verwenden`Subscribe`.

.NET

AWS SDK for .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

Abonnieren Sie eine E-Mail-Adresse für ein Thema.

```
/// <summary>
/// Creates a new subscription to a topic.
/// </summary>
/// <param name="client">The initialized Amazon SNS client object, used
/// to create an Amazon SNS subscription.</param>
/// <param name="topicArn">The ARN of the topic to subscribe to.</param>
/// <returns>A SubscribeResponse object which includes the subscription
/// ARN for the new subscription.</returns>
public static async Task<SubscribeResponse> TopicSubscribeAsync(
    IAmazonSimpleNotificationService client,
    string topicArn)
{
    SubscribeRequest request = new SubscribeRequest()
    {
        TopicArn = topicArn,
        ReturnSubscriptionArn = true,
        Protocol = "email",
        Endpoint = "recipient@example.com",
    };
```

```
            var response = await client.SubscribeAsync(request);

            return response;
        }
```

Abonnieren Sie eine Warteschlange für ein Thema mit optionalen Filtern.

```
    /// <summary>
    /// Subscribe a queue to a topic with optional filters.
    /// </summary>
    /// <param name="topicArn">The ARN of the topic.</param>
    /// <param name="useFifoTopic">The optional filtering policy for the
subscription.</param>
    /// <param name="queueArn">The ARN of the queue.</param>
    /// <returns>The ARN of the new subscription.</returns>
    public async Task<string> SubscribeTopicWithFilter(string topicArn, string?
filterPolicy, string queueArn)
    {
        var subscribeRequest = new SubscribeRequest()
        {
            TopicArn = topicArn,
            Protocol = "sqs",
            Endpoint = queueArn
        };

        if (!string.IsNullOrEmpty(filterPolicy))
        {
            subscribeRequest.Attributes = new Dictionary<string, string>
{ { "FilterPolicy", filterPolicy } };
        }

        var subscribeResponse = await
_amazonSNSClient.SubscribeAsync(subscribeRequest);
        return subscribeResponse.SubscriptionArn;
    }
```

- Details zu API finden Sie unter Abonnieren in der AWS SDK for .NET -API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

Abonnieren Sie eine E-Mail-Adresse für ein Thema.

```
//! Subscribe to an Amazon Simple Notification Service (Amazon SNS) topic with
 delivery to an email address.
/*!
  \param topicARN: An SNS topic Amazon Resource Name (ARN).
  \param emailAddress: An email address.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SNS::subscribeEmail(const Aws::String &topicARN,
                                 const Aws::String &emailAddress,
                                 const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::SNS::Model::SubscribeRequest request;
    request.SetTopicArn(topicARN);
    request.SetProtocol("email");
    request.SetEndpoint(emailAddress);

    const Aws::SNS::Model::SubscribeOutcome outcome =
 snsClient.Subscribe(request);

    if (outcome.IsSuccess()) {
        std::cout << "Subscribed successfully." << std::endl;
        std::cout << "Subscription ARN '" <<
 outcome.GetResult().GetSubscriptionArn()
                  << "'." << std::endl;
    }
    else {
        std::cerr << "Error while subscribing " <<
 outcome.GetError().GetMessage()
```

```
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

Abonnieren Sie ein Thema mit einer mobilen Anwendung.

```
//! Subscribe to an Amazon Simple Notification Service (Amazon SNS) topic with
 delivery to a mobile app.
/*!
  \param topicARN: The Amazon Resource Name (ARN) for an Amazon SNS topic.
  \param endpointARN: The ARN for a mobile app or device endpoint.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool
AwsDoc::SNS::subscribeApp(const Aws::String &topicARN,
                          const Aws::String &endpointARN,
                          const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::SNS::Model::SubscribeRequest request;
    request.SetTopicArn(topicARN);
    request.SetProtocol("application");
    request.SetEndpoint(endpointARN);

    const Aws::SNS::Model::SubscribeOutcome outcome =
 snsClient.Subscribe(request);

    if (outcome.IsSuccess()) {
        std::cout << "Subscribed successfully." << std::endl;
        std::cout << "Subscription ARN '" <<
 outcome.GetResult().GetSubscriptionArn()
                  << "'." << std::endl;
    }
    else {
        std::cerr << "Error while subscribing " <<
 outcome.GetError().GetMessage()
                  << std::endl;
    }
```

```
    return outcome.IsSuccess();
}
```

Abonnieren Sie eine Lambda-Funktion für ein Thema.

```
//! Subscribe to an Amazon Simple Notification Service (Amazon SNS) topic with
 delivery to an AWS Lambda function.
/*!
  \param topicARN: The Amazon Resource Name (ARN) for an Amazon SNS topic.
  \param lambdaFunctionARN: The ARN for an AWS Lambda function.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SNS::subscribeLambda(const Aws::String &topicARN,
                                  const Aws::String &lambdaFunctionARN,
                                  const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::SNS::Model::SubscribeRequest request;
    request.SetTopicArn(topicARN);
    request.SetProtocol("lambda");
    request.SetEndpoint(lambdaFunctionARN);

    const Aws::SNS::Model::SubscribeOutcome outcome =
 snsClient.Subscribe(request);

    if (outcome.IsSuccess()) {
        std::cout << "Subscribed successfully." << std::endl;
        std::cout << "Subscription ARN '" <<
 outcome.GetResult().GetSubscriptionArn()
                  << "'." << std::endl;
    }
    else {
        std::cerr << "Error while subscribing " <<
 outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
```

```
}
```

Abonnieren Sie eine SQS-Warteschlange für ein Thema.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::SNS::SNSClient snsClient(clientConfiguration);

            Aws::SNS::Model::SubscribeRequest request;
            request.SetTopicArn(topicARN);
            request.SetProtocol("sqs");
            request.SetEndpoint(queueARN);

            Aws::SNS::Model::SubscribeOutcome outcome =
snsClient.Subscribe(request);

            if (outcome.IsSuccess()) {
                Aws::String subscriptionARN =
outcome.GetResult().GetSubscriptionArn();
                std::cout << "The queue '" << queueName
                          << "' has been subscribed to the topic '"
                          << "'" << topicName << "'" << std::endl;
                std::cout << "with the subscription ARN '" << subscriptionARN <<
"."
                          << std::endl;
                subscriptionARNS.push_back(subscriptionARN);
            }
            else {
                std::cerr << "Error with TopicsAndQueues::Subscribe. "
                          << outcome.GetError().GetMessage()
                          << std::endl;

                cleanUp(topicARN,
                        queueURLS,
                        subscriptionARNS,
                        snsClient,
                        sqsClient);

                return false;
            }
```

Abonnieren Sie ein Thema mit einem Filter.

```
        static const Aws::String TONE_ATTRIBUTE("tone");
        static const Aws::Vector<Aws::String> TONES = {"cheerful", "funny",
 "serious",
                                                       "sincere"};

        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::SNS::SNSClient snsClient(clientConfiguration);

            Aws::SNS::Model::SubscribeRequest request;
            request.SetTopicArn(topicARN);
            request.SetProtocol("sqs");
            request.SetEndpoint(queueARN);
            if (isFifoTopic) {
                if (first) {
                    std::cout << "Subscriptions to a FIFO topic can have
 filters."
                              << std::endl;
                    std::cout
                            << "If you add a filter to this subscription, then
 only the filtered messages "
                            << "will be received in the queue." << std::endl;
                    std::cout << "For information about message filtering, "
                              << "see https://docs.aws.amazon.com/sns/latest/dg/
sns-message-filtering.html"
                              << std::endl;
                    std::cout << "For this example, you can filter messages by a
 \""
                              << TONE_ATTRIBUTE << "\" attribute." << std::endl;
                }

                std::ostringstream ostringstream;
                ostringstream << "Filter messages for \"" << queueName
                              << "\"'s subscription to the topic \""
                              << topicName << "\"?  (y/n)";

                // Add filter if user answers yes.
```

```
                if (askYesNoQuestion(ostringstream.str())) {
                    Aws::String jsonPolicy = getFilterPolicyFromUser();
                    if (!jsonPolicy.empty()) {
                        filteringMessages = true;

                        std::cout << "This is the filter policy for this
subscription."
                                  << std::endl;
                        std::cout << jsonPolicy << std::endl;

                        request.AddAttributes("FilterPolicy", jsonPolicy);
                    }
                    else {
                        std::cout
                                << "Because you did not select any attributes, no
filter "
                                << "will be added to this subscription." <<
std::endl;
                    }
                }
            }  // if (isFifoTopic)
            Aws::SNS::Model::SubscribeOutcome outcome =
snsClient.Subscribe(request);

            if (outcome.IsSuccess()) {
                Aws::String subscriptionARN =
outcome.GetResult().GetSubscriptionArn();
                std::cout << "The queue '" << queueName
                          << "' has been subscribed to the topic '"
                          << "'" << topicName << "'" << std::endl;
                std::cout << "with the subscription ARN '" << subscriptionARN <<
"."
                          << std::endl;
                subscriptionARNS.push_back(subscriptionARN);
            }
            else {
                std::cerr << "Error with TopicsAndQueues::Subscribe. "
                          << outcome.GetError().GetMessage()
                          << std::endl;

                cleanUp(topicARN,
                        queueURLS,
                        subscriptionARNS,
                        snsClient,
```

```
                        sqsClient);

                return false;
            }

//! Routine that lets the user select attributes for a subscription filter
 policy.
/*!
 \sa getFilterPolicyFromUser()
 \return Aws::String: The filter policy as JSON.
 */
Aws::String AwsDoc::TopicsAndQueues::getFilterPolicyFromUser() {
    std::cout
            << "You can filter messages by one or more of the following \""
            << TONE_ATTRIBUTE << "\" attributes." << std::endl;

    std::vector<Aws::String> filterSelections;
    int selection;
    do {
        for (size_t j = 0; j < TONES.size(); ++j) {
            std::cout << "  " << (j + 1) << ". " << TONES[j]
                      << std::endl;
        }
        selection = askQuestionForIntRange(
                "Enter a number (or enter zero to stop adding more). ",
                0, static_cast<int>(TONES.size()));

        if (selection != 0) {
            const Aws::String &selectedTone(TONES[selection - 1]);
            // Add the tone to the selection if it is not already added.
            if (std::find(filterSelections.begin(),
                          filterSelections.end(),
                          selectedTone)
                == filterSelections.end()) {
                filterSelections.push_back(selectedTone);
            }
        }
    } while (selection != 0);

    Aws::String result;
    if (!filterSelections.empty()) {
        std::ostringstream jsonPolicyStream;
        jsonPolicyStream << "{ \"" << TONE_ATTRIBUTE << "\": [";
```

```
        for (size_t j = 0; j < filterSelections.size(); ++j) {
            jsonPolicyStream << "\"" << filterSelections[j] << "\"";
            if (j < filterSelections.size() - 1) {
                jsonPolicyStream << ",";
            }
        }
        jsonPolicyStream << "] }";

        result = jsonPolicyStream.str();
    }

    return result;
}
```

- Details zu API finden Sie unter Abonnieren in der AWS SDK for C++ -API-Referenz.

CLI

AWS CLI

So abonnieren Sie ein Thema

Der folgende `subscribe`-Befehl abonniert das angegebene Thema mit eine E-Mail-Adresse.

```
aws sns subscribe \
    --topic-arn arn:aws:sns:us-west-2:123456789012:my-topic \
    --protocol email \
    --notification-endpoint my-email@example.com
```

Ausgabe:

```
{
    "SubscriptionArn": "pending confirmation"
}
```

- API-Details finden Sie unter Subscribe in der AWS CLI -Befehlsreferenz.

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

Abonnieren Sie eine Warteschlange für ein Thema mit optionalen Filtern.

```
// SnsActions encapsulates the Amazon Simple Notification Service (Amazon SNS)
 actions
// used in the examples.
type SnsActions struct {
 SnsClient *sns.Client
}



// SubscribeQueue subscribes an Amazon Simple Queue Service (Amazon SQS) queue to
 an
// Amazon SNS topic. When filterMap is not nil, it is used to specify a filter
 policy
// so that messages are only sent to the queue when the message has the specified
 attributes.
func (actor SnsActions) SubscribeQueue(topicArn string, queueArn string,
 filterMap map[string][]string) (string, error) {
 var subscriptionArn string
 var attributes map[string]string
 if filterMap != nil {
  filterBytes, err := json.Marshal(filterMap)
  if err != nil {
   log.Printf("Couldn't create filter policy, here's why: %v\n", err)
   return "", err
  }
  attributes = map[string]string{"FilterPolicy": string(filterBytes)}
 }
 output, err := actor.SnsClient.Subscribe(context.TODO(), &sns.SubscribeInput{
  Protocol:              aws.String("sqs"),
  TopicArn:              aws.String(topicArn),
```

```
  Attributes:            attributes,
  Endpoint:              aws.String(queueArn),
  ReturnSubscriptionArn: true,
 })
 if err != nil {
  log.Printf("Couldn't susbscribe queue %v to topic %v. Here's why: %v\n",
   queueArn, topicArn, err)
 } else {
  subscriptionArn = *output.SubscriptionArn
 }

 return subscriptionArn, err
}
```

- Details zu API finden Sie unter Abonnieren in der AWS SDK for Go -API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

Abonnieren Sie eine E-Mail-Adresse für ein Thema.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SnsException;
import software.amazon.awssdk.services.sns.model.SubscribeRequest;
import software.amazon.awssdk.services.sns.model.SubscribeResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
```

```
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class SubscribeEmail {
    public static void main(String[] args) {
        final String usage = """
                Usage:     <topicArn> <email>

                Where:
                   topicArn - The ARN of the topic to subscribe.
                   email - The email address to use.
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String topicArn = args[0];
        String email = args[1];
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        subEmail(snsClient, topicArn, email);
        snsClient.close();
    }

    public static void subEmail(SnsClient snsClient, String topicArn, String
 email) {
        try {
            SubscribeRequest request = SubscribeRequest.builder()
                    .protocol("email")
                    .endpoint(email)
                    .returnSubscriptionArn(true)
                    .topicArn(topicArn)
                    .build();

            SubscribeResponse result = snsClient.subscribe(request);
            System.out.println("Subscription ARN: " + result.subscriptionArn() +
 "\n\n Status is "
                    + result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
```

```
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

Abonnieren Sie ein Thema über einen HTTP-Endpunkt.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SnsException;
import software.amazon.awssdk.services.sns.model.SubscribeRequest;
import software.amazon.awssdk.services.sns.model.SubscribeResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class SubscribeHTTPS {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <topicArn> <url>

                Where:
                   topicArn - The ARN of the topic to subscribe.
                   url - The HTTPS endpoint that you want to receive
 notifications.
                """;

        if (args.length < 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String topicArn = args[0];
        String url = args[1];
```

```
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        subHTTPS(snsClient, topicArn, url);
        snsClient.close();
    }

    public static void subHTTPS(SnsClient snsClient, String topicArn, String url)
 {
        try {
            SubscribeRequest request = SubscribeRequest.builder()
                    .protocol("https")
                    .endpoint(url)
                    .returnSubscriptionArn(true)
                    .topicArn(topicArn)
                    .build();

            SubscribeResponse result = snsClient.subscribe(request);
            System.out.println("Subscription ARN is " + result.subscriptionArn()
 + "\n\n Status is "
                    + result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

Abonnieren Sie eine Lambda-Funktion für ein Thema.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SnsException;
import software.amazon.awssdk.services.sns.model.SubscribeRequest;
import software.amazon.awssdk.services.sns.model.SubscribeResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
```

```
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class SubscribeLambda {

    public static void main(String[] args) {

        final String usage = """

                Usage:    <topicArn> <lambdaArn>

                Where:
                   topicArn - The ARN of the topic to subscribe.
                   lambdaArn - The ARN of an AWS Lambda function.
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String topicArn = args[0];
        String lambdaArn = args[1];
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        String arnValue = subLambda(snsClient, topicArn, lambdaArn);
        System.out.println("Subscription ARN: " + arnValue);
        snsClient.close();
    }

    public static String subLambda(SnsClient snsClient, String topicArn, String
 lambdaArn) {
        try {
            SubscribeRequest request = SubscribeRequest.builder()
                    .protocol("lambda")
                    .endpoint(lambdaArn)
                    .returnSubscriptionArn(true)
                    .topicArn(topicArn)
                    .build();
```

```
            SubscribeResponse result = snsClient.subscribe(request);
            return result.subscriptionArn();

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
}
```

- Details zu API finden Sie unter Abonnieren in der AWS SDK for Java 2.x -API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie den Client in einem separaten Modul und exportieren Sie ihn.

```
import { SNSClient } from "@aws-sdk/client-sns";

// The AWS Region can be provided here using the `region` property. If you leave
 it blank
// the SDK will default to the region set in your AWS config.
export const snsClient = new SNSClient({});
```

Importieren Sie das SDK- und Client-Module und rufen Sie die API auf.

```
import { SubscribeCommand } from "@aws-sdk/client-sns";
import { snsClient } from "../libs/snsClient.js";

/**
```

```
 * @param {string} topicArn - The ARN of the topic for which you wish to confirm
 a subscription.
 * @param {string} emailAddress - The email address that is subscribed to the
 topic.
 */
export const subscribeEmail = async (
  topicArn = "TOPIC_ARN",
  emailAddress = "usern@me.com",
) => {
  const response = await snsClient.send(
    new SubscribeCommand({
      Protocol: "email",
      TopicArn: topicArn,
      Endpoint: emailAddress,
    }),
  );
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: 'c8e35bcd-b3c0-5940-9f66-06f6fcc108f0',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   SubscriptionArn: 'pending confirmation'
  // }
};
```

Abonnieren Sie eine mobile Anwendung für ein Thema.

```
import { SubscribeCommand } from "@aws-sdk/client-sns";
import { snsClient } from "../libs/snsClient.js";

/**
 * @param {string} topicArn - The ARN of the topic the subscriber is subscribing
 to.
 * @param {string} endpoint - The Endpoint ARN of an application. This endpoint
 is created
 *                            when an application registers for notifications.
 */
```

```
export const subscribeApp = async (
  topicArn = "TOPIC_ARN",
  endpoint = "ENDPOINT",
) => {
  const response = await snsClient.send(
    new SubscribeCommand({
      Protocol: "application",
      TopicArn: topicArn,
      Endpoint: endpoint,
    }),
  );
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: 'c8e35bcd-b3c0-5940-9f66-06f6fcc108f0',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   SubscriptionArn: 'pending confirmation'
  // }
  return response;
};
```

Abonnieren Sie eine Lambda-Funktion für ein Thema.

```
import { SubscribeCommand } from "@aws-sdk/client-sns";
import { snsClient } from "../libs/snsClient.js";

/**
 * @param {string} topicArn - The ARN of the topic the subscriber is subscribing
 to.
 * @param {string} endpoint - The Endpoint ARN of and AWS Lambda function.
 */
export const subscribeLambda = async (
  topicArn = "TOPIC_ARN",
  endpoint = "ENDPOINT",
) => {
  const response = await snsClient.send(
    new SubscribeCommand({
```

```
      Protocol: "lambda",
      TopicArn: topicArn,
      Endpoint: endpoint,
    }),
  );
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: 'c8e35bcd-b3c0-5940-9f66-06f6fcc108f0',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   SubscriptionArn: 'pending confirmation'
  // }
  return response;
};
```

Abonnieren Sie eine SQS-Warteschlange für ein Thema.

```
import { SubscribeCommand, SNSClient } from "@aws-sdk/client-sns";

const client = new SNSClient({});

export const subscribeQueue = async (
  topicArn = "TOPIC_ARN",
  queueArn = "QUEUE_ARN",
) => {
  const command = new SubscribeCommand({
    TopicArn: topicArn,
    Protocol: "sqs",
    Endpoint: queueArn,
  });

  const response = await client.send(command);
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: '931e13d9-5e2b-543f-8781-4e9e494c5ff2',
```

```
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   SubscriptionArn: 'arn:aws:sns:us-east-1:xxxxxxxxxxxx:subscribe-queue-
test-430895:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx'
  // }
  return response;
};
```

Abonnieren Sie ein Thema mit einem Filter.

```
import { SubscribeCommand, SNSClient } from "@aws-sdk/client-sns";

const client = new SNSClient({});

export const subscribeQueueFiltered = async (
  topicArn = "TOPIC_ARN",
  queueArn = "QUEUE_ARN",
) => {
  const command = new SubscribeCommand({
    TopicArn: topicArn,
    Protocol: "sqs",
    Endpoint: queueArn,
    Attributes: {
      // This subscription will only receive messages with the 'event' attribute
 set to 'order_placed'.
      FilterPolicyScope: "MessageAttributes",
      FilterPolicy: JSON.stringify({
        event: ["order_placed"],
      }),
    },
  });

  const response = await client.send(command);
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: '931e13d9-5e2b-543f-8781-4e9e494c5ff2',
  //     extendedRequestId: undefined,
```

```
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   SubscriptionArn: 'arn:aws:sns:us-east-1:xxxxxxxxxxxx:subscribe-queue-
test-430895:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx'
  // }
  return response;
};
```

- Weitere Informationen finden Sie im AWS SDK for JavaScript -Entwicklerhandbuch.
- Details zu API finden Sie unter Abonnieren in der AWS SDK for JavaScript -API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

Abonnieren Sie eine E-Mail-Adresse für ein Thema.

```
suspend fun subEmail(topicArnVal: String, email: String): String {

    val request = SubscribeRequest {
        protocol = "email"
        endpoint = email
        returnSubscriptionArn = true
        topicArn = topicArnVal
    }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.subscribe(request)
        return result.subscriptionArn.toString()
    }
}
```

Abonnieren Sie eine Lambda-Funktion für ein Thema.

```
suspend fun subLambda(topicArnVal: String?, lambdaArn: String?) {

    val request = SubscribeRequest {
        protocol = "lambda"
        endpoint = lambdaArn
        returnSubscriptionArn = true
        topicArn = topicArnVal
    }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.subscribe(request)
        println(" The subscription Arn is ${result.subscriptionArn}")
    }
}
```

- Details zu API finden Sie unter Subscribe (Abonnieren) in der AWS -SDK-für-Kotlin-API-Referenz.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

Abonnieren Sie eine E-Mail-Adresse für ein Thema.

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
```

```
 * Prepares to subscribe an endpoint by sending the endpoint a confirmation
 message.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/
guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$protocol = 'email';
$endpoint = 'sample@example.com';
$topic = 'arn:aws:sns:us-east-1:111122223333:MyTopic';

try {
    $result = $SnSclient->subscribe([
        'Protocol' => $protocol,
        'Endpoint' => $endpoint,
        'ReturnSubscriptionArn' => true,
        'TopicArn' => $topic,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

Abonnieren Sie ein Thema über einen HTTP-Endpunkt.

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
```

```
 * Prepares to subscribe an endpoint by sending the endpoint a confirmation
 message.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/
guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$protocol = 'https';
$endpoint = 'https://';
$topic = 'arn:aws:sns:us-east-1:111122223333:MyTopic';

try {
    $result = $SnSclient->subscribe([
        'Protocol' => $protocol,
        'Endpoint' => $endpoint,
        'ReturnSubscriptionArn' => true,
        'TopicArn' => $topic,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- Details zu API finden Sie unter Abonnieren in der AWS SDK for PHP -API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

Abonnieren Sie eine E-Mail-Adresse für ein Thema.

```
class SnsWrapper:
    """Encapsulates Amazon SNS topic and subscription functions."""

    def __init__(self, sns_resource):
        """
        :param sns_resource: A Boto3 Amazon SNS resource.
        """
        self.sns_resource = sns_resource


    @staticmethod
    def subscribe(topic, protocol, endpoint):
        """
        Subscribes an endpoint to the topic. Some endpoint types, such as email,
        must be confirmed before their subscriptions are active. When a
 subscription
        is not confirmed, its Amazon Resource Number (ARN) is set to
        'PendingConfirmation'.

        :param topic: The topic to subscribe to.
        :param protocol: The protocol of the endpoint, such as 'sms' or 'email'.
        :param endpoint: The endpoint that receives messages, such as a phone
 number
                         (in E.164 format) for SMS messages, or an email address
 for
                         email messages.
        :return: The newly added subscription.
        """
        try:
            subscription = topic.subscribe(
                Protocol=protocol, Endpoint=endpoint, ReturnSubscriptionArn=True
```

```
            )
            logger.info("Subscribed %s %s to topic %s.", protocol, endpoint,
 topic.arn)
        except ClientError:
            logger.exception(
                "Couldn't subscribe %s %s to topic %s.", protocol, endpoint,
 topic.arn
            )
            raise
        else:
            return subscription
```

- Details zu API finden Sie unter Abonnieren in der AWS API-Referenz zu SDK for Python (Boto3).

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

Abonnieren Sie eine E-Mail-Adresse für ein Thema.

```
require "aws-sdk-sns"
require "logger"

# Represents a service for creating subscriptions in Amazon Simple Notification
 Service (SNS)
class SubscriptionService
  # Initializes the SubscriptionService with an SNS client
  #
  # @param sns_client [Aws::SNS::Client] The SNS client
  def initialize(sns_client)
    @sns_client = sns_client
    @logger = Logger.new($stdout)
```

```
  end

  # Attempts to create a subscription to a topic
  #
  # @param topic_arn [String] The ARN of the SNS topic
  # @param protocol [String] The subscription protocol (e.g., email)
  # @param endpoint [String] The endpoint that receives the notifications (email
 address)
  # @return [Boolean] true if subscription was successfully created, false
 otherwise
  def create_subscription(topic_arn, protocol, endpoint)
    @sns_client.subscribe(topic_arn: topic_arn, protocol: protocol, endpoint:
 endpoint)
    @logger.info("Subscription created successfully.")
    true
  rescue Aws::SNS::Errors::ServiceError => e
    @logger.error("Error while creating the subscription: #{e.message}")
    false
  end
end

# Main execution if the script is run directly
if $PROGRAM_NAME == __FILE__
  protocol = "email"
  endpoint = "EMAIL_ADDRESS" # Should be replaced with a real email address
  topic_arn = "TOPIC_ARN"    # Should be replaced with a real topic ARN

  sns_client = Aws::SNS::Client.new
  subscription_service = SubscriptionService.new(sns_client)

  @logger.info("Creating the subscription.")
  unless subscription_service.create_subscription(topic_arn, protocol, endpoint)
    @logger.error("Subscription creation failed. Stopping program.")
    exit 1
  end
end
```

- Weitere Informationen finden Sie im AWS SDK for Ruby -Entwicklerhandbuch.
- Details zu API finden Sie unter Abonnieren in der AWS SDK for Ruby -API-Referenz.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

Abonnieren Sie eine E-Mail-Adresse für ein Thema.

```
async fn subscribe_and_publish(
    client: &Client,
    topic_arn: &str,
    email_address: &str,
) -> Result<(), Error> {
    println!("Receiving on topic with ARN: `{}`", topic_arn);

    let rsp = client
        .subscribe()
        .topic_arn(topic_arn)
        .protocol("email")
        .endpoint(email_address)
        .send()
        .await?;

    println!("Added a subscription: {:?}", rsp);

    let rsp = client
        .publish()
        .topic_arn(topic_arn)
        .message("hello sns!")
        .send()
        .await?;

    println!("Published message: {:?}", rsp);

    Ok(())
}
```

- Details zu API finden Sie unter Abonnieren in der AWS -SDK-für-Rust-API-Referenz.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

Abonnieren Sie eine E-Mail-Adresse für ein Thema.

```
   TRY.
       oo_result = lo_sns->subscribe(                       "oo_result is
returned for testing purposes."
               iv_topicarn = iv_topic_arn
               iv_protocol = 'email'
               iv_endpoint = iv_email_address
               iv_returnsubscriptionarn = abap_true
           ).
       MESSAGE 'Email address subscribed to SNS topic.' TYPE 'I'.
     CATCH /aws1/cx_snsnotfoundexception.
       MESSAGE 'Topic does not exist.' TYPE 'E'.
     CATCH /aws1/cx_snssubscriptionlmte00.
       MESSAGE 'Unable to create subscriptions. You have reached the maximum
number of subscriptions allowed.' TYPE 'E'.
   ENDTRY.
```

- Details zu API finden Sie unter Subscribe (Anmelden) in der API-Referenz für das AWS SDK für SAP ABAP.

Codebeispiele für Amazon SNS mit AWS SDKs

Die folgenden Codebeispiele zeigen, wie Amazon SNS mit einem AWS Software Development Kit (SDK) verwendet wird.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Servicefunktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien und serviceübergreifenden Beispiele sehen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie eine bestimmte Aufgabe ausführen können, indem Sie mehrere Funktionen innerhalb desselben Services aufrufen.

Serviceübergreifende Beispiele sind Beispielanwendungen, die über mehrere AWS-Services hinweg arbeiten.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unterAmazon SNS mit einem AWS SDK verwenden. Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Erste Schritte

Hello Amazon SNS

Die folgenden Codebeispiele veranschaulichen die ersten Schritte mit Amazon SNS.

.NET

AWS SDK for .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

```
using Amazon.SimpleNotificationService;
using Amazon.SimpleNotificationService.Model;

namespace SNSActions;
```

```
public static class HelloSNS
{
    static async Task Main(string[] args)
    {
        var snsClient = new AmazonSimpleNotificationServiceClient();

        Console.WriteLine($"Hello Amazon SNS! Following are some of your
 topics:");
        Console.WriteLine();

        // You can use await and any of the async methods to get a response.
        // Let's get a list of topics.
        var response = await snsClient.ListTopicsAsync(
            new ListTopicsRequest());

        foreach (var topic in response.Topics)
        {
            Console.WriteLine($"\tTopic ARN: {topic.TopicArn}");
            Console.WriteLine();
        }
    }
}
```

- Einzelheiten zur API finden Sie ListTopicsin der AWS SDK for .NET API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

Code für die C MakeLists .txt-CMake-Datei.

```
# Set the minimum required version of CMake for this project.
cmake_minimum_required(VERSION 3.13)
```

```
# Set the AWS service components used by this project.
set(SERVICE_COMPONENTS sns)

# Set this project's name.
project("hello_sns")

# Set the C++ standard to use to build this target.
# At least C++ 11 is required for the AWS SDK for C++.
set(CMAKE_CXX_STANDARD 11)

# Use the MSVC variable to determine if this is a Windows build.
set(WINDOWS_BUILD ${MSVC})

if (WINDOWS_BUILD) # Set the location where CMake can find the installed
 libraries for the AWS SDK.
    string(REPLACE ";" "/aws-cpp-sdk-all;" SYSTEM_MODULE_PATH
 "${CMAKE_SYSTEM_PREFIX_PATH}/aws-cpp-sdk-all")
    list(APPEND CMAKE_PREFIX_PATH ${SYSTEM_MODULE_PATH})
endif ()

# Find the AWS SDK for C++ package.
find_package(AWSSDK REQUIRED COMPONENTS ${SERVICE_COMPONENTS})

if (WINDOWS_BUILD AND AWSSDK_INSTALL_AS_SHARED_LIBS)
    # Copy relevant AWS SDK for C++ libraries into the current binary directory
 for running and debugging.

    # set(BIN_SUB_DIR "/Debug") # If you are building from the command line you
 may need to uncomment this
    # and set the proper subdirectory to the executables' location.

    AWSSDK_CPY_DYN_LIBS(SERVICE_COMPONENTS ""
 ${CMAKE_CURRENT_BINARY_DIR}${BIN_SUB_DIR})
endif ()

add_executable(${PROJECT_NAME}
        hello_sns.cpp)

target_link_libraries(${PROJECT_NAME}
        ${AWSSDK_LINK_LIBRARIES})
```

Code für die Quelldatei „hello_sns.cpp“.

```
#include <aws/core/Aws.h>
#include <aws/sns/SNSClient.h>
#include <aws/sns/model/ListTopicsRequest.h>
#include <iostream>

/*
 *  A "Hello SNS" starter application which initializes an Amazon Simple
 Notification
 *  Service (Amazon SNS) client and lists the SNS topics in the current account.
 *
 *  main function
 *
 *  Usage: 'hello_sns'
 *
 */

int main(int argc, char **argv) {
    Aws::SDKOptions options;
    // Optionally change the log level for debugging.
//   options.loggingOptions.logLevel = Utils::Logging::LogLevel::Debug;
    Aws::InitAPI(options); // Should only be called once.
    {
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

        Aws::SNS::SNSClient snsClient(clientConfig);

        Aws::Vector<Aws::SNS::Model::Topic> allTopics;
        Aws::String nextToken; // Next token is used to handle a paginated
 response.
        do {
            Aws::SNS::Model::ListTopicsRequest request;

            if (!nextToken.empty()) {
                request.SetNextToken(nextToken);
            }

            const Aws::SNS::Model::ListTopicsOutcome outcome =
 snsClient.ListTopics(
                    request);

            if (outcome.IsSuccess()) {
```

```
                const Aws::Vector<Aws::SNS::Model::Topic> &paginatedTopics =
                        outcome.GetResult().GetTopics();
                if (!paginatedTopics.empty()) {
                    allTopics.insert(allTopics.cend(), paginatedTopics.cbegin(),
                                     paginatedTopics.cend());
                }
            }
            else {
                std::cerr << "Error listing topics " <<
 outcome.GetError().GetMessage()
                          << std::endl;
                return 1;
            }

            nextToken = outcome.GetResult().GetNextToken();
        } while (!nextToken.empty());

        std::cout << "Hello Amazon SNS! You have " << allTopics.size() << "
 topic"
                  << (allTopics.size() == 1 ? "" : "s") << " in your account."
                  << std::endl;

        if (!allTopics.empty()) {
            std::cout << "Here are your topic ARNs." << std::endl;
            for (const Aws::SNS::Model::Topic &topic: allTopics) {
                std::cout << "  * " << topic.GetTopicArn() << std::endl;
            }
        }
    }


    Aws::ShutdownAPI(options); // Should only be called once.
    return 0;
}
```

- Einzelheiten zur API finden Sie unter ListTopics AWS SDK for C++API-Referenz.

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

```
package main

import (
 "context"
 "fmt"
 "log"

 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/service/sns"
 "github.com/aws/aws-sdk-go-v2/service/sns/types"
)

// main uses the AWS SDK for Go V2 to create an Amazon Simple Notification
 Service
// (Amazon SNS) client and list the topics in your account.
// This example uses the default settings specified in your shared credentials
// and config files.
func main() {
 sdkConfig, err := config.LoadDefaultConfig(context.TODO())
 if err != nil {
  fmt.Println("Couldn't load default configuration. Have you set up your AWS
 account?")
  fmt.Println(err)
  return
 }
 snsClient := sns.NewFromConfig(sdkConfig)
 fmt.Println("Let's list the topics for your account.")
 var topics []types.Topic
 paginator := sns.NewListTopicsPaginator(snsClient, &sns.ListTopicsInput{})
 for paginator.HasMorePages() {
  output, err := paginator.NextPage(context.TODO())
```

```
  if err != nil {
   log.Printf("Couldn't get topics. Here's why: %v\n", err)
   break
  } else {
   topics = append(topics, output.Topics...)
  }
 }
 if len(topics) == 0 {
  fmt.Println("You don't have any topics!")
 } else {
  for _, topic := range topics {
   fmt.Printf("\t%v\n", *topic.TopicArn)
  }
 }
}
```

- Einzelheiten zur API finden Sie ListTopicsin der AWS SDK for Go API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

```
package com.example.sns;

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SnsException;
import software.amazon.awssdk.services.sns.paginators.ListTopicsIterable;

public class HelloSNS {
    public static void main(String[] args) {
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();
```

```
        listSNSTopics(snsClient);
        snsClient.close();
    }

    public static void listSNSTopics(SnsClient snsClient) {
        try {
            ListTopicsIterable listTopics = snsClient.listTopicsPaginator();
            listTopics.stream()
                    .flatMap(r -> r.topics().stream())
                    .forEach(content -> System.out.println(" Topic ARN: " +
 content.topicArn()));

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Einzelheiten zur API finden Sie ListTopicsin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

Initialisieren Sie einen SNS-Client und listen Sie Themen in Ihrem Konto auf.

```
import { SNSClient, paginateListTopics } from "@aws-sdk/client-sns";

export const helloSns = async () => {
  // The configuration object (`{}`) is required. If the region and credentials
  // are omitted, the SDK uses your local configuration if it exists.
  const client = new SNSClient({});
```

```
  // You can also use `ListTopicsCommand`, but to use that command you must
  // handle the pagination yourself. You can do that by sending the
 `ListTopicsCommand`
  // with the `NextToken` parameter from the previous request.
  const paginatedTopics = paginateListTopics({ client }, {});
  const topics = [];

  for await (const page of paginatedTopics) {
    if (page.Topics?.length) {
      topics.push(...page.Topics);
    }
  }

  const suffix = topics.length === 1 ? "" : "s";

  console.log(
    `Hello, Amazon SNS! You have ${topics.length} topic${suffix} in your
 account.`,
  );
  console.log(topics.map((t) => `  * ${t.TopicArn}`).join("\n"));
};
```

- Einzelheiten zur API finden Sie ListTopicsin der AWS SDK for JavaScript API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

```
import aws.sdk.kotlin.services.sns.SnsClient
import aws.sdk.kotlin.services.sns.model.ListTopicsRequest
import aws.sdk.kotlin.services.sns.paginators.listTopicsPaginated
import kotlinx.coroutines.flow.transform

/**
Before running this Kotlin code example, set up your development environment,
```

```
including your credentials.

For more information, see the following documentation topic:
https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html
 */
suspend fun main() {
    listTopicsPag()
}

suspend fun listTopicsPag() {
    SnsClient { region = "us-east-1" }.use { snsClient ->
        snsClient.listTopicsPaginated(ListTopicsRequest { })
            .transform { it.topics?.forEach { topic -> emit(topic) } }
            .collect { topic ->
                println("The topic ARN is ${topic.topicArn}")
            }
    }
}
```

- API-Details finden Sie ListTopicsin der API-Referenz zum AWS SDK für Kotlin.

Codebeispiele

- Aktionen für Amazon SNS mithilfe AWS von SDKs
  - Verwendung CheckIfPhoneNumberIsOptedOut mit einem AWS SDK oder CLI
  - Verwendung ConfirmSubscription mit einem AWS SDK oder CLI
  - Verwendung CreateTopic mit einem AWS SDK oder CLI
  - Verwendung DeleteTopic mit einem AWS SDK oder CLI
  - Verwendung GetSMSAttributes mit einem AWS SDK oder CLI
  - Verwendung GetTopicAttributes mit einem AWS SDK oder CLI
  - Verwendung ListPhoneNumbersOptedOut mit einem AWS SDK oder CLI
  - Verwendung ListSubscriptions mit einem AWS SDK oder CLI
  - Verwendung ListTopics mit einem AWS SDK oder CLI
  - Verwendung Publish mit einem AWS SDK oder CLI
  - Verwendung SetSMSAttributes mit einem AWS SDK oder CLI
  - Verwendung SetSubscriptionAttributes mit einem AWS SDK oder CLI
  - Verwendung SetSubscriptionAttributesRedrivePolicy mit einem AWS SDK oder CLI

Aktionen für Amazon SNS mithilfe AWS von SDKs

Die folgenden Codebeispiele zeigen, wie einzelne Amazon SNS SNS-Aktionen mit AWS SDKs durchgeführt werden. Diese Auszüge rufen die Amazon-SNS-API auf und sind Codeauszüge aus größeren Programmen, die im Kontext ausgeführt werden müssen. Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes finden.

Die folgenden Beispiele enthalten nur die am häufigsten verwendeten Aktionen. Eine vollständige Liste finden Sie in der API-Referenz für Amazon Simple Notification Service (Amazon SNS).

Beispiele

- Verwendung CheckIfPhoneNumberIsOptedOut mit einem AWS SDK oder CLI
- Verwendung ConfirmSubscription mit einem AWS SDK oder CLI
- Verwendung CreateTopic mit einem AWS SDK oder CLI
- Verwendung DeleteTopic mit einem AWS SDK oder CLI
- Verwendung GetSMSAttributes mit einem AWS SDK oder CLI
- Verwendung GetTopicAttributes mit einem AWS SDK oder CLI
- Verwendung ListPhoneNumbersOptedOut mit einem AWS SDK oder CLI
- Verwendung ListSubscriptions mit einem AWS SDK oder CLI
- Verwendung ListTopics mit einem AWS SDK oder CLI
- Verwendung Publish mit einem AWS SDK oder CLI
- Verwendung SetSMSAttributes mit einem AWS SDK oder CLI
- Verwendung SetSubscriptionAttributes mit einem AWS SDK oder CLI
- Verwendung SetSubscriptionAttributesRedrivePolicy mit einem AWS SDK oder CLI
- Verwendung SetTopicAttributes mit einem AWS SDK oder CLI
- Verwendung Subscribe mit einem AWS SDK oder CLI
- Verwendung TagResource mit einem AWS SDK oder CLI
- Verwendung Unsubscribe mit einem AWS SDK oder CLI

Verwendung **`CheckIfPhoneNumberIsOptedOut`** mit einem AWS SDK oder CLI

Die folgenden Codebeispiele zeigen, wie es verwendet wird`CheckIfPhoneNumberIsOptedOut`.

.NET

AWS SDK for .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

```
    using System;
    using System.Threading.Tasks;
    using Amazon.SimpleNotificationService;
    using Amazon.SimpleNotificationService.Model;

    /// <summary>
    /// This example shows how to use the Amazon Simple Notification Service
    /// (Amazon SNS) to check whether a phone number has been opted out.
    /// </summary>
    public class IsPhoneNumOptedOut
    {
        public static async Task Main()
        {
            string phoneNumber = "+15551112222";

            IAmazonSimpleNotificationService client = new
AmazonSimpleNotificationServiceClient();

            await CheckIfOptedOutAsync(client, phoneNumber);
        }

        /// <summary>
        /// Checks to see if the supplied phone number has been opted out.
        /// </summary>
        /// <param name="client">The initialized Amazon SNS Client object used
        /// to check if the phone number has been opted out.</param>
        /// <param name="phoneNumber">A string representing the phone number
        /// to check.</param>
        public static async Task
CheckIfOptedOutAsync(IAmazonSimpleNotificationService client, string
phoneNumber)
        {
```

```
            var request = new CheckIfPhoneNumberIsOptedOutRequest
            {
                PhoneNumber = phoneNumber,
            };

            try
            {
                var response = await
client.CheckIfPhoneNumberIsOptedOutAsync(request);

                if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
                {
                    string optOutStatus = response.IsOptedOut ? "opted out" :
"not opted out.";
                    Console.WriteLine($"The phone number: {phoneNumber} is
{optOutStatus}");
                }
            }
            catch (AuthorizationErrorException ex)
            {
                Console.WriteLine($"{ex.Message}");
            }
        }
    }
```

- Einzelheiten zur API finden Sie CheckIfPhoneNumberIsOptedOutin der AWS SDK for .NET API-Referenz.

CLI

AWS CLI

So überprüfen Sie SMS-Nachrichten-Abmeldungen für eine Telefonnummer

Im folgenden `check-if-phone-number-is-opted-out` Beispiel wird geprüft, ob die angegebene Telefonnummer den Empfang von SMS-Nachrichten vom AWS Girokonto deaktiviert hat.

```
aws sns check-if-phone-number-is-opted-out \
    --phone-number +1555550100
```

Ausgabe:

```
{
    "isOptedOut": false
}
```

- Einzelheiten zur API finden Sie CheckIfPhoneNumberIsOptedOutunter AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import
 software.amazon.awssdk.services.sns.model.CheckIfPhoneNumberIsOptedOutRequest;
import
 software.amazon.awssdk.services.sns.model.CheckIfPhoneNumberIsOptedOutResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class CheckOptOut {
    public static void main(String[] args) {

        final String usage = """
```

```
                Usage:    <phoneNumber>

                Where:
                   phoneNumber - The mobile phone number to look up (for example,
 +1XXX5550100).

                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String phoneNumber = args[0];
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        checkPhone(snsClient, phoneNumber);
        snsClient.close();
    }

    public static void checkPhone(SnsClient snsClient, String phoneNumber) {
        try {
            CheckIfPhoneNumberIsOptedOutRequest request =
 CheckIfPhoneNumberIsOptedOutRequest.builder()
                    .phoneNumber(phoneNumber)
                    .build();

            CheckIfPhoneNumberIsOptedOutResponse result =
 snsClient.checkIfPhoneNumberIsOptedOut(request);
            System.out.println(
                    result.isOptedOut() + "Phone Number " + phoneNumber + " has
 Opted Out of receiving sns messages." +
                            "\n\nStatus was " +
 result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Einzelheiten zur API finden Sie CheckIfPhoneNumberIsOptedOutin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie den Client in einem separaten Modul und exportieren Sie ihn.

```
import { SNSClient } from "@aws-sdk/client-sns";

// The AWS Region can be provided here using the `region` property. If you leave
 it blank
// the SDK will default to the region set in your AWS config.
export const snsClient = new SNSClient({});
```

Importieren Sie das SDK- und Client-Module und rufen Sie die API auf.

```
import { CheckIfPhoneNumberIsOptedOutCommand } from "@aws-sdk/client-sns";

import { snsClient } from "../libs/snsClient.js";

export const checkIfPhoneNumberIsOptedOut = async (
  phoneNumber = "5555555555",
) => {
  const command = new CheckIfPhoneNumberIsOptedOutCommand({
    phoneNumber,
  });

  const response = await snsClient.send(command);
  console.log(response);
  // {
```

```
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: '3341c28a-cdc8-5b39-a3ee-9fb0ee125732',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   isOptedOut: false
  // }
  return response;
};
```

- Weitere Informationen finden Sie im AWS SDK for JavaScript -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie CheckIfPhoneNumberIsOptedOutin der AWS SDK for JavaScript API-Referenz.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Indicates whether the phone number owner has opted out of receiving SMS
 messages from your AWS SNS account.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/
guide_credentials.html
```

```
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$phone = '+1XXX5550100';

try {
    $result = $SnSclient->checkIfPhoneNumberIsOptedOut([
        'phoneNumber' => $phone,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- Weitere Informationen finden Sie im AWS SDK for PHP -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie CheckIfPhoneNumberIsOptedOutin der AWS SDK for PHP API-Referenz.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unterAmazon SNS mit einem AWS SDK verwenden. Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung **`ConfirmSubscription`** mit einem AWS SDK oder CLI

Die folgenden Codebeispiele zeigen, wie es verwendet wird`ConfirmSubscription`.

CLI

AWS CLI

So bestätigen Sie ein Abonnement

Mit dem folgenden `confirm-subscription`-Befehl wird der Bestätigungsvorgang abgeschlossen, der gestartet wurde, als Sie ein SNS-Thema mit dem Namen `my-topic`

abonniert haben. Der --token-Parameter stammt aus der Bestätigungsnachricht, die an den im Abonnementaufruf angegebenen Benachrichtigungsendpunkt gesendet wurde.

```
aws sns confirm-subscription \
    --topic-arn arn:aws:sns:us-west-2:123456789012:my-topic \
    --token
 2336412f37fb687f5d51e6e241d7700ae02f7124d8268910b858cb4db727ceeb2474bb937929d3bdd7ce5d0c
```

Ausgabe:

```
{
    "SubscriptionArn": "arn:aws:sns:us-west-2:123456789012:my-
topic:8a21d249-4329-4871-acc6-7be709c6ea7f"
}
```

- Einzelheiten zur API finden Sie ConfirmSubscriptionin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.ConfirmSubscriptionRequest;
import software.amazon.awssdk.services.sns.model.ConfirmSubscriptionResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
```

```
 */
public class ConfirmSubscription {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <subscriptionToken> <topicArn>

                Where:
                   subscriptionToken - A short-lived token sent to an endpoint
 during the Subscribe action.
                   topicArn - The ARN of the topic.\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String subscriptionToken = args[0];
        String topicArn = args[1];
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        confirmSub(snsClient, subscriptionToken, topicArn);
        snsClient.close();
    }

    public static void confirmSub(SnsClient snsClient, String subscriptionToken,
 String topicArn) {
        try {
            ConfirmSubscriptionRequest request =
 ConfirmSubscriptionRequest.builder()
                    .token(subscriptionToken)
                    .topicArn(topicArn)
                    .build();

            ConfirmSubscriptionResponse result =
 snsClient.confirmSubscription(request);
            System.out.println("\n\nStatus was " +
 result.sdkHttpResponse().statusCode() + "\n\nSubscription Arn: \n\n"
                    + result.subscriptionArn());

        } catch (SnsException e) {
```

```
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Einzelheiten zur API finden Sie ConfirmSubscriptionin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie den Client in einem separaten Modul und exportieren Sie ihn.

```
import { SNSClient } from "@aws-sdk/client-sns";

// The AWS Region can be provided here using the `region` property. If you leave
 it blank
// the SDK will default to the region set in your AWS config.
export const snsClient = new SNSClient({});
```

Importieren Sie das SDK- und Client-Module und rufen Sie die API auf.

```
import { ConfirmSubscriptionCommand } from "@aws-sdk/client-sns";
import { snsClient } from "../libs/snsClient.js";

/**
 * @param {string} token - This token is sent the subscriber. Only subscribers
 *                         that are not AWS services (HTTP/S, email) need to be
 confirmed.
 * @param {string} topicArn - The ARN of the topic for which you wish to confirm
 a subscription.
```

```
 */
export const confirmSubscription = async (
  token = "TOKEN",
  topicArn = "TOPIC_ARN",
) => {
  const response = await snsClient.send(
    // A subscription only needs to be confirmed if the endpoint type is
    // HTTP/S, email, or in another AWS account.
    new ConfirmSubscriptionCommand({
      Token: token,
      TopicArn: topicArn,
      // If this is true, the subscriber cannot unsubscribe while
 unauthenticated.
      AuthenticateOnUnsubscribe: "false",
    }),
  );
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: '4bb5bce9-805a-5517-8333-e1d2cface90b',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   SubscriptionArn: 'arn:aws:sns:us-east-1:xxxxxxxxxxxx:TOPIC_NAME:xxxxxxxx-
xxxx-xxxx-xxxx-xxxxxxxxxxxx'
  // }
  return response;
};
```

- Weitere Informationen finden Sie im AWS SDK for JavaScript -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie ConfirmSubscriptionin der AWS SDK for JavaScript API-Referenz.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Verifies an endpoint owner's intent to receive messages by
 * validating the token sent to the endpoint by an earlier Subscribe action.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/
guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$subscription_token = 'arn:aws:sns:us-east-1:111122223333:MyTopic:123456-
abcd-12ab-1234-12ba3dc1234a';
$topic = 'arn:aws:sns:us-east-1:111122223333:MyTopic';

try {
    $result = $SnSclient->confirmSubscription([
        'Token' => $subscription_token,
        'TopicArn' => $topic,
    ]);
    var_dump($result);
} catch (AwsException $e) {
```

```
    // output error message if fails
    error_log($e->getMessage());
}
```

- Einzelheiten zur API finden Sie ConfirmSubscriptionin der AWS SDK for PHP API-Referenz.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unterAmazon SNS mit einem AWS SDK verwenden. Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung **`CreateTopic`** mit einem AWS SDK oder CLI

Die folgenden Codebeispiele zeigen, wie es verwendet wird`CreateTopic`.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Erstellen und veröffentlichen zu einem FIFO-Thema
- Veröffentlichen Sie Nachrichten in Warteschlangen

.NET

AWS SDK for .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

Erstellen Sie ein Thema mit einem bestimmten Namen.

```
    using System;
    using System.Threading.Tasks;
    using Amazon.SimpleNotificationService;
    using Amazon.SimpleNotificationService.Model;
```

```
    /// <summary>
    /// This example shows how to use Amazon Simple Notification Service
    /// (Amazon SNS) to add a new Amazon SNS topic.
    /// </summary>
    public class CreateSNSTopic
    {
        public static async Task Main()
        {
            string topicName = "ExampleSNSTopic";

            IAmazonSimpleNotificationService client = new
 AmazonSimpleNotificationServiceClient();

            var topicArn = await CreateSNSTopicAsync(client, topicName);
            Console.WriteLine($"New topic ARN: {topicArn}");
        }

        /// <summary>
        /// Creates a new SNS topic using the supplied topic name.
        /// </summary>
        /// <param name="client">The initialized SNS client object used to
        /// create the new topic.</param>
        /// <param name="topicName">A string representing the topic name.</param>
        /// <returns>The Amazon Resource Name (ARN) of the created topic.</
returns>
        public static async Task<string>
 CreateSNSTopicAsync(IAmazonSimpleNotificationService client, string topicName)
        {
            var request = new CreateTopicRequest
            {
                Name = topicName,
            };

            var response = await client.CreateTopicAsync(request);

            return response.TopicArn;
        }
    }
```

Erstellen Sie ein neues Thema mit einem Namen und spezifischen FIFO- und Deduplizierungsattributen.

```
    /// <summary>
    /// Create a new topic with a name and specific FIFO and de-duplication
 attributes.
    /// </summary>
    /// <param name="topicName">The name for the topic.</param>
    /// <param name="useFifoTopic">True to use a FIFO topic.</param>
    /// <param name="useContentBasedDeduplication">True to use content-based de-
duplication.</param>
    /// <returns>The ARN of the new topic.</returns>
    public async Task<string> CreateTopicWithName(string topicName, bool
 useFifoTopic, bool useContentBasedDeduplication)
    {
        var createTopicRequest = new CreateTopicRequest()
        {
            Name = topicName,
        };

        if (useFifoTopic)
        {
            // Update the name if it is not correct for a FIFO topic.
            if (!topicName.EndsWith(".fifo"))
            {
                createTopicRequest.Name = topicName + ".fifo";
            }

            // Add the attributes from the method parameters.
            createTopicRequest.Attributes = new Dictionary<string, string>
            {
                { "FifoTopic", "true" }
            };
            if (useContentBasedDeduplication)
            {
                createTopicRequest.Attributes.Add("ContentBasedDeduplication",
 "true");
            }
        }

        var createResponse = await
 _amazonSNSClient.CreateTopicAsync(createTopicRequest);
        return createResponse.TopicArn;
    }
```

- Einzelheiten zur API finden Sie CreateTopicin der AWS SDK for .NET API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

```
//! Create an Amazon Simple Notification Service (Amazon SNS) topic.
/*!
  \param topicName: An Amazon SNS topic name.
  \param topicARNResult: String to return the Amazon Resource Name (ARN) for the
 topic.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SNS::createTopic(const Aws::String &topicName,
                              Aws::String &topicARNResult,
                              const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::SNS::Model::CreateTopicRequest request;
    request.SetName(topicName);

    const Aws::SNS::Model::CreateTopicOutcome outcome =
 snsClient.CreateTopic(request);

    if (outcome.IsSuccess()) {
        topicARNResult = outcome.GetResult().GetTopicArn();
        std::cout << "Successfully created an Amazon SNS topic " << topicName
                  << " with topic ARN '" << topicARNResult
                  << "'." << std::endl;

    }
    else {
        std::cerr << "Error creating topic " << topicName << ":" <<
                  outcome.GetError().GetMessage() << std::endl;
```

```
        topicARNResult.clear();
    }

    return outcome.IsSuccess();
}
```

- Einzelheiten zur API finden Sie CreateTopicin der AWS SDK for C++ API-Referenz.

CLI

AWS CLI

So erstellen Sie ein SNS-Thema

Das folgende `create-topic`-Beispiel erstellt ein SNS-Thema namens `my-topic`.

```
aws sns create-topic \
    --name my-topic
```

Ausgabe:

```
{
    "ResponseMetadata": {
        "RequestId": "1469e8d7-1642-564e-b85d-a19b4b341f83"
    },
    "TopicArn": "arn:aws:sns:us-west-2:123456789012:my-topic"
}
```

Weitere Informationen finden Sie unter Verwenden der AWS Befehlszeilenschnittstelle mit Amazon SQS und Amazon SNS im Benutzerhandbuch für die AWS Befehlszeilenschnittstelle.

- Einzelheiten zur API finden Sie unter CreateTopic AWS CLIBefehlsreferenz.

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

```
// SnsActions encapsulates the Amazon Simple Notification Service (Amazon SNS)
 actions
// used in the examples.
type SnsActions struct {
 SnsClient *sns.Client
}



// CreateTopic creates an Amazon SNS topic with the specified name. You can
 optionally
// specify that the topic is created as a FIFO topic and whether it uses content-
based
// deduplication instead of ID-based deduplication.
func (actor SnsActions) CreateTopic(topicName string, isFifoTopic bool,
 contentBasedDeduplication bool) (string, error) {
 var topicArn string
 topicAttributes := map[string]string{}
 if isFifoTopic {
  topicAttributes["FifoTopic"] = "true"
 }
 if contentBasedDeduplication {
  topicAttributes["ContentBasedDeduplication"] = "true"
 }
 topic, err := actor.SnsClient.CreateTopic(context.TODO(), &sns.CreateTopicInput{
  Name:       aws.String(topicName),
  Attributes: topicAttributes,
 })
 if err != nil {
  log.Printf("Couldn't create topic %v. Here's why: %v\n", topicName, err)
 } else {
```

```
  topicArn = *topic.TopicArn
 }

 return topicArn, err
}
```

- Einzelheiten zur API finden Sie CreateTopicin der AWS SDK for Go API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.CreateTopicRequest;
import software.amazon.awssdk.services.sns.model.CreateTopicResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class CreateTopic {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <topicName>

                Where:
```

```
                    topicName - The name of the topic to create (for example,
 mytopic).

                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String topicName = args[0];
        System.out.println("Creating a topic with name: " + topicName);
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        String arnVal = createSNSTopic(snsClient, topicName);
        System.out.println("The topic ARN is" + arnVal);
        snsClient.close();
    }

    public static String createSNSTopic(SnsClient snsClient, String topicName) {
        CreateTopicResponse result;
        try {
            CreateTopicRequest request = CreateTopicRequest.builder()
                    .name(topicName)
                    .build();

            result = snsClient.createTopic(request);
            return result.topicArn();

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
}
```

- Einzelheiten zur API finden Sie CreateTopicin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie den Client in einem separaten Modul und exportieren Sie ihn.

```
import { SNSClient } from "@aws-sdk/client-sns";

// The AWS Region can be provided here using the `region` property. If you leave
 it blank
// the SDK will default to the region set in your AWS config.
export const snsClient = new SNSClient({});
```

Importieren Sie das SDK- und Client-Module und rufen Sie die API auf.

```
import { CreateTopicCommand } from "@aws-sdk/client-sns";
import { snsClient } from "../libs/snsClient.js";

/**
 * @param {string} topicName - The name of the topic to create.
 */
export const createTopic = async (topicName = "TOPIC_NAME") => {
  const response = await snsClient.send(
    new CreateTopicCommand({ Name: topicName }),
  );
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: '087b8ad2-4593-50c4-a496-d7e90b82cf3e',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
```

```
  //   TopicArn: 'arn:aws:sns:us-east-1:xxxxxxxxxxxx:TOPIC_NAME'
  // }
  return response;
};
```

- Weitere Informationen finden Sie im AWS SDK for JavaScript -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie CreateTopicin der AWS SDK for JavaScript API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

```
suspend fun createSNSTopic(topicName: String): String {

    val request = CreateTopicRequest {
        name = topicName
    }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.createTopic(request)
        return result.topicArn.toString()
    }
}
```

- API-Details finden Sie CreateTopicin der API-Referenz zum AWS SDK für Kotlin.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Create a Simple Notification Service topics in your AWS account at the
 requested region.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/
guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$topicname = 'myTopic';

try {
    $result = $SnSclient->createTopic([
        'Name' => $topicname,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- Weitere Informationen finden Sie im AWS SDK for PHP -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie CreateTopicin der AWS SDK for PHP API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

```
class SnsWrapper:
    """Encapsulates Amazon SNS topic and subscription functions."""

    def __init__(self, sns_resource):
        """
        :param sns_resource: A Boto3 Amazon SNS resource.
        """
        self.sns_resource = sns_resource


    def create_topic(self, name):
        """
        Creates a notification topic.

        :param name: The name of the topic to create.
        :return: The newly created topic.
        """
        try:
            topic = self.sns_resource.create_topic(Name=name)
            logger.info("Created topic %s with ARN %s.", name, topic.arn)
        except ClientError:
            logger.exception("Couldn't create topic %s.", name)
            raise
        else:
            return topic
```

- Einzelheiten zur API finden Sie CreateTopicin AWS SDK for Python (Boto3) API Reference.

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu. GitHub Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

```
# This class demonstrates how to create an Amazon Simple Notification Service
 (SNS) topic.
class SNSTopicCreator
  # Initializes an SNS client.
  #
  # Utilizes the default AWS configuration for region and credentials.
  def initialize
    @sns_client = Aws::SNS::Client.new
  end

  # Attempts to create an SNS topic with the specified name.
  #
  # @param topic_name [String] The name of the SNS topic to create.
  # @return [Boolean] true if the topic was successfully created, false
 otherwise.
  def create_topic(topic_name)
    @sns_client.create_topic(name: topic_name)
    puts "The topic '#{topic_name}' was successfully created."
    true
  rescue Aws::SNS::Errors::ServiceError => e
    # Handles SNS service errors gracefully.
    puts "Error while creating the topic named '#{topic_name}': #{e.message}"
    false
  end
end

# Example usage:
```

```
if $PROGRAM_NAME == __FILE__
  topic_name = "YourTopicName" # Replace with your topic name
  sns_topic_creator = SNSTopicCreator.new

  puts "Creating the topic '#{topic_name}'..."
  unless sns_topic_creator.create_topic(topic_name)
    puts "The topic was not created. Stopping program."
    exit 1
  end
end
```

- Weitere Informationen finden Sie im AWS SDK for Ruby -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie CreateTopicin der AWS SDK for Ruby API-Referenz.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

```
async fn make_topic(client: &Client, topic_name: &str) -> Result<(), Error> {
    let resp = client.create_topic().name(topic_name).send().await?;

    println!(
        "Created topic with ARN: {}",
        resp.topic_arn().unwrap_or_default()
    );

    Ok(())
}
```

- Einzelheiten zur API finden Sie CreateTopicin der API-Referenz zum AWS SDK für Rust.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

```
   TRY.
       oo_result = lo_sns->createtopic( iv_name = iv_topic_name ). " oo_result
is returned for testing purposes. "
       MESSAGE 'SNS topic created' TYPE 'I'.
     CATCH /aws1/cx_snstopiclimitexcdex.
       MESSAGE 'Unable to create more topics. You have reached the maximum
number of topics allowed.' TYPE 'E'.
   ENDTRY.
```

- Einzelheiten zur API finden Sie CreateTopicin der API-Referenz zum AWS SDK für SAP ABAP.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unterAmazon SNS mit einem AWS SDK verwenden. Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung **`DeleteTopic`** mit einem AWS SDK oder CLI

Die folgenden Codebeispiele zeigen, wie es verwendet wird`DeleteTopic`.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Veröffentlichen Sie Nachrichten in Warteschlangen

.NET

AWS SDK for .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

Löschen Sie ein Thema mit seinem Themen-ARN.

```
/// <summary>
/// Delete a topic by its topic ARN.
/// </summary>
/// <param name="topicArn">The ARN of the topic.</param>
/// <returns>True if successful.</returns>
public async Task<bool> DeleteTopicByArn(string topicArn)
{
    var deleteResponse = await _amazonSNSClient.DeleteTopicAsync(
        new DeleteTopicRequest()
        {
            TopicArn = topicArn
        });
    return deleteResponse.HttpStatusCode == HttpStatusCode.OK;
}
```

- Einzelheiten zur API finden Sie DeleteTopicin der AWS SDK for .NET API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

```
//! Delete an Amazon Simple Notification Service (Amazon SNS) topic.
/*!
  \param topicARN: The Amazon Resource Name (ARN) for an Amazon SNS topic.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SNS::deleteTopic(const Aws::String &topicARN,
                              const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::SNS::Model::DeleteTopicRequest request;
    request.SetTopicArn(topicARN);

    const Aws::SNS::Model::DeleteTopicOutcome outcome =
 snsClient.DeleteTopic(request);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully deleted the Amazon SNS topic " << topicARN <<
 std::endl;
    }
    else {
        std::cerr << "Error deleting topic " << topicARN << ":" <<
                  outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Einzelheiten zur API finden Sie DeleteTopicin der AWS SDK for C++ API-Referenz.

CLI

AWS CLI

So löschen Sie das SNS-Thema

Das folgende `delete-topic`-Beispiel löscht die angegebene SNS-Thema.

```
aws sns delete-topic \
    --topic-arn "arn:aws:sns:us-west-2:123456789012:my-topic"
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

- Einzelheiten zur API finden Sie DeleteTopicin der AWS CLI Befehlsreferenz.

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

```
// SnsActions encapsulates the Amazon Simple Notification Service (Amazon SNS)
 actions
// used in the examples.
type SnsActions struct {
 SnsClient *sns.Client
}



// DeleteTopic delete an Amazon SNS topic.
func (actor SnsActions) DeleteTopic(topicArn string) error {
 _, err := actor.SnsClient.DeleteTopic(context.TODO(), &sns.DeleteTopicInput{
  TopicArn: aws.String(topicArn)})
 if err != nil {
  log.Printf("Couldn't delete topic %v. Here's why: %v\n", topicArn, err)
 }
 return err
}
```

- Einzelheiten zur API finden Sie DeleteTopicin der AWS SDK for Go API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.DeleteTopicRequest;
import software.amazon.awssdk.services.sns.model.DeleteTopicResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DeleteTopic {
    public static void main(String[] args) {
        final String usage = """

                Usage:     <topicArn>

                Where:
                   topicArn - The ARN of the topic to delete.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String topicArn = args[0];
        SnsClient snsClient = SnsClient.builder()
```

```
                .region(Region.US_EAST_1)
                .build();

        System.out.println("Deleting a topic with name: " + topicArn);
        deleteSNSTopic(snsClient, topicArn);
        snsClient.close();
    }

    public static void deleteSNSTopic(SnsClient snsClient, String topicArn) {
        try {
            DeleteTopicRequest request = DeleteTopicRequest.builder()
                    .topicArn(topicArn)
                    .build();

            DeleteTopicResponse result = snsClient.deleteTopic(request);
            System.out.println("\n\nStatus was " +
 result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Einzelheiten zur API finden Sie DeleteTopicin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie den Client in einem separaten Modul und exportieren Sie ihn.

```
import { SNSClient } from "@aws-sdk/client-sns";
```

```
// The AWS Region can be provided here using the `region` property. If you leave
 it blank
// the SDK will default to the region set in your AWS config.
export const snsClient = new SNSClient({});
```

Importieren Sie das SDK- und Client-Module und rufen Sie die API auf.

```
import { DeleteTopicCommand } from "@aws-sdk/client-sns";
import { snsClient } from "../libs/snsClient.js";

/**
 * @param {string} topicArn - The ARN of the topic to delete.
 */
export const deleteTopic = async (topicArn = "TOPIC_ARN") => {
  const response = await snsClient.send(
    new DeleteTopicCommand({ TopicArn: topicArn }),
  );
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: 'a10e2886-5a8f-5114-af36-75bd39498332',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   }
  // }
};
```

- Weitere Informationen finden Sie im AWS SDK for JavaScript -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie DeleteTopicin der AWS SDK for JavaScript API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

```
suspend fun deleteSNSTopic(topicArnVal: String) {

    val request = DeleteTopicRequest {
        topicArn = topicArnVal
    }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        snsClient.deleteTopic(request)
        println("$topicArnVal was successfully deleted.")
    }
}
```

- API-Details finden Sie DeleteTopicin der API-Referenz zum AWS SDK für Kotlin.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;
```

```
/**
 * Deletes an SNS topic and all its subscriptions.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/
guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$topic = 'arn:aws:sns:us-east-1:111122223333:MyTopic';

try {
    $result = $SnSclient->deleteTopic([
        'TopicArn' => $topic,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- Einzelheiten zur API finden Sie DeleteTopicin der AWS SDK for PHP API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

```
class SnsWrapper:
```

```
    """Encapsulates Amazon SNS topic and subscription functions."""

    def __init__(self, sns_resource):
        """
        :param sns_resource: A Boto3 Amazon SNS resource.
        """
        self.sns_resource = sns_resource


    @staticmethod
    def delete_topic(topic):
        """
        Deletes a topic. All subscriptions to the topic are also deleted.
        """
        try:
            topic.delete()
            logger.info("Deleted topic %s.", topic.arn)
        except ClientError:
            logger.exception("Couldn't delete topic %s.", topic.arn)
            raise
```

- Einzelheiten zur API finden Sie DeleteTopicin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

```
    TRY.
        lo_sns->deletetopic( iv_topicarn = iv_topic_arn ).
        MESSAGE 'SNS topic deleted.' TYPE 'I'.
      CATCH /aws1/cx_snsnotfoundexception.
        MESSAGE 'Topic does not exist.' TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie DeleteTopicin der API-Referenz zum AWS SDK für SAP ABAP.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unterAmazon SNS mit einem AWS SDK verwenden. Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung **`GetSMSAttributes`** mit einem AWS SDK oder CLI

Die folgenden Codebeispiele zeigen, wie es verwendet wird`GetSMSAttributes`.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

```
//! Retrieve the default settings for sending SMS messages from your AWS account
 by using
//! Amazon Simple Notification Service (Amazon SNS).
/*!
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool
AwsDoc::SNS::getSMSType(const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::SNS::Model::GetSMSAttributesRequest request;
    //Set the request to only retrieve the DefaultSMSType setting.
    //Without the following line, GetSMSAttributes would retrieve all settings.
    request.AddAttributes("DefaultSMSType");
```

```
    const Aws::SNS::Model::GetSMSAttributesOutcome outcome =
 snsClient.GetSMSAttributes(
            request);

    if (outcome.IsSuccess()) {
        const Aws::Map<Aws::String, Aws::String> attributes =
                outcome.GetResult().GetAttributes();
        if (!attributes.empty()) {
            for (auto const &att: attributes) {
                std::cout << att.first << ":  " << att.second << std::endl;
            }
        }
        else {
            std::cout
                    << "AwsDoc::SNS::getSMSType - an empty map of attributes was
 retrieved."
                    << std::endl;
        }
    }
    else {
        std::cerr << "Error while getting SMS Type: '"
                  << outcome.GetError().GetMessage()
                  << "'" << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Details zu API finden Sie unter GetSMSAttributes in der AWS SDK for C++ -API-Referenz.

CLI

AWS CLI

So führen Sie die Standard-SMS-Nachrichtenattribute auf

Das folgende `get-sms-attributes`-Beispiel führt die Standardattribute für das Senden von SMS-Nachrichten auf.

```
aws sns get-sms-attributes
```

Ausgabe:

```
{
    "attributes": {
        "DefaultSenderID": "MyName"
    }
}
```

- API-Details finden Sie unter GetSMSAttributes in der AWS CLI -Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import
 software.amazon.awssdk.services.sns.model.GetSubscriptionAttributesRequest;
import
 software.amazon.awssdk.services.sns.model.GetSubscriptionAttributesResponse;
import software.amazon.awssdk.services.sns.model.SnsException;
import java.util.Iterator;
import java.util.Map;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class GetSMSAtrributes {
    public static void main(String[] args) {
```

```
        final String usage = """

                Usage:    <topicArn>

                Where:
                   topicArn - The ARN of the topic from which to retrieve
attributes.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String topicArn = args[0];
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        getSNSAttrutes(snsClient, topicArn);
        snsClient.close();
    }

    public static void getSNSAttrutes(SnsClient snsClient, String topicArn) {
        try {
            GetSubscriptionAttributesRequest request =
GetSubscriptionAttributesRequest.builder()
                    .subscriptionArn(topicArn)
                    .build();

            // Get the Subscription attributes
            GetSubscriptionAttributesResponse res =
snsClient.getSubscriptionAttributes(request);
            Map<String, String> map = res.attributes();

            // Iterate through the map
            Iterator iter = map.entrySet().iterator();
            while (iter.hasNext()) {
                Map.Entry entry = (Map.Entry) iter.next();
                System.out.println("[Key] : " + entry.getKey() + " [Value] : " +
entry.getValue());
            }

        } catch (SnsException e) {
```

```
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }

        System.out.println("\n\nStatus was good");
    }
}
```

- Details zu API finden Sie unter GetSMSAttributes in der AWS SDK for Java 2.x -API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie den Client in einem separaten Modul und exportieren Sie ihn.

```
import { SNSClient } from "@aws-sdk/client-sns";

// The AWS Region can be provided here using the `region` property. If you leave
 it blank
// the SDK will default to the region set in your AWS config.
export const snsClient = new SNSClient({});
```

Importieren Sie das SDK- und Client-Module und rufen Sie die API auf.

```
import { GetSMSAttributesCommand } from "@aws-sdk/client-sns";
import { snsClient } from "../libs/snsClient.js";

export const getSmsAttributes = async () => {
  const response = await snsClient.send(
    // If you have not modified the account-level mobile settings of SNS,
```

```
    // the DefaultSMSType is undefined. For this example, it was set to
    // Transactional.
    new GetSMSAttributesCommand({ attributes: ["DefaultSMSType"] }),
  );

  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: '67ad8386-4169-58f1-bdb9-debd281d48d5',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   attributes: { DefaultSMSType: 'Transactional' }
  // }
  return response;
};
```

- Weitere Informationen finden Sie im AWS SDK for JavaScript -Entwicklerhandbuch.
- Details zu API finden Sie unter GetSMSAttributes in der AWS SDK for JavaScript -API-Referenz.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;
```

```
/**
 * Get the type of SMS Message sent by default from the AWS SNS service.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/
guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

try {
    $result = $SnSclient->getSMSAttributes([
        'attributes' => ['DefaultSMSType'],
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- Weitere Informationen finden Sie im AWS SDK for PHP -Entwicklerhandbuch.
- Details zu API finden Sie unter GetSMSAttributes in der AWS SDK for PHP -API-Referenz.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unterAmazon SNS mit einem AWS SDK verwenden. Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung **`GetTopicAttributes`** mit einem AWS SDK oder CLI

Die folgenden Codebeispiele zeigen, wie es verwendet wird`GetTopicAttributes`.

.NET

AWS SDK for .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

```
    using System;
    using System.Collections.Generic;
    using System.Threading.Tasks;
    using Amazon.SimpleNotificationService;

    /// <summary>
    /// This example shows how to retrieve the attributes of an Amazon Simple
    /// Notification Service (Amazon SNS) topic.
    /// </summary>
    public class GetTopicAttributes
    {
        public static async Task Main()
        {
            string topicArn = "arn:aws:sns:us-
west-2:000000000000:ExampleSNSTopic";
            IAmazonSimpleNotificationService client = new
 AmazonSimpleNotificationServiceClient();

            var attributes = await GetTopicAttributesAsync(client, topicArn);
            DisplayTopicAttributes(attributes);
        }

        /// <summary>
        /// Given the ARN of the Amazon SNS topic, this method retrieves the
 topic
        /// attributes.
        /// </summary>
        /// <param name="client">The initialized Amazon SNS client object used
        /// to retrieve the attributes for the Amazon SNS topic.</param>
        /// <param name="topicArn">The ARN of the topic for which to retrieve
        /// the attributes.</param>
        /// <returns>A Dictionary of topic attributes.</returns>
```

```
        public static async Task<Dictionary<string, string>>
GetTopicAttributesAsync(
            IAmazonSimpleNotificationService client,
            string topicArn)
        {
            var response = await client.GetTopicAttributesAsync(topicArn);

            return response.Attributes;
        }

        /// <summary>
        /// This method displays the attributes for an Amazon SNS topic.
        /// </summary>
        /// <param name="topicAttributes">A Dictionary containing the
        /// attributes for an Amazon SNS topic.</param>
        public static void DisplayTopicAttributes(Dictionary<string, string>
topicAttributes)
        {
            foreach (KeyValuePair<string, string> entry in topicAttributes)
            {
                Console.WriteLine($"{entry.Key}: {entry.Value}\n");
            }
        }
    }
```

- Einzelheiten zur API finden Sie GetTopicAttributesin der AWS SDK for .NET API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

```
//! Retrieve the properties of an Amazon Simple Notification Service (Amazon SNS)
 topic.
/*!
```

```
  \param topicARN: The Amazon Resource Name (ARN) for an Amazon SNS topic.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SNS::getTopicAttributes(const Aws::String &topicARN,
                                     const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);
    Aws::SNS::Model::GetTopicAttributesRequest request;
    request.SetTopicArn(topicARN);

    const Aws::SNS::Model::GetTopicAttributesOutcome outcome =
 snsClient.GetTopicAttributes(
            request);

    if (outcome.IsSuccess()) {
        std::cout << "Topic Attributes:" << std::endl;
        for (auto const &attribute: outcome.GetResult().GetAttributes()) {
            std::cout << "  * " << attribute.first << " : " << attribute.second
                      << std::endl;
        }
    }
    else {
        std::cerr << "Error while getting Topic attributes "
                  << outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Einzelheiten zur API finden Sie GetTopicAttributesin der AWS SDK for C++ API-Referenz.

CLI

AWS CLI

So rufen Sie die Attribute eines Themas ab

Im folgenden `get-topic-attributes`-Beispiel werden die Attribute für das angegebene Thema angezeigt.

```
aws sns get-topic-attributes \
    --topic-arn "arn:aws:sns:us-west-2:123456789012:my-topic"
```

Ausgabe:

```
{
    "Attributes": {
        "SubscriptionsConfirmed": "1",
        "DisplayName": "my-topic",
        "SubscriptionsDeleted": "0",
        "EffectiveDeliveryPolicy": "{\"http\":{\"defaultHealthyRetryPolicy
\":{\"minDelayTarget\":20,\"maxDelayTarget\":20,\"numRetries\":3,
\"numMaxDelayRetries\":0,\"numNoDelayRetries\":0,\"numMinDelayRetries\":0,
\"backoffFunction\":\"linear\"},\"disableSubscriptionOverrides\":false}}",
        "Owner": "123456789012",
        "Policy": "{\"Version\":\"2008-10-17\",\"Id\":\"__default_policy_ID
\",\"Statement\":[{\"Sid\":\"__default_statement_ID\",\"Effect\":
\"Allow\",\"Principal\":{\"AWS\":\"*\"},\"Action\":[\"SNS:Subscribe\",
\"SNS:ListSubscriptionsByTopic\",\"SNS:DeleteTopic\",\"SNS:GetTopicAttributes
\",\"SNS:Publish\",\"SNS:RemovePermission\",\"SNS:AddPermission\",
\"SNS:SetTopicAttributes\"],\"Resource\":\"arn:aws:sns:us-west-2:123456789012:my-
topic\",\"Condition\":{\"StringEquals\":{\"AWS:SourceOwner\":
\"0123456789012\"}}}]}",
        "TopicArn": "arn:aws:sns:us-west-2:123456789012:my-topic",
        "SubscriptionsPending": "0"
    }
}
```

- Einzelheiten zur API finden Sie GetTopicAttributesin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
```

```
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.GetTopicAttributesRequest;
import software.amazon.awssdk.services.sns.model.GetTopicAttributesResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class GetTopicAttributes {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <topicArn>

                Where:
                   topicArn - The ARN of the topic to look up.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String topicArn = args[0];
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        System.out.println("Getting attributes for a topic with name: " +
 topicArn);
        getSNSTopicAttributes(snsClient, topicArn);
        snsClient.close();
    }

    public static void getSNSTopicAttributes(SnsClient snsClient, String
 topicArn) {
        try {
```

```
            GetTopicAttributesRequest request =
 GetTopicAttributesRequest.builder()
                    .topicArn(topicArn)
                    .build();

            GetTopicAttributesResponse result =
 snsClient.getTopicAttributes(request);
            System.out.println("\n\nStatus is " +
 result.sdkHttpResponse().statusCode() + "\n\nAttributes: \n\n"
                    + result.attributes());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Einzelheiten zur API finden Sie GetTopicAttributesin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie den Client in einem separaten Modul und exportieren Sie ihn.

```
import { SNSClient } from "@aws-sdk/client-sns";

// The AWS Region can be provided here using the `region` property. If you leave
 it blank
// the SDK will default to the region set in your AWS config.
export const snsClient = new SNSClient({});
```

Importieren Sie das SDK- und Client-Module und rufen Sie die API auf.

```
import { GetTopicAttributesCommand } from "@aws-sdk/client-sns";
import { snsClient } from "../libs/snsClient.js";

/**
 * @param {string} topicArn - The ARN of the topic to retrieve attributes for.
 */
export const getTopicAttributes = async (topicArn = "TOPIC_ARN") => {
  const response = await snsClient.send(
    new GetTopicAttributesCommand({
      TopicArn: topicArn,
    }),
  );
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: '36b6a24e-5473-5d4e-ac32-ff72d9a73d94',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   Attributes: {
  //     Policy: '{...}',
  //     Owner: 'xxxxxxxxxxxx',
  //     SubscriptionsPending: '1',
  //     TopicArn: 'arn:aws:sns:us-east-1:xxxxxxxxxxxx:mytopic',
  //     TracingConfig: 'PassThrough',
  //     EffectiveDeliveryPolicy: '{"http":{"defaultHealthyRetryPolicy":
{"minDelayTarget":20,"maxDelayTarget":20,"numRetries":3,"numMaxDelayRetries":0,"numNoDela
{"headerContentType":"text/plain; charset=UTF-8"}}}',
  //     SubscriptionsConfirmed: '0',
  //     DisplayName: '',
  //     SubscriptionsDeleted: '1'
  //   }
  // }
  return response;
};
```

- Weitere Informationen finden Sie im AWS SDK for JavaScript -Entwicklerhandbuch.

- Einzelheiten zur API finden Sie GetTopicAttributesin der AWS SDK for JavaScript API-Referenz.

SDK für JavaScript (v2)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Importieren Sie das SDK- und Client-Module und rufen Sie die API auf.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set region
AWS.config.update({ region: "REGION" });

// Create promise and SNS service object
var getTopicAttribsPromise = new AWS.SNS({ apiVersion: "2010-03-31" })
  .getTopicAttributes({ TopicArn: "TOPIC_ARN" })
  .promise();

// Handle promise's fulfilled/rejected states
getTopicAttribsPromise
  .then(function (data) {
    console.log(data);
  })
  .catch(function (err) {
    console.error(err, err.stack);
  });
```

- Weitere Informationen finden Sie im AWS SDK for JavaScript -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie GetTopicAttributesin der AWS SDK for JavaScript API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

```
suspend fun getSNSTopicAttributes(topicArnVal: String) {

    val request = GetTopicAttributesRequest {
        topicArn = topicArnVal
    }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.getTopicAttributes(request)
        println("${result.attributes}")
    }
}
```

- API-Details finden Sie GetTopicAttributesin der API-Referenz zum AWS SDK für Kotlin.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

```
$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);
```

```
$topic = 'arn:aws:sns:us-east-1:111122223333:MyTopic';

try {
    $result = $SnSclient->getTopicAttributes([
        'TopicArn' => $topic,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- Einzelheiten zur API finden Sie GetTopicAttributesin der AWS SDK for PHP API-Referenz.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

```
    TRY.
        oo_result = lo_sns->gettopicattributes( iv_topicarn = iv_topic_arn ). "
oo_result is returned for testing purposes. "
        DATA(lt_attributes) = oo_result->get_attributes( ).
        MESSAGE 'Retrieved attributes/properties of a topic.' TYPE 'I'.
      CATCH /aws1/cx_snsnotfoundexception.
        MESSAGE 'Topic does not exist.' TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie GetTopicAttributesin der API-Referenz zum AWS SDK für SAP ABAP.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unterAmazon SNS mit einem AWS SDK verwenden. Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung **`ListPhoneNumbersOptedOut`** mit einem AWS SDK oder CLI

Die folgenden Codebeispiele zeigen, wie es verwendet wird`ListPhoneNumbersOptedOut`.

CLI

AWS CLI

So führen Sie Abmeldungen für SMS-Nachrichten auf

Das folgende `list-phone-numbers-opted-out`-Beispiel listet die Telefonnummern auf, bei denen der Empfang von SMS-Nachrichten abbestellt wurde.

```
aws sns list-phone-numbers-opted-out
```

Ausgabe:

```
{
    "phoneNumbers": [
        "+15555550100"
    ]
}
```

- Einzelheiten zur API finden Sie ListPhoneNumbersOptedOutin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.ListPhoneNumbersOptedOutRequest;
import
 software.amazon.awssdk.services.sns.model.ListPhoneNumbersOptedOutResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class ListOptOut {
    public static void main(String[] args) {
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        listOpts(snsClient);
        snsClient.close();
    }

    public static void listOpts(SnsClient snsClient) {
        try {
            ListPhoneNumbersOptedOutRequest request =
 ListPhoneNumbersOptedOutRequest.builder().build();
            ListPhoneNumbersOptedOutResponse result =
 snsClient.listPhoneNumbersOptedOut(request);
            System.out.println("Status is " +
 result.sdkHttpResponse().statusCode() + "\n\nPhone Numbers: \n\n"
                    + result.phoneNumbers());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Einzelheiten zur API finden Sie ListPhoneNumbersOptedOutin der AWS SDK for Java 2.x API-Referenz.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Returns a list of phone numbers that are opted out of receiving SMS messages
 from your AWS SNS account.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/
guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

try {
    $result = $SnSclient->listPhoneNumbersOptedOut();
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- Weitere Informationen finden Sie im AWS SDK for PHP -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie ListPhoneNumbersOptedOutin der AWS SDK for PHP API-Referenz.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unterAmazon SNS mit einem AWS SDK verwenden. Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung **`ListSubscriptions`** mit einem AWS SDK oder CLI

Die folgenden Codebeispiele zeigen, wie es verwendet wird`ListSubscriptions`.

.NET

AWS SDK for .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Amazon.SimpleNotificationService;
using Amazon.SimpleNotificationService.Model;

/// <summary>
/// This example will retrieve a list of the existing Amazon Simple
/// Notification Service (Amazon SNS) subscriptions.
/// </summary>
public class ListSubscriptions
{
    public static async Task Main()
    {
```

```
            IAmazonSimpleNotificationService client = new
 AmazonSimpleNotificationServiceClient();

            Console.WriteLine("Enter a topic ARN to list subscriptions for a
 specific topic, " +
                              "or press Enter to list subscriptions for all
 topics.");
            var topicArn = Console.ReadLine();
            Console.WriteLine();

            var subscriptions = await GetSubscriptionsListAsync(client,
 topicArn);

            DisplaySubscriptionList(subscriptions);
        }

        /// <summary>
        /// Gets a list of the existing Amazon SNS subscriptions, optionally by
 specifying a topic ARN.
        /// </summary>
        /// <param name="client">The initialized Amazon SNS client object used
        /// to obtain the list of subscriptions.</param>
        /// <param name="topicArn">The optional ARN of a specific topic. Defaults
 to null.</param>
        /// <returns>A list containing information about each subscription.</
returns>
        public static async Task<List<Subscription>>
 GetSubscriptionsListAsync(IAmazonSimpleNotificationService client, string
 topicArn = null)
        {
            var results = new List<Subscription>();

            if (!string.IsNullOrEmpty(topicArn))
            {
                var paginateByTopic = client.Paginators.ListSubscriptionsByTopic(
                    new ListSubscriptionsByTopicRequest()
                    {
                        TopicArn = topicArn,
                    });

                // Get the entire list using the paginator.
                await foreach (var subscription in paginateByTopic.Subscriptions)
                {
                    results.Add(subscription);
```

```
                }
            }
            else
            {
                var paginateAllSubscriptions =
client.Paginators.ListSubscriptions(new ListSubscriptionsRequest());

                // Get the entire list using the paginator.
                await foreach (var subscription in
paginateAllSubscriptions.Subscriptions)
                {
                    results.Add(subscription);
                }
            }

            return results;
        }

        /// <summary>
        /// Display a list of Amazon SNS subscription information.
        /// </summary>
        /// <param name="subscriptionList">A list containing details for existing
        /// Amazon SNS subscriptions.</param>
        public static void DisplaySubscriptionList(List<Subscription>
subscriptionList)
        {
            foreach (var subscription in subscriptionList)
            {
                Console.WriteLine($"Owner: {subscription.Owner}");
                Console.WriteLine($"Subscription ARN:
{subscription.SubscriptionArn}");
                Console.WriteLine($"Topic ARN: {subscription.TopicArn}");
                Console.WriteLine($"Endpoint: {subscription.Endpoint}");
                Console.WriteLine($"Protocol: {subscription.Protocol}");
                Console.WriteLine();
            }
        }
    }
```

- Einzelheiten zur API finden Sie ListSubscriptionsin der AWS SDK for .NET API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

```
//! Retrieve a list of Amazon Simple Notification Service (Amazon SNS)
 subscriptions.
/*!
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SNS::listSubscriptions(
        const Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::String nextToken; // Next token is used to handle a paginated response.
    bool result = true;
    Aws::Vector<Aws::SNS::Model::Subscription> subscriptions;
    do {
        Aws::SNS::Model::ListSubscriptionsRequest request;

        if (!nextToken.empty()) {
            request.SetNextToken(nextToken);
        }

        const Aws::SNS::Model::ListSubscriptionsOutcome outcome =
 snsClient.ListSubscriptions(
                request);

        if (outcome.IsSuccess()) {
            const Aws::Vector<Aws::SNS::Model::Subscription> &newSubscriptions =
                    outcome.GetResult().GetSubscriptions();
            subscriptions.insert(subscriptions.cend(), newSubscriptions.begin(),
                                 newSubscriptions.end());
        }
        else {
            std::cerr << "Error listing subscriptions "
```

```
                        << outcome.GetError().GetMessage()
                        <<
                        std::endl;
            result = false;
            break;
        }

        nextToken = outcome.GetResult().GetNextToken();
    } while (!nextToken.empty());

    if (result) {
        if (subscriptions.empty()) {
            std::cout << "No subscriptions found" << std::endl;
        }
        else {
            std::cout << "Subscriptions list:" << std::endl;
            for (auto const &subscription: subscriptions) {
                std::cout << "  * " << subscription.GetSubscriptionArn() <<
 std::endl;
            }
        }
    }
    return result;
}
```

- Einzelheiten zur API finden Sie ListSubscriptionsin der AWS SDK for C++ API-Referenz.

CLI

AWS CLI

So führen Sie Ihre SNS-Abonnements auf

Im folgenden `list-subscriptions` Beispiel wird eine Liste der SNS-Abonnements in Ihrem AWS Konto angezeigt.

```
aws sns list-subscriptions
```

Ausgabe:

```
{
```

```
    "Subscriptions": [
        {
            "Owner": "123456789012",
            "Endpoint": "my-email@example.com",
            "Protocol": "email",
            "TopicArn": "arn:aws:sns:us-west-2:123456789012:my-topic",
            "SubscriptionArn": "arn:aws:sns:us-west-2:123456789012:my-
topic:8a21d249-4329-4871-acc6-7be709c6ea7f"
        }
    ]
}
```

- Einzelheiten zur API finden Sie ListSubscriptionsin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.ListSubscriptionsRequest;
import software.amazon.awssdk.services.sns.model.ListSubscriptionsResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class ListSubscriptions {
    public static void main(String[] args) {
        SnsClient snsClient = SnsClient.builder()
```

```
                .region(Region.US_EAST_1)
                .build();

        listSNSSubscriptions(snsClient);
        snsClient.close();
    }

    public static void listSNSSubscriptions(SnsClient snsClient) {
        try {
            ListSubscriptionsRequest request = ListSubscriptionsRequest.builder()
                    .build();

            ListSubscriptionsResponse result =
 snsClient.listSubscriptions(request);
            System.out.println(result.subscriptions());

        } catch (SnsException e) {

            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Einzelheiten zur API finden Sie ListSubscriptionsin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie den Client in einem separaten Modul und exportieren Sie ihn.

```
import { SNSClient } from "@aws-sdk/client-sns";
```

```
// The AWS Region can be provided here using the `region` property. If you leave
 it blank
// the SDK will default to the region set in your AWS config.
export const snsClient = new SNSClient({});
```

Importieren Sie das SDK- und Client-Module und rufen Sie die API auf.

```
import { ListSubscriptionsByTopicCommand } from "@aws-sdk/client-sns";
import { snsClient } from "../libs/snsClient.js";

/**
 * @param {string} topicArn - The ARN of the topic for which you wish to list
 subscriptions.
 */
export const listSubscriptionsByTopic = async (topicArn = "TOPIC_ARN") => {
  const response = await snsClient.send(
    new ListSubscriptionsByTopicCommand({ TopicArn: topicArn }),
  );

  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: '0934fedf-0c4b-572e-9ed2-a3e38fadb0c8',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   Subscriptions: [
  //     {
  //       SubscriptionArn: 'PendingConfirmation',
  //       Owner: '901487484989',
  //       Protocol: 'email',
  //       Endpoint: 'corepyle@amazon.com',
  //       TopicArn: 'arn:aws:sns:us-east-1:901487484989:mytopic'
  //     }
  //   ]
  // }
  return response;
};
```

- Weitere Informationen finden Sie im AWS SDK for JavaScript -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie ListSubscriptionsin der AWS SDK for JavaScript API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

```
suspend fun listSNSSubscriptions() {

    SnsClient { region = "us-east-1" }.use { snsClient ->
        val response = snsClient.listSubscriptions(ListSubscriptionsRequest {})
        response.subscriptions?.forEach { sub ->
            println("Sub ARN is ${sub.subscriptionArn}")
            println("Sub protocol is ${sub.protocol}")
        }
    }
}
```

- API-Details finden Sie ListSubscriptionsin der API-Referenz zum AWS SDK für Kotlin.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

```
require 'vendor/autoload.php';
```

```
use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Returns a list of Amazon SNS subscriptions in the requested region.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/
guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

try {
    $result = $SnSclient->listSubscriptions();
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- Einzelheiten zur API finden Sie ListSubscriptionsin der AWS SDK for PHP API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

```
class SnsWrapper:
```

```
    """Encapsulates Amazon SNS topic and subscription functions."""

    def __init__(self, sns_resource):
        """
        :param sns_resource: A Boto3 Amazon SNS resource.
        """
        self.sns_resource = sns_resource


    def list_subscriptions(self, topic=None):
        """
        Lists subscriptions for the current account, optionally limited to a
        specific topic.

        :param topic: When specified, only subscriptions to this topic are
returned.
        :return: An iterator that yields the subscriptions.
        """
        try:
            if topic is None:
                subs_iter = self.sns_resource.subscriptions.all()
            else:
                subs_iter = topic.subscriptions.all()
            logger.info("Got subscriptions.")
        except ClientError:
            logger.exception("Couldn't get subscriptions.")
            raise
        else:
            return subs_iter
```

- Einzelheiten zur API finden Sie ListSubscriptionsin AWS SDK for Python (Boto3) API Reference.

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu. GitHub Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

```
# This class demonstrates how to list subscriptions to an Amazon Simple
 Notification Service (SNS) topic
class SnsSubscriptionLister
  def initialize(sns_client)
    @sns_client = sns_client
    @logger = Logger.new($stdout)
  end

  # Lists subscriptions for a given SNS topic
  # @param topic_arn [String] The ARN of the SNS topic
  # @return [Types::ListSubscriptionsResponse] subscriptions: The response object
  def list_subscriptions(topic_arn)
    @logger.info("Listing subscriptions for topic: #{topic_arn}")
    subscriptions = @sns_client.list_subscriptions_by_topic(topic_arn: topic_arn)
    subscriptions.subscriptions.each do |subscription|
      @logger.info("Subscription endpoint: #{subscription.endpoint}")
    end
    subscriptions
  rescue Aws::SNS::Errors::ServiceError => e
    @logger.error("Error listing subscriptions: #{e.message}")
    raise
  end
end

# Example usage:
if $PROGRAM_NAME == __FILE__
  sns_client = Aws::SNS::Client.new
  topic_arn = "SNS_TOPIC_ARN" # Replace with your SNS topic ARN
  lister = SnsSubscriptionLister.new(sns_client)

  begin
    lister.list_subscriptions(topic_arn)
```

```
  rescue StandardError => e
    puts "Failed to list subscriptions: #{e.message}"
    exit 1
  end
end
```

- Weitere Informationen finden Sie im AWS SDK for Ruby -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie ListSubscriptionsin der AWS SDK for Ruby API-Referenz.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

```
    TRY.
        oo_result = lo_sns->listsubscriptions( ).                " oo_result is
returned for testing purposes. "
        DATA(lt_subscriptions) = oo_result->get_subscriptions( ).
        MESSAGE 'Retrieved list of subscribers.' TYPE 'I'.
      CATCH /aws1/cx_rt_generic.
        MESSAGE 'Unable to list subscribers.' TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie ListSubscriptionsin der API-Referenz zum AWS SDK für SAP ABAP.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unterAmazon SNS mit einem AWS SDK verwenden. Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung **`ListTopics`** mit einem AWS SDK oder CLI

Die folgenden Codebeispiele zeigen, wie es verwendet wird`ListTopics`.

.NET

AWS SDK for .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

```
    using System;
    using System.Collections.Generic;
    using System.Threading.Tasks;
    using Amazon.SimpleNotificationService;
    using Amazon.SimpleNotificationService.Model;

    /// <summary>
    /// Lists the Amazon Simple Notification Service (Amazon SNS)
    /// topics for the current account.
    /// </summary>
    public class ListSNSTopics
    {
        public static async Task Main()
        {
            IAmazonSimpleNotificationService client = new
AmazonSimpleNotificationServiceClient();

            await GetTopicListAsync(client);
        }

        /// <summary>
        /// Retrieves the list of Amazon SNS topics in groups of up to 100
        /// topics.
        /// </summary>
        /// <param name="client">The initialized Amazon SNS client object used
        /// to retrieve the list of topics.</param>
        public static async Task
GetTopicListAsync(IAmazonSimpleNotificationService client)
```

```
        {
            // If there are more than 100 Amazon SNS topics, the call to
            // ListTopicsAsync will return a value to pass to the
            // method to retrieve the next 100 (or less) topics.
            string nextToken = string.Empty;

            do
            {
                var response = await client.ListTopicsAsync(nextToken);
                DisplayTopicsList(response.Topics);
                nextToken = response.NextToken;
            }
            while (!string.IsNullOrEmpty(nextToken));
        }

        /// <summary>
        /// Displays the list of Amazon SNS Topic ARNs.
        /// </summary>
        /// <param name="topicList">The list of Topic ARNs.</param>
        public static void DisplayTopicsList(List<Topic> topicList)
        {
            foreach (var topic in topicList)
            {
                Console.WriteLine($"{topic.TopicArn}");
            }
        }
    }
```

- Einzelheiten zur API finden Sie ListTopicsin der AWS SDK for .NET API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

```
//! Retrieve a list of Amazon Simple Notification Service (Amazon SNS) topics.
/*!
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool
AwsDoc::SNS::listTopics(const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::String nextToken; // Next token is used to handle a paginated response.
    bool result = true;
    do {
        Aws::SNS::Model::ListTopicsRequest request;

        if (!nextToken.empty()) {
            request.SetNextToken(nextToken);
        }

        const Aws::SNS::Model::ListTopicsOutcome outcome = snsClient.ListTopics(
                request);

        if (outcome.IsSuccess()) {
            std::cout << "Topics list:" << std::endl;
            for (auto const &topic: outcome.GetResult().GetTopics()) {
                std::cout << "  * " << topic.GetTopicArn() << std::endl;
            }
        }
        else {
            std::cerr << "Error listing topics " <<
 outcome.GetError().GetMessage() <<
                      std::endl;
            result = false;
            break;
        }

        nextToken = outcome.GetResult().GetNextToken();
    } while (!nextToken.empty());

    return result;
}
```

- Einzelheiten zur API finden Sie ListTopicsin der AWS SDK for C++ API-Referenz.

CLI

AWS CLI

So führen Sie Ihre SNS-Themen auf

Das folgende `list-topics` Beispiel listet alle SNS-Themen in Ihrem AWS Konto auf.

```
aws sns list-topics
```

Ausgabe:

```
{
    "Topics": [
        {
            "TopicArn": "arn:aws:sns:us-west-2:123456789012:my-topic"
        }
    ]
}
```

- Einzelheiten zur API finden Sie ListTopicsin der AWS CLI Befehlsreferenz.

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

```
package main

import (
 "context"
 "fmt"
```

```
 "log"

 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/service/sns"
 "github.com/aws/aws-sdk-go-v2/service/sns/types"
)

// main uses the AWS SDK for Go V2 to create an Amazon Simple Notification
 Service
// (Amazon SNS) client and list the topics in your account.
// This example uses the default settings specified in your shared credentials
// and config files.
func main() {
 sdkConfig, err := config.LoadDefaultConfig(context.TODO())
 if err != nil {
  fmt.Println("Couldn't load default configuration. Have you set up your AWS
 account?")
  fmt.Println(err)
  return
 }
 snsClient := sns.NewFromConfig(sdkConfig)
 fmt.Println("Let's list the topics for your account.")
 var topics []types.Topic
 paginator := sns.NewListTopicsPaginator(snsClient, &sns.ListTopicsInput{})
 for paginator.HasMorePages() {
  output, err := paginator.NextPage(context.TODO())
  if err != nil {
   log.Printf("Couldn't get topics. Here's why: %v\n", err)
   break
  } else {
   topics = append(topics, output.Topics...)
  }
 }
 if len(topics) == 0 {
  fmt.Println("You don't have any topics!")
 } else {
  for _, topic := range topics {
   fmt.Printf("\t%v\n", *topic.TopicArn)
  }
 }
}
```

- Einzelheiten zur API finden Sie ListTopicsin der AWS SDK for Go API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.ListTopicsRequest;
import software.amazon.awssdk.services.sns.model.ListTopicsResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class ListTopics {
    public static void main(String[] args) {
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        listSNSTopics(snsClient);
        snsClient.close();
    }

    public static void listSNSTopics(SnsClient snsClient) {
        try {
            ListTopicsRequest request = ListTopicsRequest.builder()
                    .build();
```

```
            ListTopicsResponse result = snsClient.listTopics(request);
            System.out.println(
                    "Status was " + result.sdkHttpResponse().statusCode() + "\n
\nTopics\n\n" + result.topics());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Einzelheiten zur API finden Sie ListTopicsin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie den Client in einem separaten Modul und exportieren Sie ihn.

```
import { SNSClient } from "@aws-sdk/client-sns";

// The AWS Region can be provided here using the `region` property. If you leave
 it blank
// the SDK will default to the region set in your AWS config.
export const snsClient = new SNSClient({});
```

Importieren Sie das SDK- und Client-Module und rufen Sie die API auf.

```
import { ListTopicsCommand } from "@aws-sdk/client-sns";
import { snsClient } from "../libs/snsClient.js";

export const listTopics = async () => {
```

```
  const response = await snsClient.send(new ListTopicsCommand({}));
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: '936bc5ad-83ca-53c2-b0b7-9891167b909e',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   Topics: [ { TopicArn: 'arn:aws:sns:us-east-1:xxxxxxxxxxxx:mytopic' } ]
  // }
  return response;
};
```

- Weitere Informationen finden Sie im AWS SDK for JavaScript -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie ListTopicsin der AWS SDK for JavaScript API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

```
suspend fun listSNSTopics() {

    SnsClient { region = "us-east-1" }.use { snsClient ->
        val response = snsClient.listTopics(ListTopicsRequest { })
        response.topics?.forEach { topic ->
            println("The topic ARN is ${topic.topicArn}")
        }
    }
}
```

- API-Details finden Sie ListTopicsin der API-Referenz zum AWS SDK für Kotlin.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Returns a list of the requester's topics from your AWS SNS account in the
 region specified.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/
guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

try {
    $result = $SnSclient->listTopics();
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- Einzelheiten zur API finden Sie ListTopicsin der AWS SDK for PHP API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

```
class SnsWrapper:
    """Encapsulates Amazon SNS topic and subscription functions."""

    def __init__(self, sns_resource):
        """
        :param sns_resource: A Boto3 Amazon SNS resource.
        """
        self.sns_resource = sns_resource


    def list_topics(self):
        """
        Lists topics for the current account.

        :return: An iterator that yields the topics.
        """
        try:
            topics_iter = self.sns_resource.topics.all()
            logger.info("Got topics.")
        except ClientError:
            logger.exception("Couldn't get topics.")
            raise
        else:
            return topics_iter
```

- Einzelheiten zur API finden Sie ListTopicsin AWS SDK for Python (Boto3) API Reference.

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu. GitHub Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

```
require "aws-sdk-sns"  # v2: require 'aws-sdk'

def list_topics?(sns_client)
  sns_client.topics.each do |topic|
    puts topic.arn
rescue StandardError => e
  puts "Error while listing the topics: #{e.message}"
  end
end

def run_me

  region = "REGION"
  sns_client = Aws::SNS::Resource.new(region: region)

  puts "Listing the topics."

  if list_topics?(sns_client)
  else
    puts "The bucket was not created. Stopping program."
    exit 1
  end
end

# Example usage:
run_me if $PROGRAM_NAME == __FILE__
```

- Weitere Informationen finden Sie im AWS SDK for Ruby -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie ListTopicsin der AWS SDK for Ruby API-Referenz.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

```
async fn show_topics(client: &Client) -> Result<(), Error> {
    let resp = client.list_topics().send().await?;

    println!("Topic ARNs:");

    for topic in resp.topics() {
        println!("{}", topic.topic_arn().unwrap_or_default());
    }

    Ok(())
}
```

- Einzelheiten zur API finden Sie ListTopicsin der API-Referenz zum AWS SDK für Rust.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

```
    TRY.
```

```
        oo_result = lo_sns->listtopics( ).            " oo_result is returned for
testing purposes. "
        DATA(lt_topics) = oo_result->get_topics( ).
        MESSAGE 'Retrieved list of topics.' TYPE 'I'.
      CATCH /aws1/cx_rt_generic.
        MESSAGE 'Unable to list topics.' TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie ListTopicsin der API-Referenz zum AWS SDK für SAP ABAP.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unterAmazon SNS mit einem AWS SDK verwenden. Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung **`Publish`** mit einem AWS SDK oder CLI

Die folgenden Codebeispiele zeigen, wie es verwendet wird`Publish`.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Erstellen und veröffentlichen zu einem FIFO-Thema
- Veröffentlichen einer SMS-Nachricht
- Veröffentlichen Sie Nachrichten in Warteschlangen

.NET

AWS SDK for .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

Veröffentlichen einer Nachricht für ein Thema.

```
    using System;
    using System.Threading.Tasks;
    using Amazon.SimpleNotificationService;
    using Amazon.SimpleNotificationService.Model;

    /// <summary>
    /// This example publishes a message to an Amazon Simple Notification
    /// Service (Amazon SNS) topic.
    /// </summary>
    public class PublishToSNSTopic
    {
        public static async Task Main()
        {
            string topicArn = "arn:aws:sns:us-
east-2:000000000000:ExampleSNSTopic";
            string messageText = "This is an example message to publish to the
 ExampleSNSTopic.";

            IAmazonSimpleNotificationService client = new
 AmazonSimpleNotificationServiceClient();

            await PublishToTopicAsync(client, topicArn, messageText);
        }

        /// <summary>
        /// Publishes a message to an Amazon SNS topic.
        /// </summary>
        /// <param name="client">The initialized client object used to publish
        /// to the Amazon SNS topic.</param>
        /// <param name="topicArn">The ARN of the topic.</param>
        /// <param name="messageText">The text of the message.</param>
        public static async Task PublishToTopicAsync(
            IAmazonSimpleNotificationService client,
            string topicArn,
            string messageText)
        {
            var request = new PublishRequest
            {
                TopicArn = topicArn,
                Message = messageText,
            };

            var response = await client.PublishAsync(request);
```

```
            Console.WriteLine($"Successfully published message ID:
{response.MessageId}");
        }
    }
```

Veröffentlichen Sie eine Nachricht zu einem Thema mit Gruppen-, Duplizierungs- und Attributoptionen.

```
    /// <summary>
    /// Publish messages using user settings.
    /// </summary>
    /// <returns>Async task.</returns>
    public static async Task PublishMessages()
    {
        Console.WriteLine("Now we can publish messages.");

        var keepSendingMessages = true;
        string? deduplicationId = null;
        string? toneAttribute = null;
        while (keepSendingMessages)
        {
            Console.WriteLine();
            var message = GetUserResponse("Enter a message to publish.", "This is
a sample message");

            if (_useFifoTopic)
            {
                Console.WriteLine("Because you are using a FIFO topic, you must
set a message group ID." +
                                  "\r\nAll messages within the same group will be
received in the order " +
                                  "they were published.");

                Console.WriteLine();
                var messageGroupId = GetUserResponse("Enter a message group ID
for this message:", "1");

                if (!_useContentBasedDeduplication)
                {
```

```
                    Console.WriteLine("Because you are not using content-based
deduplication, " +
                                      "you must enter a deduplication ID.");

                    Console.WriteLine("Enter a deduplication ID for this
message.");
                    deduplicationId = GetUserResponse("Enter a deduplication ID
for this message.", "1");
                }

                if (GetYesNoResponse("Add an attribute to this message?"))
                {
                    Console.WriteLine("Enter a number for an attribute.");
                    for (int i = 0; i < _tones.Length; i++)
                    {
                        Console.WriteLine($"\t{i + 1}. {_tones[i]}");
                    }

                    var selection = GetUserResponse("", "1");
                    int.TryParse(selection, out var selectionNumber);

                    if (selectionNumber > 0 && selectionNumber < _tones.Length)
                    {
                        toneAttribute = _tones[selectionNumber - 1];
                    }
                }

                var messageID = await SnsWrapper.PublishToTopicWithAttribute(
                    _topicArn, message, "tone", toneAttribute, deduplicationId,
messageGroupId);

                Console.WriteLine($"Message published with id {messageID}.");
            }

            keepSendingMessages = GetYesNoResponse("Send another message?",
false);
        }
    }
```

Wenden Sie die Benutzerauswahl auf die Veröffentlichungsaktion an.

```
    /// <summary>
```

```
    /// Publish a message to a topic with an attribute and optional deduplication
 and group IDs.
    /// </summary>
    /// <param name="topicArn">The ARN of the topic.</param>
    /// <param name="message">The message to publish.</param>
    /// <param name="attributeName">The optional attribute for the message.</
param>
    /// <param name="attributeValue">The optional attribute value for the
 message.</param>
    /// <param name="deduplicationId">The optional deduplication ID for the
 message.</param>
    /// <param name="groupId">The optional group ID for the message.</param>
    /// <returns>The ID of the message published.</returns>
    public async Task<string> PublishToTopicWithAttribute(
        string topicArn,
        string message,
        string? attributeName = null,
        string? attributeValue = null,
        string? deduplicationId = null,
        string? groupId = null)
    {
        var publishRequest = new PublishRequest()
        {
            TopicArn = topicArn,
            Message = message,
            MessageDeduplicationId = deduplicationId,
            MessageGroupId = groupId
        };

        if (attributeValue != null)
        {
            // Add the string attribute if it exists.
            publishRequest.MessageAttributes =
                new Dictionary<string, MessageAttributeValue>
                {
                    { attributeName!, new MessageAttributeValue() { StringValue =
 attributeValue, DataType = "String"} }
                };
        }

        var publishResponse = await
 _amazonSNSClient.PublishAsync(publishRequest);
        return publishResponse.MessageId;
    }
```

- Details zu API finden Sie unter Veröffentlichen in der AWS SDK for .NET -API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

```
//! Send a message to an Amazon Simple Notification Service (Amazon SNS) topic.
/*!
  \param message: The message to publish.
  \param topicARN: The Amazon Resource Name (ARN) for an Amazon SNS topic.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SNS::publishToTopic(const Aws::String &message,
                                 const Aws::String &topicARN,
                                 const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::SNS::Model::PublishRequest request;
    request.SetMessage(message);
    request.SetTopicArn(topicARN);

    const Aws::SNS::Model::PublishOutcome outcome = snsClient.Publish(request);

    if (outcome.IsSuccess()) {
        std::cout << "Message published successfully with id '"
                  << outcome.GetResult().GetMessageId() << "'." << std::endl;
    }
    else {
        std::cerr << "Error while publishing message "
                  << outcome.GetError().GetMessage()
                  << std::endl;
    }
```

```
    return outcome.IsSuccess();
}
```

Veröffentlichen Sie eine Nachricht mit einem Attribut.

```
        static const Aws::String TONE_ATTRIBUTE("tone");
        static const Aws::Vector<Aws::String> TONES = {"cheerful", "funny",
 "serious",
                                                       "sincere"};

        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::SNS::SNSClient snsClient(clientConfiguration);

        Aws::SNS::Model::PublishRequest request;
        request.SetTopicArn(topicARN);
        Aws::String message = askQuestion("Enter a message text to publish.  ");
        request.SetMessage(message);

        if (filteringMessages && askYesNoQuestion(
                "Add an attribute to this message? (y/n) ")) {
            for (size_t i = 0; i < TONES.size(); ++i) {
                std::cout << "  " << (i + 1) << ". " << TONES[i] << std::endl;
            }
            int selection = askQuestionForIntRange(
                    "Enter a number for an attribute. ",
                    1, static_cast<int>(TONES.size()));
            Aws::SNS::Model::MessageAttributeValue messageAttributeValue;
            messageAttributeValue.SetDataType("String");
            messageAttributeValue.SetStringValue(TONES[selection - 1]);
            request.AddMessageAttributes(TONE_ATTRIBUTE, messageAttributeValue);
        }

        Aws::SNS::Model::PublishOutcome outcome = snsClient.Publish(request);

        if (outcome.IsSuccess()) {
            std::cout << "Your message was successfully published." << std::endl;
        }
        else {
```

```
            std::cerr << "Error with TopicsAndQueues::Publish. "
                      << outcome.GetError().GetMessage()
                      << std::endl;

            cleanUp(topicARN,
                    queueURLS,
                    subscriptionARNS,
                    snsClient,
                    sqsClient);

            return false;
        }
```

- Details zu API finden Sie unter Veröffentlichen in der AWS SDK for C++ -API-Referenz.

CLI

AWS CLI

Beispiel 1: So veröffentlichen Sie eine Nachricht für ein Thema

Das folgende `publish`-Beispiel veröffentlicht die angegebene Nachricht im angegebenen SNS-Thema. Die Nachricht stammt aus einer Textdatei, in der Sie Zeilenumbrüche einfügen können.

```
aws sns publish \
    --topic-arn "arn:aws:sns:us-west-2:123456789012:my-topic" \
    --message file://message.txt
```

Inhalt von `message.txt`:

```
Hello World
Second Line
```

Ausgabe:

```
{
    "MessageId": "123a45b6-7890-12c3-45d6-111122223333"
}
```

Beispiel 2: So veröffentlichen Sie eine SMS-Nachricht an eine Telefonnummer

Im folgenden `publish`-Beispiel wird Nachricht `Hello world!` an Telefonnummer `+1-555-555-0100` veröffentlicht.

```
aws sns publish \
    --message "Hello world!" \
    --phone-number +1-555-555-0100
```

Ausgabe:

```
{
    "MessageId": "123a45b6-7890-12c3-45d6-333322221111"
}
```

- API-Details finden Sie unter Publish in der AWS CLI -Befehlsreferenz.

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

```
// SnsActions encapsulates the Amazon Simple Notification Service (Amazon SNS)
 actions
// used in the examples.
type SnsActions struct {
 SnsClient *sns.Client
}



// Publish publishes a message to an Amazon SNS topic. The message is then sent
 to all
// subscribers. When the topic is a FIFO topic, the message must also contain a
 group ID
```

```
// and, when ID-based deduplication is used, a deduplication ID. An optional key-
value
// filter attribute can be specified so that the message can be filtered
 according to
// a filter policy.
func (actor SnsActions) Publish(topicArn string, message string, groupId string,
 dedupId string, filterKey string, filterValue string) error {
 publishInput := sns.PublishInput{TopicArn: aws.String(topicArn), Message:
 aws.String(message)}
 if groupId != "" {
  publishInput.MessageGroupId = aws.String(groupId)
 }
 if dedupId != "" {
  publishInput.MessageDeduplicationId = aws.String(dedupId)
 }
 if filterKey != "" && filterValue != "" {
  publishInput.MessageAttributes = map[string]types.MessageAttributeValue{
   filterKey: {DataType: aws.String("String"), StringValue:
 aws.String(filterValue)},
  }
 }
 _, err := actor.SnsClient.Publish(context.TODO(), &publishInput)
 if err != nil {
  log.Printf("Couldn't publish message to topic %v. Here's why: %v", topicArn,
 err)
 }
 return err
}
```

- Details zu API finden Sie unter Veröffentlichen in der AWS SDK for Go -API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.PublishRequest;
import software.amazon.awssdk.services.sns.model.PublishResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class PublishTopic {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <message> <topicArn>

                Where:
                   message - The message text to send.
                   topicArn - The ARN of the topic to publish.
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String message = args[0];
        String topicArn = args[1];
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();
        pubTopic(snsClient, message, topicArn);
        snsClient.close();
    }

    public static void pubTopic(SnsClient snsClient, String message, String
 topicArn) {
        try {
```

```
            PublishRequest request = PublishRequest.builder()
                    .message(message)
                    .topicArn(topicArn)
                    .build();

            PublishResponse result = snsClient.publish(request);
            System.out
                    .println(result.messageId() + " Message sent. Status is " +
 result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Details zu API finden Sie unter Veröffentlichen in der AWS SDK for Java 2.x -API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie den Client in einem separaten Modul und exportieren Sie ihn.

```
import { SNSClient } from "@aws-sdk/client-sns";

// The AWS Region can be provided here using the `region` property. If you leave
 it blank
// the SDK will default to the region set in your AWS config.
export const snsClient = new SNSClient({});
```

Importieren Sie das SDK- und Client-Module und rufen Sie die API auf.

```
import { PublishCommand } from "@aws-sdk/client-sns";
import { snsClient } from "../libs/snsClient.js";

/**
 * @param {string | Record<string, any>} message - The message to send. Can be a
 plain string or an object
 *                                                  if you are using the `json`
 `MessageStructure`.
 * @param {string} topicArn - The ARN of the topic to which you would like to
 publish.
 */
export const publish = async (
  message = "Hello from SNS!",
  topicArn = "TOPIC_ARN",
) => {
  const response = await snsClient.send(
    new PublishCommand({
      Message: message,
      TopicArn: topicArn,
    }),
  );
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: 'e7f77526-e295-5325-9ee4-281a43ad1f05',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   MessageId: 'xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx'
  // }
  return response;
};
```

Veröffentlichen Sie eine Nachricht zu einem Thema mit Gruppen-, Duplizierungs- und Attributoptionen.

```
  async publishMessages() {
    const message = await this.prompter.input({
      message: MESSAGES.publishMessagePrompt,
```

```
    });

    let groupId, deduplicationId, choices;

    if (this.isFifo) {
      await this.logger.log(MESSAGES.groupIdNotice);
      groupId = await this.prompter.input({
        message: MESSAGES.groupIdPrompt,
      });

      if (this.autoDedup === false) {
        await this.logger.log(MESSAGES.deduplicationIdNotice);
        deduplicationId = await this.prompter.input({
          message: MESSAGES.deduplicationIdPrompt,
        });
      }

      choices = await this.prompter.checkbox({
        message: MESSAGES.messageAttributesPrompt,
        choices: toneChoices,
      });
    }

    await this.snsClient.send(
      new PublishCommand({
        TopicArn: this.topicArn,
        Message: message,
        ...(groupId
          ? {
              MessageGroupId: groupId,
            }
          : {}),
        ...(deduplicationId
          ? {
              MessageDeduplicationId: deduplicationId,
            }
          : {}),
        ...(choices
          ? {
              MessageAttributes: {
                tone: {
                  DataType: "String.Array",
                  StringValue: JSON.stringify(choices),
                },
```

```
              },
            }
          : {}),
      }),
    );

    const publishAnother = await this.prompter.confirm({
      message: MESSAGES.publishAnother,
    });

    if (publishAnother) {
      await this.publishMessages();
    }
  }
```

- Weitere Informationen finden Sie im AWS SDK for JavaScript -Entwicklerhandbuch.
- Details zu API finden Sie unter Veröffentlichen in der AWS SDK for JavaScript -API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

```
suspend fun pubTopic(topicArnVal: String, messageVal: String) {

    val request = PublishRequest {
        message = messageVal
        topicArn = topicArnVal
    }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.publish(request)
        println("${result.messageId} message sent.")
    }
```

```
}
```

- Details zu API finden Sie unter Publish (Veröffentlichen) in der AWS -SDK-für-Kotlin-API-Referenz.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Sends a message to an Amazon SNS topic.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/
guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$message = 'This message is sent from a Amazon SNS code sample.';
$topic = 'arn:aws:sns:us-east-1:111122223333:MyTopic';

try {
    $result = $SnSclient->publish([
```

```
        'Message' => $message,
        'TopicArn' => $topic,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- Weitere Informationen finden Sie im AWS SDK for PHP -Entwicklerhandbuch.
- Details zu API finden Sie unter Veröffentlichen in der AWS SDK for PHP -API-Referenz.

PowerShell

Tools für PowerShell

Beispiel 1: Dieses Beispiel zeigt das Veröffentlichen einer Nachricht mit einer einzigen MessageAttribute deklarierten Inline.

```
Publish-SNSMessage -TopicArn "arn:aws:sns:us-west-2:123456789012:my-topic" -
Message "Hello" -MessageAttribute
 @{'City'=[Amazon.SimpleNotificationService.Model.MessageAttributeValue]@{DataType='Strin
 StringValue ='AnyCity'}}
```

Beispiel 2: Dieses Beispiel zeigt die Veröffentlichung einer Nachricht, bei der mehrere Nachrichten im Voraus MessageAttributes deklariert wurden.

```
$cityAttributeValue = New-Object
 Amazon.SimpleNotificationService.Model.MessageAttributeValue
$cityAttributeValue.DataType = "String"
$cityAttributeValue.StringValue = "AnyCity"

$populationAttributeValue = New-Object
 Amazon.SimpleNotificationService.Model.MessageAttributeValue
$populationAttributeValue.DataType = "Number"
$populationAttributeValue.StringValue = "1250800"

$messageAttributes = New-Object System.Collections.Hashtable
$messageAttributes.Add("City", $cityAttributeValue)
```

```
$messageAttributes.Add("Population", $populationAttributeValue)

Publish-SNSMessage -TopicArn "arn:aws:sns:us-west-2:123456789012:my-topic" -
Message "Hello" -MessageAttribute $messageAttributes
```

- Einzelheiten zur API finden Sie unter In AWS Tools for PowerShell Cmdlet-Referenz veröffentlichen.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

Veröffentlichen Sie eine Nachricht mit Attributen, damit ein Abonnement basierend auf Attributen filtern kann.

```
class SnsWrapper:
    """Encapsulates Amazon SNS topic and subscription functions."""

    def __init__(self, sns_resource):
        """
        :param sns_resource: A Boto3 Amazon SNS resource.
        """
        self.sns_resource = sns_resource


    @staticmethod
    def publish_message(topic, message, attributes):
        """
        Publishes a message, with attributes, to a topic. Subscriptions can be
 filtered
        based on message attributes so that a subscription receives messages only
        when specified attributes are present.

        :param topic: The topic to publish to.
        :param message: The message to publish.
```

```
        :param attributes: The key-value attributes to attach to the message.
 Values
                           must be either `str` or `bytes`.
        :return: The ID of the message.
        """
        try:
            att_dict = {}
            for key, value in attributes.items():
                if isinstance(value, str):
                    att_dict[key] = {"DataType": "String", "StringValue": value}
                elif isinstance(value, bytes):
                    att_dict[key] = {"DataType": "Binary", "BinaryValue": value}
            response = topic.publish(Message=message, MessageAttributes=att_dict)
            message_id = response["MessageId"]
            logger.info(
                "Published message with attributes %s to topic %s.",
                attributes,
                topic.arn,
            )
        except ClientError:
            logger.exception("Couldn't publish message to topic %s.", topic.arn)
            raise
        else:
            return message_id
```

Veröffentlichen Sie eine Nachricht, die basierend auf dem Protokoll des Abonnenten unterschiedliche Formen annimmt.

```
class SnsWrapper:
    """Encapsulates Amazon SNS topic and subscription functions."""

    def __init__(self, sns_resource):
        """
        :param sns_resource: A Boto3 Amazon SNS resource.
        """
        self.sns_resource = sns_resource


    @staticmethod
    def publish_multi_message(
        topic, subject, default_message, sms_message, email_message
```

```
    ):
        """
        Publishes a multi-format message to a topic. A multi-format message takes
        different forms based on the protocol of the subscriber. For example,
        an SMS subscriber might receive a short version of the message
        while an email subscriber could receive a longer version.

        :param topic: The topic to publish to.
        :param subject: The subject of the message.
        :param default_message: The default version of the message. This version
is
                                sent to subscribers that have protocols that are
not
                                otherwise specified in the structured message.
        :param sms_message: The version of the message sent to SMS subscribers.
        :param email_message: The version of the message sent to email
subscribers.
        :return: The ID of the message.
        """
        try:
            message = {
                "default": default_message,
                "sms": sms_message,
                "email": email_message,
            }
            response = topic.publish(
                Message=json.dumps(message), Subject=subject,
MessageStructure="json"
            )
            message_id = response["MessageId"]
            logger.info("Published multi-format message to topic %s.", topic.arn)
        except ClientError:
            logger.exception("Couldn't publish message to topic %s.", topic.arn)
            raise
        else:
            return message_id
```

- Details zu API finden Sie unter Veröffentlichen in der AWS -API-Referenz zu SDK for Python (Boto3).

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

```
# Service class for sending messages using Amazon Simple Notification Service
 (SNS)
class SnsMessageSender
  # Initializes the SnsMessageSender with an SNS client
  #
  # @param sns_client [Aws::SNS::Client] The SNS client
  def initialize(sns_client)
    @sns_client = sns_client
    @logger = Logger.new($stdout)
  end

  # Sends a message to a specified SNS topic
  #
  # @param topic_arn [String] The ARN of the SNS topic
  # @param message [String] The message to send
  # @return [Boolean] true if message was successfully sent, false otherwise
  def send_message(topic_arn, message)
    @sns_client.publish(topic_arn: topic_arn, message: message)
    @logger.info("Message sent successfully to #{topic_arn}.")
    true
  rescue Aws::SNS::Errors::ServiceError => e
    @logger.error("Error while sending the message: #{e.message}")
    false
  end
end

# Example usage:
if $PROGRAM_NAME == __FILE__
  topic_arn = "SNS_TOPIC_ARN" # Should be replaced with a real topic ARN
  message = "MESSAGE"         # Should be replaced with the actual message
 content
```

```
  sns_client = Aws::SNS::Client.new
  message_sender = SnsMessageSender.new(sns_client)

  @logger.info("Sending message.")
  unless message_sender.send_message(topic_arn, message)
    @logger.error("Message sending failed. Stopping program.")
    exit 1
  end
end
```

- Weitere Informationen finden Sie im AWS SDK for Ruby -Entwicklerhandbuch.
- Details zu API finden Sie unter Veröffentlichen in der AWS SDK for Ruby -API-Referenz.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

```
async fn subscribe_and_publish(
    client: &Client,
    topic_arn: &str,
    email_address: &str,
) -> Result<(), Error> {
    println!("Receiving on topic with ARN: `{}`", topic_arn);

    let rsp = client
        .subscribe()
        .topic_arn(topic_arn)
        .protocol("email")
        .endpoint(email_address)
        .send()
        .await?;

    println!("Added a subscription: {:?}", rsp);
```

```
    let rsp = client
        .publish()
        .topic_arn(topic_arn)
        .message("hello sns!")
        .send()
        .await?;

    println!("Published message: {:?}", rsp);

    Ok(())
}
```

- Details zu API finden Sie unter Veröffentlichen in der Referenz für das AWS -SDK für Rust-API.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

```
    TRY.
        oo_result = lo_sns->publish(              " oo_result is returned for
testing purposes. "
          iv_topicarn = iv_topic_arn
          iv_message = iv_message
        ).
        MESSAGE 'Message published to SNS topic.' TYPE 'I'.
      CATCH /aws1/cx_snsnotfoundexception.
        MESSAGE 'Topic does not exist.' TYPE 'E'.
    ENDTRY.
```

- Details zu API finden Sie unter Publish (Veröffentlichen) in der API-Referenz für das AWS SDK für SAP ABAP.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unterAmazon SNS mit einem AWS SDK verwenden. Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung **`SetSMSAttributes`** mit einem AWS SDK oder CLI

Die folgenden Codebeispiele zeigen, wie es verwendet wird`SetSMSAttributes`.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

So verwenden Sie Amazon SNS zum Festlegen des DefaultSMSType-Attributs.

```
//! Set the default settings for sending SMS messages.
/*!
  \param smsType: The type of SMS message that you will send by default.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SNS::setSMSType(const Aws::String &smsType,
                             const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::SNS::Model::SetSMSAttributesRequest request;
    request.AddAttributes("DefaultSMSType", smsType);

    const Aws::SNS::Model::SetSMSAttributesOutcome outcome =
 snsClient.SetSMSAttributes(
            request);

    if (outcome.IsSuccess()) {
        std::cout << "SMS Type set successfully " << std::endl;
    }
    else {
```

```
        std::cerr << "Error while setting SMS Type: '"
                  << outcome.GetError().GetMessage()
                  << "'" << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Details zu API finden Sie unter SetSMSAttributes in der AWS SDK for C++ -API-Referenz.

CLI

AWS CLI

So legen Sie SMS-Nachrichtenattribute fest

Im folgenden `set-sms-attributes`-Beispiel wird die standardmäßige Absender-ID für SMS-Nachrichten auf `MyName` festgelegt.

```
aws sns set-sms-attributes \
    --attributes DefaultSenderID=MyName
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

- API-Details finden Sie unter SetSMSAttributes in der AWS CLI -Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SetSmsAttributesRequest;
import software.amazon.awssdk.services.sns.model.SetSmsAttributesResponse;
```

```
import software.amazon.awssdk.services.sns.model.SnsException;
import java.util.HashMap;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class SetSMSAttributes {
    public static void main(String[] args) {
        HashMap<String, String> attributes = new HashMap<>(1);
        attributes.put("DefaultSMSType", "Transactional");
        attributes.put("UsageReportS3Bucket", "janbucket");

        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();
        setSNSAttributes(snsClient, attributes);
        snsClient.close();
    }

    public static void setSNSAttributes(SnsClient snsClient, HashMap<String,
 String> attributes) {
        try {
            SetSmsAttributesRequest request = SetSmsAttributesRequest.builder()
                    .attributes(attributes)
                    .build();

            SetSmsAttributesResponse result =
 snsClient.setSMSAttributes(request);
            System.out.println("Set default Attributes to " + attributes + ".
 Status was "
                    + result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Details zu API finden Sie unter SetSMSAttributes in der AWS SDK for Java 2.x -API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie den Client in einem separaten Modul und exportieren Sie ihn.

```
import { SNSClient } from "@aws-sdk/client-sns";

// The AWS Region can be provided here using the `region` property. If you leave
 it blank
// the SDK will default to the region set in your AWS config.
export const snsClient = new SNSClient({});
```

Importieren Sie das SDK- und Client-Module und rufen Sie die API auf.

```
import { SetSMSAttributesCommand } from "@aws-sdk/client-sns";
import { snsClient } from "../libs/snsClient.js";

/**
 * @param {"Transactional" | "Promotional"} defaultSmsType
 */
export const setSmsType = async (defaultSmsType = "Transactional") => {
  const response = await snsClient.send(
    new SetSMSAttributesCommand({
      attributes: {
        // Promotional – (Default) Noncritical messages, such as marketing
 messages.
        // Transactional – Critical messages that support customer transactions,
        // such as one-time passcodes for multi-factor authentication.
```

```
        DefaultSMSType: defaultSmsType,
      },
    }),
  );
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: '1885b977-2d7e-535e-8214-e44be727e265',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   }
  // }
  return response;
};
```

- Weitere Informationen finden Sie im AWS SDK for JavaScript -Entwicklerhandbuch.
- Details zu API finden Sie unter SetSMSAttributes in der AWS SDK for JavaScript -API-Referenz.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

```
$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

try {
    $result = $SnSclient->SetSMSAttributes([
```

```
        'attributes' => [
            'DefaultSMSType' => 'Transactional',
        ],
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- Weitere Informationen finden Sie im AWS SDK for PHP -Entwicklerhandbuch.
- Details zu API finden Sie unter SetSMSAttributes in der AWS SDK for PHP -API-Referenz.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unterAmazon SNS mit einem AWS SDK verwenden. Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung **`SetSubscriptionAttributes`** mit einem AWS SDK oder CLI

Die folgenden Codebeispiele zeigen, wie es verwendet wird`SetSubscriptionAttributes`.

CLI

AWS CLI

So legen Sie Abonnementattribute fest

Im folgenden `set-subscription-attributes`-Beispiel wird das `RawMessageDelivery`-Attribut auf ein SQS-Abonnement festgelegt.

```
aws sns set-subscription-attributes \
    --subscription-arn arn:aws:sns:us-
east-1:123456789012:mytopic:f248de18-2cf6-578c-8592-b6f1eaa877dc \
    --attribute-name RawMessageDelivery \
    --attribute-value true
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Im folgenden `set-subscription-attributes`-Beispiel wird ein `FilterPolicy`-Attribut auf ein SQS-Abonnement festgelegt.

```
aws sns set-subscription-attributes \
    --subscription-arn arn:aws:sns:us-
east-1:123456789012:mytopic:f248de18-2cf6-578c-8592-b6f1eaa877dc \
    --attribute-name FilterPolicy \
    --attribute-value "{ \"anyMandatoryKey\": [\"any\", \"of\", \"these\"] }"
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Im folgenden `set-subscription-attributes`-Beispiel wird das `FilterPolicy`-Attribut von einem SQS-Abonnement entfernt.

```
aws sns set-subscription-attributes \
    --subscription-arn arn:aws:sns:us-
east-1:123456789012:mytopic:f248de18-2cf6-578c-8592-b6f1eaa877dc \
    --attribute-name FilterPolicy \
    --attribute-value "{}"
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

- Einzelheiten zur API finden Sie SetSubscriptionAttributesin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SnsException;
import java.util.ArrayList;

/**
 * Before running this Java V2 code example, set up your development
```

```
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class UseMessageFilterPolicy {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <subscriptionArn>

                Where:
                   subscriptionArn - The ARN of a subscription.

                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String subscriptionArn = args[0];
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        usePolicy(snsClient, subscriptionArn);
        snsClient.close();
    }

    public static void usePolicy(SnsClient snsClient, String subscriptionArn) {
        try {
            SNSMessageFilterPolicy fp = new SNSMessageFilterPolicy();
            // Add a filter policy attribute with a single value
            fp.addAttribute("store", "example_corp");
            fp.addAttribute("event", "order_placed");

            // Add a prefix attribute
            fp.addAttributePrefix("customer_interests", "bas");

            // Add an anything-but attribute
            fp.addAttributeAnythingBut("customer_interests", "baseball");
```

```
            // Add a filter policy attribute with a list of values
            ArrayList<String> attributeValues = new ArrayList<>();
            attributeValues.add("rugby");
            attributeValues.add("soccer");
            attributeValues.add("hockey");
            fp.addAttribute("customer_interests", attributeValues);

            // Add a numeric attribute
            fp.addAttribute("price_usd", "=", 0);

            // Add a numeric attribute with a range
            fp.addAttributeRange("price_usd", ">", 0, "<=", 100);

            // Apply the filter policy attributes to an Amazon SNS subscription
            fp.apply(snsClient, subscriptionArn);

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Einzelheiten zur API finden Sie SetSubscriptionAttributesin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

```
class SnsWrapper:
    """Encapsulates Amazon SNS topic and subscription functions."""
```

```
    def __init__(self, sns_resource):
        """
        :param sns_resource: A Boto3 Amazon SNS resource.
        """
        self.sns_resource = sns_resource


    @staticmethod
    def add_subscription_filter(subscription, attributes):
        """
        Adds a filter policy to a subscription. A filter policy is a key and a
        list of values that are allowed. When a message is published, it must
have an
        attribute that passes the filter or it will not be sent to the
subscription.

        :param subscription: The subscription the filter policy is attached to.
        :param attributes: A dictionary of key-value pairs that define the
filter.
        """
        try:
            att_policy = {key: [value] for key, value in attributes.items()}
            subscription.set_attributes(
                AttributeName="FilterPolicy",
AttributeValue=json.dumps(att_policy)
            )
            logger.info("Added filter to subscription %s.", subscription.arn)
        except ClientError:
            logger.exception(
                "Couldn't add filter to subscription %s.", subscription.arn
            )
            raise
```

- Einzelheiten zur API finden Sie SetSubscriptionAttributesin AWS SDK for Python (Boto3) API Reference.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter. Amazon SNS mit einem AWS SDK verwenden Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung **SetSubscriptionAttributesRedrivePolicy** mit einem AWS SDK oder CLI

Das folgende Codebeispiel zeigt, wie es verwendet wird`SetSubscriptionAttributesRedrivePolicy`.

Java

SDK für Java 1.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

```
// Specify the ARN of the Amazon SNS subscription.
String subscriptionArn =
    "arn:aws:sns:us-east-2:123456789012:MyEndpoint:1234a567-
bc89-012d-3e45-6fg7h890123i";

// Specify the ARN of the Amazon SQS queue to use as a dead-letter queue.
String redrivePolicy =
    "{\"deadLetterTargetArn\":\"arn:aws:sqs:us-
east-2:123456789012:MyDeadLetterQueue\"}";

// Set the specified Amazon SQS queue as a dead-letter queue
// of the specified Amazon SNS subscription by setting the RedrivePolicy
 attribute.
SetSubscriptionAttributesRequest request = new SetSubscriptionAttributesRequest()
    .withSubscriptionArn(subscriptionArn)
    .withAttributeName("RedrivePolicy")
    .withAttributeValue(redrivePolicy);
sns.setSubscriptionAttributes(request);
```

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unterAmazon SNS mit einem AWS SDK verwenden. Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung **SetTopicAttributes** mit einem AWS SDK oder CLI

Die folgenden Codebeispiele zeigen, wie es verwendet wird`SetTopicAttributes`.

CLI

AWS CLI

So legen Sie ein Attribut für ein Thema fest

Im folgenden `set-topic-attributes`-Beispiel wird das `DisplayName`-Attribute für das angegebene Thema festgelegt.

```
aws sns set-topic-attributes \
    --topic-arn arn:aws:sns:us-west-2:123456789012:MyTopic \
    --attribute-name DisplayName \
    --attribute-value MyTopicDisplayName
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

- Einzelheiten zur API finden Sie SetTopicAttributesin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SetTopicAttributesRequest;
import software.amazon.awssdk.services.sns.model.SetTopicAttributesResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
```

```
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class SetTopicAttributes {

    public static void main(String[] args) {
        final String usage = """

                Usage:    <attribute> <topicArn> <value>

                Where:
                   attribute - The attribute action to use. Valid parameters are:
 Policy | DisplayName | DeliveryPolicy .
                   topicArn - The ARN of the topic.\s
                   value - The value for the attribute.
                """;

        if (args.length < 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String attribute = args[0];
        String topicArn = args[1];
        String value = args[2];

        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        setTopAttr(snsClient, attribute, topicArn, value);
        snsClient.close();
    }

    public static void setTopAttr(SnsClient snsClient, String attribute, String
 topicArn, String value) {
        try {
            SetTopicAttributesRequest request =
 SetTopicAttributesRequest.builder()
                    .attributeName(attribute)
                    .attributeValue(value)
                    .topicArn(topicArn)
```

```
                    .build();

            SetTopicAttributesResponse result =
 snsClient.setTopicAttributes(request);
            System.out.println(
                    "\n\nStatus was " + result.sdkHttpResponse().statusCode() +
 "\n\nTopic " + request.topicArn()
                            + " updated " + request.attributeName() + " to " +
 request.attributeValue());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Einzelheiten zur API finden Sie SetTopicAttributesin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie den Client in einem separaten Modul und exportieren Sie ihn.

```
import { SNSClient } from "@aws-sdk/client-sns";

// The AWS Region can be provided here using the `region` property. If you leave
 it blank
// the SDK will default to the region set in your AWS config.
export const snsClient = new SNSClient({});
```

Importieren Sie das SDK- und Client-Module und rufen Sie die API auf.

```
import { SetTopicAttributesCommand } from "@aws-sdk/client-sns";
import { snsClient } from "../libs/snsClient.js";

export const setTopicAttributes = async (
  topicArn = "TOPIC_ARN",
  attributeName = "DisplayName",
  attributeValue = "Test Topic",
) => {
  const response = await snsClient.send(
    new SetTopicAttributesCommand({
      AttributeName: attributeName,
      AttributeValue: attributeValue,
      TopicArn: topicArn,
    }),
  );
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: 'd1b08d0e-e9a4-54c3-b8b1-d03238d2b935',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   }
  // }
  return response;
};
```

- Weitere Informationen finden Sie im AWS SDK for JavaScript -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie SetTopicAttributesin der AWS SDK for JavaScript API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

```
suspend fun setTopAttr(attribute: String?, topicArnVal: String?, value: String?)
 {

    val request = SetTopicAttributesRequest {
        attributeName = attribute
        attributeValue = value
        topicArn = topicArnVal
    }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        snsClient.setTopicAttributes(request)
        println("Topic ${request.topicArn} was updated.")
    }
}
```

- API-Details finden Sie SetTopicAttributesin der API-Referenz zum AWS SDK für Kotlin.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

```
require 'vendor/autoload.php';
```

```
use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Configure the message delivery status attributes for an Amazon SNS Topic.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/
guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);
$attribute = 'Policy | DisplayName | DeliveryPolicy';
$value = 'First Topic';
$topic = 'arn:aws:sns:us-east-1:111122223333:MyTopic';

try {
    $result = $SnSclient->setTopicAttributes([
        'AttributeName' => $attribute,
        'AttributeValue' => $value,
        'TopicArn' => $topic,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- Einzelheiten zur API finden Sie SetTopicAttributesin der AWS SDK for PHP API-Referenz.

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

```
# Service class to enable an SNS resource with a specified policy
class SnsResourceEnabler
  # Initializes the SnsResourceEnabler with an SNS resource client
  #
  # @param sns_resource [Aws::SNS::Resource] The SNS resource client
  def initialize(sns_resource)
    @sns_resource = sns_resource
    @logger = Logger.new($stdout)
  end

  # Sets a policy on a specified SNS topic
  #
  # @param topic_arn [String] The ARN of the SNS topic
  # @param resource_arn [String] The ARN of the resource to include in the policy
  # @param policy_name [String] The name of the policy attribute to set
  def enable_resource(topic_arn, resource_arn, policy_name)
    policy = generate_policy(topic_arn, resource_arn)
    topic = @sns_resource.topic(topic_arn)

    topic.set_attributes({
                           attribute_name: policy_name,
                           attribute_value: policy
                         })
    @logger.info("Policy #{policy_name} set successfully for topic
 #{topic_arn}.")
  rescue Aws::SNS::Errors::ServiceError => e
    @logger.error("Failed to set policy: #{e.message}")
  end

  private

  # Generates a policy string with dynamic resource ARNs
```

```
  #
  # @param topic_arn [String] The ARN of the SNS topic
  # @param resource_arn [String] The ARN of the resource
  # @return [String] The policy as a JSON string
  def generate_policy(topic_arn, resource_arn)
    {
      Version: "2008-10-17",
      Id: "__default_policy_ID",
      Statement: [{
                   Sid: "__default_statement_ID",
                   Effect: "Allow",
                   Principal: { "AWS": "*" },
                   Action: ["SNS:Publish"],
                   Resource: topic_arn,
                   Condition: {
                     ArnEquals: {
                       "AWS:SourceArn": resource_arn
                     }
                   }
                 }]
    }.to_json
  end
end

# Example usage:
if $PROGRAM_NAME == __FILE__
  topic_arn = "MY_TOPIC_ARN"     # Should be replaced with a real topic ARN
  resource_arn = "MY_RESOURCE_ARN" # Should be replaced with a real resource ARN
  policy_name = "POLICY_NAME"    # Typically, this is "Policy"

  sns_resource = Aws::SNS::Resource.new
  enabler = SnsResourceEnabler.new(sns_resource)

  enabler.enable_resource(topic_arn, resource_arn, policy_name)
end
```

- Weitere Informationen finden Sie im AWS SDK for Ruby -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie SetTopicAttributesin der AWS SDK for Ruby API-Referenz.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

```
TRY.
    lo_sns->settopicattributes(
        iv_topicarn = iv_topic_arn
        iv_attributename  = iv_attribute_name
        iv_attributevalue = iv_attribute_value
    ).
    MESSAGE 'Set/updated SNS topic attributes.' TYPE 'I'.
  CATCH /aws1/cx_snsnotfoundexception.
    MESSAGE 'Topic does not exist.' TYPE 'E'.
ENDTRY.
```

- Einzelheiten zur API finden Sie SetTopicAttributesin der API-Referenz zum AWS SDK für SAP ABAP.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unterAmazon SNS mit einem AWS SDK verwenden. Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung **`Subscribe`** mit einem AWS SDK oder CLI

Die folgenden Codebeispiele zeigen, wie es verwendet wird`Subscribe`.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Erstellen und veröffentlichen zu einem FIFO-Thema
- Veröffentlichen Sie Nachrichten in Warteschlangen

.NET

AWS SDK for .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

Abonnieren Sie eine E-Mail-Adresse für ein Thema.

```
/// <summary>
/// Creates a new subscription to a topic.
/// </summary>
/// <param name="client">The initialized Amazon SNS client object, used
/// to create an Amazon SNS subscription.</param>
/// <param name="topicArn">The ARN of the topic to subscribe to.</param>
/// <returns>A SubscribeResponse object which includes the subscription
/// ARN for the new subscription.</returns>
public static async Task<SubscribeResponse> TopicSubscribeAsync(
    IAmazonSimpleNotificationService client,
    string topicArn)
{
    SubscribeRequest request = new SubscribeRequest()
    {
        TopicArn = topicArn,
        ReturnSubscriptionArn = true,
        Protocol = "email",
        Endpoint = "recipient@example.com",
    };

    var response = await client.SubscribeAsync(request);

    return response;
}
```

Abonnieren Sie eine Warteschlange für ein Thema mit optionalen Filtern.

```
    /// <summary>
    /// Subscribe a queue to a topic with optional filters.
    /// </summary>
    /// <param name="topicArn">The ARN of the topic.</param>
    /// <param name="useFifoTopic">The optional filtering policy for the
subscription.</param>
    /// <param name="queueArn">The ARN of the queue.</param>
    /// <returns>The ARN of the new subscription.</returns>
    public async Task<string> SubscribeTopicWithFilter(string topicArn, string?
filterPolicy, string queueArn)
    {
        var subscribeRequest = new SubscribeRequest()
        {
            TopicArn = topicArn,
            Protocol = "sqs",
            Endpoint = queueArn
        };

        if (!string.IsNullOrEmpty(filterPolicy))
        {
            subscribeRequest.Attributes = new Dictionary<string, string>
{ { "FilterPolicy", filterPolicy } };
        }

        var subscribeResponse = await
_amazonSNSClient.SubscribeAsync(subscribeRequest);
        return subscribeResponse.SubscriptionArn;
    }
```

- Details zu API finden Sie unter Abonnieren in der AWS SDK for .NET -API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

Abonnieren Sie eine E-Mail-Adresse für ein Thema.

```
//! Subscribe to an Amazon Simple Notification Service (Amazon SNS) topic with
 delivery to an email address.
/*!
  \param topicARN: An SNS topic Amazon Resource Name (ARN).
  \param emailAddress: An email address.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SNS::subscribeEmail(const Aws::String &topicARN,
                                 const Aws::String &emailAddress,
                                 const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::SNS::Model::SubscribeRequest request;
    request.SetTopicArn(topicARN);
    request.SetProtocol("email");
    request.SetEndpoint(emailAddress);

    const Aws::SNS::Model::SubscribeOutcome outcome =
 snsClient.Subscribe(request);

    if (outcome.IsSuccess()) {
        std::cout << "Subscribed successfully." << std::endl;
        std::cout << "Subscription ARN '" <<
 outcome.GetResult().GetSubscriptionArn()
                  << "'." << std::endl;
    }
    else {
        std::cerr << "Error while subscribing " <<
 outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

Abonnieren Sie ein Thema mit einer mobilen Anwendung.

```
//! Subscribe to an Amazon Simple Notification Service (Amazon SNS) topic with
 delivery to a mobile app.
/*!
  \param topicARN: The Amazon Resource Name (ARN) for an Amazon SNS topic.
  \param endpointARN: The ARN for a mobile app or device endpoint.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool
AwsDoc::SNS::subscribeApp(const Aws::String &topicARN,
                          const Aws::String &endpointARN,
                          const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::SNS::Model::SubscribeRequest request;
    request.SetTopicArn(topicARN);
    request.SetProtocol("application");
    request.SetEndpoint(endpointARN);

    const Aws::SNS::Model::SubscribeOutcome outcome =
 snsClient.Subscribe(request);

    if (outcome.IsSuccess()) {
        std::cout << "Subscribed successfully." << std::endl;
        std::cout << "Subscription ARN '" <<
 outcome.GetResult().GetSubscriptionArn()
                  << "'." << std::endl;
    }
    else {
        std::cerr << "Error while subscribing " <<
 outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

Abonnieren Sie eine Lambda-Funktion für ein Thema.

```
//! Subscribe to an Amazon Simple Notification Service (Amazon SNS) topic with
 delivery to an AWS Lambda function.
/*!
  \param topicARN: The Amazon Resource Name (ARN) for an Amazon SNS topic.
  \param lambdaFunctionARN: The ARN for an AWS Lambda function.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SNS::subscribeLambda(const Aws::String &topicARN,
                                  const Aws::String &lambdaFunctionARN,
                                  const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::SNS::Model::SubscribeRequest request;
    request.SetTopicArn(topicARN);
    request.SetProtocol("lambda");
    request.SetEndpoint(lambdaFunctionARN);

    const Aws::SNS::Model::SubscribeOutcome outcome =
 snsClient.Subscribe(request);

    if (outcome.IsSuccess()) {
        std::cout << "Subscribed successfully." << std::endl;
        std::cout << "Subscription ARN '" <<
 outcome.GetResult().GetSubscriptionArn()
                  << "'." << std::endl;
    }
    else {
        std::cerr << "Error while subscribing " <<
 outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

Abonnieren Sie eine SQS-Warteschlange für ein Thema.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";
```

```
   Aws::SNS::SNSClient snsClient(clientConfiguration);

           Aws::SNS::Model::SubscribeRequest request;
           request.SetTopicArn(topicARN);
           request.SetProtocol("sqs");
           request.SetEndpoint(queueARN);

           Aws::SNS::Model::SubscribeOutcome outcome =
snsClient.Subscribe(request);

           if (outcome.IsSuccess()) {
               Aws::String subscriptionARN =
outcome.GetResult().GetSubscriptionArn();
               std::cout << "The queue '" << queueName
                         << "' has been subscribed to the topic '"
                         << "'" << topicName << "'" << std::endl;
               std::cout << "with the subscription ARN '" << subscriptionARN <<
"."
                         << std::endl;
               subscriptionARNS.push_back(subscriptionARN);
           }
           else {
               std::cerr << "Error with TopicsAndQueues::Subscribe. "
                         << outcome.GetError().GetMessage()
                         << std::endl;

               cleanUp(topicARN,
                       queueURLS,
                       subscriptionARNS,
                       snsClient,
                       sqsClient);

               return false;
           }
```

Abonnieren Sie ein Thema mit einem Filter.

```
       static const Aws::String TONE_ATTRIBUTE("tone");
       static const Aws::Vector<Aws::String> TONES = {"cheerful", "funny",
"serious",
                                                      "sincere"};
```

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::SNS::SNSClient snsClient(clientConfiguration);

            Aws::SNS::Model::SubscribeRequest request;
            request.SetTopicArn(topicARN);
            request.SetProtocol("sqs");
            request.SetEndpoint(queueARN);
            if (isFifoTopic) {
                if (first) {
                    std::cout << "Subscriptions to a FIFO topic can have
 filters."
                              << std::endl;
                    std::cout
                            << "If you add a filter to this subscription, then
 only the filtered messages "
                            << "will be received in the queue." << std::endl;
                    std::cout << "For information about message filtering, "
                              << "see https://docs.aws.amazon.com/sns/latest/dg/
sns-message-filtering.html"
                              << std::endl;
                    std::cout << "For this example, you can filter messages by a
 \""
                              << TONE_ATTRIBUTE << "\" attribute." << std::endl;
                }

                std::ostringstream ostringstream;
                ostringstream << "Filter messages for \"" << queueName
                              << "\"'s subscription to the topic \""
                              << topicName << "\"?  (y/n)";

                // Add filter if user answers yes.
                if (askYesNoQuestion(ostringstream.str())) {
                    Aws::String jsonPolicy = getFilterPolicyFromUser();
                    if (!jsonPolicy.empty()) {
                        filteringMessages = true;

                        std::cout << "This is the filter policy for this
 subscription."
                                  << std::endl;
                        std::cout << jsonPolicy << std::endl;
```

```
                    request.AddAttributes("FilterPolicy", jsonPolicy);
                }
                else {
                    std::cout
                            << "Because you did not select any attributes, no
 filter "
                            << "will be added to this subscription." <<
 std::endl;
                }
            }
        }  // if (isFifoTopic)
        Aws::SNS::Model::SubscribeOutcome outcome =
 snsClient.Subscribe(request);

        if (outcome.IsSuccess()) {
            Aws::String subscriptionARN =
 outcome.GetResult().GetSubscriptionArn();
            std::cout << "The queue '" << queueName
                      << "' has been subscribed to the topic '"
                      << "'" << topicName << "'" << std::endl;
            std::cout << "with the subscription ARN '" << subscriptionARN <<
 "."
                      << std::endl;
            subscriptionARNS.push_back(subscriptionARN);
        }
        else {
            std::cerr << "Error with TopicsAndQueues::Subscribe. "
                      << outcome.GetError().GetMessage()
                      << std::endl;

            cleanUp(topicARN,
                    queueURLS,
                    subscriptionARNS,
                    snsClient,
                    sqsClient);

            return false;
        }

//! Routine that lets the user select attributes for a subscription filter
 policy.
/*!
 \sa getFilterPolicyFromUser()
```

```
 \return Aws::String: The filter policy as JSON.
 */
Aws::String AwsDoc::TopicsAndQueues::getFilterPolicyFromUser() {
    std::cout
            << "You can filter messages by one or more of the following \""
            << TONE_ATTRIBUTE << "\" attributes." << std::endl;

    std::vector<Aws::String> filterSelections;
    int selection;
    do {
        for (size_t j = 0; j < TONES.size(); ++j) {
            std::cout << "  " << (j + 1) << ". " << TONES[j]
                      << std::endl;
        }
        selection = askQuestionForIntRange(
                "Enter a number (or enter zero to stop adding more). ",
                0, static_cast<int>(TONES.size()));

        if (selection != 0) {
            const Aws::String &selectedTone(TONES[selection - 1]);
            // Add the tone to the selection if it is not already added.
            if (std::find(filterSelections.begin(),
                          filterSelections.end(),
                          selectedTone)
                == filterSelections.end()) {
                filterSelections.push_back(selectedTone);
            }
        }
    } while (selection != 0);

    Aws::String result;
    if (!filterSelections.empty()) {
        std::ostringstream jsonPolicyStream;
        jsonPolicyStream << "{ \"" << TONE_ATTRIBUTE << "\": [";


        for (size_t j = 0; j < filterSelections.size(); ++j) {
            jsonPolicyStream << "\"" << filterSelections[j] << "\"";
            if (j < filterSelections.size() - 1) {
                jsonPolicyStream << ",";
            }
        }
        jsonPolicyStream << "] }";
```

```
        result = jsonPolicyStream.str();
    }

    return result;
}
```

- Details zu API finden Sie unter Abonnieren in der AWS SDK for C++ -API-Referenz.

CLI

AWS CLI

So abonnieren Sie ein Thema

Der folgende `subscribe`-Befehl abonniert das angegebene Thema mit eine E-Mail-Adresse.

```
aws sns subscribe \
    --topic-arn arn:aws:sns:us-west-2:123456789012:my-topic \
    --protocol email \
    --notification-endpoint my-email@example.com
```

Ausgabe:

```
{
    "SubscriptionArn": "pending confirmation"
}
```

- API-Details finden Sie unter Subscribe in der AWS CLI -Befehlsreferenz.

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

Abonnieren Sie eine Warteschlange für ein Thema mit optionalen Filtern.

```
// SnsActions encapsulates the Amazon Simple Notification Service (Amazon SNS)
 actions
// used in the examples.
type SnsActions struct {
 SnsClient *sns.Client
}



// SubscribeQueue subscribes an Amazon Simple Queue Service (Amazon SQS) queue to
 an
// Amazon SNS topic. When filterMap is not nil, it is used to specify a filter
 policy
// so that messages are only sent to the queue when the message has the specified
 attributes.
func (actor SnsActions) SubscribeQueue(topicArn string, queueArn string,
 filterMap map[string][]string) (string, error) {
 var subscriptionArn string
 var attributes map[string]string
 if filterMap != nil {
  filterBytes, err := json.Marshal(filterMap)
  if err != nil {
   log.Printf("Couldn't create filter policy, here's why: %v\n", err)
   return "", err
  }
  attributes = map[string]string{"FilterPolicy": string(filterBytes)}
 }
 output, err := actor.SnsClient.Subscribe(context.TODO(), &sns.SubscribeInput{
  Protocol:              aws.String("sqs"),
  TopicArn:              aws.String(topicArn),
  Attributes:            attributes,
  Endpoint:              aws.String(queueArn),
  ReturnSubscriptionArn: true,
 })
 if err != nil {
  log.Printf("Couldn't susbscribe queue %v to topic %v. Here's why: %v\n",
   queueArn, topicArn, err)
 } else {
  subscriptionArn = *output.SubscriptionArn
 }

 return subscriptionArn, err
```

```
}
```

- Details zu API finden Sie unter Abonnieren in der AWS SDK for Go -API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

Abonnieren Sie eine E-Mail-Adresse für ein Thema.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SnsException;
import software.amazon.awssdk.services.sns.model.SubscribeRequest;
import software.amazon.awssdk.services.sns.model.SubscribeResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class SubscribeEmail {
    public static void main(String[] args) {
        final String usage = """
                Usage:     <topicArn> <email>

                Where:
                   topicArn - The ARN of the topic to subscribe.
                   email - The email address to use.
                """;
```

```
        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String topicArn = args[0];
        String email = args[1];
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        subEmail(snsClient, topicArn, email);
        snsClient.close();
    }

    public static void subEmail(SnsClient snsClient, String topicArn, String
 email) {
        try {
            SubscribeRequest request = SubscribeRequest.builder()
                    .protocol("email")
                    .endpoint(email)
                    .returnSubscriptionArn(true)
                    .topicArn(topicArn)
                    .build();

            SubscribeResponse result = snsClient.subscribe(request);
            System.out.println("Subscription ARN: " + result.subscriptionArn() +
 "\n\n Status is "
                    + result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

Abonnieren Sie ein Thema über einen HTTP-Endpunkt.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
```

```
import software.amazon.awssdk.services.sns.model.SnsException;
import software.amazon.awssdk.services.sns.model.SubscribeRequest;
import software.amazon.awssdk.services.sns.model.SubscribeResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class SubscribeHTTPS {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <topicArn> <url>

                Where:
                   topicArn - The ARN of the topic to subscribe.
                   url - The HTTPS endpoint that you want to receive
 notifications.
                """;

        if (args.length < 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String topicArn = args[0];
        String url = args[1];
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        subHTTPS(snsClient, topicArn, url);
        snsClient.close();
    }

    public static void subHTTPS(SnsClient snsClient, String topicArn, String url)
 {
        try {
            SubscribeRequest request = SubscribeRequest.builder()
```

```
                    .protocol("https")
                    .endpoint(url)
                    .returnSubscriptionArn(true)
                    .topicArn(topicArn)
                    .build();

            SubscribeResponse result = snsClient.subscribe(request);
            System.out.println("Subscription ARN is " + result.subscriptionArn()
 + "\n\n Status is "
                    + result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

Abonnieren Sie eine Lambda-Funktion für ein Thema.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SnsException;
import software.amazon.awssdk.services.sns.model.SubscribeRequest;
import software.amazon.awssdk.services.sns.model.SubscribeResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class SubscribeLambda {

    public static void main(String[] args) {

        final String usage = """

                Usage:    <topicArn> <lambdaArn>
```

```
                Where:
                   topicArn - The ARN of the topic to subscribe.
                   lambdaArn - The ARN of an AWS Lambda function.
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String topicArn = args[0];
        String lambdaArn = args[1];
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        String arnValue = subLambda(snsClient, topicArn, lambdaArn);
        System.out.println("Subscription ARN: " + arnValue);
        snsClient.close();
    }

    public static String subLambda(SnsClient snsClient, String topicArn, String
 lambdaArn) {
        try {
            SubscribeRequest request = SubscribeRequest.builder()
                    .protocol("lambda")
                    .endpoint(lambdaArn)
                    .returnSubscriptionArn(true)
                    .topicArn(topicArn)
                    .build();

            SubscribeResponse result = snsClient.subscribe(request);
            return result.subscriptionArn();

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
}
```

- Details zu API finden Sie unter Abonnieren in der AWS SDK for Java 2.x -API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie den Client in einem separaten Modul und exportieren Sie ihn.

```
import { SNSClient } from "@aws-sdk/client-sns";

// The AWS Region can be provided here using the `region` property. If you leave
 it blank
// the SDK will default to the region set in your AWS config.
export const snsClient = new SNSClient({});
```

Importieren Sie das SDK- und Client-Module und rufen Sie die API auf.

```
import { SubscribeCommand } from "@aws-sdk/client-sns";
import { snsClient } from "../libs/snsClient.js";

/**
 * @param {string} topicArn - The ARN of the topic for which you wish to confirm
 a subscription.
 * @param {string} emailAddress - The email address that is subscribed to the
 topic.
 */
export const subscribeEmail = async (
  topicArn = "TOPIC_ARN",
  emailAddress = "usern@me.com",
) => {
  const response = await snsClient.send(
    new SubscribeCommand({
      Protocol: "email",
```

```
      TopicArn: topicArn,
      Endpoint: emailAddress,
    }),
  );
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: 'c8e35bcd-b3c0-5940-9f66-06f6fcc108f0',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   SubscriptionArn: 'pending confirmation'
  // }
};
```

Abonnieren Sie eine mobile Anwendung für ein Thema.

```
import { SubscribeCommand } from "@aws-sdk/client-sns";
import { snsClient } from "../libs/snsClient.js";

/**
 * @param {string} topicArn - The ARN of the topic the subscriber is subscribing
 to.
 * @param {string} endpoint - The Endpoint ARN of an application. This endpoint
 is created
 *                            when an application registers for notifications.
 */
export const subscribeApp = async (
  topicArn = "TOPIC_ARN",
  endpoint = "ENDPOINT",
) => {
  const response = await snsClient.send(
    new SubscribeCommand({
      Protocol: "application",
      TopicArn: topicArn,
      Endpoint: endpoint,
    }),
  );
  console.log(response);
```

```
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: 'c8e35bcd-b3c0-5940-9f66-06f6fcc108f0',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   SubscriptionArn: 'pending confirmation'
  // }
  return response;
};
```

Abonnieren Sie eine Lambda-Funktion für ein Thema.

```
import { SubscribeCommand } from "@aws-sdk/client-sns";
import { snsClient } from "../libs/snsClient.js";

/**
 * @param {string} topicArn - The ARN of the topic the subscriber is subscribing
 to.
 * @param {string} endpoint - The Endpoint ARN of and AWS Lambda function.
 */
export const subscribeLambda = async (
  topicArn = "TOPIC_ARN",
  endpoint = "ENDPOINT",
) => {
  const response = await snsClient.send(
    new SubscribeCommand({
      Protocol: "lambda",
      TopicArn: topicArn,
      Endpoint: endpoint,
    }),
  );
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: 'c8e35bcd-b3c0-5940-9f66-06f6fcc108f0',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
```

```
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   SubscriptionArn: 'pending confirmation'
  // }
  return response;
};
```

Abonnieren Sie eine SQS-Warteschlange für ein Thema.

```
import { SubscribeCommand, SNSClient } from "@aws-sdk/client-sns";

const client = new SNSClient({});

export const subscribeQueue = async (
  topicArn = "TOPIC_ARN",
  queueArn = "QUEUE_ARN",
) => {
  const command = new SubscribeCommand({
    TopicArn: topicArn,
    Protocol: "sqs",
    Endpoint: queueArn,
  });

  const response = await client.send(command);
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: '931e13d9-5e2b-543f-8781-4e9e494c5ff2',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   SubscriptionArn: 'arn:aws:sns:us-east-1:xxxxxxxxxxxx:subscribe-queue-
test-430895:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx'
  // }
  return response;
};
```

Abonnieren Sie ein Thema mit einem Filter.

```
import { SubscribeCommand, SNSClient } from "@aws-sdk/client-sns";

const client = new SNSClient({});

export const subscribeQueueFiltered = async (
  topicArn = "TOPIC_ARN",
  queueArn = "QUEUE_ARN",
) => {
  const command = new SubscribeCommand({
    TopicArn: topicArn,
    Protocol: "sqs",
    Endpoint: queueArn,
    Attributes: {
      // This subscription will only receive messages with the 'event' attribute
 set to 'order_placed'.
      FilterPolicyScope: "MessageAttributes",
      FilterPolicy: JSON.stringify({
        event: ["order_placed"],
      }),
    },
  });

  const response = await client.send(command);
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: '931e13d9-5e2b-543f-8781-4e9e494c5ff2',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   SubscriptionArn: 'arn:aws:sns:us-east-1:xxxxxxxxxxxx:subscribe-queue-
test-430895:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx'
  // }
  return response;
};
```

- Weitere Informationen finden Sie im AWS SDK for JavaScript -Entwicklerhandbuch.

- Details zu API finden Sie unter Abonnieren in der AWS SDK for JavaScript -API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

Abonnieren Sie eine E-Mail-Adresse für ein Thema.

```
suspend fun subEmail(topicArnVal: String, email: String): String {

    val request = SubscribeRequest {
        protocol = "email"
        endpoint = email
        returnSubscriptionArn = true
        topicArn = topicArnVal
    }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.subscribe(request)
        return result.subscriptionArn.toString()
    }
}
```

Abonnieren Sie eine Lambda-Funktion für ein Thema.

```
suspend fun subLambda(topicArnVal: String?, lambdaArn: String?) {

    val request = SubscribeRequest {
        protocol = "lambda"
        endpoint = lambdaArn
        returnSubscriptionArn = true
        topicArn = topicArnVal
    }
```

```
    SnsClient { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.subscribe(request)
        println(" The subscription Arn is ${result.subscriptionArn}")
    }
}
```

- Details zu API finden Sie unter Subscribe (Abonnieren) in der AWS -SDK-für-Kotlin-API-Referenz.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

Abonnieren Sie eine E-Mail-Adresse für ein Thema.

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Prepares to subscribe an endpoint by sending the endpoint a confirmation
 message.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/
guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);
```

```
$protocol = 'email';
$endpoint = 'sample@example.com';
$topic = 'arn:aws:sns:us-east-1:111122223333:MyTopic';

try {
    $result = $SnSclient->subscribe([
        'Protocol' => $protocol,
        'Endpoint' => $endpoint,
        'ReturnSubscriptionArn' => true,
        'TopicArn' => $topic,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

Abonnieren Sie ein Thema über einen HTTP-Endpunkt.

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Prepares to subscribe an endpoint by sending the endpoint a confirmation
 message.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/
guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);
```

```
$protocol = 'https';
$endpoint = 'https://';
$topic = 'arn:aws:sns:us-east-1:111122223333:MyTopic';

try {
    $result = $SnSclient->subscribe([
        'Protocol' => $protocol,
        'Endpoint' => $endpoint,
        'ReturnSubscriptionArn' => true,
        'TopicArn' => $topic,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- Details zu API finden Sie unter Abonnieren in der AWS SDK for PHP -API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

Abonnieren Sie eine E-Mail-Adresse für ein Thema.

```
class SnsWrapper:
    """Encapsulates Amazon SNS topic and subscription functions."""

    def __init__(self, sns_resource):
        """
        :param sns_resource: A Boto3 Amazon SNS resource.
        """
        self.sns_resource = sns_resource
```

```
    @staticmethod
    def subscribe(topic, protocol, endpoint):
        """
        Subscribes an endpoint to the topic. Some endpoint types, such as email,
        must be confirmed before their subscriptions are active. When a
subscription
        is not confirmed, its Amazon Resource Number (ARN) is set to
        'PendingConfirmation'.

        :param topic: The topic to subscribe to.
        :param protocol: The protocol of the endpoint, such as 'sms' or 'email'.
        :param endpoint: The endpoint that receives messages, such as a phone
number
                         (in E.164 format) for SMS messages, or an email address
for
                         email messages.
        :return: The newly added subscription.
        """
        try:
            subscription = topic.subscribe(
                Protocol=protocol, Endpoint=endpoint, ReturnSubscriptionArn=True
            )
            logger.info("Subscribed %s %s to topic %s.", protocol, endpoint,
topic.arn)
        except ClientError:
            logger.exception(
                "Couldn't subscribe %s %s to topic %s.", protocol, endpoint,
topic.arn
            )
            raise
        else:
            return subscription
```

- Details zu API finden Sie unter Abonnieren in der AWS API-Referenz zu SDK for Python (Boto3).

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

Abonnieren Sie eine E-Mail-Adresse für ein Thema.

```
require "aws-sdk-sns"
require "logger"

# Represents a service for creating subscriptions in Amazon Simple Notification
 Service (SNS)
class SubscriptionService
  # Initializes the SubscriptionService with an SNS client
  #
  # @param sns_client [Aws::SNS::Client] The SNS client
  def initialize(sns_client)
    @sns_client = sns_client
    @logger = Logger.new($stdout)
  end

  # Attempts to create a subscription to a topic
  #
  # @param topic_arn [String] The ARN of the SNS topic
  # @param protocol [String] The subscription protocol (e.g., email)
  # @param endpoint [String] The endpoint that receives the notifications (email
 address)
  # @return [Boolean] true if subscription was successfully created, false
 otherwise
  def create_subscription(topic_arn, protocol, endpoint)
    @sns_client.subscribe(topic_arn: topic_arn, protocol: protocol, endpoint:
 endpoint)
    @logger.info("Subscription created successfully.")
    true
  rescue Aws::SNS::Errors::ServiceError => e
    @logger.error("Error while creating the subscription: #{e.message}")
    false
  end
```

```
end

# Main execution if the script is run directly
if $PROGRAM_NAME == __FILE__
  protocol = "email"
  endpoint = "EMAIL_ADDRESS" # Should be replaced with a real email address
  topic_arn = "TOPIC_ARN"    # Should be replaced with a real topic ARN

  sns_client = Aws::SNS::Client.new
  subscription_service = SubscriptionService.new(sns_client)

  @logger.info("Creating the subscription.")
  unless subscription_service.create_subscription(topic_arn, protocol, endpoint)
    @logger.error("Subscription creation failed. Stopping program.")
    exit 1
  end
end
```

- Weitere Informationen finden Sie im AWS SDK for Ruby -Entwicklerhandbuch.
- Details zu API finden Sie unter Abonnieren in der AWS SDK for Ruby -API-Referenz.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

Abonnieren Sie eine E-Mail-Adresse für ein Thema.

```
async fn subscribe_and_publish(
    client: &Client,
    topic_arn: &str,
    email_address: &str,
) -> Result<(), Error> {
    println!("Receiving on topic with ARN: `{}`", topic_arn);
```

```
    let rsp = client
        .subscribe()
        .topic_arn(topic_arn)
        .protocol("email")
        .endpoint(email_address)
        .send()
        .await?;

    println!("Added a subscription: {:?}", rsp);

    let rsp = client
        .publish()
        .topic_arn(topic_arn)
        .message("hello sns!")
        .send()
        .await?;

    println!("Published message: {:?}", rsp);

    Ok(())
}
```

- Details zu API finden Sie unter Abonnieren in der AWS -SDK-für-Rust-API-Referenz.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

Abonnieren Sie eine E-Mail-Adresse für ein Thema.

```
    TRY.
        oo_result = lo_sns->subscribe(                      "oo_result is
returned for testing purposes."
                iv_topicarn = iv_topic_arn
                iv_protocol = 'email'
```

```
                iv_endpoint = iv_email_address
                iv_returnsubscriptionarn = abap_true
            ).
        MESSAGE 'Email address subscribed to SNS topic.' TYPE 'I'.
      CATCH /aws1/cx_snsnotfoundexception.
        MESSAGE 'Topic does not exist.' TYPE 'E'.
      CATCH /aws1/cx_snssubscriptionlmte00.
        MESSAGE 'Unable to create subscriptions. You have reached the maximum
 number of subscriptions allowed.' TYPE 'E'.
    ENDTRY.
```

- Details zu API finden Sie unter Subscribe (Anmelden) in der API-Referenz für das AWS SDK für SAP ABAP.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unterAmazon SNS mit einem AWS SDK verwenden. Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung **`TagResource`** mit einem AWS SDK oder CLI

Die folgenden Codebeispiele zeigen, wie es verwendet wird`TagResource`.

CLI

AWS CLI

So fügen Sie einem Thema ein Tag hinzu

Das folgende `tag-resource`-Beispiel fügt dem angegebenen Amazon-SNS-Thema ein Metadaten-Tag hinzu.

```
aws sns tag-resource \
    --resource-arn arn:aws:sns:us-west-2:123456789012:MyTopic \
    --tags Key=Team,Value=Alpha
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

- Einzelheiten zur API finden Sie TagResourcein der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SnsException;
import software.amazon.awssdk.services.sns.model.Tag;
import software.amazon.awssdk.services.sns.model.TagResourceRequest;
import java.util.ArrayList;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class AddTags {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <topicArn>

                Where:
                   topicArn - The ARN of the topic to which tags are added.

                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }
```

```
        String topicArn = args[0];
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        addTopicTags(snsClient, topicArn);
        snsClient.close();
    }

    public static void addTopicTags(SnsClient snsClient, String topicArn) {
        try {
            Tag tag = Tag.builder()
                    .key("Team")
                    .value("Development")
                    .build();

            Tag tag2 = Tag.builder()
                    .key("Environment")
                    .value("Gamma")
                    .build();

            List<Tag> tagList = new ArrayList<>();
            tagList.add(tag);
            tagList.add(tag2);

            TagResourceRequest tagResourceRequest = TagResourceRequest.builder()
                    .resourceArn(topicArn)
                    .tags(tagList)
                    .build();

            snsClient.tagResource(tagResourceRequest);
            System.out.println("Tags have been added to " + topicArn);

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Einzelheiten zur API finden Sie TagResourcein der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

```
suspend fun addTopicTags(topicArn: String) {

    val tag = Tag {
        key = "Team"
        value = "Development"
    }

    val tag2 = Tag {
        key = "Environment"
        value = "Gamma"
    }

    val tagList = mutableListOf<Tag>()
    tagList.add(tag)
    tagList.add(tag2)

    val request = TagResourceRequest {
        resourceArn = topicArn
        tags = tagList
    }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        snsClient.tagResource(request)
        println("Tags have been added to $topicArn")
    }
}
```

- API-Details finden Sie TagResourcein der API-Referenz zum AWS SDK für Kotlin.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unterAmazon SNS mit einem AWS SDK verwenden. Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung **`Unsubscribe`** mit einem AWS SDK oder CLI

Die folgenden Codebeispiele zeigen, wie es verwendet wird`Unsubscribe`.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Veröffentlichen Sie Nachrichten in Warteschlangen

.NET

AWS SDK for .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

Melden Sie sich mit einem Abonnement-ARN von einem Thema ab.

```
/// <summary>
/// Unsubscribe from a topic by a subscription ARN.
/// </summary>
/// <param name="subscriptionArn">The ARN of the subscription.</param>
/// <returns>True if successful.</returns>
public async Task<bool> UnsubscribeByArn(string subscriptionArn)
{
    var unsubscribeResponse = await _amazonSNSClient.UnsubscribeAsync(
        new UnsubscribeRequest()
        {
            SubscriptionArn = subscriptionArn
        });
    return unsubscribeResponse.HttpStatusCode == HttpStatusCode.OK;
}
```

- Details zu API finden Sie unter Abmelden in der AWS SDK for .NET -API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

```
//! Delete a subscription to an Amazon Simple Notification Service (Amazon SNS)
 topic.
/*!
  \param subscriptionARN: The Amazon Resource Name (ARN) for an Amazon SNS topic
 subscription.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SNS::unsubscribe(const Aws::String &subscriptionARN,
                              const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::SNS::Model::UnsubscribeRequest request;
    request.SetSubscriptionArn(subscriptionARN);

    const Aws::SNS::Model::UnsubscribeOutcome outcome =
 snsClient.Unsubscribe(request);

    if (outcome.IsSuccess()) {
        std::cout << "Unsubscribed successfully " << std::endl;
    }
    else {
        std::cerr << "Error while unsubscribing " <<
 outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Details zu API finden Sie unter Abmelden in der AWS SDK for C++ -API-Referenz.

CLI

AWS CLI

So melden Sie sich von einem Thema ab

Im folgenden `unsubscribe`-Beispiel wird das angegebene Abonnement aus einem Thema gelöscht.

```
aws sns unsubscribe \
    --subscription-arn arn:aws:sns:us-west-2:0123456789012:my-
topic:8a21d249-4329-4871-acc6-7be709c6ea7f
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

- API-Details finden Sie unter Unsubscribe in der AWS CLI -Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SnsException;
import software.amazon.awssdk.services.sns.model.UnsubscribeRequest;
import software.amazon.awssdk.services.sns.model.UnsubscribeResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
```

```
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class Unsubscribe {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <subscriptionArn>

                Where:
                   subscriptionArn - The ARN of the subscription to delete.
                """;

        if (args.length < 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String subscriptionArn = args[0];
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        unSub(snsClient, subscriptionArn);
        snsClient.close();
    }

    public static void unSub(SnsClient snsClient, String subscriptionArn) {
        try {
            UnsubscribeRequest request = UnsubscribeRequest.builder()
                    .subscriptionArn(subscriptionArn)
                    .build();

            UnsubscribeResponse result = snsClient.unsubscribe(request);
            System.out.println("\n\nStatus was " +
 result.sdkHttpResponse().statusCode()
                    + "\n\nSubscription was removed for " +
 request.subscriptionArn());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
```

```
        }
    }
}
```

- Details zu API finden Sie unter Abmelden in der AWS SDK for Java 2.x -API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie den Client in einem separaten Modul und exportieren Sie ihn.

```
import { SNSClient } from "@aws-sdk/client-sns";

// The AWS Region can be provided here using the `region` property. If you leave
 it blank
// the SDK will default to the region set in your AWS config.
export const snsClient = new SNSClient({});
```

Importieren Sie das SDK- und Client-Module und rufen Sie die API auf.

```
import { UnsubscribeCommand } from "@aws-sdk/client-sns";
import { snsClient } from "../libs/snsClient.js";

/**
 * @param {string} subscriptionArn - The ARN of the subscription to cancel.
 */
const unsubscribe = async (
  subscriptionArn = "arn:aws:sns:us-east-1:xxxxxxxxxxxx:mytopic:xxxxxxxx-xxxx-
xxxx-xxxx-xxxxxxxxxxxx",
) => {
  const response = await snsClient.send(
    new UnsubscribeCommand({
```

```
      SubscriptionArn: subscriptionArn,
    }),
  );
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: '0178259a-9204-507c-b620-78a7570a44c6',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   }
  // }
  return response;
};
```

- Weitere Informationen finden Sie im AWS SDK for JavaScript -Entwicklerhandbuch.
- Details zu API finden Sie unter Abmelden in der AWS SDK for JavaScript -API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

```
suspend fun unSub(subscriptionArnVal: String) {

    val request = UnsubscribeRequest {
        subscriptionArn = subscriptionArnVal
    }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        snsClient.unsubscribe(request)
        println("Subscription was removed for ${request.subscriptionArn}")
    }
```

```
}
```

- Details zu API finden Sie unter Unsubscribe (Abmelden) in der AWS -SDK-für-Kotlin-API-Referenz.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Deletes a subscription to an Amazon SNS topic.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/
guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$subscription = 'arn:aws:sns:us-east-1:111122223333:MySubscription';

try {
    $result = $SnSclient->unsubscribe([
        'SubscriptionArn' => $subscription,
```

```
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- Weitere Informationen finden Sie im AWS SDK for PHP -Entwicklerhandbuch.
- Details zu API finden Sie unter Abmelden in der AWS SDK for PHP -API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

```
class SnsWrapper:
    """Encapsulates Amazon SNS topic and subscription functions."""

    def __init__(self, sns_resource):
        """
        :param sns_resource: A Boto3 Amazon SNS resource.
        """
        self.sns_resource = sns_resource


    @staticmethod
    def delete_subscription(subscription):
        """
        Unsubscribes and deletes a subscription.
        """
        try:
            subscription.delete()
            logger.info("Deleted subscription %s.", subscription.arn)
        except ClientError:
```

```
            logger.exception("Couldn't delete subscription %s.",
subscription.arn)
            raise
```

- Details zu API finden Sie unter Abmelden in der AWS API-Referenz zu SDK for Python (Boto3).

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

```
    TRY.
        lo_sns->unsubscribe( iv_subscriptionarn = iv_subscription_arn ).
        MESSAGE 'Subscription deleted.' TYPE 'I'.
      CATCH /aws1/cx_snsnotfoundexception.
        MESSAGE 'Subscription does not exist.' TYPE 'E'.
      CATCH /aws1/cx_snsinvalidparameterex.
        MESSAGE 'Subscription with "PendingConfirmation" status cannot be
deleted/unsubscribed. Confirm subscription before performing unsubscribe
operation.' TYPE 'E'.
    ENDTRY.
```

- Details zu API finden Sie unter Abmelden in der API-Referenz für das AWS SDK für SAP ABAP.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unterAmazon SNS mit einem AWS SDK verwenden. Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Szenarien für Amazon SNS mit AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie gängige Szenarien in Amazon SNS mit AWS SDKs implementieren. Diese Szenarien zeigen Ihnen, wie Sie bestimmte Aufgaben ausführen können, indem Sie mehrere Funktionen in Amazon SNS aufrufen. Jedes Szenario enthält einen Link zu GitHub, wo Sie Anweisungen zur Einrichtung und Ausführung des Codes finden.

Beispiele

- Erstellen Sie mithilfe eines SDK einen Plattformendpunkt für Amazon SNS SNS-Push-Benachrichtigungen AWS
- Erstellen und Veröffentlichen in einem FIFO-Amazon-SNS-Thema mithilfe eines SDK AWS
- Veröffentlichen Sie SMS-Nachrichten mithilfe eines SDK zu einem Amazon SNS SNS-Thema AWS
- Veröffentlichen Sie mit Amazon S3 mithilfe eines SDK eine AWS große Nachricht in Amazon SNS
- Veröffentlichen Sie eine Amazon SNS SNS-SMS-Textnachricht mit einem SDK AWS
- Veröffentlichen Sie Amazon SNS SNS-Nachrichten mithilfe eines SDK in Amazon SQS SQS-Warteschlangen AWS

Erstellen Sie mithilfe eines SDK einen Plattformendpunkt für Amazon SNS SNS-Push-Benachrichtigungen AWS

Das folgende Codebeispiel zeigt, wie ein Plattformendpunkt für Amazon-SNS-Push-Benachrichtigungen erstellt werden.

CLI

AWS CLI

So erstellen Sie ein Plattformanwendungsendpunkt

Im folgenden `create-platform-endpoint`-Beispiel wird mithilfe des angegebenen Tokens ein Endpunkt für die angegebene Plattformanwendung erstellt.

```
aws sns create-platform-endpoint \
    --platform-application-arn arn:aws:sns:us-west-2:123456789012:app/GCM/
MyApplication \
    --token EXAMPLE12345...
```

Ausgabe:

```
{
      "EndpointArn": "arn:aws:sns:us-west-2:1234567890:endpoint/GCM/
MyApplication/12345678-abcd-9012-efgh-345678901234"
}
```

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.CreatePlatformEndpointRequest;
import software.amazon.awssdk.services.sns.model.CreatePlatformEndpointResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 *
 * In addition, create a platform application using the AWS Management Console.
 * See this doc topic:
 *
 * https://docs.aws.amazon.com/sns/latest/dg/mobile-push-send-register.html
 *
 * Without the values created by following the previous link, this code examples
 * does not work.
 */
```

```
public class RegistrationExample {
    public static void main(String[] args) {
        final String usage = """

                Usage:     <token> <platformApplicationArn>

                Where:
                   token - The name of the FIFO topic.\s
                   platformApplicationArn - The ARN value of platform
 application. You can get this value from the AWS Management Console.\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String token = args[0];
        String platformApplicationArn = args[1];
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        createEndpoint(snsClient, token, platformApplicationArn);
    }

    public static void createEndpoint(SnsClient snsClient, String token, String
 platformApplicationArn) {
        System.out.println("Creating platform endpoint with token " + token);
        try {
            CreatePlatformEndpointRequest endpointRequest =
 CreatePlatformEndpointRequest.builder()
                    .token(token)
                    .platformApplicationArn(platformApplicationArn)
                    .build();

            CreatePlatformEndpointResponse response =
 snsClient.createPlatformEndpoint(endpointRequest);
            System.out.println("The ARN of the endpoint is " +
 response.endpointArn());
        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
```

```
    }
}
```

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unterAmazon SNS mit einem AWS SDK verwenden. Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Erstellen und Veröffentlichen in einem FIFO-Amazon-SNS-Thema mithilfe eines SDK AWS

Die folgenden Code-Beispiele zeigen, wie man ein FIFO-Amazon-SNS-Thema erstellt.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

Dieses Beispiel

- erstellt ein Amazon-SNS-FIFO-Thema, zwei Amazon SQS-FIFO-Warteschlangen und eine Standard-Warteschlange.
- abonniert die Warteschlangen für das Thema und veröffentlicht eine Nachricht zu dem Thema.

Der Test überprüft den Eingang der Nachricht in jeder Warteschlange. Das vollständige Beispiel zeigt auch das Hinzufügen von Zugriffsrichtlinien und löscht die Ressourcen am Ende.

```
public class PriceUpdateExample {
    public final static SnsClient snsClient = SnsClient.create();
    public final static SqsClient sqsClient = SqsClient.create();

    public static void main(String[] args) {

        final String usage = "\n" +
```

```
                "Usage: " +
                "    <topicName> <wholesaleQueueFifoName> <retailQueueFifoName>
<analyticsQueueName>\n\n" +
                "Where:\n" +
                "   fifoTopicName - The name of the FIFO topic that you want to
create. \n\n" +
                "   wholesaleQueueARN - The name of a SQS FIFO queue that will be
created for the wholesale consumer. \n\n"
                +
                "   retailQueueARN - The name of a SQS FIFO queue that will
created for the retail consumer. \n\n" +
                "   analyticsQueueARN - The name of a SQS standard queue that
will be created for the analytics consumer. \n\n";
        if (args.length != 4) {
            System.out.println(usage);
            System.exit(1);
        }

        final String fifoTopicName = args[0];
        final String wholeSaleQueueName = args[1];
        final String retailQueueName = args[2];
        final String analyticsQueueName = args[3];

        // For convenience, the QueueData class holds metadata about a queue:
ARN, URL,
        // name and type.
        List<QueueData> queues = List.of(
                new QueueData(wholeSaleQueueName, QueueType.FIFO),
                new QueueData(retailQueueName, QueueType.FIFO),
                new QueueData(analyticsQueueName, QueueType.Standard));

        // Create queues.
        createQueues(queues);

        // Create a topic.
        String topicARN = createFIFOTopic(fifoTopicName);

        // Subscribe each queue to the topic.
        subscribeQueues(queues, topicARN);

        // Allow the newly created topic to send messages to the queues.
        addAccessPolicyToQueuesFINAL(queues, topicARN);

        // Publish a sample price update message with payload.
```

```
        publishPriceUpdate(topicARN, "{\"product\": 214, \"price\": 79.99}",
"Consumables");

        // Clean up resources.
        deleteSubscriptions(queues);
        deleteQueues(queues);
        deleteTopic(topicARN);
    }

    public static String createFIFOTopic(String topicName) {
        try {
            // Create a FIFO topic by using the SNS service client.
            Map<String, String> topicAttributes = Map.of(
                    "FifoTopic", "true",
                    "ContentBasedDeduplication", "false");

            CreateTopicRequest topicRequest = CreateTopicRequest.builder()
                    .name(topicName)
                    .attributes(topicAttributes)
                    .build();

            CreateTopicResponse response = snsClient.createTopic(topicRequest);
            String topicArn = response.topicArn();
            System.out.println("The topic ARN is" + topicArn);

            return topicArn;

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }

    public static void subscribeQueues(List<QueueData> queues, String topicARN) {
        queues.forEach(queue -> {
            SubscribeRequest subscribeRequest = SubscribeRequest.builder()
                    .topicArn(topicARN)
                    .endpoint(queue.queueARN)
                    .protocol("sqs")
                    .build();

            // Subscribe to the endpoint by using the SNS service client.
```

```
            // Only Amazon SQS queues can receive notifications from an Amazon
SNS FIFO
            // topic.
            SubscribeResponse subscribeResponse =
snsClient.subscribe(subscribeRequest);
            System.out.println("The queue [" + queue.queueARN + "] subscribed to
the topic [" + topicARN + "]");
            queue.subscriptionARN = subscribeResponse.subscriptionArn();
        });
    }

    public static void publishPriceUpdate(String topicArn, String payload, String
groupId) {

        try {
            // Create and publish a message that updates the wholesale price.
            String subject = "Price Update";
            String dedupId = UUID.randomUUID().toString();
            String attributeName = "business";
            String attributeValue = "wholesale";

            MessageAttributeValue msgAttValue = MessageAttributeValue.builder()
                    .dataType("String")
                    .stringValue(attributeValue)
                    .build();

            Map<String, MessageAttributeValue> attributes = new HashMap<>();
            attributes.put(attributeName, msgAttValue);
            PublishRequest pubRequest = PublishRequest.builder()
                    .topicArn(topicArn)
                    .subject(subject)
                    .message(payload)
                    .messageGroupId(groupId)
                    .messageDeduplicationId(dedupId)
                    .messageAttributes(attributes)
                    .build();

            final PublishResponse response = snsClient.publish(pubRequest);
            System.out.println(response.messageId());
            System.out.println(response.sequenceNumber());
            System.out.println("Message was published to " + topicArn);

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
```

```
            System.exit(1);
        }
    }
```

- API-Details finden Sie in den folgenden Themen der AWS SDK for Java 2.x -API-Referenz.
  - CreateTopic
  - Veröffentlichen
  - Abonnieren

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

Erstellen Sie ein Amazon-SNS-FIFO-Thema, abonnieren Sie eine Amazon-SQS-FIFO- und eine Standard-Warteschlange für das Thema und veröffentlichen Sie eine Nachricht zu dem Thema.

```
def usage_demo():
    """Shows how to subscribe queues to a FIFO topic."""
    print("-" * 88)
    print("Welcome to the `Subscribe queues to a FIFO topic` demo!")
    print("-" * 88)

    sns = boto3.resource("sns")
    sqs = boto3.resource("sqs")
    fifo_topic_wrapper = FifoTopicWrapper(sns)
    sns_wrapper = SnsWrapper(sns)

    prefix = "sqs-subscribe-demo-"
    queues = set()
    subscriptions = set()

    wholesale_queue = sqs.create_queue(
```

```
        QueueName=prefix + "wholesale.fifo",
        Attributes={
            "MaximumMessageSize": str(4096),
            "ReceiveMessageWaitTimeSeconds": str(10),
            "VisibilityTimeout": str(300),
            "FifoQueue": str(True),
            "ContentBasedDeduplication": str(True),
        },
    )
    queues.add(wholesale_queue)
    print(f"Created FIFO queue with URL: {wholesale_queue.url}.")

    retail_queue = sqs.create_queue(
        QueueName=prefix + "retail.fifo",
        Attributes={
            "MaximumMessageSize": str(4096),
            "ReceiveMessageWaitTimeSeconds": str(10),
            "VisibilityTimeout": str(300),
            "FifoQueue": str(True),
            "ContentBasedDeduplication": str(True),
        },
    )
    queues.add(retail_queue)
    print(f"Created FIFO queue with URL: {retail_queue.url}.")

    analytics_queue = sqs.create_queue(QueueName=prefix + "analytics",
Attributes={})
    queues.add(analytics_queue)
    print(f"Created standard queue with URL: {analytics_queue.url}.")

    topic = fifo_topic_wrapper.create_fifo_topic("price-updates-topic.fifo")
    print(f"Created FIFO topic: {topic.attributes['TopicArn']}.")

    for q in queues:
        fifo_topic_wrapper.add_access_policy(q, topic.attributes["TopicArn"])

    print(f"Added access policies for topic: {topic.attributes['TopicArn']}.")

    for q in queues:
        sub = fifo_topic_wrapper.subscribe_queue_to_topic(
            topic, q.attributes["QueueArn"]
        )
        subscriptions.add(sub)
```

```
    print(f"Subscribed queues to topic: {topic.attributes['TopicArn']}.")

    input("Press Enter to publish a message to the topic.")

    message_id = fifo_topic_wrapper.publish_price_update(
        topic, '{"product": 214, "price": 79.99}', "Consumables"
    )

    print(f"Published price update with message ID: {message_id}.")

    # Clean up the subscriptions, queues, and topic.
    input("Press Enter to clean up resources.")
    for s in subscriptions:
        sns_wrapper.delete_subscription(s)

    sns_wrapper.delete_topic(topic)

    for q in queues:
        fifo_topic_wrapper.delete_queue(q)

    print(f"Deleted subscriptions, queues, and topic.")

    print("Thanks for watching!")
    print("-" * 88)



class FifoTopicWrapper:
    """Encapsulates Amazon SNS FIFO topic and subscription functions."""

    def __init__(self, sns_resource):
        """
        :param sns_resource: A Boto3 Amazon SNS resource.
        """
        self.sns_resource = sns_resource

    def create_fifo_topic(self, topic_name):
        """
        Create a FIFO topic.
        Topic names must be made up of only uppercase and lowercase ASCII
 letters,
        numbers, underscores, and hyphens, and must be between 1 and 256
 characters long.
        For a FIFO topic, the name must end with the .fifo suffix.
```

```
        :param topic_name: The name for the topic.
        :return: The new topic.
        """
        try:
            topic = self.sns_resource.create_topic(
                Name=topic_name,
                Attributes={
                    "FifoTopic": str(True),
                    "ContentBasedDeduplication": str(False),
                },
            )
            logger.info("Created FIFO topic with name=%s.", topic_name)
            return topic
        except ClientError as error:
            logger.exception("Couldn't create topic with name=%s!", topic_name)
            raise error


    @staticmethod
    def add_access_policy(queue, topic_arn):
        """
        Add the necessary access policy to a queue, so
        it can receive messages from a topic.

        :param queue: The queue resource.
        :param topic_arn: The ARN of the topic.
        :return: None.
        """
        try:
            queue.set_attributes(
                Attributes={
                    "Policy": json.dumps(
                        {
                            "Version": "2012-10-17",
                            "Statement": [
                                {
                                    "Sid": "test-sid",
                                    "Effect": "Allow",
                                    "Principal": {"AWS": "*"},
                                    "Action": "SQS:SendMessage",
                                    "Resource": queue.attributes["QueueArn"],
                                    "Condition": {
                                        "ArnLike": {"aws:SourceArn": topic_arn}
```

```
                                },
                            }
                        ],
                    }
                )
            }
        )
        logger.info("Added trust policy to the queue.")
    except ClientError as error:
        logger.exception("Couldn't add trust policy to the queue!")
        raise error


@staticmethod
def subscribe_queue_to_topic(topic, queue_arn):
    """
    Subscribe a queue to a topic.

    :param topic: The topic resource.
    :param queue_arn: The ARN of the queue.
    :return: The subscription resource.
    """
    try:
        subscription = topic.subscribe(
            Protocol="sqs",
            Endpoint=queue_arn,
        )
        logger.info("The queue is subscribed to the topic.")
        return subscription
    except ClientError as error:
        logger.exception("Couldn't subscribe queue to topic!")
        raise error


@staticmethod
def publish_price_update(topic, payload, group_id):
    """
    Compose and publish a message that updates the wholesale price.

    :param topic: The topic to publish to.
    :param payload: The message to publish.
    :param group_id: The group ID for the message.
    :return: The ID of the message.
    """
```

```
        try:
            att_dict = {"business": {"DataType": "String", "StringValue":
"wholesale"}}
            dedup_id = uuid.uuid4()
            response = topic.publish(
                Subject="Price Update",
                Message=payload,
                MessageAttributes=att_dict,
                MessageGroupId=group_id,
                MessageDeduplicationId=str(dedup_id),
            )
            message_id = response["MessageId"]
            logger.info("Published message to topic %s.", topic.arn)
        except ClientError as error:
            logger.exception("Couldn't publish message to topic %s.", topic.arn)
            raise error
        return message_id


    @staticmethod
    def delete_queue(queue):
        """
        Removes an SQS queue. When run against an AWS account, it can take up to
        60 seconds before the queue is actually deleted.

        :param queue: The queue to delete.
        :return: None
        """
        try:
            queue.delete()
            logger.info("Deleted queue with URL=%s.", queue.url)
        except ClientError as error:
            logger.exception("Couldn't delete queue with URL=%s!", queue.url)
            raise error
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS -SDK für Python (Boto3).
  - CreateTopic

- Veröffentlichen
- Abonnieren

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

Erstellen Sie ein FIFO-Thema, abonnieren Sie eine Amazon-SQS-FIFO-Warteschlange für das Thema und veröffentlichen Sie eine Nachricht zu einem Amazon-SNS-Thema.

```
    " Creates a FIFO topic. "
    DATA lt_tpc_attributes TYPE /aws1/
cl_snstopicattrsmap_w=>tt_topicattributesmap.
    DATA ls_tpc_attributes TYPE /aws1/
cl_snstopicattrsmap_w=>ts_topicattributesmap_maprow.
    ls_tpc_attributes-key = 'FifoTopic'.
    ls_tpc_attributes-value = NEW /aws1/cl_snstopicattrsmap_w( iv_value =
 'true' ).
    INSERT ls_tpc_attributes INTO TABLE lt_tpc_attributes.

    TRY.
        DATA(lo_create_result) = lo_sns->createtopic(
               iv_name = iv_topic_name
               it_attributes = lt_tpc_attributes
        ).
        DATA(lv_topic_arn) = lo_create_result->get_topicarn( ).
        ov_topic_arn = lv_topic_arn.                                        "
 ov_topic_arn is returned for testing purposes. "
        MESSAGE 'FIFO topic created' TYPE 'I'.
      CATCH /aws1/cx_snstopiclimitexcdex.
        MESSAGE 'Unable to create more topics. You have reached the maximum
 number of topics allowed.' TYPE 'E'.
    ENDTRY.
```

```
    " Subscribes an endpoint to an Amazon Simple Notification Service (Amazon
 SNS) topic. "
    " Only Amazon Simple Queue Service (Amazon SQS) FIFO queues can be subscribed
 to an SNS FIFO topic. "
    TRY.
        DATA(lo_subscribe_result) = lo_sns->subscribe(
               iv_topicarn = lv_topic_arn
               iv_protocol = 'sqs'
               iv_endpoint = iv_queue_arn
           ).
        DATA(lv_subscription_arn) = lo_subscribe_result->get_subscriptionarn( ).
        ov_subscription_arn = lv_subscription_arn.                        "
 ov_subscription_arn is returned for testing purposes. "
        MESSAGE 'SQS queue was subscribed to SNS topic.' TYPE 'I'.
      CATCH /aws1/cx_snsnotfoundexception.
        MESSAGE 'Topic does not exist.' TYPE 'E'.
      CATCH /aws1/cx_snssubscriptionlmte00.
        MESSAGE 'Unable to create subscriptions. You have reached the maximum
 number of subscriptions allowed.' TYPE 'E'.
    ENDTRY.

    " Publish message to SNS topic. "
    TRY.
        DATA lt_msg_attributes TYPE /aws1/
cl_snsmessageattrvalue=>tt_messageattributemap.
        DATA ls_msg_attributes TYPE /aws1/
cl_snsmessageattrvalue=>ts_messageattributemap_maprow.
        ls_msg_attributes-key = 'Importance'.
        ls_msg_attributes-value = NEW /aws1/cl_snsmessageattrvalue( iv_datatype =
 'String' iv_stringvalue = 'High' ).
        INSERT ls_msg_attributes INTO TABLE lt_msg_attributes.

        DATA(lo_result) = lo_sns->publish(
             iv_topicarn = lv_topic_arn
             iv_message = 'The price of your mobile plan has been increased from
 $19 to $23'
             iv_subject = 'Changes to mobile plan'
             iv_messagegroupid = 'Update-2'
             iv_messagededuplicationid = 'Update-2.1'
             it_messageattributes = lt_msg_attributes
      ).
        ov_message_id = lo_result->get_messageid( ).                      "
 ov_message_id is returned for testing purposes. "
        MESSAGE 'Message was published to SNS topic.' TYPE 'I'.
```

```
      CATCH /aws1/cx_snsnotfoundexception.
        MESSAGE 'Topic does not exist.' TYPE 'E'.
    ENDTRY.
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für SAP ABAP.
  - CreateTopic
  - Veröffentlichen
  - Abonnieren

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unterAmazon SNS mit einem AWS SDK verwenden. Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Veröffentlichen Sie SMS-Nachrichten mithilfe eines SDK zu einem Amazon SNS SNS-Thema AWS

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie ein Amazon-SNS-Thema.
- Verknüpfen Sie Telefonnummern mit dem Thema.
- Veröffentlichen Sie SMS-Nachrichten im Thema, damit alle abonnierten Telefonnummern die Nachricht gleichzeitig empfangen.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

Erstellen Sie ein Thema und geben Sie seinen ARN zurück.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.CreateTopicRequest;
import software.amazon.awssdk.services.sns.model.CreateTopicResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class CreateTopic {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <topicName>

                Where:
                   topicName - The name of the topic to create (for example,
 mytopic).

                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String topicName = args[0];
        System.out.println("Creating a topic with name: " + topicName);
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        String arnVal = createSNSTopic(snsClient, topicName);
        System.out.println("The topic ARN is" + arnVal);
        snsClient.close();
    }
```

```
    public static String createSNSTopic(SnsClient snsClient, String topicName) {
        CreateTopicResponse result;
        try {
            CreateTopicRequest request = CreateTopicRequest.builder()
                    .name(topicName)
                    .build();

            result = snsClient.createTopic(request);
            return result.topicArn();

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
}
```

Abonnieren eines Endpunkts für ein Thema.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SnsException;
import software.amazon.awssdk.services.sns.model.SubscribeRequest;
import software.amazon.awssdk.services.sns.model.SubscribeResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class SubscribeTextSMS {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <topicArn> <phoneNumber>

                Where:
```

```
                    topicArn - The ARN of the topic to subscribe.
                    phoneNumber - A mobile phone number that receives
 notifications (for example, +1XXX5550100).
                """;

        if (args.length < 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String topicArn = args[0];
        String phoneNumber = args[1];
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        subTextSNS(snsClient, topicArn, phoneNumber);
        snsClient.close();
    }

    public static void subTextSNS(SnsClient snsClient, String topicArn, String
 phoneNumber) {
        try {
            SubscribeRequest request = SubscribeRequest.builder()
                    .protocol("sms")
                    .endpoint(phoneNumber)
                    .returnSubscriptionArn(true)
                    .topicArn(topicArn)
                    .build();

            SubscribeResponse result = snsClient.subscribe(request);
            System.out.println("Subscription ARN: " + result.subscriptionArn() +
 "\n\n Status is "
                    + result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

Legen Sie Attribute für die Nachricht fest, z. B. die ID des Senders, den Höchstpreis und seinen Typ. Nachrichtenattribute sind optional.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SetSmsAttributesRequest;
import software.amazon.awssdk.services.sns.model.SetSmsAttributesResponse;
import software.amazon.awssdk.services.sns.model.SnsException;
import java.util.HashMap;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class SetSMSAttributes {
    public static void main(String[] args) {
        HashMap<String, String> attributes = new HashMap<>(1);
        attributes.put("DefaultSMSType", "Transactional");
        attributes.put("UsageReportS3Bucket", "janbucket");

        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();
        setSNSAttributes(snsClient, attributes);
        snsClient.close();
    }

    public static void setSNSAttributes(SnsClient snsClient, HashMap<String,
 String> attributes) {
        try {
            SetSmsAttributesRequest request = SetSmsAttributesRequest.builder()
                    .attributes(attributes)
                    .build();

            SetSmsAttributesResponse result =
 snsClient.setSMSAttributes(request);
            System.out.println("Set default Attributes to " + attributes + ".
 Status was "
```

```
                    + result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

Veröffentlichen einer Nachricht für ein Thema. Die Nachricht wird an jeden Teilnehmer gesendet.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.PublishRequest;
import software.amazon.awssdk.services.sns.model.PublishResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class PublishTextSMS {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <message> <phoneNumber>

                Where:
                   message - The message text to send.
                   phoneNumber - The mobile phone number to which a message is
 sent (for example, +1XXX5550100).\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
```

```
        }

        String message = args[0];
        String phoneNumber = args[1];
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();
        pubTextSMS(snsClient, message, phoneNumber);
        snsClient.close();
    }

    public static void pubTextSMS(SnsClient snsClient, String message, String
 phoneNumber) {
        try {
            PublishRequest request = PublishRequest.builder()
                    .message(message)
                    .phoneNumber(phoneNumber)
                    .build();

            PublishResponse result = snsClient.publish(request);
            System.out
                    .println(result.messageId() + " Message sent. Status was " +
 result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unterAmazon SNS mit einem AWS SDK verwenden. Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Veröffentlichen Sie mit Amazon S3 mithilfe eines SDK eine AWS große Nachricht in Amazon SNS

Das folgende Code-Beispiel zeigt, wie man mit Amazon S3 eine große Nachricht an Amazon SNS veröffentlicht, um die Nachrichtennutzlast zu speichern.

Java

SDK für Java 1.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

Um eine große Nachricht zu veröffentlichen, verwenden Sie die Amazon SNS Extended Client Library für Java. Die Nachricht, die Sie senden, verweist auf ein Amazon S3-Objekt, das den tatsächlichen Nachrichteninhalt enthält.

```
import com.amazon.sqs.javamessaging.AmazonSQSExtendedClient;
import com.amazon.sqs.javamessaging.ExtendedClientConfiguration;
import com.amazonaws.regions.Region;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.sns.AmazonSNS;
import com.amazonaws.services.sns.AmazonSNSClientBuilder;
import com.amazonaws.services.sns.model.CreateTopicRequest;
import com.amazonaws.services.sns.model.PublishRequest;
import com.amazonaws.services.sns.model.SetSubscriptionAttributesRequest;
import com.amazonaws.services.sns.util.Topics;
import com.amazonaws.services.sqs.AmazonSQS;
import com.amazonaws.services.sqs.AmazonSQSClientBuilder;
import com.amazonaws.services.sqs.model.CreateQueueRequest;
import com.amazonaws.services.sqs.model.ReceiveMessageResult;
import software.amazon.sns.AmazonSNSExtendedClient;
import software.amazon.sns.SNSExtendedClientConfiguration;

public class Example {

        public static void main(String[] args) {
                final String BUCKET_NAME = "extended-client-bucket";
                final String TOPIC_NAME = "extended-client-topic";
                final String QUEUE_NAME = "extended-client-queue";
                final Regions region = Regions.DEFAULT_REGION;
```

```
                // Message threshold controls the maximum message size that will
be allowed to
                // be published
                // through SNS using the extended client. Payload of messages
exceeding this
                // value will be stored in
                // S3. The default value of this parameter is 256 KB which is the
maximum
                // message size in SNS (and SQS).
                final int EXTENDED_STORAGE_MESSAGE_SIZE_THRESHOLD = 32;

                // Initialize SNS, SQS and S3 clients
                final AmazonSNS snsClient =
AmazonSNSClientBuilder.standard().withRegion(region).build();
                final AmazonSQS sqsClient =
AmazonSQSClientBuilder.standard().withRegion(region).build();
                final AmazonS3 s3Client =
AmazonS3ClientBuilder.standard().withRegion(region).build();

                // Create bucket, topic, queue and subscription
                s3Client.createBucket(BUCKET_NAME);
                final String topicArn = snsClient.createTopic(
                                new
CreateTopicRequest().withName(TOPIC_NAME)).getTopicArn();
                final String queueUrl = sqsClient.createQueue(
                                new
CreateQueueRequest().withQueueName(QUEUE_NAME)).getQueueUrl();
                final String subscriptionArn = Topics.subscribeQueue(
                                snsClient, sqsClient, topicArn, queueUrl);

                // To read message content stored in S3 transparently through SQS
extended
                // client,
                // set the RawMessageDelivery subscription attribute to TRUE
                final SetSubscriptionAttributesRequest
subscriptionAttributesRequest = new SetSubscriptionAttributesRequest();

subscriptionAttributesRequest.setSubscriptionArn(subscriptionArn);

subscriptionAttributesRequest.setAttributeName("RawMessageDelivery");
                subscriptionAttributesRequest.setAttributeValue("TRUE");

snsClient.setSubscriptionAttributes(subscriptionAttributesRequest);
```

```
                // Initialize SNS extended client
                // PayloadSizeThreshold triggers message content storage in S3
 when the
                // threshold is exceeded
                // To store all messages content in S3, use AlwaysThroughS3 flag
                final SNSExtendedClientConfiguration
 snsExtendedClientConfiguration = new SNSExtendedClientConfiguration()
                                .withPayloadSupportEnabled(s3Client, BUCKET_NAME)

 .withPayloadSizeThreshold(EXTENDED_STORAGE_MESSAGE_SIZE_THRESHOLD);
                final AmazonSNSExtendedClient snsExtendedClient = new
 AmazonSNSExtendedClient(snsClient,
                                snsExtendedClientConfiguration);

                // Publish message via SNS with storage in S3
                final String message = "This message is stored in S3 as it
 exceeds the threshold of 32 bytes set above.";
                snsExtendedClient.publish(topicArn, message);

                // Initialize SQS extended client
                final ExtendedClientConfiguration sqsExtendedClientConfiguration
 = new ExtendedClientConfiguration()
                                .withPayloadSupportEnabled(s3Client,
 BUCKET_NAME);
                final AmazonSQSExtendedClient sqsExtendedClient = new
 AmazonSQSExtendedClient(sqsClient,
                                sqsExtendedClientConfiguration);

                // Read the message from the queue
                final ReceiveMessageResult result =
 sqsExtendedClient.receiveMessage(queueUrl);
                System.out.println("Received message is " +
 result.getMessages().get(0).getBody());
        }
}
```

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unterAmazon SNS mit einem AWS SDK verwenden. Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Veröffentlichen Sie eine Amazon SNS SNS-SMS-Textnachricht mit einem SDK AWS

In den folgenden Codebeispielen wird veranschaulicht, wie Sie SMS-Nachrichten über Amazon SNS veröffentlichen.

.NET

AWS SDK for .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

```
namespace SNSMessageExample
{
    using System;
    using System.Threading.Tasks;
    using Amazon;
    using Amazon.SimpleNotificationService;
    using Amazon.SimpleNotificationService.Model;

    public class SNSMessage
    {
        private AmazonSimpleNotificationServiceClient snsClient;

        /// <summary>
        /// Initializes a new instance of the <see cref="SNSMessage"/> class.
        /// Constructs a new SNSMessage object initializing the Amazon Simple
        /// Notification Service (Amazon SNS) client using the supplied
        /// Region endpoint.
        /// </summary>
        /// <param name="regionEndpoint">The Amazon Region endpoint to use in
        /// sending test messages with this object.</param>
        public SNSMessage(RegionEndpoint regionEndpoint)
        {
            snsClient = new
 AmazonSimpleNotificationServiceClient(regionEndpoint);
        }
```

```
        /// <summary>
        /// Sends the SMS message passed in the text parameter to the phone
 number
        /// in phoneNum.
        /// </summary>
        /// <param name="phoneNum">The ten-digit phone number to which the text
        /// message will be sent.</param>
        /// <param name="text">The text of the message to send.</param>
        /// <returns>Async task.</returns>
        public async Task SendTextMessageAsync(string phoneNum, string text)
        {
            if (string.IsNullOrEmpty(phoneNum) || string.IsNullOrEmpty(text))
            {
                return;
            }

            // Now actually send the message.
            var request = new PublishRequest
            {
                Message = text,
                PhoneNumber = phoneNum,
            };

            try
            {
                var response = await snsClient.PublishAsync(request);
            }
            catch (Exception ex)
            {
                Console.WriteLine($"Error sending message: {ex}");
            }
        }
    }
}
```

- Details zu API finden Sie unter Veröffentlichen in der AWS SDK for .NET -API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

```
/**
 * Publish SMS: use Amazon Simple Notification Service (Amazon SNS) to send an
 SMS text message to a phone number.
 * Note: This requires additional AWS configuration prior to running example.
 *
 *  NOTE: When you start using Amazon SNS to send SMS messages, your AWS account
 is in the SMS sandbox and you can only
 *  use verified destination phone numbers. See https://docs.aws.amazon.com/sns/
latest/dg/sns-sms-sandbox.html.
 *  NOTE: If destination is in the US, you also have an additional restriction
 that you have use a dedicated
 *  origination ID (phone number). You can request an origination number using
 Amazon Pinpoint for a fee.
 *  See https://aws.amazon.com/blogs/compute/provisioning-and-using-10dlc-
origination-numbers-with-amazon-sns/
 *  for more information.
 *
 *  <phone_number_value> input parameter uses E.164 format.
 *  For example, in United States, this input value should be of the form:
 +12223334444
 */

//! Send an SMS text message to a phone number.
/*!
  \param message: The message to publish.
  \param phoneNumber: The phone number of the recipient in E.164 format.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SNS::publishSms(const Aws::String &message,
                             const Aws::String &phoneNumber,
```

```
                                const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::SNS::Model::PublishRequest request;
    request.SetMessage(message);
    request.SetPhoneNumber(phoneNumber);

    const Aws::SNS::Model::PublishOutcome outcome = snsClient.Publish(request);

    if (outcome.IsSuccess()) {
        std::cout << "Message published successfully with message id, '"
                  << outcome.GetResult().GetMessageId() << "'."
                  << std::endl;
    }
    else {
        std::cerr << "Error while publishing message "
                  << outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Details zu API finden Sie unter Veröffentlichen in der AWS SDK for C++ -API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.PublishRequest;
import software.amazon.awssdk.services.sns.model.PublishResponse;
import software.amazon.awssdk.services.sns.model.SnsException;
```

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class PublishTextSMS {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <message> <phoneNumber>

                Where:
                   message - The message text to send.
                   phoneNumber - The mobile phone number to which a message is
 sent (for example, +1XXX5550100).\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String message = args[0];
        String phoneNumber = args[1];
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();
        pubTextSMS(snsClient, message, phoneNumber);
        snsClient.close();
    }

    public static void pubTextSMS(SnsClient snsClient, String message, String
 phoneNumber) {
        try {
            PublishRequest request = PublishRequest.builder()
                    .message(message)
                    .phoneNumber(phoneNumber)
                    .build();
```

```
            PublishResponse result = snsClient.publish(request);
            System.out
                    .println(result.messageId() + " Message sent. Status was " +
 result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Details zu API finden Sie unter Veröffentlichen in der AWS SDK for Java 2.x -API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

```
suspend fun pubTextSMS(messageVal: String?, phoneNumberVal: String?) {

    val request = PublishRequest {
        message = messageVal
        phoneNumber = phoneNumberVal
    }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.publish(request)
        println("${result.messageId} message sent.")
    }
}
```

- Details zu API finden Sie unter Publish (Veröffentlichen) in der AWS -SDK-für-Kotlin-API-Referenz.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Sends a text message (SMS message) directly to a phone number using Amazon
 SNS.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/
guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$message = 'This message is sent from a Amazon SNS code sample.';
$phone = '+1XXX5550100';

try {
    $result = $SnSclient->publish([
        'Message' => $message,
        'PhoneNumber' => $phone,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
```

```
    error_log($e->getMessage());
}
```

- Weitere Informationen finden Sie im AWS SDK for PHP -Entwicklerhandbuch.
- Details zu API finden Sie unter Veröffentlichen in der AWS SDK for PHP -API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

```
class SnsWrapper:
    """Encapsulates Amazon SNS topic and subscription functions."""

    def __init__(self, sns_resource):
        """
        :param sns_resource: A Boto3 Amazon SNS resource.
        """
        self.sns_resource = sns_resource


    def publish_text_message(self, phone_number, message):
        """
        Publishes a text message directly to a phone number without need for a
        subscription.

        :param phone_number: The phone number that receives the message. This
 must be
                             in E.164 format. For example, a United States phone
                             number might be +12065550101.
        :param message: The message to send.
        :return: The ID of the message.
        """
        try:
```

```
            response = self.sns_resource.meta.client.publish(
                PhoneNumber=phone_number, Message=message
            )
            message_id = response["MessageId"]
            logger.info("Published message to %s.", phone_number)
        except ClientError:
            logger.exception("Couldn't publish message to %s.", phone_number)
            raise
        else:
            return message_id
```

- Details zu API finden Sie unter Veröffentlichen in der AWS -API-Referenz zu SDK for Python (Boto3).

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unterAmazon SNS mit einem AWS SDK verwenden. Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Veröffentlichen Sie Amazon SNS SNS-Nachrichten mithilfe eines SDK in Amazon SQS SQS-Warteschlangen AWS

Die folgenden Code-Beispiele veranschaulichen Folgendes:

- Erstellen Sie ein Thema (FIFO oder Nicht-FIFO).
- Abonnieren Sie mehrere Warteschlangen für das Thema mit der Option, einen Filter anzuwenden.
- Veröffentlichen Sie eine Nachricht im Thema.
- Fragen Sie die Warteschlangen nach empfangenen Nachrichten ab.

.NET

AWS SDK for .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

Führen Sie ein interaktives Szenario an einer Eingabeaufforderung aus.

```
/// <summary>
/// Console application to run a workflow scenario for topics and queues.
/// </summary>
public static class TopicsAndQueues
{
    private static bool _useFifoTopic = false;
    private static bool _useContentBasedDeduplication = false;
    private static string _topicName = null!;
    private static string _topicArn = null!;

    private static readonly int _queueCount = 2;
    private static readonly string[] _queueUrls = new string[_queueCount];
    private static readonly string[] _subscriptionArns = new string[_queueCount];
    private static readonly string[] _tones = { "cheerful", "funny", "serious",
 "sincere" };
    public static SNSWrapper SnsWrapper { get; set; } = null!;
    public static SQSWrapper SqsWrapper { get; set; } = null!;
    public static bool UseConsole { get; set; } = true;
    static async Task Main(string[] args)
    {
        // Set up dependency injection for Amazon EventBridge.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
                logging.AddFilter("System", LogLevel.Debug)
                    .AddFilter<DebugLoggerProvider>("Microsoft",
 LogLevel.Information)
                    .AddFilter<ConsoleLoggerProvider>("Microsoft",
 LogLevel.Trace))
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonSQS>()
                    .AddAWSService<IAmazonSimpleNotificationService>()
                    .AddTransient<SNSWrapper>()
                    .AddTransient<SQSWrapper>()
            )
            .Build();

        ServicesSetup(host);
        PrintDescription();

        await RunScenario();

    }
```

```
    /// <summary>
    /// Populate the services for use within the console application.
    /// </summary>
    /// <param name="host">The services host.</param>
    private static void ServicesSetup(IHost host)
    {
        SnsWrapper = host.Services.GetRequiredService<SNSWrapper>();
        SqsWrapper = host.Services.GetRequiredService<SQSWrapper>();
    }

    /// <summary>
    /// Run the scenario for working with topics and queues.
    /// </summary>
    /// <returns>True if successful.</returns>
    public static async Task<bool> RunScenario()
    {
        try
        {
            await SetupTopic();

            await SetupQueues();

            await PublishMessages();

            foreach (var queueUrl in _queueUrls)
            {
                var messages = await PollForMessages(queueUrl);
                if (messages.Any())
                {
                    await DeleteMessages(queueUrl, messages);
                }
            }
            await CleanupResources();

            Console.WriteLine("Messaging with topics and queues workflow is
complete.");
            return true;
        }
        catch (Exception ex)
        {
            Console.WriteLine(new string('-', 80));
            Console.WriteLine($"There was a problem running the scenario:
{ex.Message}");
```

```
            await CleanupResources();
            Console.WriteLine(new string('-', 80));
            return false;
        }
    }

    /// <summary>
    /// Print a description for the tasks in the workflow.
    /// </summary>
    /// <returns>Async task.</returns>
    private static void PrintDescription()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"Welcome to messaging with topics and queues.");

        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"In this workflow, you will create an SNS topic and
 subscribe {_queueCount} SQS queues to the topic." +
                          $"\r\nYou can select from several options for
 configuring the topic and the subscriptions for the 2 queues." +
                          $"\r\nYou can then post to the topic and see the
 results in the queues.\r\n");

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Set up the SNS topic to be used with the queues.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task<string> SetupTopic()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"SNS topics can be configured as FIFO (First-In-First-
Out)." +
                          $"\r\nFIFO topics deliver messages in order and support
 deduplication and message filtering." +
                          $"\r\nYou can then post to the topic and see the
 results in the queues.\r\n");

        _useFifoTopic = GetYesNoResponse("Would you like to work with FIFO
 topics?");

        if (_useFifoTopic)
```

```
        {
            Console.WriteLine(new string('-', 80));
            _topicName = GetUserResponse("Enter a name for your SNS topic: ",
 "example-topic");
            Console.WriteLine(
                "Because you have selected a FIFO topic, '.fifo' must be appended
 to the topic name.\r\n");

            Console.WriteLine(new string('-', 80));
            Console.WriteLine($"Because you have chosen a FIFO topic,
 deduplication is supported." +
                              $"\r\nDeduplication IDs are either set in the
 message or automatically generated " +
                              $"\r\nfrom content using a hash function.\r\n" +
                              $"\r\nIf a message is successfully published to an
 SNS FIFO topic, any message " +
                              $"\r\npublished and determined to have the same
 deduplication ID, " +
                              $"\r\nwithin the five-minute deduplication
 interval, is accepted but not delivered.\r\n" +
                              $"\r\nFor more information about deduplication, " +
                              $"\r\nsee https://docs.aws.amazon.com/sns/latest/
dg/fifo-message-dedup.html.");

            _useContentBasedDeduplication = GetYesNoResponse("Use content-based
 deduplication instead of entering a deduplication ID?");
            Console.WriteLine(new string('-', 80));
        }

        _topicArn = await SnsWrapper.CreateTopicWithName(_topicName,
 _useFifoTopic, _useContentBasedDeduplication);

        Console.WriteLine($"Your new topic with the name {_topicName}" +
                          $"\r\nand Amazon Resource Name (ARN) {_topicArn}" +
                          $"\r\nhas been created.\r\n");

        Console.WriteLine(new string('-', 80));
        return _topicArn;
    }

    /// <summary>
    /// Set up the queues.
    /// </summary>
    /// <returns>Async task.</returns>
```

```
    private static async Task SetupQueues()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"Now you will create {_queueCount} Amazon Simple Queue
Service (Amazon SQS) queues to subscribe to the topic.");

        // Repeat this section for each queue.
        for (int i = 0; i < _queueCount; i++)
        {
            var queueName = GetUserResponse("Enter a name for an Amazon SQS
queue: ", $"example-queue-{i}");
            if (_useFifoTopic)
            {
                // Only explain this once.
                if (i == 0)
                {
                    Console.WriteLine(
                        "Because you have selected a FIFO topic, '.fifo' must be
appended to the queue name.");
                }

                var queueUrl = await SqsWrapper.CreateQueueWithName(queueName,
_useFifoTopic);

                _queueUrls[i] = queueUrl;

                Console.WriteLine($"Your new queue with the name {queueName}" +
                                  $"\r\nand queue URL {queueUrl}" +
                                  $"\r\nhas been created.\r\n");

                if (i == 0)
                {
                    Console.WriteLine(
                        $"The queue URL is used to retrieve the queue ARN,\r\n" +
                        $"which is used to create a subscription.");
                    Console.WriteLine(new string('-', 80));
                }

                var queueArn = await SqsWrapper.GetQueueArnByUrl(queueUrl);

                if (i == 0)
                {
                    Console.WriteLine(
```

```
                        $"An AWS Identity and Access Management (IAM) policy must
 be attached to an SQS queue, enabling it to receive\r\n" +
                        $"messages from an SNS topic");
                }

                await SqsWrapper.SetQueuePolicyForTopic(queueArn, _topicArn,
 queueUrl);

                await SetupFilters(i, queueArn, queueName);
            }
        }

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Set up filters with user options for a queue.
    /// </summary>
    /// <param name="queueCount">The number of this queue.</param>
    /// <param name="queueArn">The ARN of the queue.</param>
    /// <param name="queueName">The name of the queue.</param>
    /// <returns>Async Task.</returns>
    public static async Task SetupFilters(int queueCount, string queueArn, string
 queueName)
    {
        if (_useFifoTopic)
        {
            Console.WriteLine(new string('-', 80));
            // Only explain this once.
            if (queueCount == 0)
            {
                Console.WriteLine(
                    "Subscriptions to a FIFO topic can have filters." +
                    "If you add a filter to this subscription, then only the
 filtered messages " +
                    "will be received in the queue.");

                Console.WriteLine(
                    "For information about message filtering, " +
                    "see https://docs.aws.amazon.com/sns/latest/dg/sns-message-
filtering.html");

                Console.WriteLine(
                    "For this example, you can filter messages by a" +
```

```
                    "TONE attribute.");
            }

            var useFilter = GetYesNoResponse($"Filter messages for {queueName}'s
subscription to the topic?");

            string? filterPolicy = null;
            if (useFilter)
            {
                filterPolicy = CreateFilterPolicy();
            }
            var subscriptionArn = await
SnsWrapper.SubscribeTopicWithFilter(_topicArn, filterPolicy,
                queueArn);
            _subscriptionArns[queueCount] = subscriptionArn;

            Console.WriteLine(
                $"The queue {queueName} has been subscribed to the topic
{_topicName} " +
                $"with the subscription ARN {subscriptionArn}");
            Console.WriteLine(new string('-', 80));
        }
    }

    /// <summary>
    /// Use user input to create a filter policy for a subscription.
    /// </summary>
    /// <returns>The serialized filter policy.</returns>
    public static string CreateFilterPolicy()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine(
            $"You can filter messages by one or more of the following" +
            $"TONE attributes.");

        List<string> filterSelections = new List<string>();

        var selectionNumber = 0;
        do
        {
            Console.WriteLine(
                $"Enter a number to add a TONE filter, or enter 0 to stop adding
filters.");
            for (int i = 0; i < _tones.Length; i++)
```

```
            {
                Console.WriteLine($"\t{i + 1}. {_tones[i]}");
            }

            var selection = GetUserResponse("", filterSelections.Any() ? "0" :
 "1");
            int.TryParse(selection, out selectionNumber);
            if (selectionNumber > 0 && !
filterSelections.Contains(_tones[selectionNumber - 1]))
            {
                filterSelections.Add(_tones[selectionNumber - 1]);
            }
        } while (selectionNumber != 0);

        var filters = new Dictionary<string, List<string>>
        {
            { "tone", filterSelections }
        };
        string filterPolicy = JsonSerializer.Serialize(filters);
        return filterPolicy;
    }

    /// <summary>
    /// Publish messages using user settings.
    /// </summary>
    /// <returns>Async task.</returns>
    public static async Task PublishMessages()
    {
        Console.WriteLine("Now we can publish messages.");

        var keepSendingMessages = true;
        string? deduplicationId = null;
        string? toneAttribute = null;
        while (keepSendingMessages)
        {
            Console.WriteLine();
            var message = GetUserResponse("Enter a message to publish.", "This is
 a sample message");

            if (_useFifoTopic)
            {
                Console.WriteLine("Because you are using a FIFO topic, you must
 set a message group ID." +
```

```
                                "\r\nAll messages within the same group will be
received in the order " +
                                "they were published.");

            Console.WriteLine();
            var messageGroupId = GetUserResponse("Enter a message group ID
for this message:", "1");

            if (!_useContentBasedDeduplication)
            {
                Console.WriteLine("Because you are not using content-based
deduplication, " +
                                  "you must enter a deduplication ID.");

                Console.WriteLine("Enter a deduplication ID for this
message.");
                deduplicationId = GetUserResponse("Enter a deduplication ID
for this message.", "1");
            }

            if (GetYesNoResponse("Add an attribute to this message?"))
            {
                Console.WriteLine("Enter a number for an attribute.");
                for (int i = 0; i < _tones.Length; i++)
                {
                    Console.WriteLine($"\t{i + 1}. {_tones[i]}");
                }

                var selection = GetUserResponse("", "1");
                int.TryParse(selection, out var selectionNumber);

                if (selectionNumber > 0 && selectionNumber < _tones.Length)
                {
                    toneAttribute = _tones[selectionNumber - 1];
                }
            }

            var messageID = await SnsWrapper.PublishToTopicWithAttribute(
                _topicArn, message, "tone", toneAttribute, deduplicationId,
messageGroupId);

            Console.WriteLine($"Message published with id {messageID}.");
        }
```

```
            keepSendingMessages = GetYesNoResponse("Send another message?",
false);
        }
    }

    /// <summary>
    /// Poll for the published messages to see the results of the user's choices.
    /// </summary>
    /// <returns>Async task.</returns>
    public static async Task<List<Message>> PollForMessages(string queueUrl)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"Now the SQS queue at {queueUrl} will be polled to
retrieve the messages." +
                          "\r\nPress any key to continue.");
        if (UseConsole)
        {
            Console.ReadLine();
        }

        var moreMessages = true;
        var messages = new List<Message>();
        while (moreMessages)
        {
            var newMessages = await SqsWrapper.ReceiveMessagesByUrl(queueUrl,
10);

            moreMessages = newMessages.Any();
            if (moreMessages)
            {
                messages.AddRange(newMessages);
            }
        }

        Console.WriteLine($"{messages.Count} message(s) were received by the
queue at {queueUrl}.");

        foreach (var message in messages)
        {
            Console.WriteLine("\tMessage:" +
                              $"\n\t{message.Body}");
        }

        Console.WriteLine(new string('-', 80));
```

```
        return messages;
    }

    /// <summary>
    /// Delete the message using handles in a batch.
    /// </summary>
    /// <returns>Async task.</returns>
    public static async Task DeleteMessages(string queueUrl, List<Message>
messages)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Now we can delete the messages in this queue in a
batch.");
        await SqsWrapper.DeleteMessageBatchByUrl(queueUrl, messages);
        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Clean up the resources from the scenario.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task CleanupResources()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"Clean up resources.");

        try
        {
            foreach (var queueUrl in _queueUrls)
            {
                if (!string.IsNullOrEmpty(queueUrl))
                {
                    var deleteQueue =
                        GetYesNoResponse($"Delete queue with url {queueUrl}?");
                    if (deleteQueue)
                    {
                        await SqsWrapper.DeleteQueueByUrl(queueUrl);
                    }
                }
            }

            foreach (var subscriptionArn in _subscriptionArns)
            {
                if (!string.IsNullOrEmpty(subscriptionArn))
```

```
                {
                    await SnsWrapper.UnsubscribeByArn(subscriptionArn);
                }
            }

            var deleteTopic = GetYesNoResponse($"Delete topic {_topicName}?");
            if (deleteTopic)
            {
                await SnsWrapper.DeleteTopicByArn(_topicArn);
            }
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Unable to clean up resources. Here's why:
 {ex.Message}.");
        }

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Helper method to get a yes or no response from the user.
    /// </summary>
    /// <param name="question">The question string to print on the console.</
param>
    /// <param name="defaultAnswer">Optional default answer to use.</param>
    /// <returns>True if the user responds with a yes.</returns>
    private static bool GetYesNoResponse(string question, bool defaultAnswer =
 true)
    {
        if (UseConsole)
        {
            Console.WriteLine(question);
            var ynResponse = Console.ReadLine();
            var response = ynResponse != null &&
                           ynResponse.Equals("y",
                               StringComparison.InvariantCultureIgnoreCase);
            return response;
        }
        // If not using the console, use the default.
        return defaultAnswer;
    }

    /// <summary>
```

```
    /// Helper method to get a string response from the user through the console.
    /// </summary>
    /// <param name="question">The question string to print on the console.</
param>
    /// <param name="defaultAnswer">Optional default answer to use.</param>
    /// <returns>True if the user responds with a yes.</returns>
    private static string GetUserResponse(string question, string defaultAnswer)
    {
        if (UseConsole)
        {
            var response = "";
            while (string.IsNullOrEmpty(response))
            {
                Console.WriteLine(question);
                response = Console.ReadLine();
            }
            return response;
        }
        // If not using the console, use the default.
        return defaultAnswer;
    }
}
```

Erstellen Sie eine Klasse, die Amazon–SQS-Operationen einschließt.

```
/// <summary>
/// Wrapper for Amazon Simple Queue Service (SQS) operations.
/// </summary>
public class SQSWrapper
{
    private readonly IAmazonSQS _amazonSQSClient;

    /// <summary>
    /// Constructor for the Amazon SQS wrapper.
    /// </summary>
    /// <param name="amazonSQS">The injected Amazon SQS client.</param>
    public SQSWrapper(IAmazonSQS amazonSQS)
    {
        _amazonSQSClient = amazonSQS;
    }
```

```
    /// <summary>
    /// Create a queue with a specific name.
    /// </summary>
    /// <param name="queueName">The name for the queue.</param>
    /// <param name="useFifoQueue">True to use a FIFO queue.</param>
    /// <returns>The url for the queue.</returns>
    public async Task<string> CreateQueueWithName(string queueName, bool
useFifoQueue)
    {
        int maxMessage = 256 * 1024;
        var queueAttributes = new Dictionary<string, string>
        {
            {
                QueueAttributeName.MaximumMessageSize,
                maxMessage.ToString()
            }
        };

        var createQueueRequest = new CreateQueueRequest()
        {
            QueueName = queueName,
            Attributes = queueAttributes
        };

        if (useFifoQueue)
        {
            // Update the name if it is not correct for a FIFO queue.
            if (!queueName.EndsWith(".fifo"))
            {
                createQueueRequest.QueueName = queueName + ".fifo";
            }

            // Add an attribute for a FIFO queue.
            createQueueRequest.Attributes.Add(
                QueueAttributeName.FifoQueue, "true");
        }

        var createResponse = await _amazonSQSClient.CreateQueueAsync(
            new CreateQueueRequest()
            {
                QueueName = queueName
            });
        return createResponse.QueueUrl;
    }
```

```
   /// <summary>
   /// Get the ARN for a queue from its URL.
   /// </summary>
   /// <param name="queueUrl">The URL of the queue.</param>
   /// <returns>The ARN of the queue.</returns>
   public async Task<string> GetQueueArnByUrl(string queueUrl)
   {
       var getAttributesRequest = new GetQueueAttributesRequest()
       {
           QueueUrl = queueUrl,
           AttributeNames = new List<string>() { QueueAttributeName.QueueArn }
       };

       var getAttributesResponse = await
_amazonSQSClient.GetQueueAttributesAsync(
           getAttributesRequest);

       return getAttributesResponse.QueueARN;
   }

   /// <summary>
   /// Set the policy attribute of a queue for a topic.
   /// </summary>
   /// <param name="queueArn">The ARN of the queue.</param>
   /// <param name="topicArn">The ARN of the topic.</param>
   /// <param name="queueUrl">The url for the queue.</param>
   /// <returns>True if successful.</returns>
   public async Task<bool> SetQueuePolicyForTopic(string queueArn, string
topicArn, string queueUrl)
   {
       var queuePolicy = "{" +
                                "\"Version\": \"2012-10-17\"," +
                                "\"Statement\": [{" +
                                     "\"Effect\": \"Allow\"," +
                                     "\"Principal\": {" +
                                         $"\"Service\": " +
                                             "\"sns.amazonaws.com\"" +
                                            "}," +
                                     "\"Action\": \"sqs:SendMessage\"," +
                                     $"\"Resource\": \"{queueArn}\"," +
                                      "\"Condition\": {" +
                                           "\"ArnEquals\": {" +
```

```
                                                $"\"aws:SourceArn\":
 \"{topicArn}\"" +
                                            "}" +
                                        "}" +
                                    "}]" +
                                "}";
        var attributesResponse = await _amazonSQSClient.SetQueueAttributesAsync(
            new SetQueueAttributesRequest()
            {
                QueueUrl = queueUrl,
                Attributes = new Dictionary<string, string>() { { "Policy",
 queuePolicy } }
            });
        return attributesResponse.HttpStatusCode == HttpStatusCode.OK;
    }

    /// <summary>
    /// Receive messages from a queue by its URL.
    /// </summary>
    /// <param name="queueUrl">The url of the queue.</param>
    /// <returns>The list of messages.</returns>
    public async Task<List<Message>> ReceiveMessagesByUrl(string queueUrl, int
 maxMessages)
    {
        // Setting WaitTimeSeconds to non-zero enables long polling.
        // For information about long polling, see
        // https://docs.aws.amazon.com/AWSSimpleQueueService/latest/
SQSDeveloperGuide/sqs-short-and-long-polling.html
        var messageResponse = await _amazonSQSClient.ReceiveMessageAsync(
            new ReceiveMessageRequest()
            {
                QueueUrl = queueUrl,
                MaxNumberOfMessages = maxMessages,
                WaitTimeSeconds = 1
            });
        return messageResponse.Messages;
    }

    /// <summary>
    /// Delete a batch of messages from a queue by its url.
    /// </summary>
    /// <param name="queueUrl">The url of the queue.</param>
    /// <returns>True if successful.</returns>
```

```
    public async Task<bool> DeleteMessageBatchByUrl(string queueUrl,
 List<Message> messages)
    {
        var deleteRequest = new DeleteMessageBatchRequest()
        {
            QueueUrl = queueUrl,
            Entries = new List<DeleteMessageBatchRequestEntry>()
        };
        foreach (var message in messages)
        {
            deleteRequest.Entries.Add(new DeleteMessageBatchRequestEntry()
            {
                ReceiptHandle = message.ReceiptHandle,
                Id = message.MessageId
            });
        }

        var deleteResponse = await
 _amazonSQSClient.DeleteMessageBatchAsync(deleteRequest);

        return deleteResponse.Failed.Any();
    }

    /// <summary>
    /// Delete a queue by its URL.
    /// </summary>
    /// <param name="queueUrl">The url of the queue.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteQueueByUrl(string queueUrl)
    {
        var deleteResponse = await _amazonSQSClient.DeleteQueueAsync(
            new DeleteQueueRequest()
            {
                QueueUrl = queueUrl
            });
        return deleteResponse.HttpStatusCode == HttpStatusCode.OK;
    }
}
```

Erstellen Sie eine Klasse, die Amazon–SNS-Operationen einschließt.

```
/// <summary>
/// Wrapper for Amazon Simple Notification Service (SNS) operations.
/// </summary>
public class SNSWrapper
{
    private readonly IAmazonSimpleNotificationService _amazonSNSClient;

    /// <summary>
    /// Constructor for the Amazon SNS wrapper.
    /// </summary>
    /// <param name="amazonSQS">The injected Amazon SNS client.</param>
    public SNSWrapper(IAmazonSimpleNotificationService amazonSNS)
    {
        _amazonSNSClient = amazonSNS;
    }

    /// <summary>
    /// Create a new topic with a name and specific FIFO and de-duplication
 attributes.
    /// </summary>
    /// <param name="topicName">The name for the topic.</param>
    /// <param name="useFifoTopic">True to use a FIFO topic.</param>
    /// <param name="useContentBasedDeduplication">True to use content-based de-
duplication.</param>
    /// <returns>The ARN of the new topic.</returns>
    public async Task<string> CreateTopicWithName(string topicName, bool
 useFifoTopic, bool useContentBasedDeduplication)
    {
        var createTopicRequest = new CreateTopicRequest()
        {
            Name = topicName,
        };

        if (useFifoTopic)
        {
            // Update the name if it is not correct for a FIFO topic.
            if (!topicName.EndsWith(".fifo"))
            {
                createTopicRequest.Name = topicName + ".fifo";
            }

            // Add the attributes from the method parameters.
            createTopicRequest.Attributes = new Dictionary<string, string>
            {
```

```
                { "FifoTopic", "true" }
            };
            if (useContentBasedDeduplication)
            {
                createTopicRequest.Attributes.Add("ContentBasedDeduplication",
"true");
            }
        }

        var createResponse = await
_amazonSNSClient.CreateTopicAsync(createTopicRequest);
        return createResponse.TopicArn;
    }

    /// <summary>
    /// Subscribe a queue to a topic with optional filters.
    /// </summary>
    /// <param name="topicArn">The ARN of the topic.</param>
    /// <param name="useFifoTopic">The optional filtering policy for the
subscription.</param>
    /// <param name="queueArn">The ARN of the queue.</param>
    /// <returns>The ARN of the new subscription.</returns>
    public async Task<string> SubscribeTopicWithFilter(string topicArn, string?
filterPolicy, string queueArn)
    {
        var subscribeRequest = new SubscribeRequest()
        {
            TopicArn = topicArn,
            Protocol = "sqs",
            Endpoint = queueArn
        };

        if (!string.IsNullOrEmpty(filterPolicy))
        {
            subscribeRequest.Attributes = new Dictionary<string, string>
{ { "FilterPolicy", filterPolicy } };
        }

        var subscribeResponse = await
_amazonSNSClient.SubscribeAsync(subscribeRequest);
        return subscribeResponse.SubscriptionArn;
    }

    /// <summary>
```

```
    /// Publish a message to a topic with an attribute and optional deduplication
 and group IDs.
    /// </summary>
    /// <param name="topicArn">The ARN of the topic.</param>
    /// <param name="message">The message to publish.</param>
    /// <param name="attributeName">The optional attribute for the message.</
param>
    /// <param name="attributeValue">The optional attribute value for the
 message.</param>
    /// <param name="deduplicationId">The optional deduplication ID for the
 message.</param>
    /// <param name="groupId">The optional group ID for the message.</param>
    /// <returns>The ID of the message published.</returns>
    public async Task<string> PublishToTopicWithAttribute(
        string topicArn,
        string message,
        string? attributeName = null,
        string? attributeValue = null,
        string? deduplicationId = null,
        string? groupId = null)
    {
        var publishRequest = new PublishRequest()
        {
            TopicArn = topicArn,
            Message = message,
            MessageDeduplicationId = deduplicationId,
            MessageGroupId = groupId
        };

        if (attributeValue != null)
        {
            // Add the string attribute if it exists.
            publishRequest.MessageAttributes =
                new Dictionary<string, MessageAttributeValue>
                {
                    { attributeName!, new MessageAttributeValue() { StringValue =
 attributeValue, DataType = "String"} }
                };
        }

        var publishResponse = await
 _amazonSNSClient.PublishAsync(publishRequest);
        return publishResponse.MessageId;
    }
```

```
    /// <summary>
    /// Unsubscribe from a topic by a subscription ARN.
    /// </summary>
    /// <param name="subscriptionArn">The ARN of the subscription.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> UnsubscribeByArn(string subscriptionArn)
    {
        var unsubscribeResponse = await _amazonSNSClient.UnsubscribeAsync(
            new UnsubscribeRequest()
            {
                SubscriptionArn = subscriptionArn
            });
        return unsubscribeResponse.HttpStatusCode == HttpStatusCode.OK;
    }

    /// <summary>
    /// Delete a topic by its topic ARN.
    /// </summary>
    /// <param name="topicArn">The ARN of the topic.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteTopicByArn(string topicArn)
    {
        var deleteResponse = await _amazonSNSClient.DeleteTopicAsync(
            new DeleteTopicRequest()
            {
                TopicArn = topicArn
            });
        return deleteResponse.HttpStatusCode == HttpStatusCode.OK;
    }
}
```

- API-Details finden Sie in den folgenden Themen der AWS SDK for .NET -API-Referenz.
  - CreateQueue
  - CreateTopic
  - DeleteMessageBatch
  - DeleteQueue
  - DeleteTopic
  - GetQueueAttributes

- Veröffentlichen
- ReceiveMessage
- SetQueueAttributes
- Abonnieren
- Unsubscribe

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

//! Workflow for messaging with topics and queues using Amazon SNS and Amazon
 SQS.
/*!
 \param clientConfig Aws client configuration.
 \return bool: Successful completion.
 */
bool AwsDoc::TopicsAndQueues::messagingWithTopicsAndQueues(
        const Aws::Client::ClientConfiguration &clientConfiguration) {
    std::cout << "Welcome to messaging with topics and queues." << std::endl;
    printAsterisksLine();
    std::cout << "In this workflow, you will create an SNS topic and subscribe "
              << NUMBER_OF_QUEUES <<
              " SQS queues to the topic." << std::endl;
    std::cout
            << "You can select from several options for configuring the topic and
 the subscriptions for the "
            << NUMBER_OF_QUEUES << " queues." << std::endl;
    std::cout << "You can then post to the topic and see the results in the
 queues."
              << std::endl;
```

```
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    printAsterisksLine();

    std::cout << "SNS topics can be configured as FIFO (First-In-First-Out)."
              << std::endl;
    std::cout
            << "FIFO topics deliver messages in order and support deduplication
 and message filtering."
            << std::endl;
    bool isFifoTopic = askYesNoQuestion(
            "Would you like to work with FIFO topics? (y/n) ");

    bool contentBasedDeduplication = false;
    Aws::String topicName;
    if (isFifoTopic) {
        printAsterisksLine();
        std::cout << "Because you have chosen a FIFO topic, deduplication is
 supported."
                  << std::endl;
        std::cout
                << "Deduplication IDs are either set in the message or
 automatically generated "
                << "from content using a hash function." << std::endl;
        std::cout
                << "If a message is successfully published to an SNS FIFO topic,
 any message "
                << "published and determined to have the same deduplication ID, "
                << std::endl;
        std::cout
                << "within the five-minute deduplication interval, is accepted
 but not delivered."
                << std::endl;
        std::cout
                << "For more information about deduplication, "
                << "see https://docs.aws.amazon.com/sns/latest/dg/fifo-message-
dedup.html."
                << std::endl;
        contentBasedDeduplication = askYesNoQuestion(
                "Use content-based deduplication instead of entering a
 deduplication ID? (y/n) ");
    }
```

```
    printAsterisksLine();

    Aws::SQS::SQSClient sqsClient(clientConfiguration);
    Aws::Vector<Aws::String> queueURLS;
    Aws::Vector<Aws::String> subscriptionARNS;

    Aws::String topicARN;
    {
        topicName = askQuestion("Enter a name for your SNS topic. ");

        // 1.  Create an Amazon SNS topic, either FIFO or non-FIFO.
        Aws::SNS::Model::CreateTopicRequest request;

        if (isFifoTopic) {
            request.AddAttributes("FifoTopic", "true");
            if (contentBasedDeduplication) {
                request.AddAttributes("ContentBasedDeduplication", "true");
            }
            topicName = topicName + FIFO_SUFFIX;

            std::cout
                    << "Because you have selected a FIFO topic, '.fifo' must be
appended to the topic name."
                    << std::endl;
        }

        request.SetName(topicName);

        Aws::SNS::Model::CreateTopicOutcome outcome =
snsClient.CreateTopic(request);

        if (outcome.IsSuccess()) {
            topicARN = outcome.GetResult().GetTopicArn();
            std::cout << "Your new topic with the name '" << topicName
                      << "' and the topic Amazon Resource Name (ARN) " <<
std::endl;
            std::cout << "'" << topicARN << "' has been created." << std::endl;

        }
        else {
            std::cerr << "Error with TopicsAndQueues::CreateTopic. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
```

```
            cleanUp(topicARN,
                    queueURLS,
                    subscriptionARNS,
                    snsClient,
                    sqsClient);

            return false;
        }
    }

    printAsterisksLine();

    std::cout << "Now you will create " << NUMBER_OF_QUEUES
              << " SQS queues to subscribe to the topic." << std::endl;
    Aws::Vector<Aws::String> queueNames;
    bool filteringMessages = false;
    bool first = true;
    for (int i = 1; i <= NUMBER_OF_QUEUES; ++i) {
        Aws::String queueURL;
        Aws::String queueName;
        {
            printAsterisksLine();
            std::ostringstream ostringstream;
            ostringstream << "Enter a name for " << (first ? "an" : "the next")
                          << " SQS queue. ";
            queueName = askQuestion(ostringstream.str());

            // 2.  Create an SQS queue.
            Aws::SQS::Model::CreateQueueRequest request;
            if (isFifoTopic) {

request.AddAttributes(Aws::SQS::Model::QueueAttributeName::FifoQueue,
                                      "true");
                queueName = queueName + FIFO_SUFFIX;

                if (first) // Only explain this once.
                {
                    std::cout
                            << "Because you are creating a FIFO SQS queue,
'.fifo' must "
                            << "be appended to the queue name." << std::endl;
                }
            }
```

```
            request.SetQueueName(queueName);
            queueNames.push_back(queueName);

            Aws::SQS::Model::CreateQueueOutcome outcome =
                    sqsClient.CreateQueue(request);

            if (outcome.IsSuccess()) {
                queueURL = outcome.GetResult().GetQueueUrl();
                std::cout << "Your new SQS queue with the name '" << queueName
                          << "' and the queue URL " << std::endl;
                std::cout << "'" << queueURL << "' has been created." <<
std::endl;
            }
            else {
                std::cerr << "Error with SQS::CreateQueue. "
                          << outcome.GetError().GetMessage()
                          << std::endl;

                cleanUp(topicARN,
                        queueURLS,
                        subscriptionARNS,
                        snsClient,
                        sqsClient);

                return false;
            }
        }
        queueURLS.push_back(queueURL);

        if (first) // Only explain this once.
        {
            std::cout
                    << "The queue URL is used to retrieve the queue ARN, which is
"
                    << "used to create a subscription." << std::endl;
        }

        Aws::String queueARN;
        {
            // 3.  Get the SQS queue ARN attribute.
            Aws::SQS::Model::GetQueueAttributesRequest request;
            request.SetQueueUrl(queueURL);

request.AddAttributeNames(Aws::SQS::Model::QueueAttributeName::QueueArn);
```

```
            Aws::SQS::Model::GetQueueAttributesOutcome outcome =
                    sqsClient.GetQueueAttributes(request);

            if (outcome.IsSuccess()) {
                const Aws::Map<Aws::SQS::Model::QueueAttributeName, Aws::String>
&attributes =
                        outcome.GetResult().GetAttributes();
                const auto &iter = attributes.find(
                        Aws::SQS::Model::QueueAttributeName::QueueArn);
                if (iter != attributes.end()) {
                    queueARN = iter->second;
                    std::cout << "The queue ARN '" << queueARN
                              << "' has been retrieved."
                              << std::endl;
                }
                else {
                    std::cerr
                            << "Error ARN attribute not returned by
GetQueueAttribute."
                            << std::endl;

                    cleanUp(topicARN,
                            queueURLS,
                            subscriptionARNS,
                            snsClient,
                            sqsClient);

                    return false;
                }
            }
            else {
                std::cerr << "Error with SQS::GetQueueAttributes. "
                          << outcome.GetError().GetMessage()
                          << std::endl;

                cleanUp(topicARN,
                        queueURLS,
                        subscriptionARNS,
                        snsClient,
                        sqsClient);

                return false;
            }
```

```
        }

        if (first) {
            std::cout
                    << "An IAM policy must be attached to an SQS queue, enabling
it to receive "
                       "messages from an SNS topic." << std::endl;
        }

        {
            // 4.  Set the SQS queue policy attribute with a policy enabling the
receipt of SNS messages.
            Aws::SQS::Model::SetQueueAttributesRequest request;
            request.SetQueueUrl(queueURL);
            Aws::String policy = createPolicyForQueue(queueARN, topicARN);
            request.AddAttributes(Aws::SQS::Model::QueueAttributeName::Policy,
                                  policy);

            Aws::SQS::Model::SetQueueAttributesOutcome outcome =
                    sqsClient.SetQueueAttributes(request);

            if (outcome.IsSuccess()) {
                std::cout << "The attributes for the queue '" << queueName
                          << "' were successfully updated." << std::endl;
            }
            else {
                std::cerr << "Error with SQS::SetQueueAttributes. "
                          << outcome.GetError().GetMessage()
                          << std::endl;

                cleanUp(topicARN,
                        queueURLS,
                        subscriptionARNS,
                        snsClient,
                        sqsClient);

                return false;
            }
        }

        printAsterisksLine();

        {
            // 5.  Subscribe the SQS queue to the SNS topic.
```

```
            Aws::SNS::Model::SubscribeRequest request;
            request.SetTopicArn(topicARN);
            request.SetProtocol("sqs");
            request.SetEndpoint(queueARN);
            if (isFifoTopic) {
                if (first) {
                    std::cout << "Subscriptions to a FIFO topic can have
 filters."
                              << std::endl;
                    std::cout
                            << "If you add a filter to this subscription, then
 only the filtered messages "
                            << "will be received in the queue." << std::endl;
                    std::cout << "For information about message filtering, "
                              << "see https://docs.aws.amazon.com/sns/latest/dg/
sns-message-filtering.html"
                              << std::endl;
                    std::cout << "For this example, you can filter messages by a
 \""
                              << TONE_ATTRIBUTE << "\" attribute." << std::endl;
                }

                std::ostringstream ostringstream;
                ostringstream << "Filter messages for \"" << queueName
                              << "\"'s subscription to the topic \""
                              << topicName << "\"?  (y/n)";

                // Add filter if user answers yes.
                if (askYesNoQuestion(ostringstream.str())) {
                    Aws::String jsonPolicy = getFilterPolicyFromUser();
                    if (!jsonPolicy.empty()) {
                        filteringMessages = true;

                        std::cout << "This is the filter policy for this
 subscription."
                                  << std::endl;
                        std::cout << jsonPolicy << std::endl;

                        request.AddAttributes("FilterPolicy", jsonPolicy);
                    }
                    else {
                        std::cout
                                << "Because you did not select any attributes, no
 filter "
```

```
                                << "will be added to this subscription." <<
std::endl;
                    }
                }
            }  // if (isFifoTopic)
            Aws::SNS::Model::SubscribeOutcome outcome =
snsClient.Subscribe(request);

            if (outcome.IsSuccess()) {
                Aws::String subscriptionARN =
outcome.GetResult().GetSubscriptionArn();
                std::cout << "The queue '" << queueName
                          << "' has been subscribed to the topic '"
                          << "'" << topicName << "'" << std::endl;
                std::cout << "with the subscription ARN '" << subscriptionARN <<
"."
                          << std::endl;
                subscriptionARNS.push_back(subscriptionARN);
            }
            else {
                std::cerr << "Error with TopicsAndQueues::Subscribe. "
                          << outcome.GetError().GetMessage()
                          << std::endl;

                cleanUp(topicARN,
                        queueURLS,
                        subscriptionARNS,
                        snsClient,
                        sqsClient);

                return false;
            }
        }

        first = false;
    }

    first = true;
    do {
        printAsterisksLine();

        // 6.  Publish a message to the SNS topic.
        Aws::SNS::Model::PublishRequest request;
        request.SetTopicArn(topicARN);
```

```
        Aws::String message = askQuestion("Enter a message text to publish.  ");
        request.SetMessage(message);
        if (isFifoTopic) {
            if (first) {
                std::cout
                        << "Because you are using a FIFO topic, you must set a
message group ID."
                        << std::endl;
                std::cout
                        << "All messages within the same group will be received
in the "
                        << "order they were published." << std::endl;
            }
            Aws::String messageGroupID = askQuestion(
                    "Enter a message group ID for this message. ");
            request.SetMessageGroupId(messageGroupID);
            if (!contentBasedDeduplication) {
                if (first) {
                    std::cout
                            << "Because you are not using content-based
deduplication, "
                            << "you must enter a deduplication ID." << std::endl;
                }
                Aws::String deduplicationID = askQuestion(
                        "Enter a deduplication ID for this message. ");
                request.SetMessageDeduplicationId(deduplicationID);
            }
        }

        if (filteringMessages && askYesNoQuestion(
                "Add an attribute to this message? (y/n) ")) {
            for (size_t i = 0; i < TONES.size(); ++i) {
                std::cout << "  " << (i + 1) << ". " << TONES[i] << std::endl;
            }
            int selection = askQuestionForIntRange(
                    "Enter a number for an attribute. ",
                    1, static_cast<int>(TONES.size()));
            Aws::SNS::Model::MessageAttributeValue messageAttributeValue;
            messageAttributeValue.SetDataType("String");
            messageAttributeValue.SetStringValue(TONES[selection - 1]);
            request.AddMessageAttributes(TONE_ATTRIBUTE, messageAttributeValue);
        }

        Aws::SNS::Model::PublishOutcome outcome = snsClient.Publish(request);
```

```
        if (outcome.IsSuccess()) {
            std::cout << "Your message was successfully published." << std::endl;
        }
        else {
            std::cerr << "Error with TopicsAndQueues::Publish. "
                      << outcome.GetError().GetMessage()
                      << std::endl;

            cleanUp(topicARN,
                    queueURLS,
                    subscriptionARNS,
                    snsClient,
                    sqsClient);

            return false;
        }

        first = false;
    } while (askYesNoQuestion("Post another message? (y/n) "));

    printAsterisksLine();

    std::cout << "Now the SQS queue will be polled to retrieve the messages."
              << std::endl;
    askQuestion("Press any key to continue...", alwaysTrueTest);

    for (size_t i = 0; i < queueURLS.size(); ++i) {
        // 7.  Poll an SQS queue for its messages.
        std::vector<Aws::String> messages;
        std::vector<Aws::String> receiptHandles;
        while (true) {
            Aws::SQS::Model::ReceiveMessageRequest request;
            request.SetMaxNumberOfMessages(10);
            request.SetQueueUrl(queueURLS[i]);

            // Setting WaitTimeSeconds to non-zero enables long polling.
            // For information about long polling, see
            // https://docs.aws.amazon.com/AWSSimpleQueueService/latest/
SQSDeveloperGuide/sqs-short-and-long-polling.html
            request.SetWaitTimeSeconds(1);
            Aws::SQS::Model::ReceiveMessageOutcome outcome =
                    sqsClient.ReceiveMessage(request);
```

```
            if (outcome.IsSuccess()) {
                const Aws::Vector<Aws::SQS::Model::Message> &newMessages =
outcome.GetResult().GetMessages();
                if (newMessages.empty()) {
                    break;
                }
                else {
                    for (const Aws::SQS::Model::Message &message: newMessages) {
                        messages.push_back(message.GetBody());
                        receiptHandles.push_back(message.GetReceiptHandle());
                    }
                }
            }
            else {
                std::cerr << "Error with SQS::ReceiveMessage. "
                          << outcome.GetError().GetMessage()
                          << std::endl;

                cleanUp(topicARN,
                        queueURLS,
                        subscriptionARNS,
                        snsClient,
                        sqsClient);

                return false;
            }
        }

        printAsterisksLine();

        if (messages.empty()) {
            std::cout << "No messages were ";
        }
        else if (messages.size() == 1) {
            std::cout << "One message was ";
        }
        else {
            std::cout << messages.size() << " messages were ";
        }
        std::cout << "received by the queue '" << queueNames[i]
                  << "'." << std::endl;
        for (const Aws::String &message: messages) {
            std::cout << "  Message : '" << message << "'."
                      << std::endl;
```

```
        }

        // 8.  Delete a batch of messages from an SQS queue.
        if (!receiptHandles.empty()) {
            Aws::SQS::Model::DeleteMessageBatchRequest request;
            request.SetQueueUrl(queueURLS[i]);
            int id = 1; // Ids must be unique within a batch delete request.
            for (const Aws::String &receiptHandle: receiptHandles) {
                Aws::SQS::Model::DeleteMessageBatchRequestEntry entry;
                entry.SetId(std::to_string(id));
                ++id;
                entry.SetReceiptHandle(receiptHandle);
                request.AddEntries(entry);
            }

            Aws::SQS::Model::DeleteMessageBatchOutcome outcome =
                    sqsClient.DeleteMessageBatch(request);

            if (outcome.IsSuccess()) {
                std::cout << "The batch deletion of messages was successful."
                          << std::endl;
            }
            else {
                std::cerr << "Error with SQS::DeleteMessageBatch. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                cleanUp(topicARN,
                        queueURLS,
                        subscriptionARNS,
                        snsClient,
                        sqsClient);

                return false;
            }
        }
    }

    return cleanUp(topicARN,
                   queueURLS,
                   subscriptionARNS,
                   snsClient,
                   sqsClient,
                   true); // askUser
}
```

```
bool AwsDoc::TopicsAndQueues::cleanUp(const Aws::String &topicARN,
                                      const Aws::Vector<Aws::String> &queueURLS,
                                      const Aws::Vector<Aws::String>
 &subscriptionARNS,
                                      const Aws::SNS::SNSClient &snsClient,
                                      const Aws::SQS::SQSClient &sqsClient,
                                      bool askUser) {
    bool result = true;
    printAsterisksLine();
    if (!queueURLS.empty() && askUser &&
        askYesNoQuestion("Delete the SQS queues? (y/n) ")) {

        for (const auto &queueURL: queueURLS) {
            // 9.  Delete an SQS queue.
            Aws::SQS::Model::DeleteQueueRequest request;
            request.SetQueueUrl(queueURL);

            Aws::SQS::Model::DeleteQueueOutcome outcome =
                    sqsClient.DeleteQueue(request);

            if (outcome.IsSuccess()) {
                std::cout << "The queue with URL '" << queueURL
                          << "' was successfully deleted." << std::endl;
            }
            else {
                std::cerr << "Error with SQS::DeleteQueue. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                result = false;
            }
        }

        for (const auto &subscriptionARN: subscriptionARNS) {
            // 10. Unsubscribe an SNS subscription.
            Aws::SNS::Model::UnsubscribeRequest request;
            request.SetSubscriptionArn(subscriptionARN);

            Aws::SNS::Model::UnsubscribeOutcome outcome =
                    snsClient.Unsubscribe(request);

            if (outcome.IsSuccess()) {
```

```
                std::cout << "Unsubscribe of subscription ARN '" <<
 subscriptionARN
                          << "' was successful." << std::endl;
            }
            else {
                std::cerr << "Error with TopicsAndQueues::Unsubscribe. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                result = false;
            }
        }
    }

    printAsterisksLine();
    if (!topicARN.empty() && askUser &&
        askYesNoQuestion("Delete the SNS topic? (y/n) ")) {

        // 11. Delete an SNS topic.
        Aws::SNS::Model::DeleteTopicRequest request;
        request.SetTopicArn(topicARN);

        Aws::SNS::Model::DeleteTopicOutcome outcome =
 snsClient.DeleteTopic(request);

        if (outcome.IsSuccess()) {
            std::cout << "The topic with ARN '" << topicARN
                      << "' was successfully deleted." << std::endl;
        }
        else {
            std::cerr << "Error with TopicsAndQueues::DeleteTopicRequest. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            result = false;
        }
    }

    return result;
}

//! Create an IAM policy that gives an SQS queue permission to receive messages
 from an SNS topic.
/*!
 \sa createPolicyForQueue()
 \param queueARN: The SQS queue Amazon Resource Name (ARN).
```

```
 \param topicARN: The SNS topic ARN.
 \return Aws::String: The policy as JSON.
 */
Aws::String AwsDoc::TopicsAndQueues::createPolicyForQueue(const Aws::String
 &queueARN,
                                                           const Aws::String
 &topicARN) {
    std::ostringstream policyStream;
    policyStream << R"({
        "Statement": [
        {
            "Effect": "Allow",
                    "Principal": {
                "Service": "sns.amazonaws.com"
            },
            "Action": "sqs:SendMessage",
                    "Resource": ")" << queueARN << R"(",
                    "Condition": {
                "ArnEquals": {
                    "aws:SourceArn": ")" << topicARN << R"("
                }
            }
        }
        ]
    })";

    return policyStream.str();
}
```

- API-Details finden Sie in den folgenden Themen der AWS SDK for C++ -API-Referenz.
  - CreateQueue
  - CreateTopic
  - DeleteMessageBatch
  - DeleteQueue
  - DeleteTopic
  - GetQueueAttributes
  - Veröffentlichen
  - ReceiveMessage
  - SetQueueAttributes

- Abonnieren
- Unsubscribe

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

Führen Sie ein interaktives Szenario an einer Eingabeaufforderung aus.

```
const FIFO_SUFFIX = ".fifo"
const TONE_KEY = "tone"

var ToneChoices = []string{"cheerful", "funny", "serious", "sincere"}

// MessageBody is used to deserialize the body of a message from a JSON string.
type MessageBody struct {
 Message string
}

// ScenarioRunner separates the steps of this scenario into individual functions
 so that
// they are simpler to read and understand.
type ScenarioRunner struct {
 questioner demotools.IQuestioner
 snsActor   *actions.SnsActions
 sqsActor   *actions.SqsActions
}

func (runner ScenarioRunner) CreateTopic() (string, string, bool, bool) {
 log.Println("SNS topics can be configured as FIFO (First-In-First-Out) or
 standard.\n" +
  "FIFO topics deliver messages in order and support deduplication and message
 filtering.")
```

```
 isFifoTopic := runner.questioner.AskBool("\nWould you like to work with FIFO
 topics? (y/n) ", "y")

 contentBasedDeduplication := false
 if isFifoTopic {
  log.Println(strings.Repeat("-", 88))
  log.Println("Because you have chosen a FIFO topic, deduplication is supported.
\n" +
   "Deduplication IDs are either set in the message or are automatically
 generated\n" +
   "from content using a hash function. If a message is successfully published to
\n" +
   "an SNS FIFO topic, any message published and determined to have the same\n" +
   "deduplication ID, within the five-minute deduplication interval, is accepted
\n" +
   "but not delivered. For more information about deduplication, see:\n" +
   "\thttps://docs.aws.amazon.com/sns/latest/dg/fifo-message-dedup.html.")
  contentBasedDeduplication = runner.questioner.AskBool(
   "\nDo you want to use content-based deduplication instead of entering a
 deduplication ID? (y/n) ", "y")
 }
 log.Println(strings.Repeat("-", 88))

 topicName := runner.questioner.Ask("Enter a name for your SNS topic. ")
 if isFifoTopic {
  topicName = fmt.Sprintf("%v%v", topicName, FIFO_SUFFIX)
  log.Printf("Because you have selected a FIFO topic, '%v' must be appended to
\n"+
   "the topic name.", FIFO_SUFFIX)
 }

 topicArn, err := runner.snsActor.CreateTopic(topicName, isFifoTopic,
 contentBasedDeduplication)
 if err != nil {
  panic(err)
 }
 log.Printf("Your new topic with the name '%v' and Amazon Resource Name (ARN)
 \n"+
  "'%v' has been created.", topicName, topicArn)

 return topicName, topicArn, isFifoTopic, contentBasedDeduplication
}
```

```
func (runner ScenarioRunner) CreateQueue(ordinal string, isFifoTopic bool)
 (string, string) {
 queueName := runner.questioner.Ask(fmt.Sprintf("Enter a name for the %v SQS
 queue. ", ordinal))
 if isFifoTopic {
  queueName = fmt.Sprintf("%v%v", queueName, FIFO_SUFFIX)
  if ordinal == "first" {
   log.Printf("Because you are creating a FIFO SQS queue, '%v' must "+
    "be appended to the queue name.\n", FIFO_SUFFIX)
  }
 }
 queueUrl, err := runner.sqsActor.CreateQueue(queueName, isFifoTopic)
 if err != nil {
  panic(err)
 }
 log.Printf("Your new SQS queue with the name '%v' and the queue URL "+
  "'%v' has been created.", queueName, queueUrl)

 return queueName, queueUrl
}

func (runner ScenarioRunner) SubscribeQueueToTopic(
 queueName string, queueUrl string, topicName string, topicArn string, ordinal
 string,
 isFifoTopic bool) (string, bool) {

 queueArn, err := runner.sqsActor.GetQueueArn(queueUrl)
 if err != nil {
  panic(err)
 }
 log.Printf("The ARN of your queue is: %v.\n", queueArn)

 err = runner.sqsActor.AttachSendMessagePolicy(queueUrl, queueArn, topicArn)
 if err != nil {
  panic(err)
 }
 log.Println("Attached an IAM policy to the queue so the SNS topic can send " +
  "messages to it.")
 log.Println(strings.Repeat("-", 88))

 var filterPolicy map[string][]string
 if isFifoTopic {
  if ordinal == "first" {
   log.Println("Subscriptions to a FIFO topic can have filters.\n" +
```

```
    "If you add a filter to this subscription, then only the filtered messages\n"
 +
    "will be received in the queue.\n" +
    "For information about message filtering, see\n" +
    "\thttps://docs.aws.amazon.com/sns/latest/dg/sns-message-filtering.html\n" +
    "For this example, you can filter messages by a \"tone\" attribute.")
  }

  wantFiltering := runner.questioner.AskBool(
   fmt.Sprintf("Do you want to filter messages that are sent to \"%v\"\n"+
    "from the %v topic? (y/n) ", queueName, topicName), "y")
  if wantFiltering {
   log.Println("You can filter messages by one or more of the following \"tone\"
 attributes.")

   var toneSelections []string
   askAboutTones := true
   for askAboutTones {
    toneIndex := runner.questioner.AskChoice(
     "Enter the number of the tone you want to filter by:\n", ToneChoices)
    toneSelections = append(toneSelections, ToneChoices[toneIndex])
    askAboutTones = runner.questioner.AskBool("Do you want to add another tone to
 the filter? (y/n) ", "y")
   }
   log.Printf("Your subscription will be filtered to only pass the following
 tones: %v\n", toneSelections)
   filterPolicy = map[string][]string{TONE_KEY: toneSelections}
  }
 }

 subscriptionArn, err := runner.snsActor.SubscribeQueue(topicArn, queueArn,
 filterPolicy)
 if err != nil {
  panic(err)
 }
 log.Printf("The queue %v is now subscribed to the topic %v with the subscription
 ARN %v.\n",
  queueName, topicName, subscriptionArn)

 return subscriptionArn, filterPolicy != nil
}

func (runner ScenarioRunner) PublishMessages(topicArn string, isFifoTopic bool,
 contentBasedDeduplication bool, usingFilters bool) {
```

```
 var message string
 var groupId string
 var dedupId string
 var toneSelection string
 publishMore := true
 for publishMore {
  groupId = ""
  dedupId = ""
  toneSelection = ""
  message = runner.questioner.Ask("Enter a message to publish: ")
  if isFifoTopic {
   log.Println("Because you are using a FIFO topic, you must set a message group
 ID.\n" +
    "All messages within the same group will be received in the order they were
 published.")
   groupId = runner.questioner.Ask("Enter a message group ID: ")
   if !contentBasedDeduplication {
    log.Println("Because you are not using content-based deduplication,\n" +
     "you must enter a deduplication ID.")
    dedupId = runner.questioner.Ask("Enter a deduplication ID: ")
   }
  }
  if usingFilters {
   if runner.questioner.AskBool("Add a tone attribute so this message can be
 filtered? (y/n) ", "y") {
    toneIndex := runner.questioner.AskChoice(
     "Enter the number of the tone you want to filter by:\n", ToneChoices)
    toneSelection = ToneChoices[toneIndex]
   }
  }

  err := runner.snsActor.Publish(topicArn, message, groupId, dedupId, TONE_KEY,
 toneSelection)
  if err != nil {
   panic(err)
  }
  log.Println(("Your message was published."))

  publishMore = runner.questioner.AskBool("Do you want to publish another
 messsage? (y/n) ", "y")
 }
}

func (runner ScenarioRunner) PollForMessages(queueUrls []string) {
```

```
 log.Println("Polling queues for messages...")
 for _, queueUrl := range queueUrls {
  var messages []types.Message
  for {
   currentMsgs, err := runner.sqsActor.GetMessages(queueUrl, 10, 1)
   if err != nil {
    panic(err)
   }
   if len(currentMsgs) == 0 {
    break
   }
   messages = append(messages, currentMsgs...)
  }
  if len(messages) == 0 {
   log.Printf("No messages were received by queue %v.\n", queueUrl)
  } else if len(messages) == 1 {
   log.Printf("One message was received by queue %v:\n", queueUrl)

  } else {
   log.Printf("%v messages were received by queue %v:\n", len(messages),
 queueUrl)
  }
  for msgIndex, message := range messages {
   messageBody := MessageBody{}
   err := json.Unmarshal([]byte(*message.Body), &messageBody)
   if err != nil {
    panic(err)
   }
   log.Printf("Message %v: %v\n", msgIndex+1, messageBody.Message)
  }

  if len(messages) > 0 {
   log.Printf("Deleting %v messages from queue %v.\n", len(messages), queueUrl)
   err := runner.sqsActor.DeleteMessages(queueUrl, messages)
   if err != nil {
    panic(err)
   }
  }
 }
}

// RunTopicsAndQueuesScenario is an interactive example that shows you how to use
 the
// AWS SDK for Go to create and use Amazon SNS topics and Amazon SQS queues.
```

```
//
// 1. Create a topic (FIFO or non-FIFO).
// 2. Subscribe several queues to the topic with an option to apply a filter.
// 3. Publish messages to the topic.
// 4. Poll the queues for messages received.
// 5. Delete the topic and the queues.
//
// This example creates service clients from the specified sdkConfig so that
// you can replace it with a mocked or stubbed config for unit testing.
//
// It uses a questioner from the `demotools` package to get input during the
 example.
// This package can be found in the ..\..\demotools folder of this repo.
func RunTopicsAndQueuesScenario(
 sdkConfig aws.Config, questioner demotools.IQuestioner) {
 resources := Resources{}
 defer func() {
  if r := recover(); r != nil {
   log.Println("Something went wrong with the demo.\n" +
    "Cleaning up any resources that were created...")
   resources.Cleanup()
  }
 }()
 queueCount := 2

 log.Println(strings.Repeat("-", 88))
 log.Printf("Welcome to messaging with topics and queues.\n\n"+
  "In this workflow, you will create an SNS topic and subscribe %v SQS queues to
 the\n"+
  "topic. You can select from several options for configuring the topic and the
\n"+
  "subscriptions for the queues. You can then post to the topic and see the
 results\n"+
  "in the queues.\n", queueCount)

 log.Println(strings.Repeat("-", 88))

 runner := ScenarioRunner{
  questioner: questioner,
  snsActor:   &actions.SnsActions{SnsClient: sns.NewFromConfig(sdkConfig)},
  sqsActor:   &actions.SqsActions{SqsClient: sqs.NewFromConfig(sdkConfig)},
 }
 resources.snsActor = runner.snsActor
 resources.sqsActor = runner.sqsActor
```

```
 topicName, topicArn, isFifoTopic, contentBasedDeduplication :=
 runner.CreateTopic()
 resources.topicArn = topicArn
 log.Println(strings.Repeat("-", 88))

 log.Printf("Now you will create %v SQS queues and subscribe them to the topic.
\n", queueCount)
 ordinals := []string{"first", "next"}
 usingFilters := false
 for _, ordinal := range ordinals {
  queueName, queueUrl := runner.CreateQueue(ordinal, isFifoTopic)
  resources.queueUrls = append(resources.queueUrls, queueUrl)

  _, filtering := runner.SubscribeQueueToTopic(queueName, queueUrl, topicName,
 topicArn, ordinal, isFifoTopic)
  usingFilters = usingFilters || filtering
 }

 log.Println(strings.Repeat("-", 88))
 runner.PublishMessages(topicArn, isFifoTopic, contentBasedDeduplication,
 usingFilters)
 log.Println(strings.Repeat("-", 88))
 runner.PollForMessages(resources.queueUrls)

 log.Println(strings.Repeat("-", 88))

 wantCleanup := questioner.AskBool("Do you want to remove all AWS resources
 created for this scenario? (y/n) ", "y")
 if wantCleanup {
  log.Println("Cleaning up resources...")
  resources.Cleanup()
 }

 log.Println(strings.Repeat("-", 88))
 log.Println("Thanks for watching!")
 log.Println(strings.Repeat("-", 88))
}
```

Definieren Sie eine Struktur, die die in diesem Beispiel verwendeten Amazon SNS SNS-Aktionen umschließt.

```
// SnsActions encapsulates the Amazon Simple Notification Service (Amazon SNS)
 actions
// used in the examples.
type SnsActions struct {
 SnsClient *sns.Client
}



// CreateTopic creates an Amazon SNS topic with the specified name. You can
 optionally
// specify that the topic is created as a FIFO topic and whether it uses content-
based
// deduplication instead of ID-based deduplication.
func (actor SnsActions) CreateTopic(topicName string, isFifoTopic bool,
 contentBasedDeduplication bool) (string, error) {
 var topicArn string
 topicAttributes := map[string]string{}
 if isFifoTopic {
  topicAttributes["FifoTopic"] = "true"
 }
 if contentBasedDeduplication {
  topicAttributes["ContentBasedDeduplication"] = "true"
 }
 topic, err := actor.SnsClient.CreateTopic(context.TODO(), &sns.CreateTopicInput{
  Name:       aws.String(topicName),
  Attributes: topicAttributes,
 })
 if err != nil {
  log.Printf("Couldn't create topic %v. Here's why: %v\n", topicName, err)
 } else {
  topicArn = *topic.TopicArn
 }

 return topicArn, err
}



// DeleteTopic delete an Amazon SNS topic.
func (actor SnsActions) DeleteTopic(topicArn string) error {
 _, err := actor.SnsClient.DeleteTopic(context.TODO(), &sns.DeleteTopicInput{
```

```
  TopicArn: aws.String(topicArn)})
 if err != nil {
  log.Printf("Couldn't delete topic %v. Here's why: %v\n", topicArn, err)
 }
 return err
}



// SubscribeQueue subscribes an Amazon Simple Queue Service (Amazon SQS) queue to
 an
// Amazon SNS topic. When filterMap is not nil, it is used to specify a filter
 policy
// so that messages are only sent to the queue when the message has the specified
 attributes.
func (actor SnsActions) SubscribeQueue(topicArn string, queueArn string,
 filterMap map[string][]string) (string, error) {
 var subscriptionArn string
 var attributes map[string]string
 if filterMap != nil {
  filterBytes, err := json.Marshal(filterMap)
  if err != nil {
   log.Printf("Couldn't create filter policy, here's why: %v\n", err)
   return "", err
  }
  attributes = map[string]string{"FilterPolicy": string(filterBytes)}
 }
 output, err := actor.SnsClient.Subscribe(context.TODO(), &sns.SubscribeInput{
  Protocol:              aws.String("sqs"),
  TopicArn:              aws.String(topicArn),
  Attributes:            attributes,
  Endpoint:              aws.String(queueArn),
  ReturnSubscriptionArn: true,
 })
 if err != nil {
  log.Printf("Couldn't susbscribe queue %v to topic %v. Here's why: %v\n",
   queueArn, topicArn, err)
 } else {
  subscriptionArn = *output.SubscriptionArn
 }

 return subscriptionArn, err
}
```

```
// Publish publishes a message to an Amazon SNS topic. The message is then sent
 to all
// subscribers. When the topic is a FIFO topic, the message must also contain a
 group ID
// and, when ID-based deduplication is used, a deduplication ID. An optional key-
value
// filter attribute can be specified so that the message can be filtered
 according to
// a filter policy.
func (actor SnsActions) Publish(topicArn string, message string, groupId string,
 dedupId string, filterKey string, filterValue string) error {
 publishInput := sns.PublishInput{TopicArn: aws.String(topicArn), Message:
 aws.String(message)}
 if groupId != "" {
  publishInput.MessageGroupId = aws.String(groupId)
 }
 if dedupId != "" {
  publishInput.MessageDeduplicationId = aws.String(dedupId)
 }
 if filterKey != "" && filterValue != "" {
  publishInput.MessageAttributes = map[string]types.MessageAttributeValue{
   filterKey: {DataType: aws.String("String"), StringValue:
 aws.String(filterValue)},
  }
 }
 _, err := actor.SnsClient.Publish(context.TODO(), &publishInput)
 if err != nil {
  log.Printf("Couldn't publish message to topic %v. Here's why: %v", topicArn,
 err)
 }
 return err
}
```

Definieren Sie eine Struktur, die die in diesem Beispiel verwendeten Amazon SQS SQS-Aktionen umschließt.

```
// SqsActions encapsulates the Amazon Simple Queue Service (Amazon SQS) actions
// used in the examples.
```

```
type SqsActions struct {
 SqsClient *sqs.Client
}



// CreateQueue creates an Amazon SQS queue with the specified name. You can
 specify
// whether the queue is created as a FIFO queue.
func (actor SqsActions) CreateQueue(queueName string, isFifoQueue bool) (string,
 error) {
 var queueUrl string
 queueAttributes := map[string]string{}
 if isFifoQueue {
  queueAttributes["FifoQueue"] = "true"
 }
 queue, err := actor.SqsClient.CreateQueue(context.TODO(), &sqs.CreateQueueInput{
  QueueName:  aws.String(queueName),
  Attributes: queueAttributes,
 })
 if err != nil {
  log.Printf("Couldn't create queue %v. Here's why: %v\n", queueName, err)
 } else {
  queueUrl = *queue.QueueUrl
 }

 return queueUrl, err
}



// GetQueueArn uses the GetQueueAttributes action to get the Amazon Resource Name
 (ARN)
// of an Amazon SQS queue.
func (actor SqsActions) GetQueueArn(queueUrl string) (string, error) {
 var queueArn string
 arnAttributeName := types.QueueAttributeNameQueueArn
 attribute, err := actor.SqsClient.GetQueueAttributes(context.TODO(),
 &sqs.GetQueueAttributesInput{
  QueueUrl:       aws.String(queueUrl),
  AttributeNames: []types.QueueAttributeName{arnAttributeName},
 })
 if err != nil {
  log.Printf("Couldn't get ARN for queue %v. Here's why: %v\n", queueUrl, err)
```

```
 } else {
  queueArn = attribute.Attributes[string(arnAttributeName)]
 }
 return queueArn, err
}



// AttachSendMessagePolicy uses the SetQueueAttributes action to attach a policy
 to an
// Amazon SQS queue that allows the specified Amazon SNS topic to send messages
 to the
// queue.
func (actor SqsActions) AttachSendMessagePolicy(queueUrl string, queueArn string,
 topicArn string) error {
 policyDoc := PolicyDocument{
  Version: "2012-10-17",
  Statement: []PolicyStatement{{
   Effect:    "Allow",
   Action:    "sqs:SendMessage",
   Principal: map[string]string{"Service": "sns.amazonaws.com"},
   Resource:  aws.String(queueArn),
   Condition: PolicyCondition{"ArnEquals": map[string]string{"aws:SourceArn":
 topicArn}},
  }},
 }
 policyBytes, err := json.Marshal(policyDoc)
 if err != nil {
  log.Printf("Couldn't create policy document. Here's why: %v\n", err)
  return err
 }
 _, err = actor.SqsClient.SetQueueAttributes(context.TODO(),
 &sqs.SetQueueAttributesInput{
  Attributes: map[string]string{
   string(types.QueueAttributeNamePolicy): string(policyBytes),
  },
  QueueUrl: aws.String(queueUrl),
 })
 if err != nil {
  log.Printf("Couldn't set send message policy on queue %v. Here's why: %v\n",
 queueUrl, err)
 }
 return err
}
```

```
// PolicyDocument defines a policy document as a Go struct that can be serialized
// to JSON.
type PolicyDocument struct {
 Version   string
 Statement []PolicyStatement
}

// PolicyStatement defines a statement in a policy document.
type PolicyStatement struct {
 Effect    string
 Action    string
 Principal map[string]string `json:",omitempty"`
 Resource  *string           `json:",omitempty"`
 Condition PolicyCondition   `json:",omitempty"`
}

// PolicyCondition defines a condition in a policy.
type PolicyCondition map[string]map[string]string



// GetMessages uses the ReceiveMessage action to get messages from an Amazon SQS
 queue.
func (actor SqsActions) GetMessages(queueUrl string, maxMessages int32, waitTime
 int32) ([]types.Message, error) {
 var messages []types.Message
 result, err := actor.SqsClient.ReceiveMessage(context.TODO(),
 &sqs.ReceiveMessageInput{
  QueueUrl:            aws.String(queueUrl),
  MaxNumberOfMessages: maxMessages,
  WaitTimeSeconds:     waitTime,
 })
 if err != nil {
  log.Printf("Couldn't get messages from queue %v. Here's why: %v\n", queueUrl,
 err)
 } else {
  messages = result.Messages
 }
 return messages, err
}
```

```
// DeleteMessages uses the DeleteMessageBatch action to delete a batch of
 messages from
// an Amazon SQS queue.
func (actor SqsActions) DeleteMessages(queueUrl string, messages []types.Message)
 error {
 entries := make([]types.DeleteMessageBatchRequestEntry, len(messages))
 for msgIndex := range messages {
  entries[msgIndex].Id = aws.String(fmt.Sprintf("%v", msgIndex))
  entries[msgIndex].ReceiptHandle = messages[msgIndex].ReceiptHandle
 }
 _, err := actor.SqsClient.DeleteMessageBatch(context.TODO(),
 &sqs.DeleteMessageBatchInput{
  Entries:  entries,
  QueueUrl: aws.String(queueUrl),
 })
 if err != nil {
  log.Printf("Couldn't delete messages from queue %v. Here's why: %v\n",
 queueUrl, err)
 }
 return err
}



// DeleteQueue deletes an Amazon SQS queue.
func (actor SqsActions) DeleteQueue(queueUrl string) error {
 _, err := actor.SqsClient.DeleteQueue(context.TODO(), &sqs.DeleteQueueInput{
  QueueUrl: aws.String(queueUrl)})
 if err != nil {
  log.Printf("Couldn't delete queue %v. Here's why: %v\n", queueUrl, err)
 }
 return err
}
```

- API-Details finden Sie in den folgenden Themen der AWS SDK for Go -API-Referenz.
  - CreateQueue
  - CreateTopic
  - DeleteMessageBatch
  - DeleteQueue

- DeleteTopic
- GetQueueAttributes
- Veröffentlichen
- ReceiveMessage
- SetQueueAttributes
- Abonnieren
- Unsubscribe

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

Dies ist der Einstiegspunkt für diesen Workflow.

```
import { SNSClient } from "@aws-sdk/client-sns";
import { SQSClient } from "@aws-sdk/client-sqs";

import { TopicsQueuesWkflw } from "./TopicsQueuesWkflw.js";
import { Prompter } from "@aws-doc-sdk-examples/lib/prompter.js";
import { SlowLogger } from "@aws-doc-sdk-examples/lib/slow-logger.js";

export const startSnsWorkflow = () => {
  const noLoggerDelay = process.argv.find((arg) => arg === "--no-logger-delay");
  const snsClient = new SNSClient({});
  const sqsClient = new SQSClient({});
  const prompter = new Prompter();
  const logger = noLoggerDelay ? console : new SlowLogger(25);

  const wkflw = new TopicsQueuesWkflw(snsClient, sqsClient, prompter, logger);

  wkflw.start();
};
```

Der obige Code stellt die erforderlichen Abhängigkeiten bereit und startet den Workflow. Der nächste Abschnitt enthält den Großteil des Beispiels.

```
const toneChoices = [
  { name: "cheerful", value: "cheerful" },
  { name: "funny", value: "funny" },
  { name: "serious", value: "serious" },
  { name: "sincere", value: "sincere" },
];

export class TopicsQueuesWkflw {
  // SNS topic is configured as First-In-First-Out
  isFifo = true;

  // Automatic content-based deduplication is enabled.
  autoDedup = false;

  snsClient;
  sqsClient;
  topicName;
  topicArn;
  subscriptionArns = [];
  /**
   * @type {{ queueName: string, queueArn: string, queueUrl: string, policy?:
 string }[]}
   */
  queues = [];
  prompter;

  /**
   * @param {import('@aws-sdk/client-sns').SNSClient} snsClient
   * @param {import('@aws-sdk/client-sqs').SQSClient} sqsClient
   * @param {import('../../libs/prompter.js').Prompter} prompter
   * @param {import('../../libs/logger.js').Logger} logger
   */
  constructor(snsClient, sqsClient, prompter, logger) {
    this.snsClient = snsClient;
    this.sqsClient = sqsClient;
    this.prompter = prompter;
    this.logger = logger;
```

```
  }

  async welcome() {
    await this.logger.log(MESSAGES.description);
  }

  async confirmFifo() {
    await this.logger.log(MESSAGES.snsFifoDescription);
    this.isFifo = await this.prompter.confirm({
      message: MESSAGES.snsFifoPrompt,
    });

    if (this.isFifo) {
      this.logger.logSeparator(MESSAGES.headerDedup);
      await this.logger.log(MESSAGES.deduplicationNotice);
      await this.logger.log(MESSAGES.deduplicationDescription);
      this.autoDedup = await this.prompter.confirm({
        message: MESSAGES.deduplicationPrompt,
      });
    }
  }

  async createTopic() {
    await this.logger.log(MESSAGES.creatingTopics);
    this.topicName = await this.prompter.input({
      message: MESSAGES.topicNamePrompt,
    });
    if (this.isFifo) {
      this.topicName += ".fifo";
      this.logger.logSeparator(MESSAGES.headerFifoNaming);
      await this.logger.log(MESSAGES.appendFifoNotice);
    }

    const response = await this.snsClient.send(
      new CreateTopicCommand({
        Name: this.topicName,
        Attributes: {
          FifoTopic: this.isFifo ? "true" : "false",
          ...(this.autoDedup ? { ContentBasedDeduplication: "true" } : {}),
        },
      }),
    );

    this.topicArn = response.TopicArn;
```

```
    await this.logger.log(
      MESSAGES.topicCreatedNotice
        .replace("${TOPIC_NAME}", this.topicName)
        .replace("${TOPIC_ARN}", this.topicArn),
    );
  }

  async createQueues() {
    await this.logger.log(MESSAGES.createQueuesNotice);
    // Increase this number to add more queues.
    let maxQueues = 2;

    for (let i = 0; i < maxQueues; i++) {
      await this.logger.log(MESSAGES.queueCount.replace("${COUNT}", i + 1));
      let queueName = await this.prompter.input({
        message: MESSAGES.queueNamePrompt.replace(
          "${EXAMPLE_NAME}",
          i === 0 ? "good-news" : "bad-news",
        ),
      });

      if (this.isFifo) {
        queueName += ".fifo";
        await this.logger.log(MESSAGES.appendFifoNotice);
      }

      const response = await this.sqsClient.send(
        new CreateQueueCommand({
          QueueName: queueName,
          Attributes: { ...(this.isFifo ? { FifoQueue: "true" } : {}) },
        }),
      );

      const { Attributes } = await this.sqsClient.send(
        new GetQueueAttributesCommand({
          QueueUrl: response.QueueUrl,
          AttributeNames: ["QueueArn"],
        }),
      );

      this.queues.push({
        queueName,
        queueArn: Attributes.QueueArn,
```

```
        queueUrl: response.QueueUrl,
      });

      await this.logger.log(
        MESSAGES.queueCreatedNotice
          .replace("${QUEUE_NAME}", queueName)
          .replace("${QUEUE_URL}", response.QueueUrl)
          .replace("${QUEUE_ARN}", Attributes.QueueArn),
      );
    }
  }

  async attachQueueIamPolicies() {
    for (const [index, queue] of this.queues.entries()) {
      const policy = JSON.stringify(
        {
          Statement: [
            {
              Effect: "Allow",
              Principal: {
                Service: "sns.amazonaws.com",
              },
              Action: "sqs:SendMessage",
              Resource: queue.queueArn,
              Condition: {
                ArnEquals: {
                  "aws:SourceArn": this.topicArn,
                },
              },
            },
          ],
        },
        null,
        2,
      );

      if (index !== 0) {
        this.logger.logSeparator();
      }

      await this.logger.log(MESSAGES.attachPolicyNotice);
      console.log(policy);
      const addPolicy = await this.prompter.confirm({
        message: MESSAGES.addPolicyConfirmation.replace(
```

```
          "${QUEUE_NAME}",
          queue.queueName,
        ),
      });

      if (addPolicy) {
        await this.sqsClient.send(
          new SetQueueAttributesCommand({
            QueueUrl: queue.queueUrl,
            Attributes: {
              Policy: policy,
            },
          }),
        );
        queue.policy = policy;
      } else {
        await this.logger.log(
          MESSAGES.policyNotAttachedNotice.replace(
            "${QUEUE_NAME}",
            queue.queueName,
          ),
        );
      }
    }
  }

  async subscribeQueuesToTopic() {
    for (const [index, queue] of this.queues.entries()) {
      /**
       * @type {import('@aws-sdk/client-sns').SubscribeCommandInput}
       */
      const subscribeParams = {
        TopicArn: this.topicArn,
        Protocol: "sqs",
        Endpoint: queue.queueArn,
      };
      let tones = [];

      if (this.isFifo) {
        if (index === 0) {
          await this.logger.log(MESSAGES.fifoFilterNotice);
        }
        tones = await this.prompter.checkbox({
          message: MESSAGES.fifoFilterSelect.replace(
```

```
            "${QUEUE_NAME}",
            queue.queueName,
          ),
          choices: toneChoices,
        });

        if (tones.length) {
          subscribeParams.Attributes = {
            FilterPolicyScope: "MessageAttributes",
            FilterPolicy: JSON.stringify({
              tone: tones,
            }),
          };
        }
      }

      const { SubscriptionArn } = await this.snsClient.send(
        new SubscribeCommand(subscribeParams),
      );

      this.subscriptionArns.push(SubscriptionArn);

      await this.logger.log(
        MESSAGES.queueSubscribedNotice
          .replace("${QUEUE_NAME}", queue.queueName)
          .replace("${TOPIC_NAME}", this.topicName)
          .replace("${TONES}", tones.length ? tones.join(", ") : "none"),
      );
    }
  }

  async publishMessages() {
    const message = await this.prompter.input({
      message: MESSAGES.publishMessagePrompt,
    });

    let groupId, deduplicationId, choices;

    if (this.isFifo) {
      await this.logger.log(MESSAGES.groupIdNotice);
      groupId = await this.prompter.input({
        message: MESSAGES.groupIdPrompt,
      });
```

```
      if (this.autoDedup === false) {
        await this.logger.log(MESSAGES.deduplicationIdNotice);
        deduplicationId = await this.prompter.input({
          message: MESSAGES.deduplicationIdPrompt,
        });
      }

      choices = await this.prompter.checkbox({
        message: MESSAGES.messageAttributesPrompt,
        choices: toneChoices,
      });
    }

    await this.snsClient.send(
      new PublishCommand({
        TopicArn: this.topicArn,
        Message: message,
        ...(groupId
          ? {
              MessageGroupId: groupId,
            }
          : {}),
        ...(deduplicationId
          ? {
              MessageDeduplicationId: deduplicationId,
            }
          : {}),
        ...(choices
          ? {
              MessageAttributes: {
                tone: {
                  DataType: "String.Array",
                  StringValue: JSON.stringify(choices),
                },
              },
            }
          : {}),
      }),
    );

    const publishAnother = await this.prompter.confirm({
      message: MESSAGES.publishAnother,
    });
```

```
    if (publishAnother) {
      await this.publishMessages();
    }
  }

  async receiveAndDeleteMessages() {
    for (const queue of this.queues) {
      const { Messages } = await this.sqsClient.send(
        new ReceiveMessageCommand({
          QueueUrl: queue.queueUrl,
        }),
      );

      if (Messages) {
        await this.logger.log(
          MESSAGES.messagesReceivedNotice.replace(
            "${QUEUE_NAME}",
            queue.queueName,
          ),
        );
        console.log(Messages);

        await this.sqsClient.send(
          new DeleteMessageBatchCommand({
            QueueUrl: queue.queueUrl,
            Entries: Messages.map((message) => ({
              Id: message.MessageId,
              ReceiptHandle: message.ReceiptHandle,
            })),
          }),
        );
      } else {
        await this.logger.log(
          MESSAGES.noMessagesReceivedNotice.replace(
            "${QUEUE_NAME}",
            queue.queueName,
          ),
        );
      }
    }

    const deleteAndPoll = await this.prompter.confirm({
      message: MESSAGES.deleteAndPollConfirmation,
    });
```

```
    if (deleteAndPoll) {
      await this.receiveAndDeleteMessages();
    }
  }

  async destroyResources() {
    for (const subscriptionArn of this.subscriptionArns) {
      await this.snsClient.send(
        new UnsubscribeCommand({ SubscriptionArn: subscriptionArn }),
      );
    }

    for (const queue of this.queues) {
      await this.sqsClient.send(
        new DeleteQueueCommand({ QueueUrl: queue.queueUrl }),
      );
    }

    if (this.topicArn) {
      await this.snsClient.send(
        new DeleteTopicCommand({ TopicArn: this.topicArn }),
      );
    }
  }

  async start() {
    console.clear();

    try {
      this.logger.logSeparator(MESSAGES.headerWelcome);
      await this.welcome();
      this.logger.logSeparator(MESSAGES.headerFifo);
      await this.confirmFifo();
      this.logger.logSeparator(MESSAGES.headerCreateTopic);
      await this.createTopic();
      this.logger.logSeparator(MESSAGES.headerCreateQueues);
      await this.createQueues();
      this.logger.logSeparator(MESSAGES.headerAttachPolicy);
      await this.attachQueueIamPolicies();
      this.logger.logSeparator(MESSAGES.headerSubscribeQueues);
      await this.subscribeQueuesToTopic();
      this.logger.logSeparator(MESSAGES.headerPublishMessage);
      await this.publishMessages();
```

```
      this.logger.logSeparator(MESSAGES.headerReceiveMessages);
      await this.receiveAndDeleteMessages();
    } catch (err) {
      console.error(err);
    } finally {
      await this.destroyResources();
    }
  }
}
```

- API-Details finden Sie in den folgenden Themen der AWS SDK for JavaScript -API-Referenz.
  - CreateQueue
  - CreateTopic
  - DeleteMessageBatch
  - DeleteQueue
  - DeleteTopic
  - GetQueueAttributes
  - Veröffentlichen
  - ReceiveMessage
  - SetQueueAttributes
  - Abonnieren
  - Unsubscribe

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unterAmazon SNS mit einem AWS SDK verwenden. Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Serverlose Beispiele für Amazon SNS mit SDKs AWS

Die folgenden Codebeispiele zeigen, wie Amazon SNS mit AWS SDKs verwendet wird.

Beispiele

- Eine Lambda-Funktion über einen Amazon-SNS-Trigger aufrufen

Eine Lambda-Funktion über einen Amazon-SNS-Trigger aufrufen

In den folgenden Codebeispiele wird die Implementierung einer Lambda-Funktion veranschaulicht, die ein Ereignis empfängt, das durch das Empfangen von Nachrichten aus einem SNS-Thema ausgelöst wird. Die Funktion ruft die Nachrichten aus dem Ereignisparameter ab und protokolliert den Inhalt jeder Nachricht.

.NET

AWS SDK for .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines SNS-Ereignisses mit Lambda unter Verwendung von .NET.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
using Amazon.Lambda.Core;
using Amazon.Lambda.SNSEvents;


// Assembly attribute to enable the Lambda function's JSON input to be converted
 into a .NET class.
[assembly:
 LambdaSerializer(typeof(Amazon.Lambda.Serialization.SystemTextJson.DefaultLambdaJsonSeri

namespace SnsIntegration;

public class Function
{
    public async Task FunctionHandler(SNSEvent evnt, ILambdaContext context)
    {
        foreach (var record in evnt.Records)
        {
            await ProcessRecordAsync(record, context);
        }
        context.Logger.LogInformation("done");
    }
```

```
    private async Task ProcessRecordAsync(SNSEvent.SNSRecord record,
 ILambdaContext context)
    {
        try
        {
            context.Logger.LogInformation($"Processed record
 {record.Sns.Message}");

            // TODO: Do interesting work based on the new message
            await Task.CompletedTask;
        }
        catch (Exception e)
        {
            //You can use Dead Letter Queue to handle failures. By configuring a
 Lambda DLQ.
            context.Logger.LogError($"An error occurred");
            throw;
        }
    }
}
```

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines SNS-Ereignisses mit Lambda unter Verwendung von Go.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package main

import (
 "context"
 "fmt"
```

```
 "github.com/aws/aws-lambda-go/events"
 "github.com/aws/aws-lambda-go/lambda"
)

func handler(ctx context.Context, snsEvent events.SNSEvent) {
 for _, record := range snsEvent.Records {
  processMessage(record)
 }
 fmt.Println("done")
}

func processMessage(record events.SNSEventRecord) {
 message := record.SNS.Message
 fmt.Printf("Processed message: %s\n", message)
 // TODO: Process your record here
}

func main() {
 lambda.Start(handler)
}
```

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Verwenden eines SNS-Ereignisses mit Lambda unter Verwendung von Java.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package example;

import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.LambdaLogger;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.SNSEvent;
```

```
import com.amazonaws.services.lambda.runtime.events.SNSEvent.SNSRecord;


import java.util.Iterator;
import java.util.List;

public class SNSEventHandler implements RequestHandler<SNSEvent, Boolean> {
    LambdaLogger logger;

    @Override
    public Boolean handleRequest(SNSEvent event, Context context) {
        logger = context.getLogger();
        List<SNSRecord> records = event.getRecords();
        if (!records.isEmpty()) {
            Iterator<SNSRecord> recordsIter = records.iterator();
            while (recordsIter.hasNext()) {
                processRecord(recordsIter.next());
            }
        }
        return Boolean.TRUE;
    }

    public void processRecord(SNSRecord record) {
        try {
            String message = record.getSNS().getMessage();
            logger.log("message: " + message);
        } catch (Exception e) {
            throw new RuntimeException(e);
        }
    }

}
```

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Konsumieren eines SNS-Ereignisses mit Lambda unter Verwendung. JavaScript

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
exports.handler = async (event, context) => {
  for (const record of event.Records) {
    await processMessageAsync(record);
  }
  console.info("done");
};

async function processMessageAsync(record) {
  try {
    const message = JSON.stringify(record.Sns.Message);
    console.log(`Processed message ${message}`);
    await Promise.resolve(1); //Placeholder for actual async work
  } catch (err) {
    console.error("An error occurred");
    throw err;
  }
}
```

Konsumieren eines SNS-Ereignisses mit Lambda unter Verwendung. TypeScript

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import { SNSEvent, Context, SNSHandler, SNSEventRecord } from "aws-lambda";

export const functionHandler: SNSHandler = async (
  event: SNSEvent,
  context: Context
```

```
): Promise<void> => {
  for (const record of event.Records) {
    await processMessageAsync(record);
  }
  console.info("done");
};

async function processMessageAsync(record: SNSEventRecord): Promise<any> {
  try {
    const message: string = JSON.stringify(record.Sns.Message);
    console.log(`Processed message ${message}`);
    await Promise.resolve(1); //Placeholder for actual async work
  } catch (err) {
    console.error("An error occurred");
    throw err;
  }
}
```

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines SNS-Ereignisses mit Lambda unter Verwendung von PHP

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
<?php

/*
Since native PHP support for AWS Lambda is not available, we are utilizing Bref's
 PHP functions runtime for AWS Lambda.
For more information on Bref's PHP runtime for Lambda, refer to: https://bref.sh/
docs/runtimes/function

Another approach would be to create a custom runtime.
```

```
A practical example can be found here: https://aws.amazon.com/blogs/apn/aws-
lambda-custom-runtime-for-php-a-practical-example/
*/

// Additional composer packages may be required when using Bref or any other PHP
 functions runtime.
// require __DIR__ . '/vendor/autoload.php';

use Bref\Context\Context;
use Bref\Event\Sns\SnsEvent;
use Bref\Event\Sns\SnsHandler;

class Handler extends SnsHandler
{
    public function handleSns(SnsEvent $event, Context $context): void
    {
        foreach ($event->getRecords() as $record) {
            $message = $record->getMessage();

            // TODO: Implement your custom processing logic here
            // Any exception thrown will be logged and the invocation will be
 marked as failed

            echo "Processed Message: $message" . PHP_EOL;
        }
    }
}

return new Handler();
```

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines SNS-Ereignisses mit Lambda unter Verwendung von Python.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
def lambda_handler(event, context):
    for record in event['Records']:
        process_message(record)
    print("done")

def process_message(record):
    try:
        message = record['Sns']['Message']
        print(f"Processed message {message}")
        # TODO; Process your record here

    except Exception as e:
        print("An error occurred")
        raise e
```

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Verwenden eines SNS-Ereignisses mit Lambda unter Verwendung von Ruby.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
def lambda_handler(event:, context:)
  event['Records'].map { |record| process_message(record) }
end

def process_message(record)
  message = record['Sns']['Message']
  puts("Processing message: #{message}")
rescue StandardError => e
  puts("Error processing message: #{e}")
```

```
  raise
end
```

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu. GitHub Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines SNS-Ereignisses mit Lambda unter Verwendung von Rust

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
use aws_lambda_events::event::sns::SnsEvent;
use aws_lambda_events::sns::SnsRecord;
use lambda_runtime::{run, service_fn, Error, LambdaEvent};
use tracing::info;

// Built with the following dependencies:
//  aws_lambda_events = { version = "0.10.0", default-features = false, features
 = ["sns"] }
//  lambda_runtime = "0.8.1"
//  tokio = { version = "1", features = ["macros"] }
//  tracing = { version = "0.1", features = ["log"] }
//  tracing-subscriber = { version = "0.3", default-features = false, features =
 ["fmt"] }

async fn function_handler(event: LambdaEvent<SnsEvent>) -> Result<(), Error> {
    for event in event.payload.records {
        process_record(&event)?;
    }

    Ok(())
}

fn process_record(record: &SnsRecord) -> Result<(), Error> {
    info!("Processing SNS Message: {}", record.sns.message);
```

```
    // Implement your record handling code here.

    Ok(())
}

#[tokio::main]
async fn main() -> Result<(), Error> {
    tracing_subscriber::fmt()
        .with_max_level(tracing::Level::INFO)
        .with_target(false)
        .without_time()
        .init();

    run(service_fn(function_handler)).await
}
```

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unterAmazon SNS mit einem AWS SDK verwenden. Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Serviceübergreifende Beispiele für Amazon SNS mit SDKs AWS

Die folgenden Beispielanwendungen verwenden AWS SDKs, um Amazon SNS mit anderen zu kombinieren. AWS-Services Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zur Einrichtung und Ausführung der Anwendung finden.

Beispiele

- Erstellen Sie eine Anwendung zum Senden von Daten an eine DynamoDB-Tabelle
- Erstellen einer Publish- und Abonnement-Anwendung, die Nachrichten übersetzt
- Eine Anwendung für Foto-Asset-Management erstellen, mit der Benutzer Fotos mithilfe von Labels verwalten können
- Erstellen Sie eine Amazon-Textract-Explorer-Anwendung
- Erkennen Sie Personen und Objekte in einem Video mit Amazon Rekognition mithilfe eines SDK AWS
- Veröffentlichen Sie Amazon SNS SNS-Nachrichten mithilfe eines SDK in Amazon SQS SQS-Warteschlangen AWS

- Verwenden von API Gateway zum Aufrufen einer Lambda-Funktion
- Verwendung geplanter Ereignisse zum Aufrufen einer Lambda-Funktion

Erstellen Sie eine Anwendung zum Senden von Daten an eine DynamoDB-Tabelle

Die folgenden Code-Beispiele zeigen, wie man eine Anwendung erstellt, die Daten an eine Amazon-DynamoDB-Tabelle übermittelt und Sie benachrichtigt, wenn ein Benutzer die Tabelle aktualisiert.

Java

SDK für Java 2.x

Zeigt, wie man eine dynamische Webanwendung erstellt, die Daten über die Amazon-DynamoDB-Java-API übermittelt und eine Textnachricht über die Amazon Simple Notification Service Java API sendet.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- DynamoDB
- Amazon SNS

JavaScript

SDK für JavaScript (v3)

Das Beispiel zeigt, wie man eine App erstellt, die es Benutzern ermöglicht, Daten an eine Amazon-DynamoDB-Tabelle zu übermitteln und eine Textnachricht an den Administrator mit Amazon Simple Notification Service (Amazon SNS) zu senden.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

Dieses Beispiel ist auch verfügbar im AWS SDK for JavaScript Entwicklerhandbuch für v3.

In diesem Beispiel verwendete Dienste

- DynamoDB

- Amazon SNS

Kotlin

SDK für Kotlin

Zeigt, wie man eine native Android-Anwendung erstellt, die Daten über die Amazon-DynamoDB-Kotlin-API übermittelt und eine Textnachricht über die Amazon-SNS-Kotlin-API sendet.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- DynamoDB
- Amazon SNS

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unterAmazon SNS mit einem AWS SDK verwenden. Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Erstellen einer Publish- und Abonnement-Anwendung, die Nachrichten übersetzt

Die folgende Code-Beispiele zeigen, wie man eine Anwendung erstellt, die über Abonnements- und Veröffentlichungsfunktionen verfügt und Nachrichten übersetzt.

.NET

AWS SDK for .NET

Zeigt, wie man die .NET-API für Amazon Simple Notification Service verwendet, um eine Webanwendung zu erstellen, die über Abonnement- und Veröffentlichungsfunktionalität verfügt. Darüber hinaus übersetzt diese Beispielanwendung auch Nachrichten.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon SNS
- Amazon Translate

Java

SDK für Java 2.x

Zeigt, wie man die Java-API für Amazon Simple Notification Service verwendet, um eine Webanwendung zu erstellen, die über Abonnement- und Veröffentlichungsfunktionen verfügt. Darüber hinaus übersetzt diese Beispielanwendung auch Nachrichten.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung des Beispiels, das die Java Async API verwendet, finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon SNS
- Amazon Translate

Kotlin

SDK für Kotlin

Zeigt, wie man die Kotlin-API für Amazon SNS verwendet, um eine Anwendung zu erstellen, die über Abonnement- und Veröffentlichungsfunktionen verfügt. Darüber hinaus übersetzt diese Beispielanwendung auch Nachrichten.

Den vollständigen Quellcode und Anweisungen zum Erstellen einer Web-App finden Sie im vollständigen Beispiel unter GitHub.

Den vollständigen Quellcode und Anweisungen zum Erstellen einer nativen Android-App finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon SNS
- Amazon Translate

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unterAmazon SNS mit einem AWS SDK verwenden. Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Eine Anwendung für Foto-Asset-Management erstellen, mit der Benutzer Fotos mithilfe von Labels verwalten können

Die folgenden Codebeispiele zeigen, wie eine Serverless-Anwendung erstellt wird, mit der Benutzer Fotos mithilfe von Labels verwalten können.

.NET

AWS SDK for .NET

Zeigt, wie eine Anwendung zur Verwaltung von Fotobeständen entwickelt wird, die mithilfe von Amazon Rekognition Labels in Bildern erkennt und sie für einen späteren Abruf speichert.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

Einen tiefen Einblick in den Ursprung dieses Beispiels finden Sie im Beitrag in der AWS - Community.

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

C++

SDK für C++

Zeigt, wie eine Anwendung zur Verwaltung von Fotobeständen entwickelt wird, die mithilfe von Amazon Rekognition Labels in Bildern erkennt und sie für einen späteren Abruf speichert.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

Einen tiefen Einblick in den Ursprung dieses Beispiels finden Sie im Beitrag in der AWS - Community.

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Java

SDK für Java 2.x

Zeigt, wie eine Anwendung zur Verwaltung von Fotobeständen entwickelt wird, die mithilfe von Amazon Rekognition Labels in Bildern erkennt und sie für einen späteren Abruf speichert.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

Einen tiefen Einblick in den Ursprung dieses Beispiels finden Sie im Beitrag in der AWS - Community.

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

JavaScript

SDK für JavaScript (v3)

Zeigt, wie eine Anwendung zur Verwaltung von Fotobeständen entwickelt wird, die mithilfe von Amazon Rekognition Labels in Bildern erkennt und sie für einen späteren Abruf speichert.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

Einen tiefen Einblick in den Ursprung dieses Beispiels finden Sie im Beitrag in der AWS - Community.

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Kotlin

SDK für Kotlin

Zeigt, wie eine Anwendung zur Verwaltung von Fotobeständen entwickelt wird, die mithilfe von Amazon Rekognition Labels in Bildern erkennt und sie für einen späteren Abruf speichert.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

Einen tiefen Einblick in den Ursprung dieses Beispiels finden Sie im Beitrag in der AWS - Community.

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB

- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

PHP

SDK für PHP

Zeigt, wie eine Anwendung zur Verwaltung von Fotobeständen entwickelt wird, die mithilfe von Amazon Rekognition Labels in Bildern erkennt und sie für einen späteren Abruf speichert.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

Einen tiefen Einblick in den Ursprung dieses Beispiels finden Sie im Beitrag in der AWS - Community.

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Rust

SDK für Rust

Zeigt, wie eine Anwendung zur Verwaltung von Fotobeständen entwickelt wird, die mithilfe von Amazon Rekognition Labels in Bildern erkennt und sie für einen späteren Abruf speichert.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

Einen tiefen Einblick in den Ursprung dieses Beispiels finden Sie im Beitrag in der AWS - Community.

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unterAmazon SNS mit einem AWS SDK verwenden. Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Erstellen Sie eine Amazon-Textract-Explorer-Anwendung

Die folgenden Code-Beispiele zeigen, wie man die Amazon-Textract-Ausgabe in einer interaktiven Anwendung untersuchen kann.

JavaScript

SDK für JavaScript (v3)

Zeigt, wie Sie mit AWS SDK for JavaScript dem eine React-Anwendung erstellen, die Amazon Textract verwendet, um Daten aus einem Dokumentbild zu extrahieren und auf einer interaktiven Webseite anzuzeigen. Dieses Beispiel wird in einem Webbrowser ausgeführt und erfordert eine authentifizierte Amazon-Cognito-Identität für Anmeldeinformationen. Es verwendet Amazon Simple Storage Service (Amazon S3) zur Speicherung und fragt für Benachrichtigungen eine Amazon Simple Queue Service (Amazon SQS)-Warteschlange ab, die ein Amazon Simple Notification Service (Amazon SNS)-Thema abonniert hat.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon Cognito Identity

- Amazon S3
- Amazon SNS
- Amazon SQS
- Amazon Textract

Python

SDK für Python (Boto3)

Zeigt, wie Sie AWS SDK for Python (Boto3) mit Amazon Textract Text-, Formular- und Tabellenelemente in einem Dokumentbild erkennen können. Das Eingabe-Image und die Amazon-Textract-Ausgabe werden in einer Tkinter-Anwendung angezeigt, mit der Sie die erkannten Elemente untersuchen können.

- Senden Sie ein Dokument-Image an Amazon Textract und untersuchen Sie die Ausgabe erkannter Elemente.
- Senden Sie Images direkt an Amazon Textract oder über einen Amazon Simple Storage Service (Amazon S3)-Bucket.
- Verwenden Sie asynchrone APIs, um einen Auftrag zu starten, der eine Benachrichtigung an ein Amazon Simple Notification Service (Amazon SNS)-Thema veröffentlicht.
- Stellen Sie eine Amazon Simple Queue Service (Amazon SQS)-Warteschlange ab, um eine Meldung zum Abschluss des Auftrags zu erhalten.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon S3
- Amazon SNS
- Amazon SQS
- Amazon Textract

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unterAmazon SNS mit einem AWS SDK verwenden. Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Erkennen Sie Personen und Objekte in einem Video mit Amazon Rekognition mithilfe eines SDK AWS

Die folgenden Code-Beispiele zeigen, wie man Personen und Objekte in einem Video mit Amazon Rekognition erkennt.

Python

SDK für Python (Boto3)

Verwenden Sie Amazon Rekognition, um Gesichter, Objekte und Personen in Videos zu erkennen, indem Sie asynchrone Erkennungsaufträge starten. In diesem Beispiel wird Amazon Rekognition auch so konfiguriert, dass es ein Amazon Simple Notification Service (Amazon SNS)-Thema benachrichtigt, wenn Aufträge abgeschlossen sind, und eine Amazon Simple Queue Service (Amazon SQS)-Warteschlange bei dem Thema abonniert. Wenn die Warteschlange eine Meldung über einen Job erhält, wird der Job abgerufen und die Ergebnisse werden ausgegeben.

Dieses Beispiel lässt sich am besten unter ansehen GitHub. Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon Rekognition
- Amazon SNS
- Amazon SQS

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unterAmazon SNS mit einem AWS SDK verwenden. Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Veröffentlichen Sie Amazon SNS SNS-Nachrichten mithilfe eines SDK in Amazon SQS SQS-Warteschlangen AWS

Die folgenden Code-Beispiele veranschaulichen Folgendes:

- Erstellen Sie ein Thema (FIFO oder Nicht-FIFO).

- Abonnieren Sie mehrere Warteschlangen für das Thema mit der Option, einen Filter anzuwenden.
- Veröffentlichen Sie eine Nachricht im Thema.
- Fragen Sie die Warteschlangen nach empfangenen Nachrichten ab.

Java

SDK für Java 2.x

Zeigt das Messaging mit Themen und Warteschlangen mithilfe von Amazon Simple Notification Service (Amazon SNS) und Amazon Simple Queue Service (Amazon SQS).

Den vollständigen Quellcode und Anweisungen, die das Versenden von Nachrichten mit Themen und Warteschlangen in Amazon SNS und Amazon SQS demonstrieren, finden Sie im vollständigen Beispiel unter. GitHub

In diesem Beispiel verwendete Dienste

- Amazon SNS
- Amazon SQS

Kotlin

SDK für Kotlin

Zeigt das Messaging mit Themen und Warteschlangen mithilfe von Amazon Simple Notification Service (Amazon SNS) und Amazon Simple Queue Service (Amazon SQS).

Den vollständigen Quellcode und Anweisungen, die das Versenden von Nachrichten mit Themen und Warteschlangen in Amazon SNS und Amazon SQS demonstrieren, finden Sie im vollständigen Beispiel unter. GitHub

In diesem Beispiel verwendete Dienste

- Amazon SNS
- Amazon SQS

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter. Amazon SNS mit einem AWS SDK verwenden Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwenden von API Gateway zum Aufrufen einer Lambda-Funktion

Die folgenden Codebeispiele zeigen, wie eine von Amazon API Gateway aufgerufene AWS Lambda Funktion erstellt wird.

Java

SDK für Java 2.x

Zeigt, wie eine AWS Lambda Funktion mithilfe der Lambda Java Runtime API erstellt wird. In diesem Beispiel werden verschiedene AWS Dienste aufgerufen, um einen bestimmten Anwendungsfall auszuführen. Dieses Beispiel zeigt, wie man eine Lambda-Funktion erstellt, die von Amazon API Gateway aufgerufen wird und eine Amazon-DynamoDB-Tabelle nach Arbeitsjubiläen durchsucht und Amazon Simple Notification Service (Amazon SNS) verwendet, um eine Textnachricht an Ihre Mitarbeiter zu senden, die ihnen zu ihrem einjährigen Jubiläum gratuliert.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda
- Amazon SNS

JavaScript

SDK für JavaScript (v3)

Zeigt, wie eine AWS Lambda Funktion mithilfe der JavaScript Lambda-Laufzeit-API erstellt wird. In diesem Beispiel werden verschiedene AWS Dienste aufgerufen, um einen bestimmten Anwendungsfall auszuführen. Dieses Beispiel zeigt, wie man eine Lambda-Funktion erstellt, die von Amazon API Gateway aufgerufen wird und eine Amazon-DynamoDB-Tabelle nach Arbeitsjubiläen durchsucht und Amazon Simple Notification Service (Amazon SNS) verwendet, um eine Textnachricht an Ihre Mitarbeiter zu senden, die ihnen zu ihrem einjährigen Jubiläum gratuliert.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

Dieses Beispiel ist auch verfügbar im AWS SDK for JavaScript Entwicklerhandbuch für v3.

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda
- Amazon SNS

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unterAmazon SNS mit einem AWS SDK verwenden. Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung geplanter Ereignisse zum Aufrufen einer Lambda-Funktion

Die folgenden Codebeispiele zeigen, wie eine AWS Lambda Funktion erstellt wird, die durch ein von Amazon EventBridge geplantes Ereignis aufgerufen wird.

Java

SDK für Java 2.x

Zeigt, wie ein von Amazon EventBridge geplantes Ereignis erstellt wird, das eine AWS Lambda Funktion aufruft. Konfigurieren Sie so EventBridge , dass ein Cron-Ausdruck verwendet wird, um zu planen, wann die Lambda-Funktion aufgerufen wird. In diesem Beispiel erstellen Sie eine Lambda-Funktion mithilfe der Lambda-Java-Laufzeit-API. In diesem Beispiel werden verschiedene AWS Dienste aufgerufen, um einen bestimmten Anwendungsfall auszuführen. Dieses Beispiel zeigt, wie man eine App erstellt, die eine mobile Textnachricht an Ihre Mitarbeiter sendet, um ihnen zum einjährigen Jubiläum zu gratulieren.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- DynamoDB
- EventBridge

- Lambda
- Amazon SNS

JavaScript

SDK für JavaScript (v3)

Zeigt, wie ein von Amazon EventBridge geplantes Ereignis erstellt wird, das eine AWS Lambda Funktion aufruft. Konfigurieren Sie so EventBridge , dass ein Cron-Ausdruck verwendet wird, um zu planen, wann die Lambda-Funktion aufgerufen wird. In diesem Beispiel erstellen Sie eine Lambda-Funktion mithilfe der JavaScript Lambda-Laufzeit-API. In diesem Beispiel werden verschiedene AWS Dienste aufgerufen, um einen bestimmten Anwendungsfall auszuführen. Dieses Beispiel zeigt, wie man eine App erstellt, die eine mobile Textnachricht an Ihre Mitarbeiter sendet, um ihnen zum einjährigen Jubiläum zu gratulieren.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

Dieses Beispiel ist auch verfügbar im AWS SDK for JavaScript Entwicklerhandbuch für v3.

In diesem Beispiel verwendete Dienste

- DynamoDB
- EventBridge
- Lambda
- Amazon SNS

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unterAmazon SNS mit einem AWS SDK verwenden. Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Amazon-SNS-Sicherheit

Dieser Abschnitt enthält Informationen zur Sicherheit, Authentifizierung und Zugriffskontrolle des Amazon SNS, sowie zur Amazon SNS Access Policy Language.

Themen

Datenschutz

Das AWSModell der wird auch auf den in Amazon Relational Database Service angewendet. Wie in diesem Modell beschrieben, ist AWS verantwortlich für den Schutz der globalen Infrastruktur, in der die gesamte AWS Cloud ausgeführt wird. Sie sind dafür verantwortlich, die Kontrolle über Ihre in dieser Infrastruktur gehosteten Inhalte zu behalten. Dieser Inhalt enthält die Sicherheitskonfigurations- und Verwaltungsaufgaben für die von Ihnen verwendeten AWS-Services. Weitere Informationen zum Datenschutz finden Sie unter Häufig gestellte Fragen zum Datenschutz. Informationen zum Datenschutz in Europa finden Sie im Blog-Beitrag AWS-Modell der geteilten Verantwortung und die GDPR im Blog zur AWS-Sicherheit.

Wir empfehlen aus Gründen des Datenschutzes, dass Sie AWS-Konto-Anmeldeinformationen schützen und die Benutzerkonten jeweils mit AWS Identity and Access Management (IAM) einrichten. So erhält jeder Benutzer nur die Berechtigungen, die zum Durchführen seiner Aufgaben erforderlich sind. Außerdem sollten Sie die Daten mit folgenden Methoden schützen:

- Verwenden Sie für jedes Konto die Multi-Factor Authentication (MFA).
- Verwenden Sie SSL/TLS für die Kommunikation mit AWS-Ressourcen. Wir empfehlen TLS 1.2 oder höher.
- Richten Sie die API und die Protokollierung von Benutzeraktivitäten mit ein AWS CloudTrail.

- Verwenden Sie AWS-Verschlüsselungslösungen zusammen mit allen Standardsicherheitskontrollen in AWS-Services.
- Verwenden Sie erweiterte verwaltete Sicherheitsservices wie Amazon Macie, die dabei helfen, in Amazon S3 gespeicherte persönliche Daten zu erkennen und zu sichern.
- Wenn Sie für den Zugriff auf AWS über eine Befehlszeilenschnittstelle oder über eine API FIPS 140-2-validierte kryptografische Module benötigen, verwenden Sie einen FIPS-Endpunkt. Weitere Informationen über verfügbare FIPS-Endpunkte finden Sie unter Federal Information Processing Standard (FIPS) 140-2.
- Nachrichtendatenschutz
  - Der Nachrichtendatenschutz ist eine neue Hauptfunktion von Amazon SNS.
  - Verwenden Sie MDP, um Nachrichten nach vertraulichen oder sensiblen Informationen zu durchsuchen.
  - Stellen Sie Nachrichtenüberwachung für den gesamten Inhalt bereit, der das Thema passiert.
  - Stellen Sie Inhaltszugriffskontrollen für Nachrichten zur Verfügung, die im Thema veröffentlicht wurden, und für Nachrichten, die vom Thema zugestellt wurden.

> **Important**
>
> Wir empfehlen dringend, in Freitextfeldern wie z. B. im Feld Name keine vertraulichen oder sensiblen Informationen wie die E-Mail-Adressen Ihrer Kunden einzugeben. Dies gilt auch, wenn Sie mit Amazon SNS oder andere Amazon Web Services über die Konsole, API, AWS CLI oder AWS SDKs arbeiten. Alle Daten, die Sie in Tags (Markierungen) oder Freiformfelder eingeben, die für Namen verwendet werden, können für Abrechnungs- oder Diagnoseprotokolle verwendet werden. Wenn Sie eine URL für einen externen Server bereitstellen, empfehlen wir dringend, Sie keine Anmeldeinformationen zur Validierung Ihrer Anforderung an den betreffenden Server in die URL einzuschließen.

Die folgenden Abschnitte enthalten zusätzliche Informationen zum Inhaltsschutz in Amazon SNS.

Themen

Datenverschlüsselung

Dieser Datenschutz bezieht sich auf Daten bei der Übertragung (wenn sie zu Amazon SNS oder von diesem geschickt werden), ebenso wie auf ruhende Daten (die in Amazon-S3-Rechenzentren auf Datenträgern gespeichert sind). Sie können Daten mithilfe von Secure Sockets Layer (SSL) oder einer clientseitigen Verschlüsselung während der Übertragung schützen. Standardmäßig speichert Amazon SNS Nachrichten und Dateien mit Festplattenverschlüsselung. Sie können Daten im Ruhezustand schützen, indem Sie Amazon SNS auffordern, Ihre Nachrichten zu verschlüsseln, bevor Sie sie im verschlüsselten Dateisystem in seinen Rechenzentren speichern. Amazon SNS empfiehlt die Verwendung von SSE für eine optimierte Datenverschlüsselung.

Themen

Verschlüsselung im Ruhezustand

Mit der serverseitigen Verschlüsselung (SSE) können Sie sensible Daten in verschlüsselten Themen speichern, indem Sie den Inhalt von Nachrichten in Amazon SNSThemen mithilfe von Schlüsseln schützen, die in AWS Key Management Service () verwaltet werdenAWS KMS.

SSE verschlüsselt Nachrichten, sobald sie bei Amazon SNS eingehen. Die Nachrichten werden in verschlüsselter Form gespeichert und nur entschlüsselt, wenn sie gesendet werden.

- Informationen zum Verwalten von SSE über die AWS Management Console oder den AWS SDK for Java (durch Festlegen des Attributs `KmsMasterKeyId` über die API-Aktionen `CreateTopic` und `SetTopicAttributes`) finden Sie unter Aktivieren der serverseitigen Verschlüsselung (SSE) für ein Amazon-SNS-Thema.
- Informationen zum Erstellen von verschlüsselten Themen mit AWS CloudFormation (durch Setzen der Eigenschaft `KmsMasterKeyId` mit der Ressource `AWS::SNS::Topic`) finden Sie unter AWS CloudFormation-Benutzerhandbuch.

> **Important**
>
> Alle Anfragen zu Themen mit aktiviertem SSE müssen HTTPS und Signature Version 4 verwenden.
> Weitere Informationen zur Kompatibilität anderer Dienste mit verschlüsselten Themen finden Sie in Ihrer Service-Dokumentation.
> Amazon SNS unterstützt nur KMS-Schlüssel mit symmetrischer Verschlüsselung.
> Sie können keinen anderen KMS-Schlüsseltyp für die Verschlüsselung Ihre Service-Ressourcen verwenden. Wie Sie feststellen, ob ein KMS-Schlüssel ein symmetrisch Verschlüsselungsschlüssel ist, erfahren Sie unter Erkennen asymmetrischer Schlüssel.

AWS KMS kombiniert sichere, hoch verfügbare Hard- und Software, um ein System für die Schlüsselverwaltung bereitzustellen, das für die Cloud skaliert ist. Wenn Sie Amazon SNS mit AWS KMS, verwenden, werden die Datenschlüssel, mit denen Ihre Nachrichtendaten verschlüsselt werden, ebenfalls verschlüsselt und mit den Daten, die sie schützen, gespeichert.

Vorteile von AWS KMS:

- Sie können den AWS KMS key selbst erstellen und verwalten.
- Sie können auch AWS-verwaltete KMS-Schlüssel für Amazon SNS verwenden, die für jedes Konto und jede Region eindeutig sind.
- Die Sicherheitsstandards von AWS KMS tragen dazu bei, dass Sie Ihre Compliance-Anforderungen bezüglich der Verschlüsselung erfüllen.

Weitere Informationen finden Sie unter Was ist AWS Key Management Service? im AWS Key Management Service-Entwicklerhandbuch.

Themen

- Verschlüsselungsumfang
- Wichtige Begriffe

Verschlüsselungsumfang

SSE verschlüsselt den Nachrichtentext in einer Amazon SNS-Thema.

Folgendes wird von SSE nicht verschlüsselt:

- Themenmetadaten (Name und Attribute des Themas)
- Metadaten der Nachrichten (Betreff, ID, Zeitstempel und Attribute)
- Datenschutzrichtlinie
- Metriken nach Themen

> **Note**
>
> - Eine Nachricht ist nur dann verschlüsselt, wenn sie gesendet wurde, nachdem die Verschlüsselung einer Queue aktiviert wurde. Amazon SNS verschlüsselt keine Nachrichten, die sich bereits in der Queue befinden.
> - Verschlüsselte Nachrichten bleiben auch dann verschlüsselt, wenn die Verschlüsselung des Themas deaktiviert wird.

Wichtige Begriffe

Die folgenden wichtigen Begriffe vermitteln Ihnen ein besseres Verständnis für die Funktionalität von SSE. Detaillierte Beschreibungen finden Sie in derAmazon Simple Notification Service-API-Referenz.

Datenschlüssel

Der Datenverschlüsselungsschlüssel (Data Encryption Key, DEK), der dafür zuständig ist, den Inhalt von Amazon SNS-Nachrichten zu verschlüsseln.

Weitere Informationen finden Sie unter Datenschlüssel im AWS Key Management ServiceEntwicklerhandbuch und Envelope-Verschlüsselung im AWS Encryption SDKEntwicklerhandbuch.

AWS KMS key--ID

Der Alias, Alias ARN, die Schlüssel-ID oder der Schlüssel-ARN von einem AWS KMS key oder einem benutzerdefinierten AWS KMS in Ihrem Konto oder in einem anderen Konto. Während der Alias des von AWS verwalteten AWS KMS für Amazon SNS immer `alias/aws/sns` ist, kann der Alias eines benutzerdefinierten AWS KMS beispielsweise `alias/MyAlias` lauten. Sie können diese AWS KMS-Schlüssel zum Schutz der Nachrichten in Amazon-SNS-Themen verwenden.

> **Note**
>
> Beachten Sie Folgendes:

- Wenn Sie die AWS Management Console das erste Mal nutzen, um den AWS-verwalteten KMS für Amazon SNS für ein Thema anzugeben, erstellt AWS KMS den AWS-verwalteten KMS für Amazon SNS.
- Alternativ erstellt AWS KMS bei der ersten Verwendung der `Publish`-Aktion zu einem Thema mit aktivierter SSE den AWS-verwalteten KMS für Amazon SNS.

Sie können AWS KMS-Schlüssel erstellen, die Richtlinien, die die Verwendung von AWS KMS-Schlüsseln steuern, definieren und die AWS KMS-Nutzung mithilfe des Abschnitts AWS KMS keys der der AWS KMS-Konsole oder der `CreateKey` AWS KMS-Aktion überprüfen. Weitere Informationen finden Sie unter AWS KMS keys und Erstellen von Schlüsseln im AWS Key Management Service-Entwicklerhandbuch. Weitere Beispiele für AWS KMS Kennungen finden Sie unter KeyId in der APIAWS Key Management Service-Referenz zu . Weitere Informationen zum Suchen von AWS KMS-IDs finden Sie unter Suchen der Schlüssel-ID und des ARN im AWS Key Management Service-Entwicklerhandbuch.

Important

Für AWS KMS fallen zusätzliche Gebühren an. Weitere Informationen finden Sie unter Einschätzen der AWS KMS-Kosten und AWS Key Management Service-Preise.

Schlüsselverwaltung

Die folgenden Abschnitte enthalten Informationen zum Arbeiten mit Schlüsseln, die in AWS Key Management Service (AWS KMS) verwaltet werden. Mehr Informationen über

Note

Amazon SNS unterstützt nur KMS-Schlüssel mit symmetrischer Verschlüsselung. Sie können keinen anderen KMS-Schlüsseltyp für die Verschlüsselung Ihre Service-Ressourcen verwenden. Wie Sie feststellen, ob ein KMS-Schlüssel ein symmetrisch Verschlüsselungsschlüssel ist, erfahren Sie unter Erkennen asymmetrischer Schlüssel.

Themen

- Konfigurieren von AWS KMS-Berechtigungen
- AWS KMS-Fehler

Einschätzen der AWS KMS-Kosten

Möglicherweise möchten Sie Ihre Kosten genauer vorhersagen und Ihre AWS-Rechnung besser nachvollziehen können und wollen daher wissen, wie oft Amazon SNS Ihren AWS KMS key verwendet.

> **Note**
>
> Mit der nachstehenden Formel erhalten Sie eine gute Vorstellung davon, welche Kosten auf Sie zukommen. Allerdings können die tatsächlichen Kosten aufgrund der verteilten Struktur von Amazon SNS höher liegen.

Zur Berechnung der Anzahl der API-Anfragen (`R`) pro Thema verwenden Sie folgende Formel:

```
R = B / D * (2 * P)
```

`B` ist der Abrechnungszeitraum (in Sekunden).

`D` ist die Wiederverwendungszeit des Datenschlüssels (in Sekunden—Amazon SNS verwendet einen Datenschlüssel bis zu 5 Minuten lang).

`P` ist die Anzahl der Veröffentlichungen der Prinzipale, die Nachrichten an das Amazon-SNS-Thema senden.

Es folgen Beispielberechnungen: Preisinformationen finden Sie unter AWS Key Management Service-Preise.

Beispiel 1: Berechnung der Anzahl der AWS KMS-API-Aufrufe für 1 Herausgeber und 1 Thema

In diesem Beispiel wird Folgendes angenommen:

- Der Abrechnungszeitraum ist 1. bis 31. Januar (2 678 400 Sekunden).
- Der Wiederverwendungszeitraum für den Datenschlüssel ist 5 Minuten (300 Sekunden).
- Es gibt 1 Thema.

- Es gibt 1 Veröffentlichungsprinzipal.

```
2,678,400 / 300 * (2 * 1) = 17,856
```

Beispiel 2: Berechnung der Anzahl der AWS KMS-API-Aufrufe für mehrere Herausgeber und 2 Themen

In diesem Beispiel wird Folgendes angenommen:

- Der Abrechnungszeitraum ist 1. bis 28. Februar (2 419 200 Sekunden).
- Der Wiederverwendungszeitraum für den Datenschlüssel ist 5 Minuten (300 Sekunden).
- Es gibt 2 Themen.
- Das erste Thema hat 3 Veröffentlichungsprinzipale.
- Das zweite Thema hat 5 Veröffentlichungsprinzipale.

```
(2,419,200 / 300 * (2 * 3)) + (2,419,200 / 300 * (2 * 5)) = 129,024
```

Konfigurieren von AWS KMS-Berechtigungen

Bevor Sie SSE verwenden können, müssen Sie AWS KMS key-Richtlinien konfigurieren, um eine Verschlüsselung von Themen und eine Verschlüsselung sowie Entschlüsselung von Nachrichten zu ermöglichen. Weitere Informationen zu Berechtigungen für AWS KMS-Aktionen finden Sie unter AWS KMS Glue-API-Berechtigungen: Referenz für Aktionen und Ressourcen im AWS Key Management Service Entwicklerhandbuch. Einzelheiten zur Einrichtung eines Amazon-SNS-Themas mit serverseitiger Verschlüsselung finden Sie unter Ein Amazon-SNS-Thema mit serverseitiger Verschlüsselung einrichten.

> **Note**
>
> Sie können Berechtigungen für symmetrische KMS-Verschlüsselungsschlüssel auch mit IAM-Richtlinien verwalten. Weitere Informationen finden Sie unter Verwenden von IAM-Richtlinien mitAWS KMS.
> Sie können globale Berechtigungen zum Senden und Empfangen von Amazon SNS konfigurieren. Für AWS KMS ist jedoch eine explizite Benennung des vollständigen ARN von KMSs in bestimmten Regionen im Abschnitt `Resource` der IAM-Richtlinie erforderlich.

Außerdem müssen Sie sicherstellen, dass die Schlüsselrichtlinien des AWS KMS key die erforderlichen Berechtigungen gewähren. Geben Sie dazu die Namen der Prinzipale an, die verschlüsselte Nachrichten in Amazon SNS als Benutzer in der KMS-Schlüsselrichtlinie produzieren und verbrauchen.

Alternativ können Sie die erforderlichen AWS KMS-Aktionen und den KMS-ARN in einer IAM-Richtlinie angeben, die den Prinzipalen zugewiesen ist, die verschlüsselte Nachrichten in Amazon SNS veröffentlichen und abonnieren. Weitere Informationen finden Sie unter Verwalten des Zugriffs auf AWS KMS im AWS Key Management Service-Entwicklerhandbuch.

Wenn Sie einen vom Kunden verwalteten Schlüssel für Ihr Amazon-SNS-Thema auswählen und Sie Aliase verwenden, um den Zugriff auf KMS-Schlüssel mithilfe von IAM-Richtlinien oder KMS-Schlüsselrichtlinien mit dem Bedingungsschlüssel `kms:ResourceAliases` zu steuern, stellen Sie sicher, dass dem ausgewählten vom Kunden verwalteten Schlüssel auch ein Alias zugeordnet ist. Weitere Informationen zur Verwendung von Aliasen zur Steuerung des Zugriffs auf KMS-Schlüssel finden Sie unter Verwenden von Aliasen zur Steuerung des Zugriffs auf KMS-Schlüssel im Entwicklerhandbuch zu AWS Key Management Service.

Einem Benutzer erlauben, Nachrichten an ein bestimmtes Thema mit SSE zu senden

Der Publisher muss die Berechtigungen `kms:GenerateDataKey*` und `kms:Decrypt` für den AWS KMS key besitzen.

```
{
  "Statement": [{
    "Effect": "Allow",
    "Action": [
      "kms:GenerateDataKey*",
      "kms:Decrypt"
    ],
    "Resource": "arn:aws:kms:us-
east-2:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab"
  }, {
    "Effect": "Allow",
    "Action": [
      "sns:Publish"
    ],
    "Resource": "arn:aws:sns:*:123456789012:MyTopic"
  }]
}
```

Ermöglichen der Kompatibilität zwischen Ereignisquellen aus AWS Services und verschlüsselten Themen

Mehrere AWS-Services veröffentlichen Ereignisse zu Amazon SNS Themen. Damit diese Ereignisquellen mit verschlüsselten Themen arbeiten können, müssen Sie die folgenden Schritte ausführen:

1. Verwenden Sie einen vom Kunden verwalteten Schlüssel. Weitere Informationen finden Sie unter Erstellen von Schlüsseln im AWS Key Management Service-Entwicklerhandbuch.
2. Damit der AWS-Service über die Berechtigungen `kms:GenerateDataKey*` und `kms:Decrypt` verfügt, fügen Sie die folgende Anweisung zur KMS-Richtlinie hinzu.

```
{
  "Statement": [{
    "Effect": "Allow",
    "Principal": {
      "Service": "service.amazonaws.com"
    },
    "Action": [
      "kms:GenerateDataKey*",
      "kms:Decrypt"
    ],
    "Resource": "*"
  }]
}
```

Ereignisquelle	Dienstauftraggeber
Amazon CloudWatch	`cloudwatch.amazonaws.com`
Amazon CloudWatch Events	`events.amazonaws.com`
AWS CodeCommit	`codecommit.amazonaws.com`
AWS CodeStar	`codestar-notifications.amazonaws.com`
AWS Database Migration Service	`dms.amazonaws.com`
AWS Directory Service	`ds.amazonaws.com`

Ereignisquelle	Dienstauftraggeber
Amazon DynamoDB	`dynamodb.amazonaws.com`
Amazon Inspector	`inspector.amazonaws.com`
Amazon Redshift	`redshift.amazonaws.com`
Amazon RDS	`events.rds.amazonaws.com`
Amazon S3 Glacier	`glacier.amazonaws.com`
Amazon Simple Email Service	`ses.amazonaws.com`
Amazon Simple Storage Service	`s3.amazonaws.com`
AWS Snowball	`importexport.amazonaws.com`
AWS Systems Manager Incident Manager	AWS Systems Manager Incident Manager besteht aus zwei Service-Prinzipien: `ssm-incidents.amazonaws.com` ; `ssm-contacts.amazonaws.com`

Note

Bei einigen Ereignisquellen von Amazon SNS müssen Sie eine IAM-Rolle(anstelle des Dienstprinzips) in der AWS KMS key-Richtlinie bereitstellen:

- Amazon EC2 Auto Scaling
- Amazon Elastic Transcoder
- AWS CodePipeline
- AWS Config
- AWS Elastic Beanstalk
- AWS IoT
- EC2 Image Builder

3. Fügen Sie die `aws:SourceAccount`- und `aws:SourceArn`-Bedingungsschlüssel der KMS-Ressourcenrichtlinie hinzu, um den KMS-Schlüssel vor Confused-Deputy-Angriffen zu schützen. Genaue Einzelheiten zu jedem Fall finden Sie in der jeweiligen Liste der servicespezifischen Dokumentation (oben).

   > **Important**
   >
   > Das Hinzufügen von `aws:SourceAccount` und `aws:SourceArn` zu einer AWS KMS-Richtlinie wird für Themen „EventBridge-zu-verschlüsselt“ nicht unterstützt.

```
{
  "Effect": "Allow",
  "Principal": {
    "Service": "service.amazonaws.com"
  },
  "Action": [
    "kms:GenerateDataKey*",
    "kms:Decrypt"
  ],
  "Resource": "*",
  "Condition": {
    "StringEquals": {
      "aws:SourceAccount": "customer-account-id"
    },
    "ArnLike": {
      "aws:SourceArn": "arn:aws:service:region:customer-account-id:resource-
type:customer-resource-id"
    }
  }
}
```

4. Aktivieren Sie SSE für Ihr Thema mithilfe Ihres KMS.
5. Stellen Sie der Ereignisquelle den ARN des verschlüsselten Themas zur Verfügung.

AWS KMS-Fehler

Wenn Sie mit Amazon SNS und AWS KMS, arbeiten, können Fehler auftreten. Die folgende Liste beschreibt die Fehler und möglichen Lösungen.

KMSAccessDeniedException

Der Verschlüsselungstext verweist auf einen Schlüssel, der nicht vorhanden ist oder auf den Sie keinen Zugriff haben.

HTTP Status Code: 400

KMSDisabledException

Die Anforderung wurde abgelehnt, da der angegebene KMS nicht aktiviert ist.

HTTP Status Code: 400

KMSInvalidStateException

Die Anforderung wurde abgelehnt, da der Status der angegebenen Ressource für diese Anforderung ungültig ist. Weitere Informationen finden Sie unter Schlüsselstatus von AWS KMS keys im AWS Key Management Service-Entwicklerhandbuch.

HTTP Status Code: 400

KMSNotFoundException

Die Anforderung wurde abgelehnt, da die angegebene Entity oder Ressource nicht gefunden wurde.

HTTP Status Code: 400

KMSOptInRequired

Die AWS-Zugriffsschlüssel-ID benötigt ein Abonnement für den Service.

HTTP Status Code: 403

KMSThrottlingException

Die Anforderung wurde aufgrund der Drosselung von Anforderungen abgelehnt. Weitere Informationen zur Drosselung finden Sie unter Kontingente im AWS Key Management Service-Entwicklerhandbuch.

HTTP Status Code: 400

Aktivieren der serverseitigen Verschlüsselung (SSE) für ein Amazon-SNS-Thema

Mit der serverseitigen Verschlüsselung (Server-Side Encryption, SSE) können Sie vertrauliche Daten in verschlüsselten Themen speichern. SSE schützt den Inhalt von Nachrichten in Amazon SNS-

Themen mit Schlüsseln, die in AWS Key Management Service (AWS KMS) verwaltetet werden. Weitere Informationen zur serverseitigen Verschlüsselung mit Amazon SNS finden Sie unter Verschlüsselung im Ruhezustand. Weiteres zum Erstellen von AWS KMS-Schlüsseln finden Sie unter Erstellen von Schlüsseln im AWS Key Management Service-Entwicklerhandbuch.

> **Important**
>
> Alle Anfragen zu Themen mit aktiviertem SSE müssen HTTPS und Signature Version 4 verwenden.

Serverseitige Verschlüsselung (SSE) für ein Amazon-SNS-Thema mit der AWS Management Console aktivieren

1. Melden Sie sich bei der Amazon-SNS-Konsole an.
2. Wählen Sie im Navigationsbereich Topics (Themen) aus.
3. Wählen Sie auf der Seite Topics (Themen) ein Thema und danach Actions (Aktionen), Edit (Bearbeiten) aus.
4. Erweitern Sie den Abschnitt Encryption (Verschlüsselung) und gehen Sie wie folgt vor:

   a. Wählen Sie Enable encryption (Verschlüsselung aktivieren) aus.

   b. Geben Sie den AWS KMS-Schlüssel an. Weitere Informationen finden Sie unter Wichtige Begriffe.

      Für jeden KMS-Typ werden Description (Beschreibung), Account (Konto) und KMS ARN (KMS-ARN) angezeigt.

      > **Important**
      >
      > Wenn Sie nicht der Besitzer des KMS-Schlüssels sind oder wenn Sie sich mit einem Konto anmelden, das über keine `kms:ListAliases`- und `kms:DescribeKey`-Berechtigungen verfügt, können Sie auf der Amazon-SNS-Konsole keine Informationen über den KMS aufrufen.
      > Bitten Sie den Inhaber des KMS, Ihnen diese Berechtigungen zu erteilen. Beispiele und weitere Informationen zu AWS KMS-Berechtigungen finden Sie unter API-Berechtigungen: Referenztabelle für Aktionen und Ressourcen im AWS Key Management Service Benutzerhandbuch.

- Der AWS-verwaltete KMS für Amazon SNS (Standard) aws/sns ist standardmäßig ausgewählt.

  > **Note**
  >
  > Beachten Sie Folgendes:
  >
  > - Wenn Sie die AWS Management Console das erste Mal nutzen, um den AWS-verwalteten KMS für Amazon SNS für ein Thema anzugeben, erstellt AWS KMS den AWS-verwalteten KMS für Amazon SNS.
  > - Alternativ erstellt AWS KMS bei der ersten Verwendung der `Publish`-Aktion zu einem Thema mit aktivierter SSE den AWS-verwalteten KMS für Amazon SNS.

- Um einen benutzerdefinierten KMS aus Ihrem AWS-Konto zu nutzen, wählen Sie das Feld KMS-Schlüssel und dann den benutzerdefinierten KMS aus der Liste aus.

  > **Note**
  >
  > Weitere Informationen zum Erstellen benutzerdefinierter KMSs finden Sie unter Erstellen von Schlüsseln im AWS Key Management Service-Entwicklerhandbuch.

- Um einen benutzerdefinierten KMS-ARN aus Ihrem AWS-Konto oder von einem anderen AWS-Konto zu nutzen, geben Sie ihn in das Feld KMS-Schlüssel ein.

5. Wählen Sie Save Changes.

   SSE wird für Ihr Thema aktiviert und die Seite ***MeinThema*** wird angezeigt.

   Auf der Registerkarte Verschlüsselung werden der Verschlüsselungsstatus des Themas, das AWS-Konto, der Kundenhauptschlüssel (CMK), der CMK-ARN und die Beschreibung angezeigt.

Ein Amazon-SNS-Thema mit serverseitiger Verschlüsselung einrichten

Verwenden Sie beim Erstellen Ihres KMS-Schlüssels die folgende KMS-Schlüsselrichtlinie:

```
{
    "Effect": "Allow",
    "Principal": {
        "Service": "service.amazonaws.com"
```

```
     },
    "Action": [
        "kms:Decrypt",
        "kms:GenerateDataKey"
    ],
    "Resource": "*",
    "Condition": {
        "ArnLike": {
            "aws:SourceArn": "arn:aws:service:region:customer-account-id:resource-
type/customer-resource-id"
        },
        "StringEquals": {
            "kms:EncryptionContext:aws:sns:topicArn":
 "arn:aws:sns:your_region:customer-account-id:your_sns_topic_name"
        }
    }
}
```

Aktivieren der serverseitigen Verschlüsselung (SSE) für ein Amazon-SNS-Thema mit abonnierter verschlüsselter Amazon-SQS-Warteschlange

Sie können die serverseitige Verschlüsselung (SSE) für ein Thema aktivieren, um die entsprechenden Daten zu schützen. Damit Amazon SNS Nachrichten an verschlüsselte Amazon-SQS-Warteschlangen senden kann, muss der mit der Amazon-SQS-Warteschlange verknüpfte verwaltete Schlüssel über eine Richtlinienanweisung verfügen, die dem Amazon-SNS-Dienstprinzipal Zugriff auf die AWS KMS-API-Aktionen `GenerateDataKey` und `Decrypt` gewährt. Weitere Informationen zur Verwendung von der SSE finden Sie unter Verschlüsselung im Ruhezustand.

Auf dieser Seite erfahren Sie, wie Sie SSE für ein Amazon SNS Amazon SNS-Thema aktivieren, für das eine verschlüsselte Amazon SQS Queue mit der abonniert ist.AWS Management Consoleaus.

Schritt 1: Erstellen eines benutzerdefinierten KMS-Schlüssels

1. Melden Sie sich bei der AWS KMS-Konsole mit einem Benutzer an, der mindestens über die `AWSKeyManagementServicePowerUser`-Richtlinie verfügt.
2. Klicken Sie auf Create a key.
3. Um einen KMS-Schlüssel mit symmetrischer Verschlüsselung zu erstellen, wählen Sie für Key type (Schlüsseltyp) die Option Symmetric (Symmetrisch) aus.

   Weitere Informationen zum Erstellen eines asymmetrischen KMS-Schlüssels in der AWS KMS-Konsole finden Sie unter Erstellen asymmetrischer KMS-Schlüssel (Konsole).

4. Unter Key usage (Schlüsselverwendung) ist die Option Encrypt and decrypt (Verschlüsseln und Entschlüsseln) für Sie ausgewählt.

   Weitere Informationen wie Sie KMS-Schlüssel erstellen, die MAC-Codes generieren und überprüfen, finden Sie unter Erstellen von HMAC-KMS-Schlüssel.

   Weitere Hinweise zu den Erweiterten Optionen finden Sie unter Schlüssel für spezielle Zwecke.

5. Wählen Sie Next (Weiter).

6. Geben Sie einen Alias für den Replikatschlüssel ein. Der Aliasname darf nicht mit **`aws/`** beginnen. Das Präfix **`aws/`** ist von Amazon Web Services reserviert und steht für Von AWS verwaltete Schlüssel in Ihrem Konto.

   > **Note**
   >
   > Wenn Sie einen Alias hinzufügen, löschen oder aktualisieren, wird dadurch möglicherweise eine Berechtigung für den KMS-Schlüssel erteilt oder verweigert. Einzelheiten finden Sie unter ABAC für AWS KMS und Verwenden von Aliassen zur Steuerung des Zugriffs auf KMS-Schlüssel.

   Ein Alias ist ein Anzeigename, den Sie verwenden können, um einen KMS-Schlüssel zu identifizieren. Wir empfehlen, dass Sie einen Alias wählen, der auf die Art von Daten, die Sie schützen möchten, oder die Anwendung, die Sie mit dem KMS-Schlüssel verwenden möchten, hindeutet.

   Zum Erstellen eines KMS-Schlüssels in der AWS Management Console benötigen Sie Aliasse. Sie sind hingegen optional, wenn Sie die CreateKey-Produktion verwenden.

7. (Optional) Geben Sie eine Beschreibung für den KMS-Schlüssel ein.

   Sie können jetzt eine Beschreibung hinzufügen oder sie jederzeit aktualisieren, es sei denn, der Schlüsselstatus lautet `Pending Deletion` oder `Pending Replica Deletion`. Um die Beschreibung eines vorhandenen kundenverwalteten KMS-Schlüssels hinzuzufügen, zu ändern oder zu löschen, bearbeiten Sie die Beschreibung in der AWS Management Console oder verwenden Sie die UpdateKeyDescription-Produktion.

8. (Optional) Geben Sie einen Tag-Schlüssel und einen optionalen Tag-Wert ein. Wählen Sie Add tag (Tag hinzufügen), wenn Sie mehr als ein Tag zum KMS-Schlüssel hinzufügen möchten.

> **Note**
>
> Wenn Sie einen KMS-Schlüssel markieren oder entmarkieren, wird dadurch möglicherweise die Berechtigung für den KMS-Schlüssel erteilt oder verweigert. Einzelheiten finden Sie unter ABAC für AWS KMS und Verwenden von Tags zur Steuerung des Zugriffs auf KMS-Schlüssel.

Wenn Sie Tags auf AWS-Ressourcen anwenden, erzeugt AWS einen Kostenzuordnungsbericht mit Nutzungs- und Kostendaten der Tags. Markierungen können auch verwendet werden, um den Zugriff auf einen KMS-Schlüssel zu steuern. Weitere Informationen über das Markieren von KMS-Schlüsseln finden Sie unter Markieren von Schlüsseln und ABAC für AWS KMS.

9. Wählen Sie Next (Weiter).
10. Wählen Sie die IAM-Benutzer und -Rollen aus, die den KMS-Schlüssel verwalten können.

> **Note**
>
> Diese wichtige Richtlinie gibt AWS-Konto volle Kontrolle über diesen KMS-Schlüssel. Kontoadministratoren können damit anderen Prinzipalen mithilfe von IAM-Richtlinien die Berechtigung zum Verwalten des KMS-Schlüssels erteilen. Details dazu finden Sie unter Standardschlüsselrichtlinie.
>
> Bewährte IAM-Methoden raten von der Verwendung von IAM-Benutzern mit langfristigen Anmeldeinformationen ab. Verwenden Sie nach Möglichkeit IAM-Rollen, die temporäre Anmeldeinformationen bereitstellen. Weitere Informationen finden Sie unter Bewährte Methoden für die Sicherheit in IAM im IAM-Benutzerhandbuch.

11. (Optional) Um zu verhindern, dass die ausgewählten IAM-Benutzer und -Rollen diesen KMS-Schlüssel löschen, deaktivieren Sie unten auf der Seite im Abschnitt Key deletion (Schlüssellöschung) das Kontrollkästchen Allow key administrators to delete this key (Administratoren erlauben, diesen Schlüssel zu löschen).
12. Wählen Sie Next (Weiter).
13. Wählen Sie die IAM-Benutzer und -Rollen aus, die den KMS-Schlüssel für kryptographische Operationen verwenden können. Wählen Sie Next (Weiter).

14. Fügen Sie auf der Seite Review and edit key policy (Schlüsselrichtlinie überprüfen und bearbeiten) die folgende Anweisung zur Schlüsselrichtlinie hinzu. Wählen Sie anschließend Finish (Fertig stellen) aus.

```
{
    "Sid": "Allow Amazon SNS to use this key",
    "Effect": "Allow",
    "Principal": {
        "Service": "sns.amazonaws.com"
    },
    "Action": [
        "kms:Decrypt",
        "kms:GenerateDataKey*"
    ],
    "Resource": "*"
}
```

Ihr neues benutzerdefinierter kundenverwalteter Schlüssel wird in der Liste der Schlüssel angezeigt.

Schritt 2: Erstellen eines verschlüsselten Amazon SNS-Themas

1. Melden Sie sich bei der Amazon-SNS-Konsole an.
2. Wählen Sie im Navigationsbereich Topics (Themen) aus.
3. Wählen Sie Create topic (Thema erstellen) aus.
4. Geben Sie auf der Seite Create new topic (Neues Thema erstellen) in Topic name (Name des Themas) einen Namen für das Thema ein (z. B. `MyEncryptedTopic`). Wählen Sie anschließend Create topic (Thema erstellen) aus.
5. Erweitern Sie den Abschnitt Encryption (Verschlüsselung) und gehen Sie wie folgt vor:

   a. Wählen Sie Enable server-side encryption.

   b. Erstellen des kundenverwalteten Schlüssels. Weitere Informationen finden Sie unter Wichtige Begriffe.

      Für jeden kundenverwalteten Schlüsseltyp werden die Beschreibung, das Konto und der ARN des kundenverwalteten Schlüssels angezeigt.

> **Important**
>
> Wenn Sie nicht der Besitzer des kundenverwalteten Schlüssels sind oder wenn Sie sich mit einem Konto anmelden, das über keine `kms:ListAliases`- und `kms:DescribeKey`-Berechtigungen verfügt, können Sie auf der Amazon-SNS-Konsole keine Informationen über den kundenverwalteten Schlüssel aufrufen. Bitten Sie den Inhaber des kundenverwalteten Schlüssels, Ihnen diese Berechtigungen zu erteilen. Beispiele und weitere Informationen zu AWS KMS-Berechtigungen finden Sie unter API-Berechtigungen: Referenztabelle für Aktionen und Ressourcen im AWS Key Management Service Benutzerhandbuch.

c. Wählen Sie bei Kundenverwalteter Schlüssel das MyCustomCMK-Element aus, das Sie zuvor erstellt haben. Wählen Sie dann Serverseitige Verschlüsselung aktivieren aus.

6. Wählen Sie Save Changes.

   SSE wird für Ihr Thema aktiviert und die Seite MeinThema wird angezeigt.

   Auf der Registerkarte Verschlüsselung werden der Verschlüsselungsstatus des Themas, das AWS-Konto, der kundenverwaltete Schlüssel, der CMK-ARN und die Beschreibung angezeigt.

Ihr neues verschlüsseltes Thema wird in der Themenliste angezeigt.

Schritt 3: Erstellen und Abonnieren verschlüsselter Amazon-SQS-Warteschlangen

1. Melden Sie sich bei der Amazon SQS-Konsole an.
2. Klicken Sie auf Create New Queue (Neue Queue erstellen).
3. Führen Sie auf der Seite Create New Queue (Neue Queue erstellen) die folgenden Schritte aus:

   a. Geben Sie einen Queue Name (Queuesname) ein (z. B. `MyEncryptedQueue1`).

   b. Klicken Sie auf Standard Queue (StandardQueue) und wählen Sie dann Configure Queue (Queue konfigurieren) aus.

   c. Wählen Sie Use SSE.

   d. Wählen Sie als AWS KMS key das MyCustomCMK-Element aus, das Sie zuvor erstellt haben. Wählen Sie anschließend Warteschlange erstellen aus.

4. Wiederholen Sie den Vorgang, um eine zweite Queue (z. B. namens) zu erstellen. `MyEncryptedQueue2`).

Ihre neuen verschlüsselten Queues werden in der Liste der Queues angezeigt.

5. Wählen Sie in der Amazon SQS-Konsole `MyEncryptedQueue1` und `MyEncryptedQueue2` aus. Wählen Sie anschließend (Queuesaktionen) und Subscribe Queues to SNS Topic (Abonniere SNS-Thema für Queues erstellen) aus.
6. Wählen Sie im Dialogfeld Subscribe to a Topic (Ein Thema abonnieren) in Choose a Topic (Ein Thema wählen) MyEncryptedTopic (Mein verschlüsseltes Thema) und anschließend Subscribe (Abonnieren) aus.

   Die Abonnements Ihrer verschlüsselten Queues für Ihr verschlüsseltes Thema werden im Dialogfeld Topic Subscription Result (Ergebnis für Thema-Abonnement angezeigt.
7. Wählen Sie OK.

Schritt 4: Veröffentlichen einer Nachricht in Ihrem verschlüsselten Thema

1. Melden Sie sich bei der Amazon-SNS-Konsole an.
2. Wählen Sie im Navigationsbereich Topics (Themen) aus.
3. Wählen Sie aus der Liste der Themen MyEncryptedTopic und anschließend Publish to topic (Zu Thema veröffentlichen) aus.
4. Führen Sie auf der Seite Publish a message (Ziel auswählen) folgende Schritte durch:

   a. (Optional) Geben Sie im Abschnitt Message details (Nachrichtendetails) den Subject (Betreff) ein (z. B. `Testing message publishing`).

   b. Geben Sie im Abschnitt Message body (Nachrichtentext) den Nachrichtentext ein (z. B. `My message body is encrypted at rest.`).

   c. Wählen Sie Publish message (Nachricht veröffentlichen) aus.

Ihre Nachricht wird in Ihren abonnierten verschlüsselten Queues veröffentlicht.

Schritt 5: Überprüfen der Zustellung Ihrer Nachrichten

1. Melden Sie sich bei der Amazon SQS-Konsole an.
2. Wählen Sie in der Liste der Warteschlangen MyEncryptedQueue1 und dann Send and receive messages (Nachrichten senden und empfangen) aus.

3. Wählen Sie auf der Seite Send and receive messages in MyEncryptedQueue1 (Nachrichten in „Meine verschlüsselte Warteschlange 1“ senden/empfangen) Poll for messages (Abrufen von Nachrichten) aus.

   Die Nachricht die Sie zuvor gesendet haben wird angezeigt.
4. Wählen Sie Weitere Details, um Ihre Nachricht anzuzeigen.
5. Wählen Sie Close, wenn Sie fertig sind.
6. Wiederholen Sie den Prozess für MyEncryptedQueue2 (Meine verschlüsselte Queue 2).

Richtlinie für den Datenverkehr zwischen Netzwerken

Ein Amazon Virtual Private Cloud (Amazon VPC)-Endpunkt für Amazon SNS ist eine logische Einheit innerhalb einer VPC, die nur Konnektivität mit Amazon SNS ermöglicht. Die VPC leitet Anforderungen an Amazon SNS weiter und Antworten an die VPC zurück. In den folgenden Abschnitten finden Sie Informationen zum Arbeiten mit VPC-Endpunkten und zum Erstellen von VPC-Endpunktrichtlinien.

Wenn Sie Amazon Virtual Private Cloud (Amazon VPC) zum Hosten Ihrer AWS-Ressourcen verwenden, können Sie eine Verbindung zwischen Ihrer VPC und Amazon SNS herstellen. Mit dieser Verbindung können Sie Nachrichten in Ihren Amazon SNS-Themen veröffentlichen, ohne sie über das öffentliche Internet senden zu müssen.

Amazon VPC ist ein AWS-Service, den Sie verwenden können, um AWS-Ressourcen in einem von Ihnen definierten virtuellen Netzwerk auszuführen. Mit einer VPC haben Sie die Kontrolle über Ihre Netzwerkeinstellungen, wie IP-Adressbereich, Subnetze, Routing-Tabellen und Netzwerk-Gateways. Zum Herstellen einer Verbindung Ihrer VPC mit Amazon SNS definieren Sie einen Schnittstellen-VPC-Endpunkt. Mit dieser Art Endpunkt können Sie Ihre VPC mit AWS-Services verbinden. Der Endpunkt bietet zuverlässige, skalierbare Konnektivität zu Amazon SNS, ohne dass ein Internet-Gateway, eine NAT-Instance (Network Address Translation) oder eine VPN-Verbindung erforderlich ist. Weitere Informationen finden Sie unter Interface VPC Endpoints (Interface VPC-Endpunkte) im Amazon VPC-Benutzerhandbuch.

Die Informationen in diesem Abschnitt sind für Benutzer von Amazon VPC vorgesehen. Weitere Informationen und erste Schritte zur Erstellung einer VPC finden Sie unter Getting Started With Amazon VPC (Erste Schritte mit Amazon VPC) im Amazon VPC-Benutzerhandbuch.

> **Note**
>
> VPC-Endpunkte erlauben es nicht, ein Amazon SNS-Thema an eine private IP-Adresse zu abonnieren.

Themen

- Erstellen eines Amazon VPC Endpunkts für Amazon SNS
- Erstellen einer Amazon-VPC-Endpunktrichtlinie für Amazon SNS
- Veröffentlichen einer Amazon SNS Nachricht aus Amazon VPC

Erstellen eines Amazon VPC Endpunkts für Amazon SNS

Zum Veröffentlichen von Nachrichten in Ihren Amazon SNS-Themen von einer Amazon VPC aus, erstellen Sie einen VPC-Endpunkt. Anschließend können Sie Nachrichten in Ihren Themen veröffentlichen und gleichzeitig den Datenverkehr innerhalb des Netzwerks behalten, das Sie mit der VPC verwalten.

Verwenden Sie die folgenden Informationen, um den Endpunkt zu erstellen und die Verbindung zwischen Ihrer VPC und Amazon SNS zu testen. Außerdem finden Sie ein Tutorial, mit dem Sie von Grund auf neu starten, unter Veröffentlichen einer Amazon SNS Nachricht aus Amazon VPC.

Erstellen des Endpunkts

Sie können einen Amazon SNS SNS-Endpunkt in Ihrer VPC mit dem AWS Management Console, dem, einem AWS SDK AWS CLI, der Amazon SNS SNS-API oder erstellen. AWS CloudFormation

Informationen zum Erstellen und Konfigurieren eines Endpunkts über die Amazon-VPC-Konsole oder die AWS CLI finden Sie unter Creating an Interface Endpoint (Erstellen eines Schnittstellenendpunkts) im Amazon-VPC-Benutzerhandbuch.

> **Important**
>
> Sie können Amazon Virtual Private Cloud nur mit HTTPS-Amazon-SNS-Endpunkten verwenden.
> Wenn Sie einen Endpunkt erstellen, geben Sie an, dass Amazon SNS der Service ist, mit dem Ihre VPC eine Verbindung einrichten soll. In der Amazon VPC-Konsole variieren die

Servicenamen abhängig von der ausgewählten Region. Wenn Sie beispielsweise USA Ost (Nord-Virginia) wählen, lautet der Servicename com.amazonaws.us-east-1.states.
Wenn Sie Amazon SNS konfigurieren, um Nachrichten von Amazon VPC zu senden, müssen Sie privates DNS aktivieren und Endpunkte im `sns.us-east-2.amazonaws.com`-Format angeben.
Ein privates DNS unterstützt keine Legacy-Endpunkte wie z. B. `queue.amazonaws.com` oder `us-east-2.queue.amazonaws.com`.

Informationen zum Erstellen und Konfigurieren eines Endpunkts mithilfe AWS CloudFormation von finden Sie in der *AWS::EC2::VPCEndpoint*Ressource im AWS CloudFormation Benutzerhandbuch.

Testen der Verbindung zwischen Ihrer VPC und Amazon SNS

Nachdem Sie einen Endpunkt für Amazon SNS erstellt haben, können Sie Nachrichten von Ihrer VPC zu Ihrem Amazon SNS-Thema senden. Zum Testen dieser Verbindung gehen Sie wie folgt vor:

1. Stellen Sie eine Verbindung mit einer Amazon EC2-Instance in Ihrer VPC her. Weitere Informationen zum Herstellen einer Verbindung finden Sie unter Herstellen einer Verbindung mit Ihrer Linux-Instance or Herstellen einer Verbindung mit Ihrer Windows-Instance in der Amazon EC2-Dokumentation.

   Um Beispielsweise eine Verbindung mit einer Linux-Instance über einen SSH-Client herzustellen, führen Sie von einem Terminal aus den folgenden Befehl aus:

   ```
   $ ssh -i ec2-key-pair.pem ec2-user@instance-hostname
   ```

   Wobei gilt:

   - ec2-key-pair.pem ist die Datei mit dem Schlüsselpaar, das Amazon EC2 beim Erstellen der Instance bereitgestellt hat.
   - instance-hostname ist der öffentliche Hostname der Instance. So laden Sie den Hostnamen in der Amazon-EC2-Konsole: Wählen Sie Instances, wählen Sie Ihre Instance aus und suchen Sie den Wert für Public DNS (IPv4) (Öffentlicher DNS (IPv4)).
2. Verwenden Sie von Ihrer Instance aus Amazon SNS-Befehl `publish` in der AWS CLI. Mit dem folgenden Befehl können eine einfache Nachricht für ein Thema senden:

```
$ aws sns publish --region aws-region --topic-arn sns-topic-arn --message "Hello"
```

Wobei gilt:

- aws-region ist die AWS Region, in der sich das Thema befindet.
- sns-topic-arnist der Amazon-Ressourcenname (ARN) des Themas. So erhalten Sie den ARN von der Amazon-SNS-Konsole: Wählen Sie Topics (Themen), suchen Sie Ihr Thema und verwenden Sie den Wert in der Spalte ARN.

Wenn die Nachricht vom Amazon SNS erfolgreich empfangen wurde, gibt das Terminal eine Nachrichten-ID wie die folgende aus:

```
{
   "MessageId": "6c96dfff-0fdf-5b37-88d7-8cba910a8b64"
}
```

Erstellen einer Amazon-VPC-Endpunktrichtlinie für Amazon SNS

Sie können eine Richtlinie für Amazon VPC Endpunkte für Amazon SNS erstellen, in der Sie Folgendes angeben:

- Prinzipal, der die Aktionen ausführen kann
- Aktionen, die ausgeführt werden können
- Die Ressourcen, für die Aktionen ausgeführt werden können.

Weitere Informationen finden Sie unter Steuerung des Zugriffs auf Services mit VPC-Endpunkten im Amazon VPC User Guide.

Im folgenden Beispiel für eine VPC-Endpunktrichtlinie wird angegeben, dass der IAM-Benutzer`MyUser` zum Amazon SNS-Thema `MyTopic` veröffentlichen darf.

```
{
  "Statement": [{
    "Action": ["sns:Publish"],
    "Effect": "Allow",
    "Resource": "arn:aws:sns:us-east-2:123456789012:MyTopic",
```

```
    "Principal": {
      "AWS": "arn:aws:iam:123456789012:user/MyUser"
    }
  }]
}
```

Folgendes wird abgelehnt:

- Andere Amazon-SNS-API-Aktionen, z. B. `sns:Subscribe` und `sns:Unsubscribe`.
- Andere IAM-Benutzer und -Regeln, die versuchen, diesen VPC-Endpunkt zu verwenden.
- `MyUser`-Veröffentlichungen in einem anderen Amazon SNS Amazon SNS-Thema.

> **Note**
>
> Der IAM-Benutzer kann nach wie vor andere Amazon SNS-API-Aktionen von außerhalb der VPC verwenden.

Veröffentlichen einer Amazon SNS Nachricht aus Amazon VPC

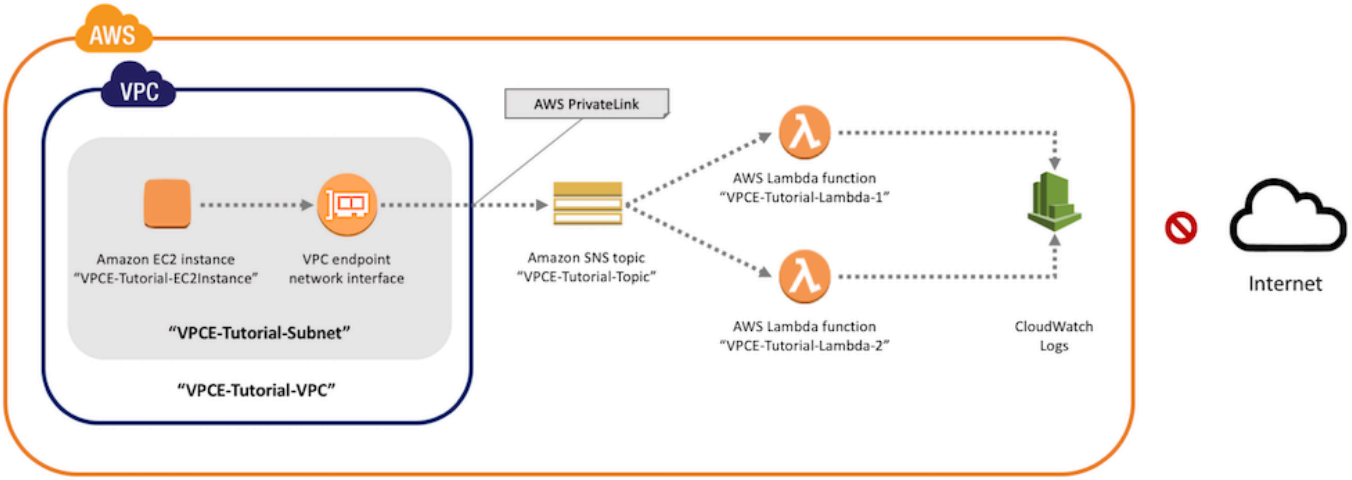
In diesem Abschnitt erfahren Sie, wie Sie ein Amazon SNS Amazon SNS-Thema veröffentlichen und gleichzeitig die Nachrichten in einem privaten Netzwerk geschützt halten. Sie veröffentlichen eine Nachricht von einer Amazon EC2 Instance, die in Amazon Virtual Private Cloud (Amazon VPC) gehostet wird. Die Nachricht bleibt innerhalb des AWS-Netzwerks, ohne dass sie ins öffentliche Internet gelangt. Durch die private Veröffentlichung von Nachrichten von einer VPC aus können Sie die Sicherheit des Datenverkehrs zwischen Ihren Anwendungen und Amazon SNS verbessern. Diese Sicherheit ist wichtig, wenn Sie personenbezogene Informationen (PII) über Ihre Kunden veröffentlichen oder wenn Ihre Anwendung Marktregulierungen unterliegt. Beispiel: Die private Veröffentlichung ist hilfreich bei einem Gesundheitssystem, das den Health Insurance Portability and Accountability Act (HIPAA) erfüllen muss, oder bei einem Finanzsystem, das den Payment Card Industry Data Security Standard (PCI DSS) erfüllen muss.

Gehen Sie wie folgt vor:

- Verwenden Sie eine AWS CloudFormation-Vorlage, um automatisch ein temporäres privates Netzwerk in Ihrem AWS-Konto-Konto zu erstellen.
- Erstellen Sie einen VPC-Endpunkt, der die VPC mit Amazon SNS verbindet.

- Melden Sie sich bei einer Amazon EC2-Instance an und veröffentlichen Sie eine Nachricht privat in einem Amazon SNS-Thema.
- Vergewissern Sie sich, dass die Nachricht erfolgreich zugestellt wurde.
- Löschen Sie die Ressourcen, die Sie für dieses Tutorial erstellt haben, damit sie nicht in Ihrem -Konto bleiben AWS-Konto.

Das folgende Diagramm zeigt das private Netzwerk, das Sie in Ihrem AWS-Konto erstellt haben, während Sie dieses Tutorial abschließen:



Dieses Netzwerk besteht aus einer VPC, die eine Amazon-EC2-Instance enthält. Die Instance stellt über einen Schnittstellen-VPC-Endpunkt eine Verbindung mit Amazon SNS her. Dieser Endpunkttyp stellt eine Verbindung zu Services von AWS PrivateLink her. Nach dem Herstellen dieser Verbindung können Sie sich bei der Amazon-EC2-Instance anmelden und Nachrichten im Amazon SNS-Thema veröffentlichen, auch wenn das Netzwerk nicht mit dem öffentlichen Internet verbunden ist. Das Thema verbreitet die empfangenen Nachrichten an zwei AWS Lambda-Funktionen mit Abonnement. Diese Funktionen protokollieren die empfangenen Nachrichten in Amazon CloudWatch Logs.

Es dauert ungefähr 20 Minuten, bis diese Schritte fertig ausgeführt sind.

Themen

Bevor Sie beginnen

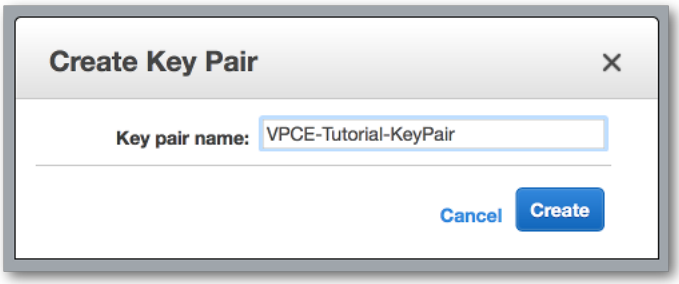
Bevor Sie beginnen, benötigen Sie einen Amazon Web Services (AWS)-Konto. Beim Anmelden wird Ihr Konto automatisch für alle Dienste in AWS, einschließlich Amazon SNS und Amazon VPC, angemeldet. Wenn Sie nicht bereits ein Konto erstellt haben, wechseln Sie zu https://aws.amazon.com/ und wählen Sie Create a Free Account (Kostenloses Konto erstellen) aus.

Schritt 1: Erstellen eines Amazon EC2-Schlüsselpaares

Ein Schlüsselpaar wird verwendet, um sich an einer Amazon EC2-Instance anzumelden. Es besteht aus einem öffentlichen Schlüssel zum Verschlüsseln Ihrer Anmeldeinformationen und einem privaten Schlüssel zum Entschlüsseln. Wenn Sie ein Schlüsselpaar erstellen, laden eine Kopie des privaten Schlüssels herunter. Später verwenden Sie das Schlüsselpaar zum Anmelden an einer Amazon EC2 Instance. Zum Anmelden geben Sie den Namen des Schlüsselpaares und den privaten Schlüssel an.

So erstellen Sie ein Schlüsselpaar

1. Melden Sie sich bei der AWS Management Console an und öffnen Sie die Amazon-EC2-Konsole unter https://console.aws.amazon.com/EC2/.
2. Suchen Sie im Navigationsmenü auf der linken Seite den Abschnitt Network & Security (Netzwerk und Sicherheit). Wählen Sie dann Key Pairs (Schlüsselpaare).
3. Wählen Sie Create Key Pair aus.
4. Geben Sie im Fenster Create Key Pair (Schlüsselpaar erstellen) für Key pair name (Schlüsselpaar-Name) den Wert **`VPCE-Tutorial-KeyPair`** ein. Wählen Sie dann die Option Create (Erstellen) aus.



5. Die private Schlüsseldatei wird von Ihrem Browser automatisch runtergeladen. Bewahren Sie es an einem sicheren Ort auf. Amazon EC2 gibt der Datei die Erweiterung `.pem`.
6. (Optional) Wenn Sie einen SSH-Client auf Ihrem Mac- oder Linux-Computer verwenden, um eine Verbindung mit der Instance herzustellen, legen Sie die Berechtigungen für die private Schlüsseldatei mit dem Befehl `chmod` fest, sodass nur Sie diese lesen können:

   a. Öffnen Sie ein Terminal und navigieren Sie zu dem Verzeichnis, das den privaten Schlüssel enthält:

   ```
   $ cd /filepath_to_private_key/
   ```

   b. Legen Sie mit folgendem Befehl die Berechtigungen fest:

   ```
   $ chmod 400 VPCE-Tutorial-KeyPair.pem
   ```

Schritt 2: Erstellen der AWS-Ressourcen

Verwenden Sie zum Einrichten der Infrastruktur, die dieses Tutorial unterstützt, eine AWS CloudFormation-Vorlage. Bei einer Vorlage handelt es sich um eine Datei, die als Vorlage für die Erstellung von AWS-Ressourcen dient, wie z. B. Amazon EC2-Instances und Amazon SNS-Themen. Die Vorlage für dieses Tutorial steht auf GitHub zum Download bereit.

Sie stellen die Vorlage für AWS CloudFormation bereit und AWS CloudFormation stellt die benötigten Ressourcen als Stack in Ihrem AWS-Konto-Konto bereit. Bei einem Stack handelt es sich um eine Sammlung von Ressourcen, die Sie als einzelne Einheit verwalten. Wenn Sie das Tutorial beendet haben, können Sie AWS CloudFormation verwenden, um alle Ressourcen im Stack auf einmal zu löschen. Diese Ressourcen bleiben nicht in Ihrem AWS-Konto-Konto, wenn Sie dies nicht möchten.

Der Stack für dieses Tutorial enthält die folgenden Ressourcen:

- Eine VPC und die zugehörigen Netzwerkressourcen, einschließlich eines Subnetzes, einer Sicherheitsgruppe, eines Internet-Gateways und einer Routing-Tabelle.
- Eine Amazon EC2-Instance, die im Subnetz in der VPC gestartet wird.
- Amazon SNS-Thema.
- Zwei AWS Lambda-Funktionen. Diese Funktionen empfangen Nachrichten, die im Amazon-SNS-Thema veröffentlicht werden, und protokollieren Ereignisse in CloudWatch Logs.
- Amazon CloudWatch-Metriken und -Protokolle
- Eine IAM-Rolle, die der Amazon EC2-Instance die Verwendung von Amazon SNS erlaubt, und eine IAM-Rolle,, die den Lambda-Funktionen das Schreiben von Daten in CloudWatch Logs erlaubt.

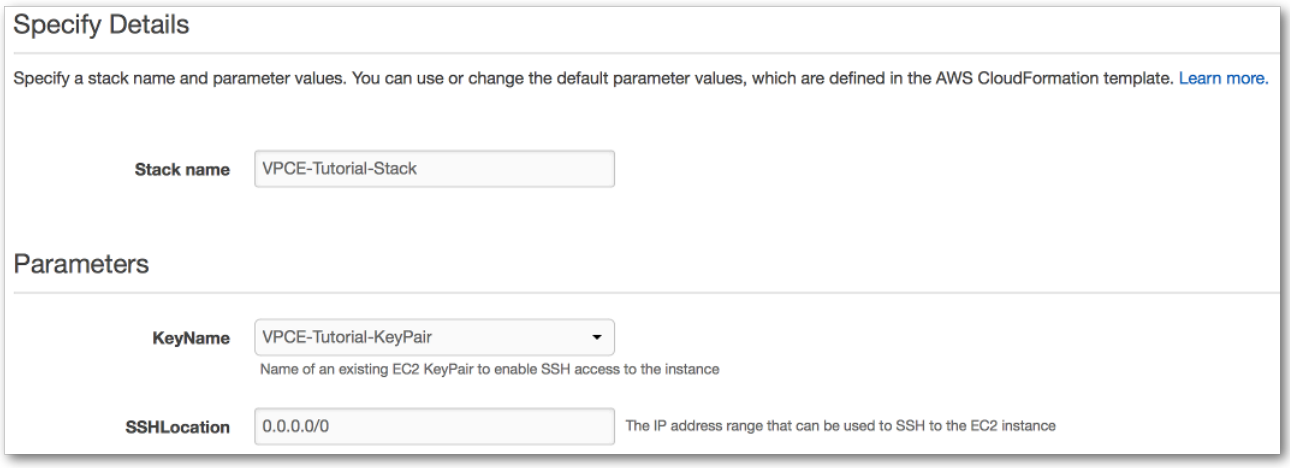
So erstellen Sie AWS-Ressourcen

1. Laden Sie die Vorlagedatei von der GitHub-Website herunter.
2. Melden Sie sich an der AWS CloudFormation-Konsole an.
3. Wählen Sie Create Stack aus.
4. Wählen Sie auf der Seite Select Template (Vorlage auswählen) die Option Upload a template to Amazon S3 (Eine Vorlage zu Amazon S3 hochladen) aus. Wählen Sie dann Ihre Datei und Next (Weiter) aus.
5. Geben Sie auf der Seite Specify Details (Details angeben) Stack- und Schlüsselnamen an:

   a. Geben Sie für Stack name **`VPCE-Tutorial-Stack`** ein.

   b. Wählen Sie für KeyName VPCE-Tutorial-KeyPair.

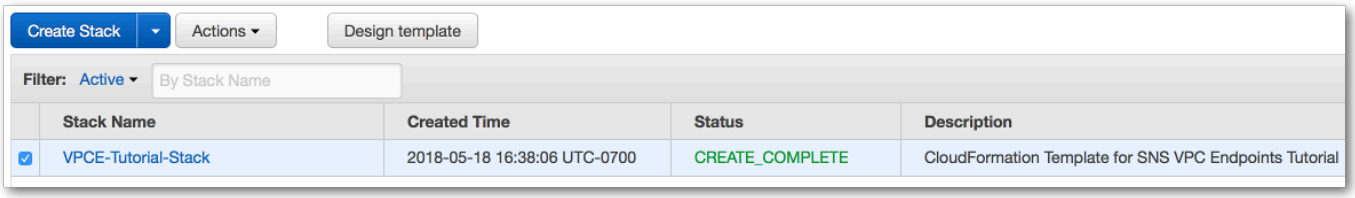
   c. Behalten Sie für SSHLocation den Standardwert **`0.0.0.0/0`** bei.



   d. Wählen Sie Next (Weiter).

6. Behalten Sie auf der Seite Options (Optionen) alle Standardwerte bei, und wählen Sie dann Next (Weiter).
7. Überprüfen Sie die Einstellungen auf der Seite Review (Überprüfen) die Stack-Details.
8. Bestätigen Sie unter Capabilities (Funktionen), dass AWS CloudFormation IAM-Ressourcen mit benutzerdefinierten Namen erstellen kann.
9. Wählen Sie Create (Erstellen) aus.

   Die AWS CloudFormation-Konsole öffnet die Seite Stacks. Der VPCE-Tutorial-Stack weist den Status CREATE_IN_PROGRESS auf. Nach Abschluss des Erstellungsprozesses ändert sich der Status in wenigen Minuten in CREATE_COMPLETE.



> **Tip**
>
> Wählen Sie die Schaltfläche Refresh (Aktualisieren), um den neuesten Stack-Status zu sehen.

Schritt 3: Bestätigen Sie, dass Ihre Amazon EC2-Instance über keinen Internetzugang verfügt

Die im vorherigen Schritt in Ihrer VPC gestartete Amazon EC2-Instance hat keinen Internetzugang. Sie verbietet ausgehenden Datenverkehr und kann keine Nachrichten in Amazon SNS veröffentlichen. Überprüfen Sie dies, indem Sie sich bei der Instance anmelden. Anschließend versuchen Sie, eine Verbindung zu einem öffentlichen Endpunkt herzustellen, und versuchen, eine Nachricht an Amazon SNS zu senden.

An diesem Punkt schlägt der Veröffentlichungsversuch fehl. In einem späteren Schritt nach dem Erstellen eines VPC-Endpunkts für Amazon SNS ist der Veröffentlichungsversuch erfolgreich.

Stellen Sie eine Verbindung zu Ihrer Amazon EC2-Instance her

1. Öffnen Sie die Amazon EC2-Konsole unter https://console.aws.amazon.com/ec2/.
2. Suchen Sie im Navigationsmenü auf der linken Seite den Abschnitt Instances. Wählen Sie anschließend Instances.

3. Wählen Sie in der Liste der Instances VPCE-Tutorial-EC2Instance aus.
4. Kopieren Sie den Hostnamen, der in der Spalte Public DNS (IPv4) (Öffentlicher DNS (IPv4)) angegeben ist.

5. Öffnen Sie ein Terminalfenster. Stellen Sie im Verzeichnis, das das Schlüsselpaar enthält, eine Verbindung mit der Instance her, indem Sie den folgenden Befehl verwenden, wobei *instance-hostname* der Hostname ist, den Sie von der Amazon EC2-Konsole kopiert haben:

```
$ ssh -i VPCE-Tutorial-KeyPair.pem ec2-user@instance-hostname
```

So überprüfen Sie, ob die Instance eine Verbindung zum Internet hat

- Versuchen Sie in Ihrem Terminal, eine Verbindung mit einem öffentlichen Endpunkt herzustellen, wie z. B. amazon.com:

```
$ ping amazon.com
```

Da der Verbindungsversuch fehlschlägt, können Sie ihn jederzeit stornieren (Strg+C unter Windows oder Befehl+C unter macOS).

Überprüfen, dass die Instance keine Verbindung zu Amazon SNS hat

1. Melden Sie sich bei der Amazon SNS-Konsole an.
2. Wählen Sie im Navigationsmenü auf der linken Seite Topics (Themen) aus.
3. Kopieren Sie auf der Seite Topics (Themen) den Amazon-Ressourcennamen (ARN) für das Thema VPCE-Tutorial-Topic.
4. Versuchen Sie in Ihrem Terminal eine Nachricht an das Thema zu veröffentlichen:

```
$ aws sns publish --region aws-region --topic-arn sns-topic-arn --message "Hello"
```

Da der Veröffentlichungsversuch fehlschlägt, können Sie ihn jederzeit abbrechen.

Schritt 4: Erstellen eines Amazon VPC-Endpunkts für Amazon SNS

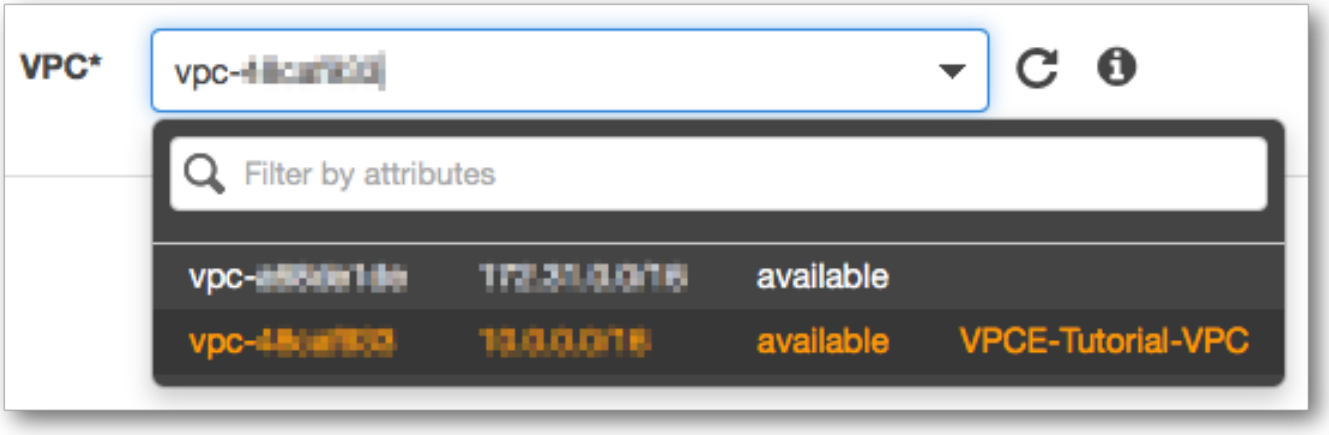
Zum Herstellen einer Verbindung der VPC mit Amazon SNS definieren Sie einen Schnittstellen-VPC-Endpunkt. Nach dem Hinzufügen des Endpunkts können Sie sich bei der Amazon EC2-Instance in Ihrer VPC anmelden und von dort aus die Amazon SNS-API verwenden. Sie können Nachrichten in dem Thema veröffentlichen. Die Nachrichten werden privat veröffentlicht. Sie bleiben innerhalb des AWS-Netzwerks und bewegen sich nicht im öffentlichen Internet.

> **Note**
>
> Die Instance hat immer noch keinen Zugriff auf andere AWS-Services und -Endpunkte im Internet.

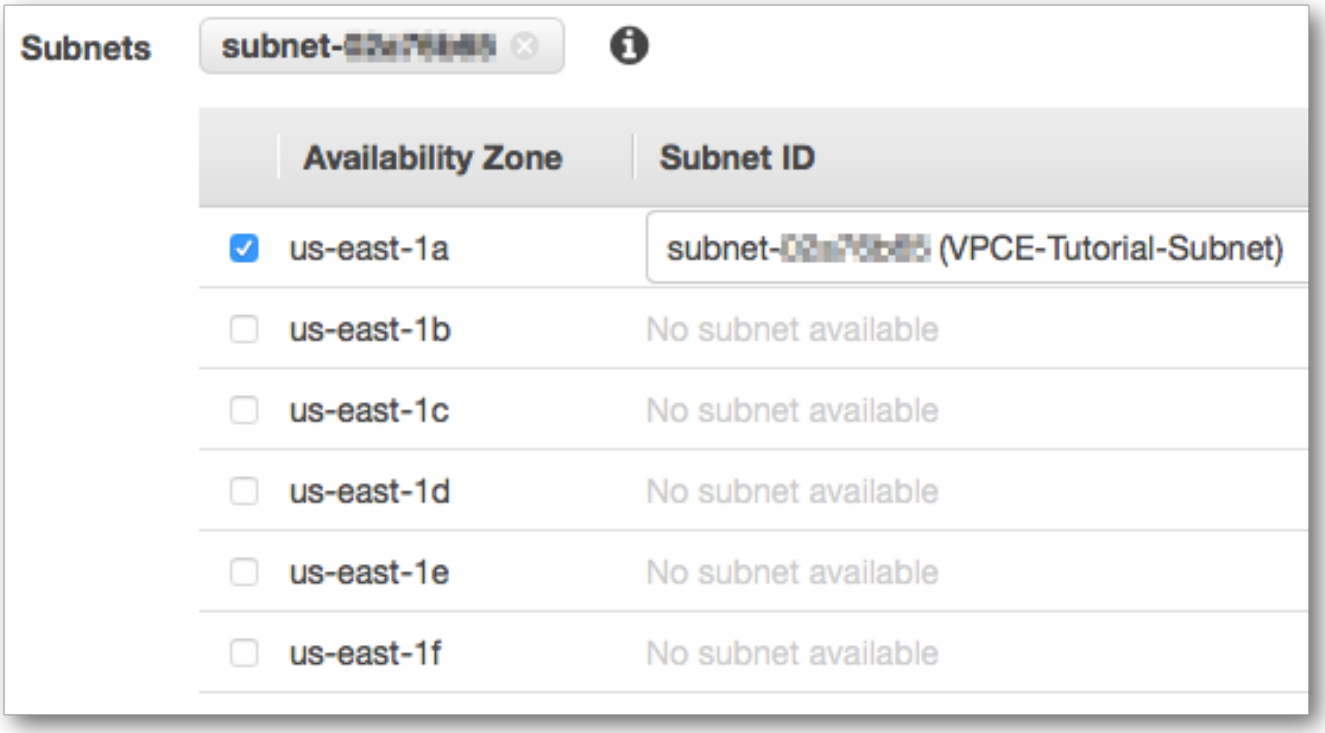
So erstellen Sie den Endpunkt

1. Öffnen Sie die Amazon VPC-Konsole unter https://console.aws.amazon.com/vpc/.
2. Wählen Sie im Navigationsmenü auf der linken Seite Endpoints (Endpunkte) aus.
3. Klicken Sie auf Create Endpoint.
4. Behalten Sie auf der Seite Create Endpoint (Endpunkt erstellen) für Service category (Servicekategorie) die Standardauswahl von AWS Services bei.
5. Wählen Sie für Service Name (Service-Name) den Service-Namen für Amazon SNS.

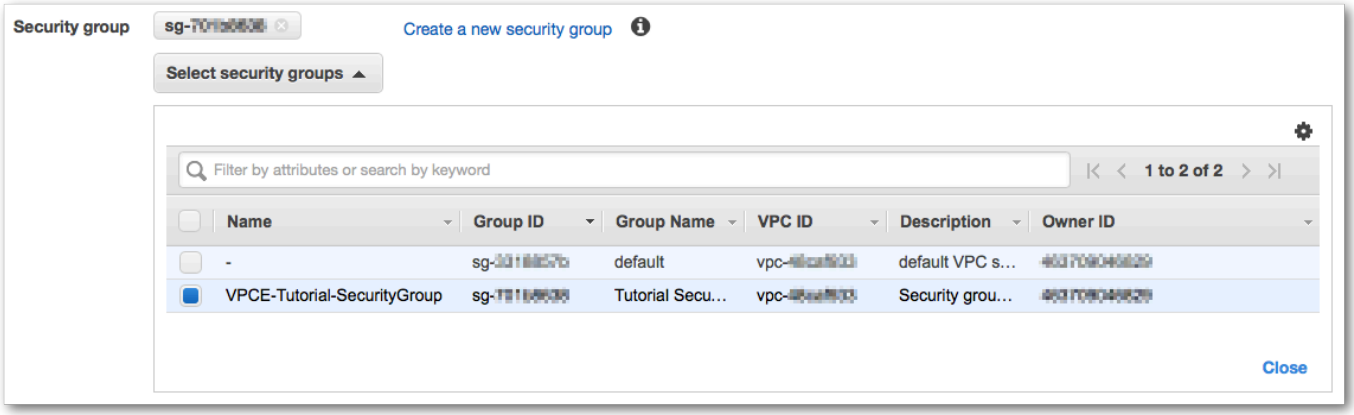
   Die Servicenamen variieren abhängig von der ausgewählten Region. Wenn Sie beispielsweise USA Ost (Nord-Virginia) wählen, lautet der Servicename com.amazonaws.***us-east-1***.sns.
6. Wählen Sie für VPC die VPC mit dem Namen VPCE-Tutorial-VPC.



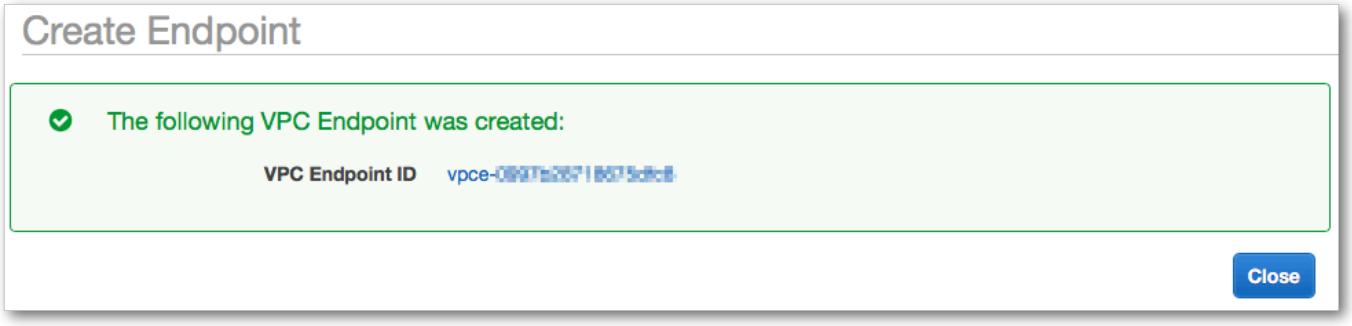
7. Wählen Sie unter Subnets (Subnetze) die Subnetze aus, deren Subnetz-ID VPCE-Tutorial-Subnet enthält.



8. Wählen Sie für Enable Private DNS Name (Privaten DNS-Namen aktivieren) die Option Enable for this endpoint (Für diesen Endpunkt aktivieren) aus.
9. Wählen Sie für Security group (Sicherheitsgruppe) die Option Select security group (Sicherheitsgruppe auswählen) und danach VPCE-Tutorial-SecurityGroup aus.

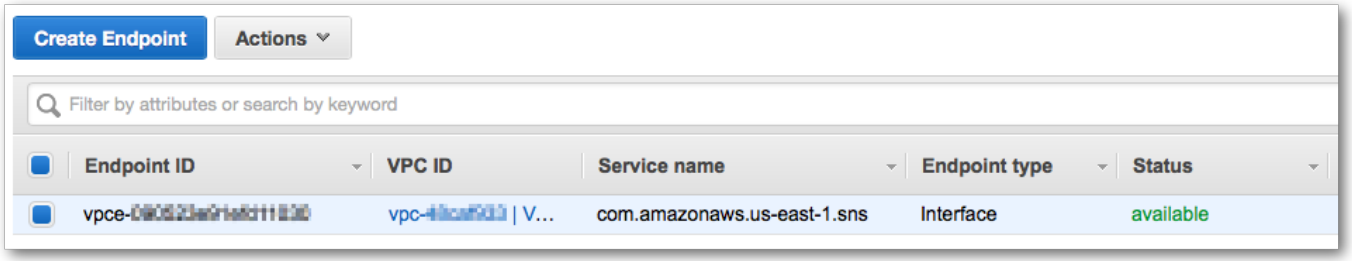


10. Wählen Sie Create endpoint. Die Amazon VPC-Konsole bestätigt, dass ein VPC-Endpunkt erstellt wurde.



11. Klicken Sie auf Close.

    Die Amazon-VPC-Konsole öffnet die Seite Endpunkte. Der neue Endpunkt weist den Status pending (ausstehend) auf. Nach Abschluss des Erstellungsprozesses ändert sich der Status in wenigen Minuten in available (verfügbar).



Schritt 5: Veröffentlichen einer Nachricht in Ihrem Amazon SNS-Thema

Jetzt, da Ihre VPC einen Endpunkt für Amazon SNS umfasst, können Sie sich an der Amazon EC2-Instance anmelden und Nachrichten im Thema veröffentlichen.

So veröffentlichen Sie eine Nachricht

1. Falls Ihr Terminal nicht mehr mit Ihrer Amazon EC2-Instance verbunden ist, stellen Sie die Verbindung wieder her:

```
$ ssh -i VPCE-Tutorial-KeyPair.pem ec2-user@instance-hostname
```

2. Führen Sie den gleichen Befehl aus wie zuvor, um eine Nachricht in Ihrem Amazon SNS-Thema zu veröffentlichen. Dieses Mal ist der Veröffentlichungsversuch erfolgreich und Amazon SNS gibt eine Mitteilungs-ID zurück:

```
$ aws sns publish --region aws-region --topic-arn sns-topic-arn --message "Hello"
```

```
{
    "MessageId": "5b111270-d169-5be6-9042-410dfc9e86de"
}
```

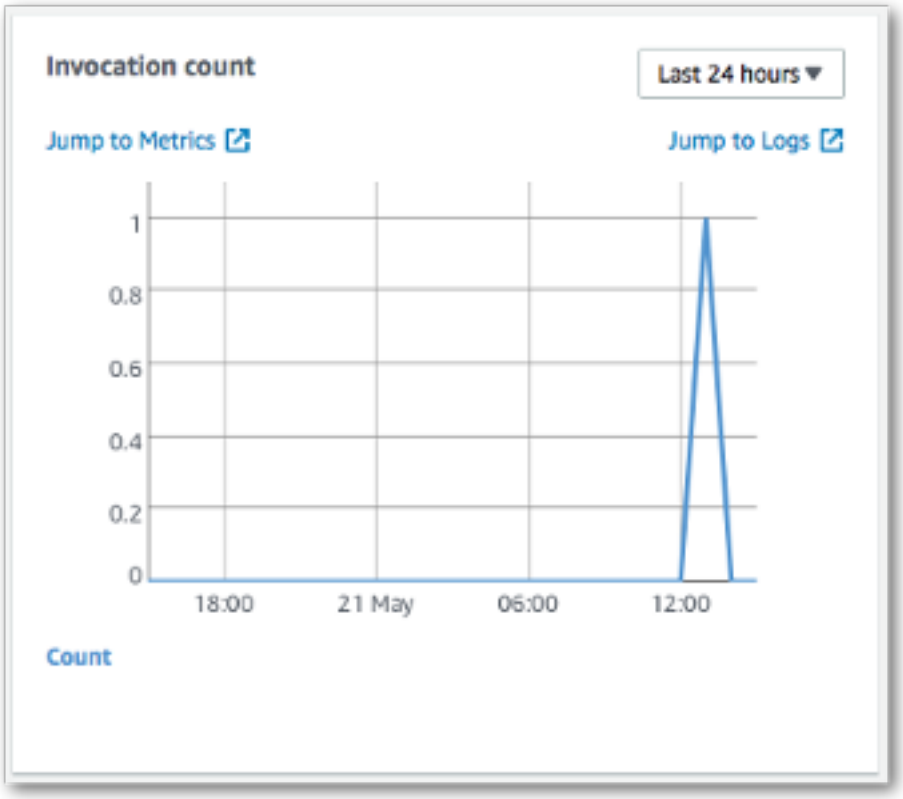
Schritt 6: Überprüfen Sie die Zustellung Ihrer Nachrichten

Wenn das Amazon SNS-Thema eine Nachricht empfängt, wird die Nachricht durch Senden an die zwei Lambda-Funktionen mit Abonnement verbreitet. Wenn diese Funktionen die Nachricht empfangen, protokollieren Sie das Ereignis in CloudWatch Logs. Um sicherzustellen, dass Ihre Nachrichtenzustellung erfolgreich war, überprüfen Sie, dass die Funktionen aufgerufen wurden und dass die CloudWatch Logs aktualisiert wurden.

So überprüfen Sie, dass die Lambda-Funktionen aufgerufen wurden

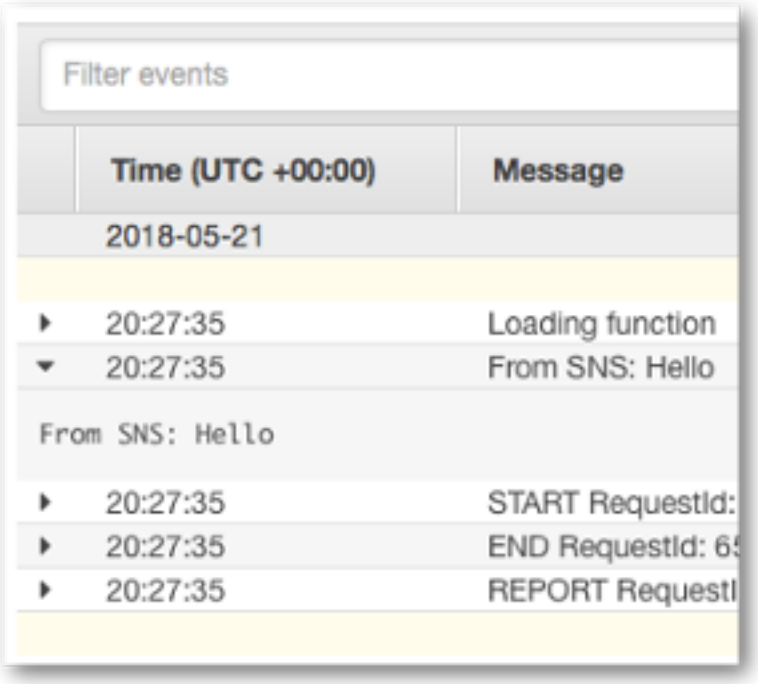
1. Öffnen Sie die AWS Lambda-Konsole unter https://console.aws.amazon.com/lambda/.
2. Wählen Sie auf der Seite Functions (Funktionen) die Option VPCE-Tutorial-Lambda-1.
3. Wählen Sie Monitoring.
4. Überprüfen Sie das Diagramm Invocation count (Anzahl der Aufrufe). Dieses Diagramm zeigt die Anzahl der Ausführungen der Lambda-Funktion.

   Die Anzahl der Aufrufe entspricht der Anzahl der Veröffentlichungen einer Nachricht im Thema.



So überprüfen Sie, dass die CloudWatch Logs aktualisiert wurden

1. Öffnen Sie die CloudWatch-Konsole unter https://console.aws.amazon.com/cloudwatch/.
2. Wählen Sie im Navigationsmenü links Logs (Protokolle).
3. Überprüfen Sie die Protokolle, die von den Lambda-Funktionen geschrieben wurden:
   a. Wählen Sie die Protokollgruppe /aws/lambda/VPCE-Tutorial-Lambda-1/.
   b. Wählen Sie den Protokollstream.
   c. Prüfen Sie, dass das Protokoll den Eintrag enthält `From SNS: Hello`.



d. Wählen Sie Log Groups (Protokollgruppen) oben auf der Konsole, um zur Seite Log Groups (Protokollgruppen) zurückzukehren. Wiederholen Sie dann die vorangegangenen Schritte für die Protokollgruppe /aws/lambda/VPCE-Tutorial-Lambda-2/.

Herzlichen Glückwunsch! Durch das Hinzufügen eines Endpunkts für Amazon SNS zu einer VPC konnten Sie eine Nachricht in einem Thema innerhalb des Netzwerks veröffentlichen, das von der VPC verwaltet wird. Die Nachricht wurde privat veröffentlicht, ohne im Internet veröffentlicht zu werden.

Schritt 7: Bereinigen

Sofern Sie die Ressourcen, die Sie für dieses Tutorial erstellt haben, nicht behalten möchten, können Sie sie nun löschen. Durch das Löschen von AWS-Ressourcen, die Sie nicht mehr verwenden, können Sie verhindern, dass unnötige Gebühren in Ihrem AWS-Konto-Konto anfallen.

Löschen Sie zuerst Ihren VPC-Endpunkt mithilfe der Amazon VPC-Konsole. Löschen Sie anschließend die anderen Ressourcen, die Sie erstellt haben, indem Sie in der AWS CloudFormation-Konsole den Stack löschen. Wenn Sie einen Stack löschen, entfernt AWS CloudFormation die Stack-Ressourcen aus Ihrem AWS-Konto-Konto.

So löschen Sie Ihren VPC-Endpunkt

1. Öffnen Sie die Amazon VPC-Konsole unter https://console.aws.amazon.com/vpc/.
2. Wählen Sie im Navigationsmenü auf der linken Seite Endpoints (Endpunkte) aus.
3. Wählen Sie den Endpunkt aus, den Sie erstellt haben.
4. Wählen Sie Actions (Aktionen) und anschließend Delete Endpoint (Endpunkt löschen).
5. Wählen Sie im Fenster Delete Endpoint (Endpunkt löschen) Yes, Delete (Ja, löschen).

   Der Status des Endpunkts wird in deleting (Wird gelöscht...) geändert. Wenn der Löschvorgang abgeschlossen ist, wird der Endpunkt von der Seite entfernt.

So löschen Sie Ihren AWS CloudFormation-Stack

1. Öffnen Sie die AWS CloudFormation-Konsole unter https://console.aws.amazon.com/cloudformation.
2. Wählen Sie den Stack VPCE-Tutorial-Stack aus.
3. Wählen Sie die Option Actions (Aktionen) und anschließend Delete Stack (Stack löschen) aus.
4. Wählen Sie im Fenster Delete Stack (Stack löschen) Yes, Delete (Ja, löschen).

   Der Stack-Status ändert sich in DELETE_IN_PROGRESS. Wenn der Löschvorgang abgeschlossen ist, wird der Stack von der Seite entfernt.

Zugehörige Ressourcen

Weitere Informationen finden Sie in den folgenden Ressourcen.

- AWS-Sicherheitsblog: Schutz von Nachrichten, die in Amazon SNS veröffentlicht werden, mit AWS PrivateLink
- Was ist Amazon VPC?
- VPC-Endpunkte
- Was ist Amazon EC2?
- AWS CloudFormation Konzepte

Sicherheit des Nachrichtendatenschutzes

- Der Nachrichtendatenschutz ist eine Funktion in Amazon SNS, mit der Sie Ihre eigenen Regeln und Richtlinien definieren können, um den Inhalt für Daten in Bewegung im Gegensatz zu Daten im Ruhezustand zu überprüfen und zu kontrollieren.
- Der Nachrichtendatenschutz bietet Governance-, Compliance- und Auditing-Services für Unternehmensanwendungen, die auf Nachrichten ausgerichtet sind, sodass der Dateneingang und -ausgang vom Amazon-SNS-Themenbesitzer kontrolliert und Inhaltsabläufe verfolgt und protokolliert werden können.
- Sie können nutzlastbasierte Governance-Regeln erstellen, um zu verhindern, dass nicht autorisierte Nutzlastinhalte in Ihre Nachrichten-Streams gelangen.
- Sie können einzelnen Abonnenten unterschiedliche Zugriffsberechtigungen für Inhalte erteilen und den gesamten Inhaltsablaufprozess überwachen.

Identity and Access Management in Amazon SNS

Für den Zugriff auf Amazon SNS werden Anmeldeinformationen benötigt, die AWS zur Authentifizierung Ihrer Anforderungen verwenden kann. Diese Anmeldeinformationen müssen über Berechtigungen für den Zugriff auf AWS-Ressourcen, wie beispielsweise Amazon SNS-Themen und -Nachrichten, verfügen. In den folgenden Abschnitten erfahren Sie, wie Sie Ihre Ressourcen mithilfe von AWS Identity and Access Management (IAM) und Amazon SNS schützen können, indem Sie den Zugriff auf sie kontrollieren.

AWS Identity and Access Management (IAM) ist ein AWS-Service, mit dem Administratoren den Zugriff auf AWS-Ressourcen sicher steuern können. IAM-Administratoren steuern, wer authentifiziert (angemeldet) und autorisiert (im Besitz von Berechtigungen) ist, Amazon-SNS-Ressourcen zu nutzen. IAM ist ein AWS-Service, den Sie ohne zusätzliche Kosten verwenden können.

Zielgruppe

Wie Sie AWS Identity and Access Management (IAM) verwenden, unterscheidet sich je nach Ihrer Arbeit in Amazon SNS.

Service-Benutzer – wenn Sie den Amazon–SNS-Service zur Ausführung von Aufgaben verwenden, stellt Ihnen Ihr Administrator die Anmeldeinformationen und Berechtigungen bereit, die Sie benötigen. Wenn Sie zur Ausführung von Aufgaben weitere Amazon-SNS-Funktionen verwenden,

benötigen Sie möglicherweise zusätzliche Berechtigungen. Wenn Sie die Featuresweise der Zugriffskontrolle nachvollziehen, wissen Sie bereits, welche Berechtigungen Sie von Ihrem Administrator anzufordern müssen. Wenn Sie auf eine Funktion in Amazon SNS nicht zugreifen können, finden Sie Informationen dazu unter Beheben von Identitäts- und Zugriffsfehlern bei Amazon Simple Notification Service.

Service-Administrator – wenn Sie in Ihrem Unternehmen für die Amazon-SNS-Ressourcen zuständig sind, haben Sie wahrscheinlich vollen Zugriff auf Amazon SNS. Ihre Aufgabe besteht darin, zu bestimmen, auf welche Amazon-SNS-Funktionen und -Ressourcen Ihre Servicebenutzer zugreifen sollen. Sie müssen dann Anträge an Ihren IAM-Administrator stellen, um die Berechtigungen Ihrer Servicenutzer zu ändern. Lesen Sie die Informationen auf dieser Seite, um die Grundkonzepte von IAM nachzuvollziehen. Weitere Informationen dazu, wie Ihr Unternehmen IAM mit Amazon SNS verwenden kann, finden Sie unter So funktioniert Amazon Simple Notification Service mit IAM.

IAM-Administrator – wenn Sie ein IAM-Administrator sind, möchten Sie vielleicht Details darüber erfahren, wie Sie Richtlinien zur Verwaltung des Zugriffs auf Amazon SNS erstellen können. Beispiele für identitätsbasierte Amazon-SNS-Richtlinien, die Sie in IAM verwenden können, finden Sie unter Beispiele für identitätsbasierte Richtlinien für Amazon Simple Notification Service.

Authentifizierung mit Identitäten

Die Authentifizierung ist die Art und Weise, wie Sie sich mit Ihren Anmeldeinformationen bei AWS anmelden. Die Authentifizierung (Anmeldung bei AWS) muss als AWS-Konto-Root-Benutzer, als IAM-Benutzer oder durch Übernahme einer IAM-Rolle erfolgen.

Sie können sich bei AWS als Verbundidentität mit Anmeldeinformationen anmelden, die über eine Identitätsquelle bereitgestellt werden. Benutzer von AWS IAM Identity Center. (IAM Identity Center), die Single-Sign-on-Authentifizierung Ihres Unternehmens und Anmeldeinformationen für Google oder Facebook sind Beispiele für Verbundidentitäten. Wenn Sie sich als Verbundidentität anmelden, hat der Administrator vorher mithilfe von IAM-Rollen einen Identitätsverbund eingerichtet. Wenn Sie auf AWS mithilfe des Verbunds zugreifen, übernehmen Sie indirekt eine Rolle.

Je nachdem, welcher Benutzertyp Sie sind, können Sie sich bei der AWS Management Console oder beim AWS-Zugriffsportal anmelden. Weitere Informationen zum Anmelden bei AWS finden Sie unter So melden Sie sich bei Ihrem AWS-Konto an im Benutzerhandbuch von AWS-Anmeldung.

Bei programmgesteuertem Zugriff auf AWS bietet AWS ein Software Development Kit (SDK) und eine Command Line Interface (CLI, Befehlszeilenschnittstelle) zum kryptographischen Signieren Ihrer Anfragen mit Ihren Anmeldeinformationen. Wenn Sie keine AWS-Tools verwenden, müssen

Sie Anforderungen selbst signieren. Weitere Informationen zur Verwendung der empfohlenen Methode zum eigenen Signieren von Anforderungen finden Sie unter Signieren von AWS-API-Anforderungen im IAM-Benutzerhandbuch.

Unabhängig von der verwendeten Authentifizierungsmethode müssen Sie möglicherweise zusätzliche Sicherheitsinformationen angeben. AWS empfiehlt beispielsweise die Verwendung von Multi-Faktor Authentifizierung (MFA), um die Sicherheit Ihres Kontos zu verbessern. Weitere Informationen finden Sie unter Multi-Faktor-Authentifizierung im AWS IAM Identity Center-Benutzerhandbuch und Verwenden der Multi-Faktor-Authentifizierung (MFA) in AWS im IAM-Benutzerhandbuch.

AWS-Konto-Root-Benutzer

Wenn Sie ein AWS-Konto neu erstellen, beginnen Sie mit einer Anmeldeidentität, die vollständigen Zugriff auf alle AWS-Services und Ressourcen des Kontos hat. Diese Identität wird als AWS-Konto-Root-Benutzer bezeichnet. Für den Zugriff auf den Root-Benutzer müssen Sie sich mit der E-Mail-Adresse und dem Passwort anmelden, die zur Erstellung des Kontos verwendet wurden. Wir raten ausdrücklich davon ab, den Root-Benutzer für Alltagsaufgaben zu verwenden. Schützen Sie Ihre Root-Benutzer-Anmeldeinformationen und verwenden Sie diese, um die Aufgaben auszuführen, die nur der Root-Benutzer ausführen kann. Eine vollständige Liste der Aufgaben, für die Sie sich als Root-Benutzer anmelden müssen, finden Sie unter Aufgaben, die Root-Benutzer-Anmeldeinformationen erfordern im IAM-Benutzerhandbuch.

Verbundidentität

Als bewährte Methode empfiehlt es sich, menschliche Benutzer, einschließlich Benutzer, die Administratorzugriff benötigen, aufzufordern, den Verbund mit einem Identitätsanbieter zu verwenden, um auf AWS-Services mit temporären Anmeldeinformationen zuzugreifen.

Eine Verbundidentität ist ein Benutzer aus dem Benutzerverzeichnis Ihres Unternehmens, ein Web Identity Provider, AWS Directory Service, das Identity-Center-Verzeichnis oder jeder Benutzer, der mit Anmeldeinformationen, die über eine Identitätsquelle bereitgestellt werden, auf AWS-Services zugreift. Wenn Verbundidentitäten auf AWS-Konten zugreifen, übernehmen sie Rollen und die Rollen stellen temporäre Anmeldeinformationen bereit.

Für die zentrale Zugriffsverwaltung empfehlen wir Ihnen, AWS IAM Identity Center zu verwenden. Sie können Benutzer und Gruppen im IAM Identity Center erstellen oder Sie können eine Verbindung mit einer Gruppe von Benutzern und Gruppen in Ihrer eigenen Identitätsquelle herstellen und synchronisieren, um sie in allen AWS-Konten und Anwendungen zu verwenden. Informationen zu

IAM Identity Center finden Sie unter Was ist IAM Identity Center? im AWS IAM Identity Center-Benutzerhandbuch.

IAM-Benutzer und -Gruppen

Ein IAM-Benutzer ist eine Identität in Ihrem AWS-Konto mit bestimmten Berechtigungen für eine einzelne Person oder eine einzelne Anwendung. Wenn möglich, empfehlen wir, temporäre Anmeldeinformationen zu verwenden, anstatt IAM-Benutzer zu erstellen, die langfristige Anmeldeinformationen wie Passwörter und Zugriffsschlüssel haben. Bei speziellen Anwendungsfällen, die langfristige Anmeldeinformationen mit IAM-Benutzern erfordern, empfehlen wir jedoch, die Zugriffsschlüssel zu rotieren. Weitere Informationen finden Sie unter Regelmäßiges Rotieren von Zugriffsschlüsseln für Anwendungsfälle, die langfristige Anmeldeinformationen erfordern im IAM-Benutzerhandbuch.

Eine IAM-Gruppe ist eine Identität, die eine Sammlung von IAM-Benutzern angibt. Sie können sich nicht als Gruppe anmelden. Mithilfe von Gruppen können Sie Berechtigungen für mehrere Benutzer gleichzeitig angeben. Gruppen vereinfachen die Verwaltung von Berechtigungen, wenn es zahlreiche Benutzer gibt. Sie könnten beispielsweise einer Gruppe mit dem Namen IAMAdmins Berechtigungen zum Verwalten von IAM-Ressourcen erteilen.

Benutzer unterscheiden sich von Rollen. Ein Benutzer ist einer einzigen Person oder Anwendung eindeutig zugeordnet. Eine Rolle kann von allen Personen angenommen werden, die sie benötigen. Benutzer besitzen dauerhafte Anmeldeinformationen. Rollen stellen temporäre Anmeldeinformationen bereit. Weitere Informationen finden Sie unter Erstellen eines IAM-Benutzers (anstatt einer Rolle) im IAM-Benutzerhandbuch.

IAM-Rollen

Eine IAM-Rolle ist eine Identität in Ihrem AWS-Konto mit spezifischen Berechtigungen. Sie ist einem IAM-Benutzer vergleichbar, ist aber nicht mit einer bestimmten Person verknüpft. Sie können vorübergehend eine IAM-Rolle in der AWS Management Console übernehmen, indem Sie Rollen wechseln. Sie können eine Rolle annehmen, indem Sie eine AWS CLI oder AWS-API-Operation aufrufen oder eine benutzerdefinierte URL verwenden. Weitere Informationen zu Methoden für die Verwendung von Rollen finden Sie unter Verwenden von IAM-Rollen im IAM-Benutzerhandbuch.

IAM-Rollen mit temporären Anmeldeinformationen sind in folgenden Situationen hilfreich:

- Verbundbenutzerzugriff: Um einer Verbundidentität Berechtigungen zuzuweisen, erstellen Sie eine Rolle und definieren Berechtigungen für die Rolle. Wird eine Verbundidentität authentifiziert, so

wird die Identität der Rolle zugeordnet und erhält die von der Rolle definierten Berechtigungen. Informationen zu Rollen für den Verbund finden Sie unter Erstellen von Rollen für externe Identitätsanbieter im IAM-Benutzerhandbuch. Wenn Sie IAM Identity Center verwenden, konfigurieren Sie einen Berechtigungssatz. Wenn Sie steuern möchten, worauf Ihre Identitäten nach der Authentifizierung zugreifen können, korreliert IAM Identity Center den Berechtigungssatz mit einer Rolle in IAM. Informationen zu Berechtigungssätzen finden Sie unter Berechtigungssätze im AWS IAM Identity Center-Benutzerhandbuch.

- Temporäre IAM-Benutzerberechtigungen: Ein IAM-Benutzer oder eine -Rolle kann eine IAM-Rolle übernehmen, um vorübergehend andere Berechtigungen für eine bestimmte Aufgabe zu erhalten.
- Kontoübergreifender Zugriff – Sie können eine IAM-Rolle verwenden, um einem vertrauenswürdigen Prinzipal in einem anderen Konto den Zugriff auf Ressourcen in Ihrem Konto zu ermöglichen. Rollen stellen die primäre Möglichkeit dar, um kontoübergreifendem Zugriff zu gewähren. In einigen AWS-Services können Sie jedoch eine Richtlinie direkt an eine Ressource anfügen (anstatt eine Rolle als Proxy zu verwenden). Informationen zu den Unterschieden zwischen Rollen und ressourcenbasierten Richtlinien für den kontoübergreifenden Zugriff finden Sie unter So unterscheiden sich IAM-Rollen von ressourcenbasierten Richtlinien im IAM-Benutzerhandbuch.
- Serviceübergreifender Zugriff: Einige AWS-Services verwenden Features in anderen AWS-Services. Wenn Sie beispielsweise einen Aufruf in einem Service tätigen, führt dieser Service häufig Anwendungen in Amazon EC2 aus oder speichert Objekte in Amazon S3. Ein Dienst kann dies mit den Berechtigungen des aufrufenden Prinzipals mit einer Servicerolle oder mit einer serviceverknüpften Rolle tun.
  - Forward access sessions (FAS) – Wenn Sie einen IAM-Benutzer oder eine IAM-Rolle zum Ausführen von Aktionen in AWS verwenden, gelten Sie als Prinzipal. Bei einigen Services könnte es Aktionen geben, die dann eine andere Aktion in einem anderen Service auslösen. FAS verwendet die Berechtigungen des Prinzipals, der einen AWS-Service aufruft, in Kombination mit der Anforderung an den AWS-Service, Anforderungen an nachgelagerte Services zu stellen. FAS-Anfragen werden nur dann gestellt, wenn ein Service eine Anfrage erhält, die eine Interaktion mit anderen AWS-Services oder -Ressourcen erfordert. In diesem Fall müssen Sie über Berechtigungen zum Ausführen beider Aktionen verfügen. Einzelheiten zu den Richtlinien für FAS-Anfragen finden Sie unter Zugriffssitzungen weiterleiten.
  - Servicerolle: Eine Servicerolle ist eine IAM-Rolle, die ein Service übernimmt, um Aktionen in Ihrem Namen auszuführen. Ein IAM-Administrator kann eine Servicerolle innerhalb von IAM erstellen, ändern und löschen. Weitere Informationen finden Sie unter Erstellen einer Rolle zum Delegieren von Berechtigungen an einen AWS-Service im IAM-Benutzerhandbuch.

  - Serviceverknüpfte Rolle: Eine serviceverknüpfte Rolle ist ein Typ von Servicerolle, die mit einem AWS-Service verknüpft ist. Der Service kann die Rolle übernehmen, um eine Aktion in Ihrem Namen auszuführen. Serviceverknüpfte Rollen werden in Ihrem AWS-Konto angezeigt und gehören zum Service. Ein IAM-Administrator kann die Berechtigungen für serviceverbundene Rollen anzeigen, aber nicht bearbeiten.
- Anwendungen in Amazon EC2 – Sie können eine IAM-Rolle verwenden, um temporäre Anmeldeinformationen für Anwendungen zu verwalten, die auf einer EC2-Instance ausgeführt werden und – AWS CLIoder AWS-API-Anforderungen durchführen. Das ist eher zu empfehlen, als Zugriffsschlüssel innerhalb der EC2-Instance zu speichern. Erstellen Sie ein Instance-Profil, das an die Instance angefügt ist, um eine AWS-Rolle einer EC2-Instance zuzuweisen und die Rolle für sämtliche Anwendungen der Instance bereitzustellen. Ein Instance-Profil enthält die Rolle und ermöglicht, dass Programme, die in der EC2-Instance ausgeführt werden, temporäre Anmeldeinformationen erhalten. Weitere Informationen finden Sie unter Verwenden einer IAM-Rolle zum Erteilen von Berechtigungen für Anwendungen, die auf Amazon EC2-Instances ausgeführt werden im IAM-Benutzerhandbuch.

Informationen dazu, wann Sie IAM-Rollen oder IAM-Benutzer verwenden sollten, finden Sie unter Erstellen einer IAM-Rolle (anstatt eines Benutzers) im IAM-Benutzerhandbuch.

Verwalten des Zugriffs mit Richtlinien

Für die Zugriffssteuerung in AWS erstellen Sie Richtlinien und weisen diese den AWS-Identitäten oder -Ressourcen zu. Eine Richtlinie ist ein Objekt in AWS, das, wenn es einer Identität oder Ressource zugeordnet wird, deren Berechtigungen definiert. AWS wertet diese Richtlinien aus, wenn ein Prinzipal (Benutzer, Root-Benutzer oder Rollensitzung) eine Anforderung stellt. Berechtigungen in den Richtlinien bestimmen, ob die Anforderung zugelassen oder abgelehnt wird. Die meisten Richtlinien werden in AWS als JSON-Dokumente gespeichert. Weitere Informationen zu Struktur und Inhalten von JSON-Richtliniendokumenten finden Sie unter Übersicht über JSON-Richtlinien im IAM-Benutzerhandbuch.

Administratoren können mithilfe von AWS-JSON-Richtlinien festlegen, wer zum Zugriff auf was berechtigt ist. Das bedeutet, welcher Prinzipal kann Aktionen für welche Ressourcen und unter welchen Bedingungen ausführen.

Standardmäßig haben Benutzer, Gruppen und Rollen keine Berechtigungen. Ein IAM-Administrator muss IAM-Richtlinien erstellen, die Benutzern die Berechtigung erteilen, Aktionen für die Ressourcen

auszuführen, die sie benötigen. Der Administrator kann dann die IAM-Richtlinien zu Rollen hinzufügen, und Benutzer können die Rollen annehmen.

IAM-Richtlinien definieren Berechtigungen für eine Aktion unabhängig von der Methode, die Sie zur Ausführung der Aktion verwenden. Angenommen, es gibt eine Richtlinie, die Berechtigungen für die `iam:GetRole`-Aktion erteilt. Ein Benutzer mit dieser Richtlinie kann Benutzerinformationen über die AWS Management Console, die AWS CLI oder die AWS -API abrufen.

Identitätsbasierte Richtlinien

Identitätsbasierte Richtlinien sind JSON-Berechtigungsrichtliniendokumente, die Sie einer Identität anfügen können, wie z. B. IAM-Benutzern, -Benutzergruppen oder -Rollen. Diese Richtlinien steuern, welche Aktionen die Benutzer und Rollen für welche Ressourcen und unter welchen Bedingungen ausführen können. Informationen zum Erstellen identitätsbasierter Richtlinien finden Sie unter Erstellen von IAM-Richtlinien im IAM-Benutzerhandbuch.

Identitätsbasierte Richtlinien können weiter als Inline-Richtlinien oder verwaltete Richtlinien kategorisiert werden. Inline-Richtlinien sind direkt in einen einzelnen Benutzer, eine einzelne Gruppe oder eine einzelne Rolle eingebettet. Verwaltete Richtlinien sind eigenständige Richtlinien, die Sie mehreren Benutzern, Gruppen und Rollen in Ihrem AWS-Konto anfügen können. Verwaltete Richtlinien umfassen von AWS verwaltete und von Kunden verwaltete Richtlinien. Informationen dazu, wie Sie zwischen einer verwalteten Richtlinie und einer eingebundenen Richtlinie wählen, finden Sie unter Auswahl zwischen verwalteten und eingebundenen Richtlinien im IAM-Benutzerhandbuch.

Ressourcenbasierte Richtlinien

Ressourcenbasierte Richtlinien sind JSON-Richtliniendokumente, die Sie an eine Ressource anfügen. Beispiele für ressourcenbasierte Richtlinien sind IAM-Rollen-Vertrauensrichtlinien und Amazon-S3-Bucket-Richtlinien. In Services, die ressourcenbasierte Richtlinien unterstützen, können Service-Administratoren sie verwenden, um den Zugriff auf eine bestimmte Ressource zu steuern. Für die Ressource, an welche die Richtlinie angehängt ist, legt die Richtlinie fest, welche Aktionen ein bestimmter Prinzipal unter welchen Bedingungen für diese Ressource ausführen kann. Sie müssen in einer ressourcenbasierten Richtlinie einen Prinzipal angeben. Prinzipale können Konten, Benutzer, Rollen, Verbundbenutzer oder AWS-Services umfassen.

Ressourcenbasierte Richtlinien sind Richtlinien innerhalb dieses Diensts. Sie können verwaltete AWS-Richtlinien von IAM nicht in einer ressourcenbasierten Richtlinie verwenden.

Zugriffssteuerungslisten (ACLs)

Zugriffssteuerungslisten (ACLs) steuern, welche Prinzipale (Kontomitglieder, Benutzer oder Rollen) auf eine Ressource zugreifen können. ACLs sind ähnlich wie ressourcenbasierte Richtlinien, verwenden jedoch nicht das JSON-Richtliniendokumentformat.

Amazon S3, AWS WAF und Amazon VPC sind Beispiele für Dienste, die ACLs unterstützen. Weitere Informationen“ zu ACLs finden Sie unter Zugriffskontrollliste (ACL) – Übersicht (Access Control List) im Amazon-Simple-Storage-Service-Entwicklerhandbuch.

Weitere Richtlinientypen

AWS unterstützt zusätzliche, weniger häufig verwendete Richtlinientypen. Diese Richtlinientypen können die maximalen Berechtigungen festlegen, die Ihnen von den häufiger verwendeten Richtlinientypen erteilt werden können.

- Berechtigungsgrenzen:Eine Berechtigungsgrenze ist ein erweitertes Feature, mit der Sie die maximalen Berechtigungen festlegen können, die eine identitätsbasierte Richtlinie einer IAM-Entität (IAM-Benutzer oder -Rolle) erteilen kann. Sie können eine Berechtigungsgrenze für eine Entität festlegen. Die daraus resultierenden Berechtigungen sind der Schnittpunkt der identitätsbasierten Richtlinien einer Entität und ihrer Berechtigungsgrenzen. Ressourcenbasierte Richtlinien, die den Benutzer oder die Rolle im Feld `Principal` angeben, werden nicht durch Berechtigungsgrenzen eingeschränkt. Eine explizite Zugriffsverweigerung in einer dieser Richtlinien setzt eine Zugriffserlaubnis außer Kraft. Weitere Informationen über Berechtigungsgrenzen finden Sie unter Berechtigungsgrenzen für IAM-Entitäten im IAM-Benutzerhandbuch.
- Service-Kontrollrichtlinien (SCPs) – SCPs sind JSON-Richtlinien, die die maximalen Berechtigungen für eine Organisation oder Organisationseinheit (OE) in AWS Organizations angeben. AWS Organizations ist ein Dienst für die Gruppierung und zentrale Verwaltung mehrerer AWS-Konten Ihres Unternehmens. Wenn Sie innerhalb einer Organisation alle Funktionen aktivieren, können Sie Service-Kontrollrichtlinien (SCPs) auf alle oder einzelne Ihrer Konten anwenden. Eine SCP beschränkt die Berechtigungen für Entitäten in Mitgliedskonten, einschließlich aller AWS-Konto-Stammbenutzer. Weitere Informationen zu Organizations und SCPs finden Sie unter Funktionsweise von SCPs im AWS Organizations-Benutzerhandbuch.
- Sitzungsrichtlinien:Sitzungsrichtlinien sind erweiterte Richtlinien, die Sie als Parameter übergeben, wenn Sie eine temporäre Sitzung für eine Rolle oder einen verbundenen Benutzer programmgesteuert erstellen. Die resultierenden Sitzungsberechtigungen sind eine Schnittmenge der auf der Identität des Benutzers oder der Rolle basierenden Richtlinien und der Sitzungsrichtlinien. Berechtigungen können auch aus einer ressourcenbasierten Richtlinie

stammen. Eine explizite Zugriffsverweigerung in einer dieser Richtlinien setzt eine Zugriffserlaubnis außer Kraft. Weitere Informationen finden Sie unter Sitzungsrichtlinien im IAM-Benutzerhandbuch.

Mehrere Richtlinientypen

Wenn mehrere auf eine Anforderung mehrere Richtlinientypen angewendet werden können, sind die entsprechenden Berechtigungen komplizierter. Informationen dazu, wie AWS die Zulässigkeit einer Anforderung ermittelt, wenn mehrere Richtlinientypen beteiligt sind, finden Sie unter Logik für die Richtlinienauswertung im IAM-Benutzerhandbuch.

Zugriffskontrolle

Amazon SNS verfügt über ein eigenes ressourcenbasiertes Berechtigungssystem. Dieses verwendet Richtlinien, die in derselben Sprache abgefasst sind, wie die AWS Identity and Access Management (IAM)-Richtlinien. Das bedeutet, dass Sie mit Amazon SNS- und IAM-Richtlinien ähnliche Ziele erreichen können.

> **Note**
>
> Es ist wichtig zu verstehen, dass alle AWS-Konten ihre Berechtigungen an Benutzer, die ihren Konten angehören, delegieren können. Der kontenübergreifende Zugriff ermöglicht Ihnen den gemeinsamen Zugriff auf AWS-Ressourcen, ohne zusätzliche Benutzer verwalten zu müssen. Weitere Informationen über die Verwendung des kontenübergreifenden Zugriffs finden Sie unter Enabling Cross-Account Access im IAM-Benutzerhandbuch.

Übersicht über die Zugriffsverwaltung in Amazon SNS

In diesem Abschnitt werden die grundlegenden Konzepte in Bezug auf die Verwendung von Zugriffskontrollsprachen zum Schreiben von Richtlinien beschrieben. Außerdem wird der allgemeine Prozess erörtert, wie die Zugriffskontrolle mit Zugriffskontrollsprachen funktioniert und wie Richtlinien bewertet werden.

Themen

- Wann die Zugriffskontrolle verwendet werden sollte
- Die wichtigsten Konzepte
- Übersicht über die Architektur

- Verwenden der Zugriffskontrollsprache
- Auswertungslogik
- Beispiele für die Zugriffskontrolle in Amazon SNS

Wann die Zugriffskontrolle verwendet werden sollte

Sie sind sehr flexibel, was die Erteilung bzw. Verweigerung des Zugriffs auf eine Ressource anbelangt. Die typischen Anwendungsfälle sind jedoch recht einfach:

- Sie möchten einem anderen AWS-Konto eine bestimmte Themenaktion erlauben (z. B. Veröffentlichen). Weitere Informationen finden Sie unter AWS-Konto-Zugriff auf ein Thema gewähren.
- Sie möchten die Abonnements für Ihr Thema nur auf das HTTPS-Protokoll beschränken. Weitere Informationen finden Sie unter Beschränken von Abonnements auf HTTPS.
- Sie möchten, dass Amazon SNS in der Lage ist, Nachrichten für Ihre Amazon SQS-Queue zu veröffentlichen. Weitere Informationen finden Sie unter Veröffentlichen in einer Amazon SQS-Queue..

Die wichtigsten Konzepte

In den folgenden Abschnitten werden die Konzepte beschrieben, die Sie in Bezug auf die Verwendung der Zugriffskontrollsprache kennen sollten. Sie sind logisch angeordnet, das heißt, die wichtigsten Konzepte befinden sich ganz oben in der Liste.

Themen

- Berechtigung
- Statement
- Richtlinie
- Aussteller
- Auftraggeber
- Aktion
- Ressource
- Bedingungen und Schlüssel
- Auftraggeber

- Bewertung
- Auswirkung
- Standardmäßige Ablehnung
- Sobald Sie die Details auf dieser Seite überprüft haben, klicken Sie auf
- Explizite Zugriffsverweigerung

Berechtigung

Eine Berechtigung ist das Konzept, eine bestimmte Art des Zugriffs auf eine bestimmte Ressource zu erteilen bzw. zu verweigern. Berechtigungen folgen im Allgemeinen dieser Form: "A darf (bzw. darf nicht) B bei C tun, wenn D gilt". Beispiel: Jane (A) hat die Berechtigung zum Veröffentlichen (B) zum Thema A (C), sofern sie das HTTP-Protokoll verwendet (D). Jedes Mal, wenn Jane eine Veröffentlichung zum Thema A vornimmt, prüft der Service, ob Jane dazu berechtigt ist und ob die Anforderung die Bedingungen erfüllt, die für die Berechtigung vorgegeben sind.

Statement

Eine Anweisungist die formelle Beschreibung einer einzelnen Berechtigung, die in der Zugriffskontrollsprache verfasst wird. Sie schreiben eine Anweisung stets als Teil eines umfassenderen Containerdokuments, das als Richtlinie bezeichnet wird (siehe nächstes Konzept).

Richtlinie

Eine Richtlinie ist ein Dokument (verfasst in der Zugriffskontrollsprache), das als Container für eine oder mehrere Anweisungen fungiert. Eine Richtlinie könnte beispielsweise zwei Anweisungen enthalten: eine Anweisung, die besagt, dass Jane unter Verwendung des E-Mail-Protokolls ein Abonnement durchführen darf, und eine andere, die besagt, dass Bob nicht zum Thema A veröffentlichen darf. Die folgende Abbildung zeigt ein analoges Szenario mit zwei Richtlinien: Eine besagt, dass Jane unter Verwendung des E-Mail-Protokolls abonnieren darf, und eine andere, dass Bob nicht zum Thema A veröffentlichen darf.

In Richtliniendokumenten sind nur ASCII-Zeichen zulässig. Sie können `aws:SourceAccount` und `aws:SourceOwner` verwenden, um das Szenario zu umgehen, in dem Sie die ARNs anderer AWS-Services integrieren müssen, die Nicht-ASCII-Zeichen enthalten. Sehen Sie sich die Unterschiede zwischen `aws:SourceAccount` im Vergleich zu `aws:SourceOwner` an.

Aussteller

Der Aussteller ist die Person, die eine Richtlinie schreibt, um Berechtigungen für eine Ressource zu gewähren. Der Aussteller ist definitionsgemäß immer der Eigentümer der Ressource. AWS gestattet AWS-Benutzern nicht, Richtlinien für Ressourcen zu erstellen, deren Eigentümer sie nicht sind. Angenommen, John besitzt eine Ressource. Wenn John die von ihm erstellte Richtlinie übermittelt, mit der er Berechtigungen für diese Ressource erteilt, wird seine Identität von AWS authentifiziert.

Auftraggeber

Der Prinzipal ist die Person bzw. sind die Personen, die die Berechtigung in der Richtlinie erhalten. In der Anweisung "A ist berechtigt, B bei C zu tun, sofern D gilt" ist der Prinzipal "A". Sie können den Prinzipal in einer Richtlinie auf "alle" festlegen (d. h. Sie können einen Platzhalter angeben, der alle Personen umfasst). Sie können dies z. B. tun, wenn Sie den Zugriff nicht basierend auf der tatsächlichen Identität des Anforderers einschränken möchten, sondern aufgrund anderer Identifikationsmerkmale (z. B. der IP-Adresse des Anforderers).

Aktion

Die Aktion ist die Aktivität, die der Prinzipal ausführen darf. Die Aktion ist das „B“ in der Aussage „A ist berechtigt, B bei C zu tun, sofern D gilt“. Normalerweise ist die Aktion nur der Vorgang in der Anfrage an AWS. Beispiel: Jane sendet mit `Action=Subscribe`. eine Anfrage an Amazon SNS Sie können in einer Richtlinie eine oder mehrere Aktionen angeben.

Ressource

Die Ressource ist das Objekt, für die der Prinzipal Zugriff anfordert. In der Anweisung "A ist berechtigt, B bei C zu tun, sofern D gilt" ist die Ressource das "C".

Bedingungen und Schlüssel

Bedingungen sind jegliche Einschränkungen oder Details hinsichtlich der Berechtigung. Die Bedingung ist das D in der Aussage „A ist berechtigt, B bei C zu tun, sofern D gilt“. Der Teil der Richtlinie, der die Bedingungen festlegt, kann der zugleich der ausführlichste und komplexeste aller Bestandteile sein. Typische Bedingungen betreffen:

- Datum und Uhrzeit (z. B.: Die Anfrage muss vor einem bestimmten Tag eintreffen.)
- IP-Adresse (z. B. muss die IP-Adresse des Anforderers Teil eines bestimmten CIDR-Bereichs sein)

Ein Schlüssel ist das besondere Merkmal, das die Grundlage für die Einschränkung des Zugriffs bildet, z. B. das Datum und die Uhrzeit der Anfrage.

Sie verwenden Bedingungen und Schlüssel zusammen, um die Einschränkung auszudrücken. Am besten lässt sich anhand eines Beispiel zeigen, wie Sie eine Einschränkung implementieren: Wenn Sie den Zugriff auf die Zeit vor dem 30. Mai 2010 einschränken möchten, verwenden Sie die Bedingung `DateLessThan`. Sie verwenden den Schlüssel namens`aws:CurrentTime`und setzen Sie es auf den Wert`2010-05-30T00:00:00Z`.AWSdefiniert die Bedingungen und Schlüssel, die Sie verwenden können. Der AWS-Service selbst (z. B. Amazon SQS oder Amazon SNS) kann auch servicespezifische Schlüssel definieren. Weitere Informationen finden Sie unter Amazon SNS-API-Berechtigungen: Referenztabelle für Aktionen und Ressourcen.

Auftraggeber

Der Anforderer ist die Person, die eine Anfrage an einen AWS-Service sendet und um Zugriff auf eine bestimmte Ressource bittet. Der Anforderer sendet eine Anfrage an AWS, die Folgendes besagt: "Erlaubst du mir, B bei C auszuführen, wenn D gilt?".

Bewertung

Die Bewertung ist der Prozess, mit dem der AWS-Service bestimmt, ob eine eingehende Anfrage basierend auf den geltenden Richtlinien abgelehnt oder zugelassen werden soll. Informationen zur Bewertungslogik finden Sie unter Auswertungslogik.

Auswirkung

Die Auswirkung ist das Ergebnis, das eine Richtlinienanweisung zum Bewertungszeitpunkt zurückgeben soll. Sie geben diesen Wert beim Schreiben der Anweisungen in einer Richtlinie an. Die mögliche Werte sind Zugriffsverweigerung und Zugriffserlaubnis.

Sie können beispielsweise eine Richtlinie schreiben, die eine Anweisung enthält, die alle Anfragen von der Antarktis (Auswirkung = Zugriffsverweigerung, da in der Anfrage eine IP-Adresse verwendet wird, die der Antarktis zugewiesen ist) ablehnt. Sie könnten auch eine Richtlinie schreiben, die eine Anweisung enthält, die alle Anfragen, die nicht von der Antarktis stammen, zulässt (Auswirkung = Zugriffserlaubnis, da die Anfrage nicht von der Antarktis stammt). Diese beiden Anweisungen scheinen zwar das Gleiche zu tun, unterscheiden sich aber nach der Logik der Zugriffskontrollsprache. Weitere Informationen finden Sie unter Auswertungslogik.

Zwar gibt es für die Angabe der Auswirkung (Zulassen oder Ablehnen) nur zwei mögliche Werte, dennoch kann es bei der Bewertung der Richtlinie drei unterschiedliche Ergebnisse geben: standardmäßige Ablehnung, Zugriffserlaubnis oder explizite Zugriffsverweigerung. Weitere Informationen finden Sie in den folgenden Konzepten und unter Auswertungslogik.

Standardmäßige Ablehnung

Eine standardmäßige Ablehnung ist das Standardergebnis einer Richtlinie, wenn weder "Zugriffserlaubnis" noch "explizite Zugriffsverweigerung" vorhanden sind.

Sobald Sie die Details auf dieser Seite überprüft haben, klicken Sie auf

Zugriffserlaubnis

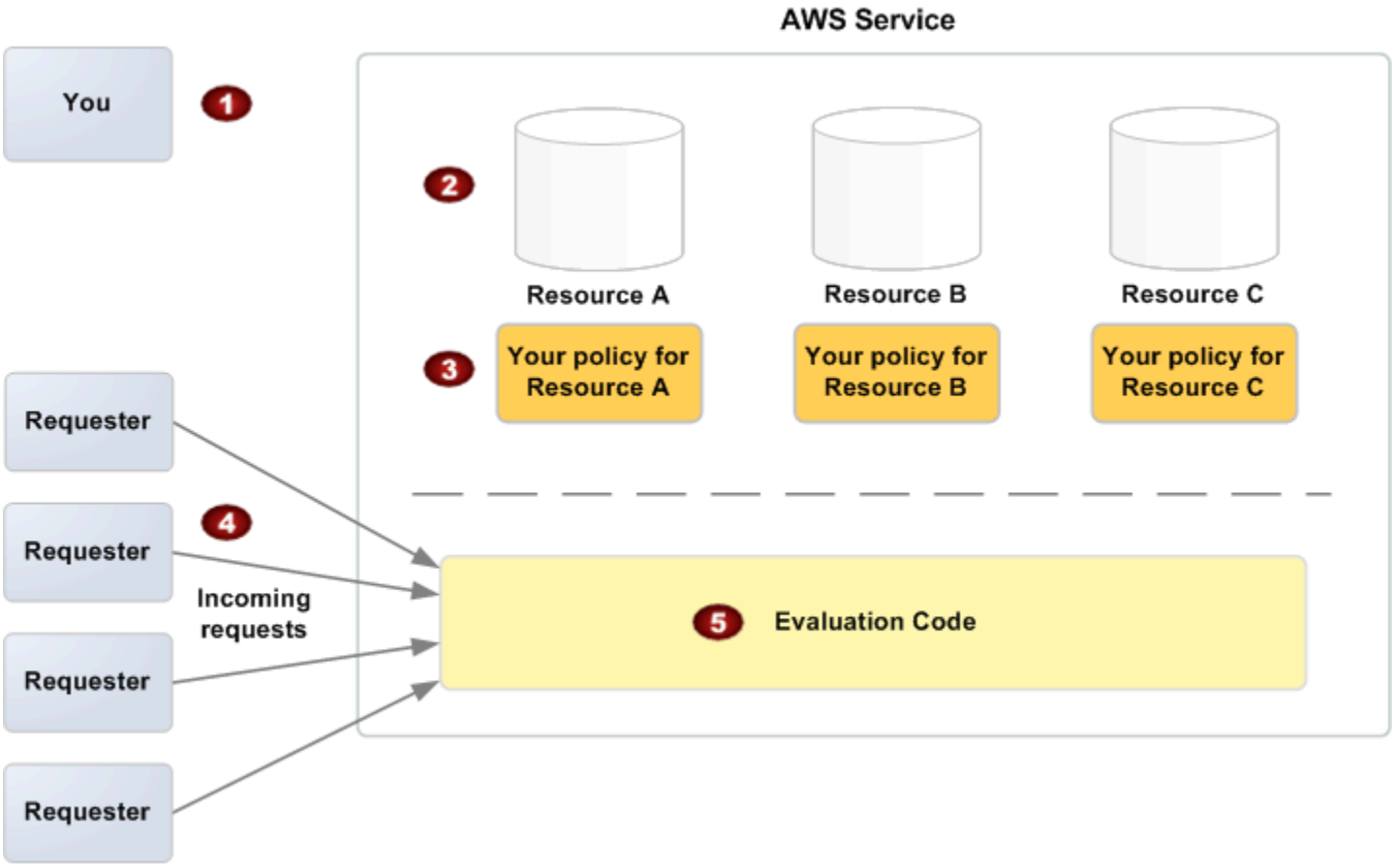
Zugriffserlaubnis resultiert aus einer Anweisung, die die Auswirkung "Zugriffserlaubnis" hat, davon ausgehend, dass alle angegebenen Bedingungen erfüllt sind. Beispiel: Alle Anfragen zulassen, wenn sie am 30. April 2010 vor 13:00 Uhr eingehen. Durch "Zugriffserlaubnis" werden alle standardmäßigen Ablehnungen überschrieben, niemals jedoch eine explizite Zugriffsverweigerung.

Explizite Zugriffsverweigerung

Eine explizite Zugriffsverweigerung resultiert aus einer Anweisung, die die Auswirkung "Zugriffsverweigerung" hat, davon ausgehend, dass alle angegebenen Bedingungen erfüllt sind. Beispiel: Alle Anfragen ablehnen, wenn sie von der Antarktis stammen. Alle Anfragen von der Antarktis werden immer abgelehnt, unabhängig davon, was andere Richtlinien ggf. erlauben.

Übersicht über die Architektur

In der folgenden Abbildung und Tabelle werden die Hauptkomponenten beschrieben, die interagieren, um eine Zugriffskontrolle für Ihre Ressourcen zu bieten.

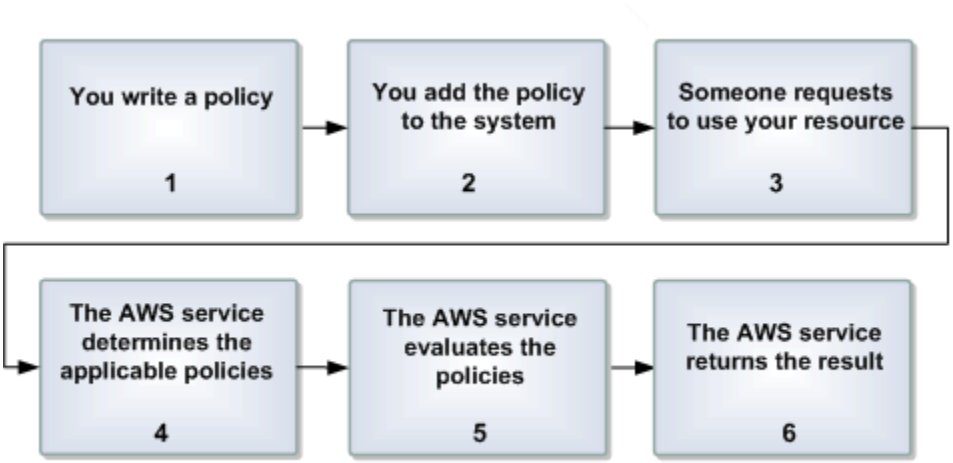


1	Sie selbst als Ressourceneigentümer.
2	Ihre Ressourcen (die im AWS-Service; z. B. Amazon-SQS-Warteschlange enthalten sind).
3	Ihre Richtlinien. In der Regel verfügen Sie über eine Richtlinie pro Ressource. Sie können aber auch mehrere haben. Der AWS-Service selbst stellt eine API bereit, mit der Sie Ihre Richtlinien hochladen und verwalten können.
4	Anforderer und deren eingehende Anforderungen an den AWS-Service.
5	Code für die Auswertung der Zugriffskontrollsprache

	Dies ist der Codesatz im AWS-Service, mit dem eingehende Anforderungen hinsichtlich der zutreffenden Richtlinien ausgewertet werden und bestimmt wird, ob dem Anforderer Zugriff auf die Ressource gewährt wird. Weitere Informationen dazu, wie der Service die Entscheidung trifft, finden Sie unter Auswertungslogik.

Verwenden der Zugriffskontrollsprache

Die folgende Abbildung und Tabelle beschreiben den allgemeinen Prozess der Zugriffskontrolle in der Zugriffskontrollsprache.



Verfahren für die Verwendung der Zugriffsrichtliniensprache

1	Sie schreiben eine Richtlinie für Ihre Ressource. Sie schreiben z. B. eine Richtlinie, um die Berechtigungen für Ihre Amazon-SNS-Themen anzugeben.
2	Sie laden Ihre Richtlinien in AWS hoch. Der AWS-Service selbst stellt eine API bereit, mit der Sie Ihre Richtlinien hochladen können. Sie verwenden beispielsweise die Amazon SNS `SetTopicAttributes` -Aktion zum Hochladen einer Richtlinie für ein bestimmtes Amazon SNS-Thema.
3	Jemand sendet eine Anfrage zur Verwendung Ihrer Ressource. Beispielsweise sendet ein Benutzer eine Anfrage an Amazon SNS zur Verwendung eines Ihrer Themen.

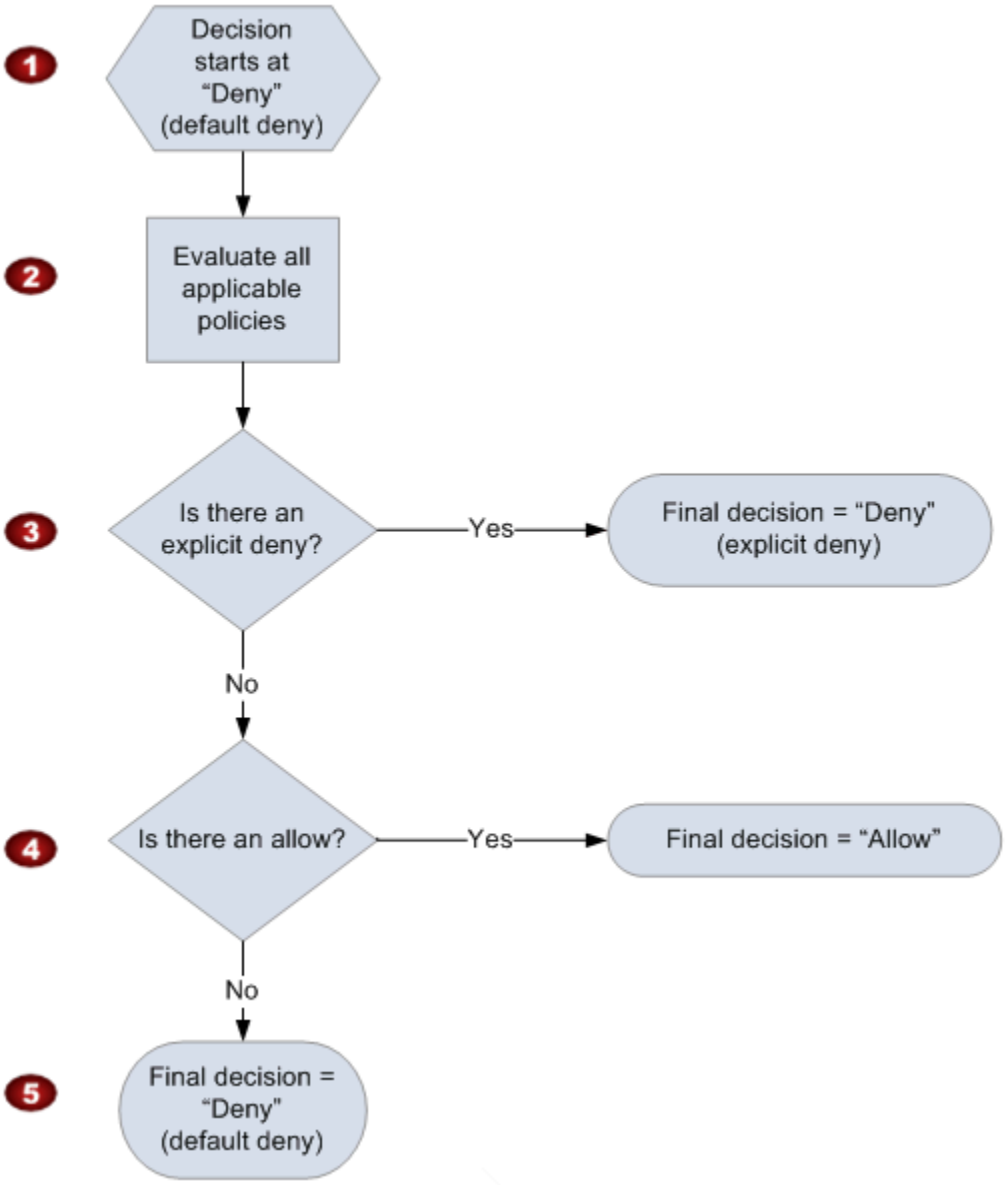
4	Der AWS-Service bestimmt, welche Richtlinien für die Anfrage gelten. Zum Beispiel überprüft Amazon SNS alle verfügbaren Amazon SNS-Richtlinien und bestimmt dann, welche anwendbar sind (abhängig davon, um welche Ressource es sich handelt, wer der Anforderer ist usw.).
5	Der AWS-Service bewertet die Richtlinien. Zum Beispiel bewertet Amazon SNS die Richtlinien und bestimmt, ob der Anforderer Ihr Thema verwenden darf oder nicht. Informationen zur Entscheidungslogik finden Sie unter Auswertungslogik.
6	Der AWS-Service lehnt die Anfrage ab oder fährt mit ihrer Verarbeitung fort. Basierend auf dem Ergebnis der Richtlinienbewertung gibt der Service z. B. entweder den Fehler "Zugriff verweigert" an den Anforderer zurück oder fährt mit der Verarbeitung der Anfrage fort.

Auswertungslogik

Das Ziel zum Bewertungszeitpunkt ist, zu entscheiden, ob eine bestimmte Anfrage gewährt oder abgelehnt werden soll. Die Auswertungslogik unterliegt mehreren Grundregeln:

- Standardmäßig werden alle Anforderungen zur Verwendung Ihrer Ressourcen, die nicht von Ihnen stammen, verweigert
- Eine Zugriffserlaubnis überschreibt jedwede standardmäßige Ablehnung.
- Eine explizite Zugriffsverweigerung überschreibt jedwede Zugriffserlaubnis
- Es spielt keine Rolle, in welcher Reihenfolge die Richtlinien ausgewertet werden

Das folgende Flussdiagramm stellt genauer dar, wie die Entscheidung getroffen wird.



1	Die Entscheidung beginnt mit einer standardmäßigen Ablehnung.
2	Durch den Durchführungscode werden alle Richtlinien ausgewertet, die auf die Anfrage anwendbar sind (basierend auf der Ressource, dem Prinzipal, der Aktion und den Bedingungen). Es spielt keine Rolle, in welcher Reihenfolge die Richtlinien ausgewertet werden.
3	Es wird in allen Richtlinien nach expliziten Anweisungen für Zugriffsverweigerungen gesucht, die auf die Anforderung zutreffen.

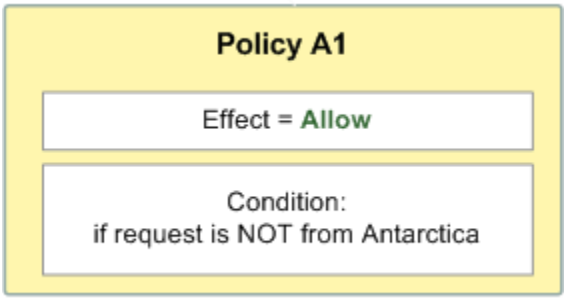
	Wenn auch nur eine gefunden wird, gibt der Durchführungscode eine Ablehnungsentschei dung zurück und der Prozess wird beendet (dies ist eine explizite Zugriffsverweigerung; weitere Informationen siehe Explizite Zugriffsverweigerung).
4	Wenn keine explizite Zugriffsverweigerung gefunden wird, wird nach "allow"-Anweisungen gesucht, die auf die Anfrage zutreffen. Wenn auch nur eine gefunden wird, gibt der Durchführungscode eine Zugriffserlaubnis zurück und der Prozess ist beendet (bzw. der Service setzt die Verarbeitung der Anfrage fort).
5	Wenn kein "allow" gefunden wird, ist die endgültige Entscheidung "deny" (da es keine explizite Zugriffsverweigerung bzw. Zugriffserlaubnis gab, gilt dies als standardmäßige Ablehnung) (weitere Informationen siehe Standardmäßige Ablehnung).

Das Zusammenspiel von expliziten und standardmäßigen Zugriffsverweigerungen

Wenn eine Richtlinie nicht direkt auf eine Anfrage anwendbar ist, wird der Zugriff standardmäßig verweigert ("standardmäßige Ablehnung"). Wenn z. B. ein Benutzer eine Anfrage über die Verwendung von Amazon SNS sendet, die Richtlinie für das Thema aber nicht auf das AWS-Konto-Konto des Benutzers verweist, führt diese Richtlinie zu einer standardmäßigen Ablehnung.

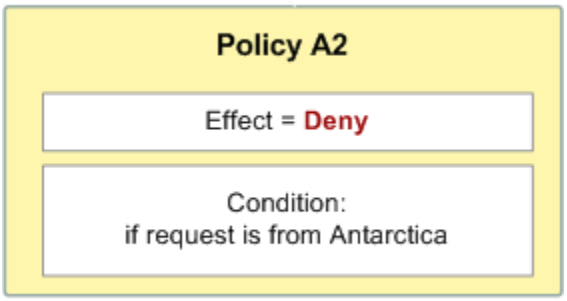
Eine Richtlinie hat außerdem eine standardmäßige Ablehnung zur Folge, wenn eine Bedingung in einer Anweisung nicht erfüllt ist. Wenn alle Bedingungen einer Anweisung erfüllt sind, hat dies für diese Richtlinie abhängig vom Wert des Auswirkungselements in der Richtlinie entweder eine Zugriffserlaubnis oder eine explizite Zugriffsverweigerung zur Folge. In Richtlinien ist nicht festgelegt, was geschieht, wenn eine Bedingung nicht erfüllt ist, daher ist das Ergebnis in diesem Fall eine standardmäßige Ablehnung.

Angenommen, Sie möchten Anforderungen verhindern, die aus der Antarktis stammen. Sie erstellen eine Richtlinie ("Richtlinie A1"), um Zugriff nur dann zu gewähren, wenn Anfragen von außerhalb von Antarktis kommen. Diese Richtlinie ist in der folgenden Abbildung dargestellt.



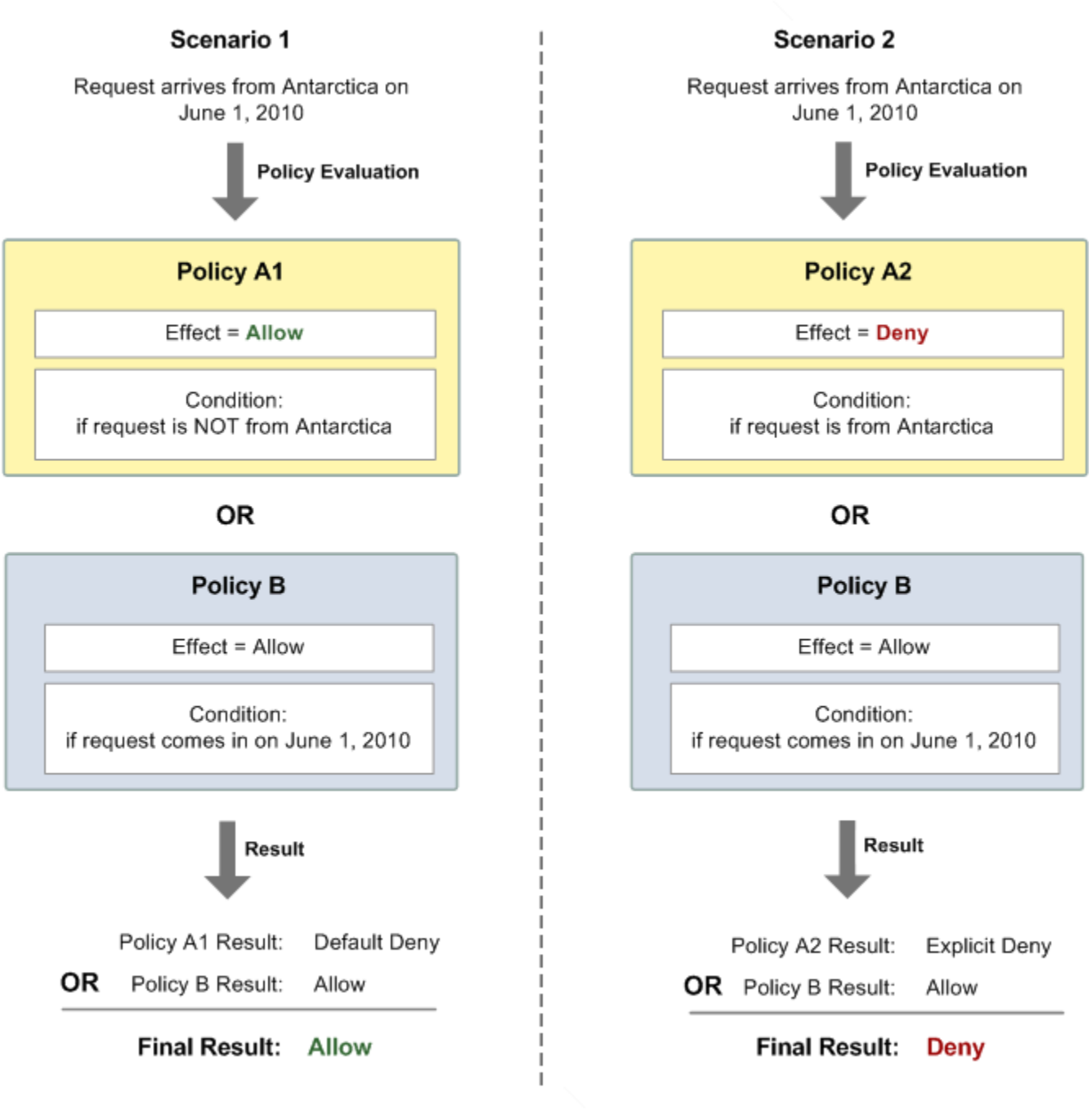
Wenn eine Anforderung aus den USA gesendet wird, ist die Bedingung erfüllt (die Anforderung stammt nicht aus der Antarktis). Daher wird die Anforderung zugelassen. Wenn jedoch eine Anforderung aus der Antarktis eingeht, ist die Bedingung nicht erfüllt und der Zugriff wird standardmäßig abgelehnt.

Sie könnten das Ergebnis in eine explizite Zugriffsverweigerung umwandeln, indem Sie die Richtlinie ("Richtlinie A2") wie im folgenden Diagramm dargestellt neu schreiben. Hier lehnt die Richtlinie eine Anfrage explizit ab, wenn sie von der Antarktis stammt.



Wenn eine Anforderung aus der Antarktis eingeht, ist die Bedingung erfüllt und das Richtlinienergebnis ist daher eine explizite Zugriffsverweigerung.

Die Unterscheidung zwischen "standardmäßiger Ablehnung" und "expliziter Zugriffsverweigerung" ist wichtig, da eine standardmäßige Ablehnung durch eine Zugriffserlaubnis überschrieben werden kann, eine explizite Zugriffsverweigerung jedoch nicht. Angenommen, es gibt eine andere Richtlinie, die Anforderungen zulässt, wenn diese am 1. Juni 2010 gesendet werden. Zu welchem Gesamtergebnis führt diese Richtlinie, wenn sie gemeinsam mit der Richtlinie ausgewertet wird, die Zugriff aus der Antarktis verweigert? Wir vergleichen das Gesamtergebnis, wenn diese datenbasierte Richtlinie (nennen wir sie "Richtlinie B") zusammen mit den vorherigen Richtlinien (A1 und A2) ausgewertet wird. In Szenario 1 werden die Richtlinien A1 und B zusammen ausgewertet, in Szenario 2 die Richtlinien A2 und B. Die folgende Abbildung zeigt die Ergebnisse, wenn eine Anfrage am 1. Juni 2010 aus der Antarktis eingeht.



In Szenario 1 gibt Richtlinie A1 eine standardmäßige Ablehnung wie zuvor beschrieben zurück. Richtlinie B gibt eine Zugriffserlaubnis zurück, da die Richtlinie (laut Definition) Anfragen zulässt, die am 1. Juni 2010 eingehen. Die Zugriffserlaubnis aus Richtlinie B überschreibt die standardmäßige Zugriffsverweigerung aus Richtlinie A1 und die Anforderung wird zugelassen.

In Szenario 2 gibt Richtlinie A2 eine explizite Zugriffsverweigerung wie zuvor beschrieben zurück. Richtlinie B gibt auch hier "allow" zurück. Die Zugriffserlaubnis aus Richtlinie B wird durch die explizite Zugriffsverweigerung aus Richtlinie A2 überschrieben und die Anforderung wird abgelehnt.

Beispiele für die Zugriffskontrolle in Amazon SNS

In diesem Abschnitt finden Sie einige typische Anwendungsfälle für die Zugriffskontrolle.

Themen

AWS-Konto-Zugriff auf ein Thema gewähren

Angenommen, Sie verfügen über ein Thema im Amazon SNS-System. Im einfachsten Fall möchten Sie einem oder mehreren AWS-Konten-Konten Zugriff auf eine bestimmte Themenaktion (z. B. Veröffentlichen) erlauben.

Dies können Sie über die Amazon SNS -API-Aktion `AddPermission` implementieren. Diese nimmt ein Thema, eine Liste der AWS-Konto-IDs, eine Liste der Aktionen und eine Bezeichnung und erstellt automatisch eine neue Anweisung in der Zugriffskontrollrichtlinie des Themas. In diesem Fall müssen Sie nicht selbst eine Richtlinie schreiben, da Amazon SNS die neue Richtlinienanweisung automatisch generiert. Sie können die Richtlinienanweisung zu einem späteren Zeitpunkt entfernen, indem Sie `RemovePermission` mit der zugehörigen Bezeichnung aufrufen.

Wenn Sie beispielsweise `AddPermission` für das Thema arn:aws:sns:us-east-2:444455556666: MyTopicmit der AWS-Konto ID 1111-2222-3333, der `Publish` Aktion und der Bezeichnung aufgerufen haben`grant-1234-publish`, würde Amazon SNS die folgende Zugriffskontrollrichtlinienanweisung generieren und einfügen:

```
{
  "Statement": [{
```

```
    "Sid": "grant-1234-publish",
    "Effect": "Allow",
    "Principal": {
      "AWS": "111122223333"
    },
    "Action": ["sns:Publish"],
    "Resource": "arn:aws:sns:us-east-2:444455556666:MyTopic"
  }]
}
```

Nachdem diese Anweisung hinzugefügt wurde, kann der Benutzer mit AWS-Konto 1111-2222-3333 Nachrichten zu dem Thema veröffentlichen.

Beschränken von Abonnements auf HTTPS

Im folgenden Beispiel schränken Sie das Protokoll für die Benachrichtigungsübermittlung auf HTTPS ein.

Sie müssen wissen, wie Sie eine eigene Richtlinie für das Thema schreiben, da Sie mit der Amazon SNS `AddPermission`-Aktion keine Protokollbeschränkung angeben können, wenn Sie jemandem Zugriff auf Ihr Thema gewähren. In diesem Fall würden Sie eine eigene Richtlinie schreiben und dann die Aktion `SetTopicAttributes` verwenden, um das `Policy`-Attribut des Themas für Ihre neue Richtlinie festzulegen.

Im folgenden Beispiel einer vollständigen Richtlinie wird der AWS-Konto-ID 1111-2222-3333 die Fähigkeit gewährt, Benachrichtigungen von einem Thema zu abonnieren.

```
{
  "Statement": [{
    "Sid": "Statement1",
    "Effect": "Allow",
    "Principal": {
      "AWS": "111122223333"
    },
    "Action": ["sns:Subscribe"],
    "Resource": "arn:aws:sns:us-east-2:444455556666:MyTopic",
    "Condition": {
      "StringEquals": {
        "sns:Protocol": "https"
      }
    }
  }]
```

```
}
```

Veröffentlichen in einer Amazon SQS-Queue.

In diesem Anwendungsfall möchten Sie Nachrichten von Ihrem Thema an Ihre Amazon-SQS-Warteschlange veröffentlichen. Wie Amazon SNS verwendet Amazon SQS die Zugriffskontrollsprache von Amazon für die Zugriffskontrolle. Um es Amazon SNS zu erlauben, Nachrichten zu senden, müssen Sie mithilfe der Amazon SQS-Aktion `SetQueueAttributes` eine Richtlinie für die Queue einrichten.

Auch hier müssen Sie wissen, wie Sie eine eigene Richtlinie schreiben, da die Amazon SQS `AddPermission`-Aktion keine Richtlinienanweisungen mit Bedingungen erstellt.

> **Note**
>
> Das unten aufgeführte Beispiel ist eine Amazon SQS -Richtlinie (zur Steuerung des Zugriffs auf Ihre Queue) und keine Amazon SNS-Richtlinie (zur Steuerung des Zugriffs auf Ihr Thema). Bei den Aktionen handelt es sich um Amazon SQS-Aktionen und die Ressource ist der Amazon-Ressourcenname (ARN) der Queue. Sie können den ARN der Queue durch Abrufen des `QueueArn`-Attributs der Queue mit der Aktion `GetQueueAttributes` ermitteln.

```
{
  "Statement": [{
    "Sid": "Allow-SNS-SendMessage",
    "Effect": "Allow",
    "Principal": {
      "Service": "sns.amazonaws.com"
    },
    "Action": ["sqs:SendMessage"],
    "Resource": "arn:aws:sqs:us-east-2:444455556666:MyQueue",
    "Condition": {
      "ArnEquals": {
        "aws:SourceArn": "arn:aws:sns:us-east-2:444455556666:MyTopic"
      }
    }
  }]
}
```

Diese Richtlinie verwendet die Bedingung `aws:SourceArn`, um den Zugriff auf die Queue basierend auf der Quelle der Nachricht, die an die Queue gesendet wird, zu beschränken. Sie können diese Art von Richtlinie verwenden, um es Amazon SNS zu erlauben, Nachrichten an Ihre Queue zu senden, allerdings nur dann, wenn die Nachrichten von einem Ihrer eigenen Themen stammen. In diesem Fall geben Sie ein bestimmtes Ihrer Themen an, dessen ARN arn:aws:sns:us-east-2:444455556666 ist:MyTopic.

Die vorangegangene Richtlinie ist ein Beispiel für die Amazon SQS-Richtlinie, die Sie schreiben und zu einer bestimmten Queue hinzufügen könnten. Es würde Zugriff auf Amazon SNS und andere AWS-Services. Amazon SNS stellt für alle neu erstellten Themen eine Standardrichtlinie bereit. Die Standardrichtlinie gewährt allen anderen AWS-Services Zugriff auf Ihr Thema. Diese Standardrichtlinie verwendet eine `aws:SourceArn`-Bedingung, um sicherzustellen, dass AWS-Services nur im Namen von AWS-Ressourcen, die Sie besitzen, auf Ihr Thema zugreifen.

Erlauben Sie Amazon S3 Ereignisbenachrichtigungen für ein Thema

In diesem Fall können Sie die Richtlinie eines Themas so konfigurieren, dass der Amazon S3-Bucket eines anderen AWS-Kontos eine Veröffentlichung zu Ihrem Thema vornehmen kann. Weitere Informationen zum Veröffentlichen von Benachrichtigungen von Amazon S3 finden Sie unter Setting Up Notifications of Bucket Events.

Bei diesem Beispiel wird davon ausgegangen, dass Sie eine eigene Richtlinie schreiben und dann die Aktion `SetTopicAttributes` verwenden, um das `Policy`-Attribut des Themas für Ihre neue Richtlinie festzulegen.

Die folgende Beispielanweisung verwendet die Bedingung `SourceAccount`, um sicherzustellen, dass nur das Amazon S3-Eigentümerkonto auf das Thema zugreifen kann. In diesem Beispiel ist der Themeneigentümer „111122223333“ und der Amazon S3 Eigentümer „444455556666“. Das Beispiel besagt, dass jeder Amazon S3-Bucket, der zu 444455556666 gehört, in veröffentlichen darf MyTopic.

```
{
  "Statement": [{
    "Effect": "Allow",
     "Principal": {
      "Service": "s3.amazonaws.com"
    },
    "Action": "sns:Publish",
    "Resource": "arn:aws:sns:us-east-2:111122223333:MyTopic",
    "Condition": {
      "StringEquals": {
```

```
        "AWS:SourceAccount": "444455556666"
      }
    }
  }]
}
```

Beim Veröffentlichen von Ereignissen in Amazon SNS unterstützen die folgenden Services `aws:SourceAccount`:

- Amazon API Gateway
- Amazon CloudWatch
- Amazon DevOpsGuru
- Amazon ElastiCache
- Amazon GameLift
- Amazon Pinpoint-SMS- und Sprachnachrichten-API
- Amazon RDS
- Amazon Redshift
- Amazon S3 Glacier
- Amazon SES
- Amazon Simple Storage Service
- AWS CodeCommit
- AWS Directory Service
- AWS Lambda
- AWS Systems Manager Incident Manager

Erlauben Sie Amazon SES, in einem Thema zu veröffentlichen, das einem anderen Konto gehört

Sie können eine andereAWS-Dienst verwenden, um in einem Thema zu veröffentlichen, das im Besitz eines anderenAWS-Kontoist. Angenommen, Sie haben sich beim 111122223333 Konto angemeldet, Amazon SES geöffnet und eine E-Mail erstellt. Um Benachrichtigungen zu dieser E-Mail an ein Amazon SNS-Thema zu veröffentlichen, das das Konto 444455556666 besitzt, erstellen Sie eine Richtlinie wie die folgende. Um dies zu tun, müssen Sie Informationen über den Prinzipal (den anderen Dienst) und den Besitz jeder Ressource angeben. Die`Resource`-Anweisung enthält das Thema ARN, das die Konto-ID des Topic-Besitzers 444455556666 enthält.

Die`"aws:SourceOwner": "111122223333"`-Anweisung gibt an, dass Ihr Konto Eigentümer der E-Mail ist.

```
{
  "Version": "2008-10-17",
  "Id": "__default_policy_ID",
  "Statement": [
    {
      "Sid": "__default_statement_ID",
      "Effect": "Allow",
      "Principal": {
        "Service": "ses.amazonaws.com"
      },
      "Action": "SNS:Publish",
      "Resource": "arn:aws:sns:us-east-2:444455556666:MyTopic",
      "Condition": {
        "StringEquals": {
          "aws:SourceOwner": "111122223333"
        }
      }
    }
  ]
}
```

Beim Veröffentlichen von Ereignissen in Amazon SNS unterstützen die folgenden Services `aws:SourceOwner`:

- Amazon API Gateway
- Amazon CloudWatch
- Amazon DevOpsGuru
- Amazon ElastiCache
- Amazon GameLift
- Amazon Pinpoint-SMS- und Sprachnachrichten-API
- Amazon RDS
- Amazon Redshift
- Amazon SES
- AWS CodeCommit
- AWS Directory Service

- AWS Lambda
- AWS Systems Manager Incident Manager

`aws:SourceAccount` im Vergleich zu `aws:SourceOwner`

> **Important**
>
> `aws:SourceOwner` ist veraltet, und neue Services können nur über `aws:SourceArn` und `aws:SourceAccount` in Amazon SNS integriert werden. Amazon SNS behält weiterhin die Abwärtskompatibilität für bestehende Services bei, die derzeit `aws:SourceOwner` unterstützen.

Die Bedingungsschlüssel `aws:SourceAccount` und `aws:SourceOwner` werden jeweils von einigen AWS-Services-Services verwendet, wenn sie in einem Amazon-SNS-Thema veröffentlicht werden. Wenn dies unterstützt wird, ist der Wert der 12-stellige AWS-Konto-ID, in dessen Auftrag der Dienstdaten veröffentlicht. Einige Dienste unterstützen einen, andere unterstützen den anderen.

- Siehe Erlauben Sie Amazon S3 Ereignisbenachrichtigungen für ein Thema für die Verwendung von `aws:SourceAccount` durch Amazon S3-Benachrichtigungen und eine Liste von AWS-Diensten, die diese Bedingung unterstützen.
- Siehe Erlauben Sie Amazon SES, in einem Thema zu veröffentlichen, das einem anderen Konto gehört für die Verwendung von Amazon SES `aws:SourceOwner` und eine Liste von AWS-Diensten, die diese Bedingung unterstützen.

Zulassen, dass Konten in einer AWS Organizations-Organisation Veröffentlichungen zu einem Thema in einem anderen Konto vornehmen können

Der AWS Organizations-Service hilft Ihnen dabei, Abrechnungen zentral zu verwalten, den Zugriff und die Sicherheit zu kontrollieren und Ressourcen über Ihre AWS-Konten hinweg freizugeben.

Sie finden Ihre Organisations-ID in der Organisationskonsole. Weitere Informationen finden Sie unter Anzeigen von Details zu einer Organisation vom Masterkonto aus.

In diesem Beispiel kann jedes AWS-Konto-Konto in der `myOrgId` der Organisation Veröffentlichungen zum Amazon-SNS-Thema `MyTopic` im Konto `444455556666` vornehmen. Die Richtlinie überprüft den Organisations-ID-Wert mithilfe des globalen Bedingungsschlüssels `aws:PrincipalOrgID`.

```
{
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": "*"
            },
            "Action": "SNS:Publish",
            "Resource": "arn:aws:sns:us-east-2:444455556666:MyTopic",
            "Condition": {
                "StringEquals": {
                    "aws:PrincipalOrgID": "myOrgId"
                }
            }
        }
    ]
}
```

Jedem CloudWatch Alarm erlauben, zu einem Thema in einem anderen Konto zu veröffentlichen

In diesem Fall 111122223333 dürfen alle CloudWatch Alarme im Konto in einem Amazon SNS-Thema im Konto veröffentlichen444455556666.

```
{
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": "*"
            },
            "Action": "SNS:Publish",
            "Resource": "arn:aws:sns:us-east-2:444455556666:MyTopic",
            "Condition": {
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:cloudwatch:us-
east-2:111122223333:alarm:*"
                }
            }
        }
    ]
}
```

Die Veröffentlichung von einem bestimmten VPC-Endpunkt aus auf nur ein Amazon SNS -Thema beschränken

In diesem Fall darf das Thema im Konto 444455556666 nur vom VPC-Endpunkt mit der ID `vpce-1ab2c34d` veröffentlicht werden.

```
{
  "Statement": [{
    "Effect": "Deny",
    "Principal": "*",
    "Action": "SNS:Publish",
    "Resource": "arn:aws:sns:us-east-2:444455556666:MyTopic",
    "Condition": {
      "StringNotEquals": {
        "aws:sourceVpce": "vpce-1ab2c34d"
      }
    }
  }]
}
```

So funktioniert Amazon Simple Notification Service mit IAM

Bevor Sie mit IAM den Zugriff auf Amazon SNS verwalten können, sollten Sie sich darüber informieren, welche IAM-Funktionen Sie mit Amazon SNS verwenden können.

IAM-Funktionen, die Sie mit Amazon Simple Notification Service verwenden können

IAM-Feature	Amazon-SNS-Support
Identitätsbasierte Richtlinien	Ja
Ressourcenbasierte Richtlinien	Ja
Richtlinienaktionen	Ja
Richtlinienressourcen	Ja
Richtlinienbedingungsschlüssel (services pezifisch)	Ja
ACLs	Nein

IAM-Feature	Amazon-SNS-Support
ABAC (Tags in Richtlinien)	Teilweise
Temporäre Anmeldeinformationen	Ja
Hauptberechtigungen	Ja
Servicerollen	Ja
Service-verknüpfte Rollen	Nein

Einen Überblick über das Zusammenwirken von Amazon SNS und anderen AWS-Services mit den meisten IAM-Funktionen finden Sie unter AWS-Services, die mit IAM funktionieren im IAM-Benutzerhandbuch.

Richtlinienaktionen für Amazon SNS

Unterstützt Richtlinienaktionen	Ja

Administratoren können mithilfe von AWS-JSON-Richtlinien festlegen, wer zum Zugriff auf was berechtigt ist. Das heißt, welcher Prinzipal kann Aktionen für welche Ressourcen und unter welchen Bedingungen ausführen.

Das Element `Action` einer JSON-Richtlinie beschreibt die Aktionen, mit denen Sie den Zugriff in einer Richtlinie zulassen oder verweigern können. Richtlinienaktionen haben normalerweise denselben Namen wie die zugehörige AWS-API-Operation. Es gibt einige Ausnahmen, z. B. Aktionen, die nur mit Genehmigung durchgeführt werden können und für die es keine passende API-Operation gibt. Es gibt auch einige Operationen, die mehrere Aktionen in einer Richtlinie erfordern. Diese zusätzlichen Aktionen werden als abhängige Aktionen bezeichnet.

Schließen Sie Aktionen in eine Richtlinie ein, um Berechtigungen zur Durchführung der zugeordneten Operation zu erteilen.

Eine Liste der Amazon-SNS-Aktionen finden Sie unter Von Amazon Simple Notification Service definierte Ressourcen in der Service-Autorisierungs-Referenz.

Richtlinienaktionen in Amazon SNS verwenden das folgende Präfix vor der Aktion:

```
sns
```

Um mehrere Aktionen in einer einzigen Anweisung anzugeben, trennen Sie sie mit Kommata:

```
"Action": [
      "sns:action1",
      "sns:action2"
         ]
```

Beispiele für identitätsbasierte Amazon–SNS-Richtlinien finden Sie unter Beispiele für identitätsbasierte Richtlinien für Amazon Simple Notification Service.

Richtlinienressourcen für Amazon SNS

Unterstützt Richtlinienressourcen	Ja

Administratoren können mithilfe von AWS-JSON-Richtlinien festlegen, wer zum Zugriff auf was berechtigt ist. Das bedeutet die Festlegung, welcher Prinzipal Aktionen für welche Ressourcen unter welchen Bedingungen ausführen kann.

Das JSON-Richtlinienelement `Resource` gibt die Objekte an, auf welche die Aktion angewendet wird. Anweisungen müssen entweder ein – `Resource`oder ein `NotResource`-Element enthalten. Als bewährte Methode geben Sie eine Ressource mit dem zugehörigen Amazon-Ressourcennamen (ARN) an. Sie können dies für Aktionen tun, die einen bestimmten Ressourcentyp unterstützen, der als Berechtigungen auf Ressourcenebene bezeichnet wird.

Verwenden Sie für Aktionen, die keine Berechtigungen auf Ressourcenebene unterstützen, z. B. Auflistungsoperationen, einen Platzhalter (*), um anzugeben, dass die Anweisung für alle Ressourcen gilt.

```
"Resource": "*"
```

Eine Liste der Amazon-SNS-Ressourcentypen und ihre ARNs finden Sie unter Von Amazon Simple Notification Service definierte Aktionen in der Service-Autorisierungs-Referenz. Um zu erfahren, mit welchen Aktionen Sie den ARN jeder Ressource angeben können, lesen Sie von Von Amazon Simple Notification Service definierte Ressourcen.

Beispiele für identitätsbasierte Amazon–SNS-Richtlinien finden Sie unter Beispiele für identitätsbasierte Richtlinien für Amazon Simple Notification Service.

Richtlinien-Bedingungsschlüssel für Amazon SNS

Unterstützt servicespezifische Richtlinienbedingungsschlüssel	Ja

Administratoren können mithilfe von AWS-JSON-Richtlinien festlegen, wer zum Zugriff auf was berechtigt ist. Das heißt, welcher Prinzipal kann Aktionen für welche Ressourcen und unter welchen Bedingungen ausführen.

Das Element `Condition` (oder `Condition` block) ermöglicht Ihnen die Angabe der Bedingungen, unter denen eine Anweisung wirksam ist. Das Element `Condition` ist optional. Sie können bedingte Ausdrücke erstellen, die Bedingungsoperatoren verwenden, z. B. ist gleich oder kleiner als, damit die Bedingung in der Richtlinie mit Werten in der Anforderung übereinstimmt.

Wenn Sie mehrere `Condition`-Elemente in einer Anweisung oder mehrere Schlüssel in einem einzelnen `Condition`-Element angeben, wertet AWS diese mittels einer logischen `AND`-Operation aus. Wenn Sie mehrere Werte für einen einzelnen Bedingungsschlüssel angeben, wertet AWS die Bedingung mittels einer logischen `OR`-Operation aus. Alle Bedingungen müssen erfüllt werden, bevor die Berechtigungen der Anweisung gewährt werden.

Sie können auch Platzhaltervariablen verwenden, wenn Sie Bedingungen angeben. Beispielsweise können Sie einem IAM-Benutzer die Berechtigung für den Zugriff auf eine Ressource nur dann gewähren, wenn sie mit dessen IAM-Benutzernamen gekennzeichnet ist. Weitere Informationen finden Sie unter IAM-Richtlinienelemente: Variablen und Tags im IAM-Benutzerhandbuch.

AWS unterstützt globale Bedingungsschlüssel und servicespezifische Bedingungsschlüssel. Eine Liste aller globalen AWS-Bedingungsschlüssel finden Sie unter Globale AWS-Bedingungskontextschlüssel im IAM-Benutzerhandbuch.

Eine Liste der Amazon-SNS-Bedingungsschlüssel finden Sie unter Bedingungsschlüssel für Amazon Simple Notification Service in der Service-Authorization-Referenz. Informationen dazu, mit welchen Aktionen und Ressourcen Sie einen Bedingungsschlüssel verwenden können, finden Sie unter Von Amazon Simple Notification Service definierte Ressourcen.

Beispiele für identitätsbasierte Amazon–SNS-Richtlinien finden Sie unter Beispiele für identitätsbasierte Richtlinien für Amazon Simple Notification Service.

ACLs in Amazon SNS

Unterstützt ACLs	Nein

Zugriffssteuerungslisten (ACLs) steuern, welche Prinzipale (Kontomitglieder, Benutzer oder Rollen) auf eine Ressource zugreifen können. ACLs sind ähnlich wie ressourcenbasierte Richtlinien, verwenden jedoch nicht das JSON-Richtliniendokumentformat.

ABAC mit Amazon SNS

Unterstützt ABAC (Tags in Richtlinien)	Teilweise

Die attributbasierte Zugriffskontrolle (ABAC) ist eine Autorisierungsstrategie, bei der Berechtigungen basierend auf Attributen definiert werden. In AWS werden diese Attribute als Tags bezeichnet. Sie können Tags an IAM-Entitäten (Benutzer oder Rollen) und mehrere AWS-Ressourcen anfügen. Das Markieren von Entitäten und Ressourcen ist der erste Schritt von ABAC. Anschließend entwerfen Sie ABAC-Richtlinien, um Operationen zuzulassen, wenn das Tag des Prinzipals mit dem Tag der Ressource übereinstimmt, auf die sie zugreifen möchten.

ABAC ist in Umgebungen hilfreich, die schnell wachsen, und unterstützt Sie in Situationen, in denen die Richtlinienverwaltung mühsam wird.

Um den Zugriff auf der Grundlage von Tags zu steuern, geben Sie im Bedingungselement einer Richtlinie Tag-Informationen an, indem Sie die Schlüssel `aws:ResourceTag/`*`key-name`*, `aws:RequestTag/`*`key-name`*, oder Bedingung `aws:TagKeys` verwenden.

Wenn ein Service alle drei Bedingungsschlüssel für jeden Ressourcentyp unterstützt, lautet der Wert für den Service Ja. Wenn ein Service alle drei Bedingungsschlüssel für nur einige Ressourcentypen unterstützt, lautet der Wert Teilweise.

Weitere Informationen zu ABAC finden Sie unter Was ist ABAC? im IAM-Benutzerhandbuch. Um ein Tutorial mit Schritten zur Einstellung von ABAC anzuzeigen, siehe Attributbasierte Zugriffskontrolle (ABAC) verwenden im IAM-Benutzerhandbuch.

Verwenden temporärer Anmeldeinformationen mit Amazon SNS

Unterstützt temporäre Anmeldeinformationen	Ja

Einige AWS-Services Featureieren nicht, wenn Sie sich mit temporären Anmeldeinformationen anmelden. Weitere Informationen, unter anderem darüber, welche AWS-Services mit temporären Anmeldeinformationen arbeiten, finden Sie unter AWS-Services, die mit IAM arbeiten im IAM-Benutzerhandbuch.

Sie verwenden temporäre Anmeldeinformationen, wenn Sie sich mit einer anderen Methode als einem Benutzernamen und einem Passwort bei der AWS Management Console anmelden. Wenn Sie beispielsweise über den Single Sign-On (SSO)-Link Ihres Unternehmens auf AWS zugreifen, erstellt dieser Prozess automatisch temporäre Anmeldeinformationen. Sie erstellen auch automatisch temporäre Anmeldeinformationen, wenn Sie sich als Benutzer bei der Konsole anmelden und dann die Rollen wechseln. Weitere Informationen zum Wechseln von Rollen finden Sie unter Wechseln zu einer Rolle (Konsole) im IAM-Benutzerhandbuch.

Sie können mithilfe der AWS CLI- oder AWS-API manuell temporäre Anmeldeinformationen erstellen. Sie können dann diese temporären Anmeldeinformationen verwenden, um auf AWS zuzugreifen. AWS empfiehlt, dass Sie temporäre Anmeldeinformationen dynamisch generieren, anstatt langfristige Zugriffsschlüssel zu verwenden. Weitere Informationen finden Sie unter Temporäre Sicherheitsanmeldeinformationen in IAM.

Serviceübergreifende Prinzipalberechtigungen für Amazon SNS

Unterstützt Forward Access Sessions (FAS)	Ja

Wenn Sie einen IAM-Benutzer oder eine IAM-Rolle zum Ausführen von Aktionen in AWS verwenden, gelten Sie als Prinzipal. Bei einigen Services könnte es Aktionen geben, die dann eine andere Aktion in einem anderen Service auslösen. FAS verwendet die Berechtigungen des Prinzipals, der einen AWS-Service aufruft, in Kombination mit der Anforderung an den AWS-Service, Anforderungen an nachgelagerte Services zu stellen. FAS-Anfragen werden nur dann gestellt, wenn ein Service eine Anfrage erhält, die eine Interaktion mit anderen AWS-Services oder -Ressourcen erfordert. In diesem Fall müssen Sie über Berechtigungen zum Ausführen beider Aktionen verfügen. Einzelheiten zu den Richtlinien für FAS-Anfragen finden Sie unter Zugriffssitzungen weiterleiten.

Servicerollen für Amazon SNS

Unterstützt Servicerollen	Ja

Eine Servicerolle ist eine IAM-Rolle, die ein Service annimmt, um Aktionen in Ihrem Namen auszuführen. Ein IAM-Administrator kann eine Servicerolle innerhalb von IAM erstellen, ändern und löschen. Weitere Informationen finden Sie unter Erstellen einer Rolle zum Delegieren von Berechtigungen an einen AWS-Service im IAM-Benutzerhandbuch.

> **Warning**
>
> Das Ändern der Berechtigungen für eine Servicerolle könnte die Funktionalität von Amazon SNS beeinträchtigen. Bearbeiten Sie Servicerollen nur, wenn Amazon SNS dazu Anleitungen gibt.

Serviceverknüpfte Rollen für Amazon SNS

Unterstützt serviceverknüpfte Rollen	Nein

Eine serviceverknüpfte Rolle ist eine Art von Servicerolle, die mit einem AWS-Service verknüpft ist. Der Service kann die Rolle übernehmen, um eine Aktion in Ihrem Namen auszuführen. Serviceverknüpfte Rollen werden in Ihrem AWS-Konto angezeigt und gehören zum Service. Ein IAM-Administrator kann die Berechtigungen für Service-verknüpfte Rollen anzeigen, aber nicht bearbeiten.

Details zum Erstellen oder Verwalten von serviceverknüpften Rollen finden Sie unter AWS-Services, die mit IAM funktionieren. Suchen Sie in der Tabelle nach einem Service mit einem `Yes` in der Spalte Service-linked role (Serviceverknüpfte Rolle). Wählen Sie den Link Yes (Ja) aus, um die Dokumentation für die serviceverknüpfte Rolle für diesen Service anzuzeigen.

Beispiele für identitätsbasierte Richtlinien für Amazon Simple Notification Service

Benutzer besitzen standardmäßig keine Berechtigungen zum Erstellen oder Ändern von Amazon-SNS-Ressourcen. Sie können auch keine Aufgaben über die AWS Management Console, die AWS

Command Line Interface (AWS CLI) oder die AWS-API ausführen. Ein IAM-Administrator muss IAM-Richtlinien erstellen, die Benutzern die Berechtigung erteilen, Aktionen für die Ressourcen auszuführen, die sie benötigen. Der Administrator kann dann die IAM-Richtlinien zu Rollen hinzufügen, und Benutzer können die Rollen annehmen.

Informationen dazu, wie Sie unter Verwendung dieser beispielhaften JSON-Richtliniendokumente eine identitätsbasierte IAM-Richtlinie erstellen, finden Sie unter Erstellen von IAM-Richtlinien im IAM-Benutzerhandbuch.

Einzelheiten zu Aktionen und Ressourcentypen, die von Amazon SNS definiert werden, einschließlich des Formats der ARNs für die einzelnen Ressourcentypen, finden Sie unter Aktionen, Ressourcen und Bedingungsschlüssel für Amazon Simple Notification Service in der Service-Authorisierungs-Referenz.

Themen

Bewährte Methoden für Richtlinien

Identitätsbasierte Richtlinien können festlegen, ob jemand Amazon-SNS-Ressourcen in Ihrem Konto erstellen, darauf zugreifen oder daraus löschen kann. Dies kann zusätzliche Kosten für Ihr verursachen AWS-Konto. Befolgen Sie beim Erstellen oder Bearbeiten identitätsbasierter Richtlinien die folgenden Anleitungen und Empfehlungen:

- Erste Schritte mit AWS-verwaltete Richtlinien und Umstellung auf Berechtigungen mit den geringsten Berechtigungen:Um Ihren Benutzern und Workloads Berechtigungen zu gewähren, verwenden Sie die AWS-verwaltete Richtlinien die Berechtigungen für viele allgemeine Anwendungsfälle gewähren. Sie sind in Ihrem AWS-Konto verfügbar. Wir empfehlen Ihnen, die Berechtigungen weiter zu reduzieren, indem Sie vom Kunden verwaltete AWS-Richtlinien definieren, die speziell auf Ihre Anwendungsfälle zugeschnitten sind. Weitere Informationen finden Sie unter AWS-verwaltete Richtlinien oder AWS-verwaltete Richtlinien für Auftragsfunktionen im IAM-Benutzerhandbuch.

- Anwendung von Berechtigungen mit den geringsten Rechten:Wenn Sie mit IAM-Richtlinien Berechtigungen festlegen, gewähren Sie nur die Berechtigungen, die für die Durchführung einer Aufgabe erforderlich sind. Sie tun dies, indem Sie die Aktionen definieren, die für bestimmte Ressourcen unter bestimmten Bedingungen durchgeführt werden können, auch bekannt als die geringsten Berechtigungen. Weitere Informationen zur Verwendung von IAM zum Anwenden von Berechtigungen finden Sie unter Richtlinien und Berechtigungen in IAM im IAM-Benutzerhandbuch.
- Verwenden von Bedingungen in IAM-Richtlinien zur weiteren Einschränkung des Zugriffs:Sie können Ihren Richtlinien eine Bedingung hinzufügen, um den Zugriff auf Aktionen und Ressourcen zu beschränken. Sie können beispielsweise eine Richtlinienbedingung schreiben, um festzulegen, dass alle Anforderungen mithilfe von SSL gesendet werden müssen. Sie können auch Bedingungen verwenden, um Zugriff auf Service-Aktionen zu gewähren, wenn diese durch ein bestimmtes AWS-Service, wie beispielsweise AWS CloudFormation, verwendet werden. Weitere Informationen finden Sie unter IAM-JSON-Richtlinienelemente: Bedingung im IAM-Benutzerhandbuch.
- Verwenden von IAM Access Analyzer zur Validierung Ihrer IAM-Richtlinien, um sichere und funktionale Berechtigungen zu gewährleisten:IAM Access Analyzer validiert neue und vorhandene Richtlinien, damit die Richtlinien der IAM-Richtliniensprache (JSON) und den bewährten IAM-Methoden entsprechen. IAM Access Analyzer stellt mehr als 100 Richtlinienprüfungen und umsetzbare Empfehlungen zur Verfügung, damit Sie sichere und funktionale Richtlinien erstellen können. Weitere Informationen finden Sie unter Richtlinienvalidierung zum IAM Access Analyzer im IAM-Benutzerhandbuch.
- Bedarf einer Multi-Faktor-Authentifizierung (MFA):Wenn Sie ein Szenario haben, das IAM-Benutzer oder Root-Benutzer in Ihrem AWS-Konto erfordert, aktivieren Sie MFA für zusätzliche Sicherheit. Um MFA beim Aufrufen von API-Vorgängen anzufordern, fügen Sie Ihren Richtlinien MFA-Bedingungen hinzu. Weitere Informationen finden Sie unter Konfigurieren eines MFA-geschützten API-Zugriffs im IAM-Benutzerhandbuch.

Weitere Informationen zu bewährten Methoden in IAM finden Sie unter Bewährte Methoden für die Sicherheit in IAM im IAM-Benutzerhandbuch.

Verwenden der Amazon-SNS-Konsole

Um auf die Konsole von Amazon Simple Notification Service zugreifen zu können, müssen Sie über einen Mindestsatz von Berechtigungen verfügen. Diese Berechtigungen müssen es Ihnen ermöglichen, Details über die Amazon-SNS-Ressourcen in Ihrem AWS-Konto aufzulisten und

anzuzeigen. Wenn Sie eine identitätsbasierte Richtlinie erstellen, die strenger ist als die mindestens erforderlichen Berechtigungen, funktioniert die Konsole nicht wie vorgesehen für Entitäten (Benutzer oder Rollen) mit dieser Richtlinie.

Für Benutzer, die nur Aufrufe an die AWS CLI oder AWS-API durchführen, müssen Sie keine Mindestberechtigungen in der Konsole erteilen. Stattdessen sollten Sie nur Zugriff auf die Aktionen zulassen, die der API-Operation entsprechen, die die Benutzer ausführen möchten.

Um sicherzustellen, dass Benutzer und Rollen weiterhin die Amazon-SNS-Konsole verwenden können, fügen Sie den Entitäten auch die von Amazon SNS `ConsoleAccess` oder `ReadOnly` AWS verwaltete Richtlinie hinzu. Weitere Informationen finden Sie unter Hinzufügen von Berechtigungen zu einem Benutzer im IAM-Benutzerhandbuch.

Weitere Richtlinientypen

AWS unterstützt zusätzliche, weniger häufig verwendete Richtlinientypen. Diese Richtlinientypen können die maximalen Berechtigungen festlegen, die Ihnen von den häufiger verwendeten Richtlinientypen erteilt werden können.

- Berechtigungsgrenzen:Eine Berechtigungsgrenze ist ein erweitertes Feature, mit der Sie die maximalen Berechtigungen festlegen können, die eine identitätsbasierte Richtlinie einer IAM-Entität (IAM-Benutzer oder -Rolle) erteilen kann. Sie können eine Berechtigungsgrenze für eine Entität festlegen. Die daraus resultierenden Berechtigungen sind der Schnittpunkt der identitätsbasierten Richtlinien einer Entität und ihrer Berechtigungsgrenzen. Ressourcenbasierte Richtlinien, die den Benutzer oder die Rolle im Feld `Principal` angeben, werden nicht durch Berechtigungsgrenzen eingeschränkt. Eine explizite Zugriffsverweigerung in einer dieser Richtlinien setzt eine Zugriffserlaubnis außer Kraft. Weitere Informationen über Berechtigungsgrenzen finden Sie unter Berechtigungsgrenzen für IAM-Entitäten im IAM-Benutzerhandbuch.
- Service-Kontrollrichtlinien (SCPs) – SCPs sind JSON-Richtlinien, die die maximalen Berechtigungen für eine Organisation oder Organisationseinheit (OE) in AWS Organizations angeben. AWS Organizations ist ein Dienst für die Gruppierung und zentrale Verwaltung mehrerer AWS-Konten Ihres Unternehmens. Wenn Sie innerhalb einer Organisation alle Funktionen aktivieren, können Sie Service-Kontrollrichtlinien (SCPs) auf alle oder einzelne Ihrer Konten anwenden. Eine SCP beschränkt die Berechtigungen für Entitäten in Mitgliedskonten, einschließlich aller AWS-Konto-Stammbenutzer. Weitere Informationen zu Organizations und SCPs finden Sie unter Funktionsweise von SCPs im AWS Organizations-Benutzerhandbuch.
- Sitzungsrichtlinien:Sitzungsrichtlinien sind erweiterte Richtlinien, die Sie als Parameter übergeben, wenn Sie eine temporäre Sitzung für eine Rolle oder einen verbundenen

Benutzer programmgesteuert erstellen. Die resultierenden Sitzungsberechtigungen sind eine Schnittmenge der auf der Identität des Benutzers oder der Rolle basierenden Richtlinien und der Sitzungsrichtlinien. Berechtigungen können auch aus einer ressourcenbasierten Richtlinie stammen. Eine explizite Zugriffsverweigerung in einer dieser Richtlinien setzt eine Zugriffserlaubnis außer Kraft. Weitere Informationen finden Sie unter Sitzungsrichtlinien im IAM-Benutzerhandbuch.

Mehrere Richtlinientypen

Wenn mehrere auf eine Anforderung mehrere Richtlinientypen angewendet werden können, sind die entsprechenden Berechtigungen komplizierter. Informationen dazu, wie AWS die Zulässigkeit einer Anforderung ermittelt, wenn mehrere Richtlinientypen beteiligt sind, finden Sie unter Logik für die Richtlinienauswertung im IAM-Benutzerhandbuch.

Gewähren der Berechtigung zur Anzeige der eigenen Berechtigungen für Benutzer

In diesem Beispiel wird gezeigt, wie Sie eine Richtlinie erstellen, die IAM-Benutzern die Berechtigung zum Anzeigen der eingebundenen Richtlinien und verwalteten Richtlinien gewährt, die ihrer Benutzeridentität angefügt sind. Diese Richtlinie enthält Berechtigungen für die Ausführung dieser Aktion auf der Konsole oder für die programmgesteuerte Ausführung über die AWS CLI oder die AWS-API.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
                "iam:GetUser"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
        {
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
```

```
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
            "Resource": "*"
        }
    ]
}
```

Identitätsbasierte Richtlinien für Amazon SNS

Unterstützt Richtlinien auf Identitätsbasis.	Ja

Identitätsbasierte Richtlinien sind JSON-Berechtigungsrichtliniendokumente, die Sie einer Identität anfügen können, wie z. B. IAM-Benutzern, -Benutzergruppen oder -Rollen. Diese Richtlinien steuern, welche Aktionen die Benutzer und Rollen für welche Ressourcen und unter welchen Bedingungen ausführen können. Informationen zum Erstellen identitätsbasierter Richtlinien finden Sie unter Erstellen von IAM-Richtlinien im IAM-Benutzerhandbuch.

Mit identitätsbasierten IAM-Richtlinien können Sie angeben, welche Aktionen und Ressourcen zugelassen oder abgelehnt werden. Darüber hinaus können Sie die Bedingungen festlegen, unter denen Aktionen zugelassen oder abgelehnt werden. Sie können den Prinzipal nicht in einer identitätsbasierten Richtlinie angeben, da er für den Benutzer oder die Rolle gilt, dem er zugeordnet ist. Informationen zu sämtlichen Elementen, die Sie in einer JSON-Richtlinie verwenden, finden Sie in der IAM-Referenz für JSON-Richtlinienelemente im IAM-Benutzerhandbuch.

Beispiele für identitätsbasierte Richtlinien für Amazon SNS

Beispiele für identitätsbasierte Amazon–SNS-Richtlinien finden Sie unter Beispiele für identitätsbasierte Richtlinien für Amazon Simple Notification Service.

Ressourcenbasierte Richtlinien in Amazon ECS

Unterstützt ressourcenbasierte Richtlinien	Ja

Ressourcenbasierte Richtlinien sind JSON-Richtliniendokumente, die Sie an eine Ressource anfügen. Beispiele für ressourcenbasierte Richtlinien sind IAM-Rollen-Vertrauensrichtlinien und Amazon-S3-Bucket-Richtlinien. In Services, die ressourcenbasierte Richtlinien unterstützen, können Service-Administratoren sie verwenden, um den Zugriff auf eine bestimmte Ressource zu steuern. Für die Ressource, an welche die Richtlinie angehängt ist, legt die Richtlinie fest, welche Aktionen ein bestimmter Prinzipal unter welchen Bedingungen für diese Ressource ausführen kann. Sie müssen in einer ressourcenbasierten Richtlinie einen Prinzipal angeben. Prinzipale können Konten, Benutzer, Rollen, Verbundbenutzer oder AWS-Services umfassen.

Um kontoübergreifenden Zugriff zu ermöglichen, können Sie ein gesamtes Konto oder IAM-Entitäten in einem anderen Konto als Prinzipal in einer ressourcenbasierten Richtlinie angeben. Durch das Hinzufügen eines kontoübergreifenden Auftraggebers zu einer ressourcenbasierten Richtlinie ist nur die halbe Vertrauensbeziehung eingerichtet. Wenn sich der Prinzipal und die Ressource in unterschiedlichen AWS-Konten befinden, muss ein IAM-Administrator im vertrauenswürdigen Konto auch der Prinzipalentität (Benutzer oder Rolle) die Berechtigung zum Zugriff auf die Ressource erteilen. Sie erteilen Berechtigungen, indem Sie der juristischen Stelle eine identitätsbasierte Richtlinie anfügen. Wenn jedoch eine ressourcenbasierte Richtlinie Zugriff auf einen Prinzipal in demselben Konto gewährt, ist keine zusätzliche identitätsbasierte Richtlinie erforderlich. Weitere Informationen finden Sie unter Wie sich IAM-Rollen von ressourcenbasierten Richtlinien unterscheiden im IAM-Benutzerhandbuch.

Verwenden von identitätsbasierten Richtlinien mit Amazon SNS

Themen

- Gemeinsame Nutzung von IAM- und Amazon SNS-Richtlinien
- Amazon SNS Ressourcen - ARN Format
- Amazon SNS API-Aktionen
- Amazon SNS Richtlinienschlüssel
- Beispielrichtlinien für Amazon SNS

Amazon Simple Notification Service ist integriert inAWS Identity and Access Management (IAM), sodass Sie angeben können, welche Amazon SNS Aktionen ein Benutzer in Ihrem AWS-Konto kann mit Amazon SNS-Ressourcen arbeiten. Sie können ein bestimmtes Thema in der Richtlinie festlegen. Sie können z. B. beim Erstellen einer IAM-Richtlinie Variablen verwenden, um bestimmten Benutzern in Ihrer Organisation die Verwendung der `Publish`-Aktion für bestimmte Themen in Ihrem AWS-

Konto zu ermöglichen. Weitere Informationen finden Sie unter Richtlinienvariablen im Handbuch Verwenden von IAM.

> **Important**
>
> Durch die Nutzung von IAM mit Amazon SNS ändert sich die Verwendung von Amazon SNS nicht. Es gibt keine geänderten Amazon SNS-Aktionen und keine neuen -Aktionen im Zusammenhang mit Benutzern und Zugriffskontrolle.

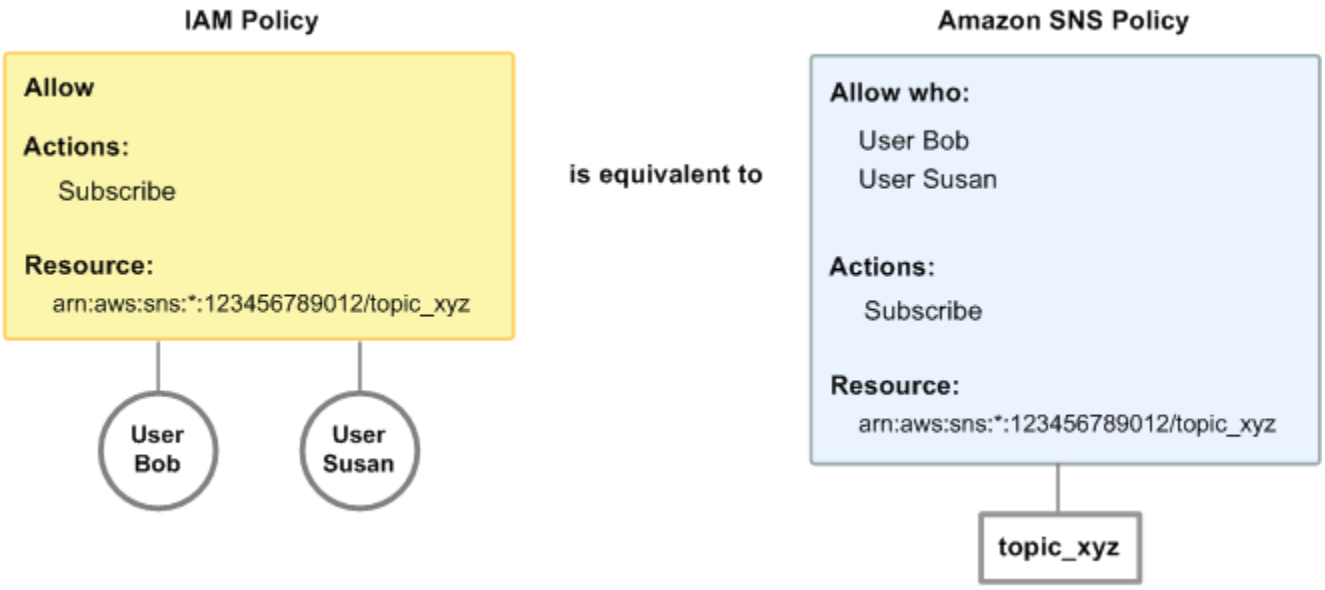
Beispiele von Richtlinien, die Amazon SNS Aktionen und Ressourcen abdecken, finden Sie unter Beispielrichtlinien für Amazon SNS.

Gemeinsame Nutzung von IAM- und Amazon SNS-Richtlinien

Sie verwenden eine IAM-Richtlinie, um den Zugriff des Benutzers auf Amazon SNS-Aktionen und -Themen einzuschränken. Eine IAM-Richtlinie kann den Zugriff nur für Benutzer in Ihrem AWS und nicht für andere AWS-Konten regeln.

Sie können eine Amazon SNS-Richtlinie mit einem bestimmten Thema verwenden, um einzuschränken, wer mit diesem Thema arbeiten kann (z. B., wer Nachrichten veröffentlichen kann, wer es abonnieren kann usw.). Amazon SNS-Richtlinien können Zugriff auf andere AWS-Konten oder an Benutzer innerhalb Ihrer eigenen AWS-Konto aus.

Um Ihren Benutzern Berechtigungen für Ihre Amazon SNS-Themen zu gewähren, können Sie IAM-Richtlinien, Amazon SNS-Richtlinien oder beide verwenden. In den meisten Fällen erzielen Sie mit beiden Systemen dieselben Ergebnisse. Die folgende Abbildung zeigt eine IAM-Richtlinie und eine entsprechende Amazon SNS-Richtlinie. Die IAM-Richtlinie lässt die Amazon-SNS-Aktion `Subscribe` für das Thema namens "topic_xyz" in Ihrem AWS zu. Die IAM-Richtlinie ist den Benutzern Bob und Susan zugeordnet (was bedeutet, dass Bob und Susan die Berechtigungen aus der Richtlinie haben). Die Amazon SNS-Richtlinie bietet Bob und Susan ebenfalls die Berechtigung für den Zugriff auf `Subscribe` für "topic_xyz".



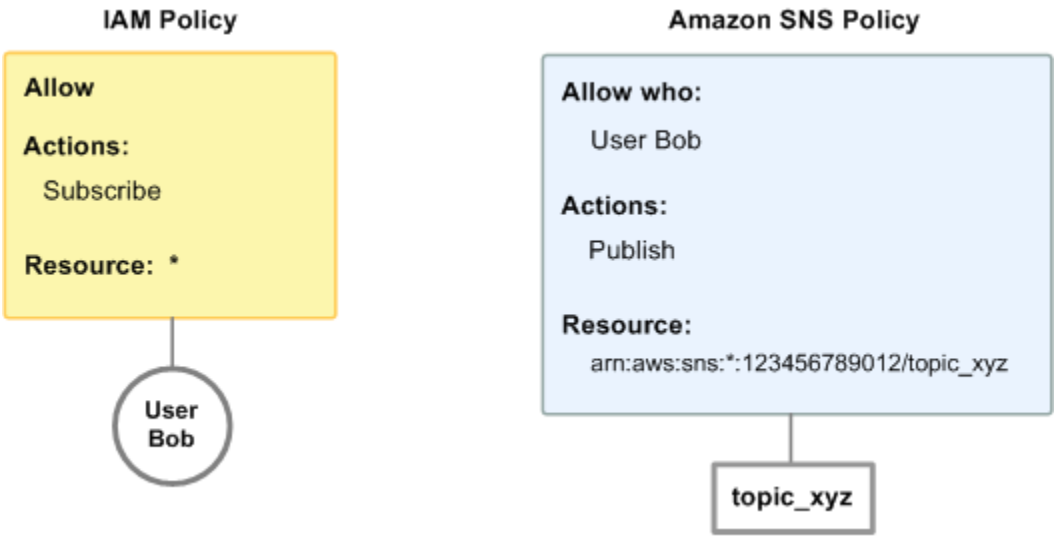
> **Note**
>
> Das vorherige Beispiel veranschaulicht einfache Richtlinien ohne Bedingungen. Sie können eine bestimmte Bedingung in einer der Richtlinien angeben und erhalten dasselbe Ergebnis.

Es gibt einen großen Unterschied zwischen AWS-IAM- und Amazon SNS-Richtlinien: Im Gegensatz zum IAM-Richtliniensystem können Sie mit dem Amazon SNS-Richtliniensystem Berechtigungen für andere AWS-Konten erteilen.

Sie entscheiden je nach Ihren Anforderungen, wie Sie beide Systeme zum Verwalten von Berechtigungen verwenden. Die folgenden Beispiele zeigen, wie die beiden Richtliniensysteme zusammenarbeiten.

Example 1

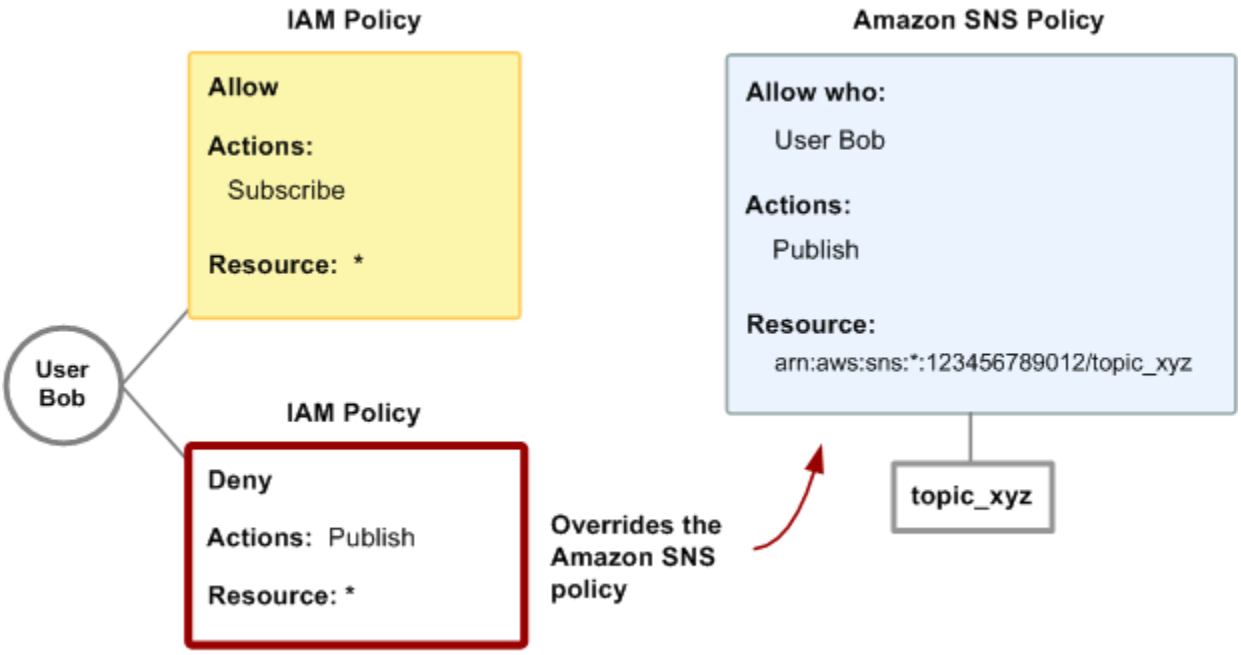
In diesem Beispiel wird eine IAM-Richtlinie und eine Amazon SNS-Richtlinie auf Bob angewendet. Die IAM-Richtlinie gewährt ihm Berechtigungen für `Subscribe` zu jedem Thema im AWS-Konto. Die Amazon SNS-Richtlinie gewährt ihm Berechtigungen für `Publish` zu einem bestimmten Thema („topic_xyz"). Das folgende Diagramm verdeutlicht das Konzept.



Wenn Bob eine Anforderung zum Abonnieren an ein einzelnes Thema im AWS-Konto sendet, lässt die IAM-Richtlinie diese Aktion zu. Wenn Bob eine Anforderung zum Veröffentlichen einer Nachricht an „topic_xyz“ sendet, lässt die Amazon-SNS-Richtlinie diese Aktion zu.

Example 2

In diesem Beispiel bauen wir auf Beispiel 1 auf (wobei Bob zwei für ihn geltende Richtlinien hat). Angenommen, Bob veröffentlicht Nachrichten für das Thema „topic_xyz“, und sollte daran gehindert werden. In diesem Fall entfernen Sie seine Möglichkeit zur Veröffentlichung für diese Themen vollständig. Der einfachste Weg ist, eine IAM-Richtlinie hinzuzufügen, die ihm den Zugriff auf die `Publish`-Aktionen für alle Themen verweigert. Diese dritte Richtlinie überschreibt die Amazon SNS-Richtlinie, die ihm ursprünglich die Berechtigung zur Veröffentlichung in das Thema „topic_xyz“ gewährt hat. Eine explizite Verweigerung überschreibt grundsätzlich eine gewährte Berechtigung (weitere Informationen über die Logik der Richtlinienauswertung finden Sie unter Auswertungslogik). Das folgende Diagramm verdeutlicht das Konzept.



Beispiele von Richtlinien, die Amazon SNS-Aktionen und -Ressourcen abdecken, finden Sie unter Beispielrichtlinien für Amazon SNS. Weitere Informationen zum Erstellen von Amazon SNS - Richtlinien finden Sie in der technischen Dokumentation zu Amazon SNS .

Amazon SNS Ressourcen - ARN Format

Bei Amazon SNS, können Sie in einer Richtlinie nur den Ressourcentyp „Thema“ festlegen. Im Folgenden finden Sie das Amazon-Ressourcenname (ARN)-Format für Themen:

```
arn:aws:sns:region:account_ID:topic_name
```

Weitere Informationen zu ARNs finden Sie unter ARNs im IAM-Benutzerhandbuch.

Example

Im Folgenden finden Sie einen ARN für ein Thema mit dem Namen MyTopic in der Region us-east-2, der zu AWS-Konto 123456789012 gehört.

```
arn:aws:sns:us-east-2:123456789012:MyTopic
```

Example

Wenn Sie MyTopic in jeder der verschiedenen Regionen, die Amazon SNS unterstützt, ein Thema mit dem Namen hätten, könnten Sie die Themen mit dem folgenden ARN angeben.

```
arn:aws:sns:*:123456789012:MyTopic
```

Sie können im Namen des Themas die Platzhalterzeichen * und ? verwenden. Der folgende Namen könnte z. B. alle Themen bezeichnen, die von Bob erstellt wurden und den Präfix bob_. haben.

```
arn:aws:sns:*:123456789012:bob_*
```

Zur Vereinfachung gibt Amazon SNS beim Erstellen eines Themas den ARN in der Antwort zurück.

Amazon SNS API-Aktionen

In einer IAM-Richtlinie können Sie eine beliebige, von Amazon SNS unterstützte Aktion angeben. Die Aktionen `ConfirmSubscription` und `Unsubscribe` erfordern jedoch keine Authentifizierung. Dies bedeutet, dass &IAM; die Benutzer auch dann nicht beim Zugriff beschränkt, wenn Sie diese Aktionen in einer Richtlinie festlegen.

Jeder in einer Richtlinie angegebenen Aktion muss die Zeichenfolge `sns:` in Kleinbuchstaben vorangestellt werden. Zum Angeben von allen Amazon SNS -Aktionen würden Sie beispielsweise `sns:*` verwenden. Eine Liste der Aktionen finden Sie unterAmazon Simple Notification Service-API-Referenz.

Amazon SNS Richtlinienschlüssel

Amazon SNS implementiert die folgenden AWS-weiten Richtlinienschlüssel sowie einige servicespezifische Schlüssel.

Eine Liste der Bedingungsschlüssel, die von jedem AWS-Service unterstützt werden, finden Sie unter Aktionen, Ressourcen und Bedingungsschlüssel für AWS-Services im IAM-Benutzerhandbuch. Eine Liste der Bedingungsschlüssel, die in mehreren AWS-Services verwendet werden können, finden Sie unter Globale AWS-Bedingungskontextschlüssel im IAM-Benutzerhandbuch.

Amazon SNS verwendet die folgenden servicespezifischen Schlüssel. Verwenden Sie diese Schlüssel in Richtlinien, die den Zugriff auf `Subscribe`-Anfragen beschränken.

- sns:Endpoint: Die URL, die E-Mail-Adresse oder der ARN von einer `Subscribe`-Anfrage oder eines zuvor bestätigten Abonnements. Verwendung mit Bedingungen im Zusammenhang mit

Zeichenfolgen (siehe Beispielrichtlinien für Amazon SNS), um den Zugriff auf bestimmte Endpunkte (z. B. *@yourcompany.com) zu beschränken.

- sns:Protocol: Der `protocol`-Wert einer `Subscribe`-Anfrage oder eines zuvor bestätigten Abonnements. Verwendung mit Bedingungen im Zusammenhang mit Zeichenfolgen (siehe Beispielrichtlinien für Amazon SNS), um die Veröffentlichung auf bestimmte Bereitstellungsprotokolle (z. B. HTTPS) zu beschränken.

Beispielrichtlinien für Amazon SNS

Dieser Abschnitt zeigt mehrere einfache Richtlinien zum Steuern des Benutzerzugriffs auf Amazon SNS .

> **Note**
>
> In Zukunft können in Amazon SNS neue Aktionen hinzugefügt werden, die logisch in einer der folgenden Richtlinien eingeschlossen werden müssen, basierend auf den formulierten Zielen der Richtlinie.

Example 1: Zulassen der Erstellung und Verwaltung von Themen für eine Gruppe

In diesem Beispiel erstellen wir eine Richtlinie, die Zugriff auf `CreateTopic`, `ListTopics`, `SetTopicAttributes` und `DeleteTopic` gewährt.

```
{
  "Statement": [{
    "Effect": "Allow",
    "Action": ["sns:CreateTopic", "sns:ListTopics", "sns:SetTopicAttributes",
 "sns:DeleteTopic"],
    "Resource": "*"
  }]
}
```

Example 2: Zulassen der Veröffentlichung von Nachrichten zu einem bestimmten Thema für die IT-Gruppe

In diesem Beispiel erstellen wir eine Gruppe für die IT, die eine Richtlinie zuweist, die Zugriff auf `Publish` für das definierte Thema gewährt.

```
{
  "Statement": [{
    "Effect": "Allow",
    "Action": "sns:Publish",
    "Resource": "arn:aws:sns:*:123456789012:MyTopic"
  }]
}
```

Example 3: Benutzern im AWS-Konto die Möglichkeit zum Abonnieren von Themen gewähren

In diesem Beispiel erstellen wir eine Richtlinie mit der Bedingungen für die Zeichenfolgenübereinstimmungen für die Richtlinienschlüssel `sns:Protocol` und `sns:Endpoint`, die den Zugriff auf die `Subscribe`-Aktion gewährt.

```
{
  "Statement": [{
    "Effect": "Allow",
    "Action": ["sns:Subscribe"],
    "Resource": "*",
    "Condition": {
      "StringLike": {
        "SNS:Endpoint": "*@example.com"
      },
      "StringEquals": {
        "sns:Protocol": "email"
      }
    }
  }]
}
```

Example 4: Zulassen der Veröffentlichung von Nachrichten zu einem bestimmten Thema für einen Partner

Sie können eine Amazon SNS-Richtlinie oder eine IAM-Richtlinie verwenden, um einem Partner das Veröffentlichen zu einem bestimmten Thema zu gewähren. Wenn Ihr Partner über ein AWS-Konto verfügt, empfiehlt sich die Verwendung einer Amazon SNS-Richtlinie. Es könnte jedoch jeder Benutzer im Unternehmen des Partners, der über die AWS-Anmeldeinformationen verfügt, Nachrichten für das Thema veröffentlichen. In diesem Beispiel wird davon ausgegangen, dass Sie den Zugriff auf eine bestimmte Person (oder Anwendung) begrenzen möchten. Zu diesem Zweck müssen Sie den Partner wie einen Benutzer innerhalb Ihres eigenen Unternehmens behandeln. Verwenden Sie eine &IAM;-Richtlinie anstelle einer Amazon SNS-Richtlinie.

In diesem Beispiel erstellen wir eine Gruppe namens , WidgetCo die das Partnerunternehmen repräsentiert. Wir erstellen einen Benutzer für die spezifische Person (oder Anwendung) des Partnerunternehmens, die Zugriff benötigt. Anschließend fügen wir den Benutzer in die Gruppe ein.

Anschließend fügen wir eine Richtlinie an, die der Gruppe `Publish` Zugriff auf das spezifische Thema mit dem Namen gewährtWidgetPartnerTopic.

Wir möchten auch verhindern, dass die WidgetCo Gruppe etwas anderes mit Themen tut. Daher fügen wir eine Anweisung hinzu, die die Berechtigung für alle anderen Amazon SNS-Aktionen als `Publish` für andere Themen als verweigert WidgetPartnerTopic. Dies ist nur erforderlich, wenn eine umfassendere Richtlinie im System verwendet wird, die Benutzern einen breiten Zugriff auf Amazon SNS-Ressourcen ermöglicht.

```
{
  "Statement": [{
      "Effect": "Allow",
      "Action": "sns:Publish",
      "Resource": "arn:aws:sns:*:123456789012:WidgetPartnerTopic"
    },
    {
      "Effect": "Deny",
      "NotAction": "sns:Publish",
      "NotResource": "arn:aws:sns:*:123456789012:WidgetPartnerTopic"
    }
  ]
}
```

Verwenden von temporären Sicherheitsanmeldeinformationen mit Amazon SNS

IAM ermöglicht Ihnen nicht nur das Erstellen von -Benutzern mit eigenen Sicherheitsanmeldeinformationen, sondern auch die Gewährung temporärer Sicherheitsanmeldeinformationen für jeden beliebigen Benutzer, um auf Ihre AWS-Services und -Ressourcen zuzugreifen. Sie können Benutzer mit AWS-Konten verwalten (IAM-Benutzer). Sie können auch Benutzer für Ihr System verwalten, die nicht über AWS-Konten verfügen (verbundene Benutzer). Zusätzlich kann es sich bei „Benutzern“ um Anwendungen handeln, die Sie erstellen, um auf Ihre AWS-Ressourcen zuzugreifen.

Sie können diese temporären Sicherheitsanmeldeinformationen beim Senden von Anfragen an Amazon SNS verwenden. Die API-Bibliotheken berechnen anhand dieser Anmeldeinformationen

den notwendigen Signaturwert, um Ihre Anforderung zu authentifizieren. Wenn Sie beim Senden von Anfragen abgelaufene Anmeldeinformationen verwenden, lehnt Amazon SNS die Anfrage ab.

Weitere Informationen zum IAM-Support von temporären Sicherheitsanmeldeinformationen finden Sie unter Granting Temporary Access to Your AWS Resources unter Verwenden von IAM.

Example Verwendung temporärer Sicherheitsanmeldeinformationen zur Authentifizierung einer Amazon SNS-Anforderung

Das folgende Beispiel zeigt, wie Sie temporäre Sicherheitsanmeldeinformationen zum Authentifizieren einer Amazon SNS-Anforderung abrufen.

```
http://sns.us-east-2.amazonaws.com/
?Name=My-Topic
&Action=CreateTopic
&Signature=gfzIF53exFVdpSNb8AiwN3Lv%2FNYXh6S%2Br3yySK70oX4%3D
&SignatureVersion=2
&SignatureMethod=HmacSHA256
&Timestamp=2010-03-31T12%3A00%3A00.000Z
&SecurityToken=SecurityTokenValue
&AWSAccessKeyId=Access Key ID provided by AWS Security Token Service
```

Amazon SNS-API-Berechtigungen: Referenztabelle für Aktionen und Ressourcen

Die folgende Liste enthält Informationen zur Amazon SNS-Implementierung der Zugriffskontrolle:

- Jede Richtlinie darf nur ein einzelnes Thema betreffen (bei der Erstellung einer Richtlinie dürfen Sie keine Anweisungen zu anderen Themen schreiben).
- Jede Richtlinie muss eine eindeutige `Id` haben.
- Jede Anweisung in einer Richtlinie muss eine eindeutige Anweisungs-`sid` haben.

Richtlinienkontingente

In der folgenden Tabelle sind die Kontingente für eine Richtlinienanweisung aufgeführt.

Name	Höchstkontingent
Bytes	30 KB

Name	Höchstkontingent
Anweisungen	100
Prinzipale	1 bis 200 (0 ist ungültig)
Ressource	1 (0 ist ungültig. Der Wert muss mit dem ARN des Themas der Richtlinie übereinstimmen.)

Gültige Amazon SNS-Richtlinienaktionen

Amazon SNS unterstützt die in der folgenden Tabelle aufgeführten Aktionen.

Aktion	Beschreibung
sns:AddPermission	Erteilt die Berechtigung zum Hinzufügen von Berechtigungen zur Themarichtlinie.
sns:DeleteTopic	Erteilt die Berechtigung zum Löschen eines Themas.
sns:GetDataProtectionPolicy	Erteilt die Berechtigung zum Abrufen der Datenschutzrichtlinie eines Themas.
sns:GetTopicAttributes	Erteilt die Berechtigung zum Empfangen aller Themenattribute.
sns:ListSubscriptionsByTopic	Erteilt die Berechtigung zum Abrufen aller Abonnements für ein bestimmtes Thema.
sns:ListTagsForResource	Gewährt die Berechtigung zum Auflisten aller Tags, die einem bestimmten Thema hinzugefügt wurden.
sns:Publish	Erteilt die Berechtigung zum Batch-Veröffentlichen in einem Thema oder Endpunkt. Weitere Informationen finden Sie unter Veröffentlichen und PublishBatch in der API-Referenz zum Amazon Simple Notification Service.
sns:PutDataProtectionPolicy	Erteilt die Berechtigung zum Festlegen der Datenschutzrichtlinie eines Themas.

Aktion	Beschreibung
sns:RemovePermission	Erteilt die Berechtigung zum Entfernen von Berechtigungen in der Themarichtlinie.
sns:SetTopicAttributes	Erteilt die Berechtigung zum Festlegen der Attribute eines Themas.
sns:Subscribe	Erteilt die Berechtigung zum Abonnieren eines Themas.

Servicespezifische Schlüssel

Amazon SNS verwendet die folgenden servicespezifischen Schlüssel. Sie können diese in Richtlinien verwenden, die den Zugriff auf `Subscribe`-Anfragen beschränken.

- sns:Endpoint: Die URL, die E-Mail-Adresse oder der ARN von einer `Subscribe`-Anfrage oder eines zuvor bestätigten Abonnements. Verwendung mit Bedingungen im Zusammenhang mit Zeichenfolgen (siehe Beispielrichtlinien für Amazon SNS), um den Zugriff auf bestimmte Endpunkte (z. B. *@beispiel.de) zu beschränken.
- sns:Protocol: Der `protocol`-Wert einer `Subscribe`-Anfrage oder eines zuvor bestätigten Abonnements. Verwendung mit Bedingungen im Zusammenhang mit Zeichenfolgen (siehe Beispielrichtlinien für Amazon SNS), um die Veröffentlichung auf bestimmte Bereitstellungsprotokolle (z. B. HTTPS) zu beschränken.

> **Important**
>
> Wenn Sie eine Richtlinie verwenden, um den Zugriff nach sns:Endpoint zu beschränken, beachten Sie, dass künftig DNS-Probleme die Namensauflösung des Endpunkts beeinträchtigen können.

Beheben von Identitäts- und Zugriffsfehlern bei Amazon Simple Notification Service

Diagnostizieren und beheben Sie mithilfe der folgenden Informationen gängige Probleme, die bei der Verwendung von Amazon SNS und IAM auftreten können.

Themen

- Ich bin nicht autorisiert, eine Aktion in Amazon SNS auszuführen
- Ich bin nicht autorisiert, iam durchzuführen:PassRole
- Ich möchte Personen außerhalb meines AWS-Konto Zugriff auf meine Amazon-SNS-Ressourcen gewähren

Ich bin nicht autorisiert, eine Aktion in Amazon SNS auszuführen

Wenn Sie die Fehlermeldung erhalten, dass Sie nicht zum Durchführen einer Aktion autorisiert sind, müssen Ihre Richtlinien aktualisiert werden, um die Aktion durchführen zu können.

Der folgende Beispielfehler tritt auf, wenn der `mateojackson`-Benutzer versucht, die Konsole zum Anzeigen von Details zu einer fiktiven *`my-example-widget`*-Ressource zu verwenden, jedoch nicht über `sns:`*`GetWidget`*-Berechtigungen verfügt.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 sns:GetWidget on resource: my-example-widget
```

In diesem Fall muss die Mateo-Richtlinie aktualisiert werden, damit er mit der `sns:`*`GetWidget`*-Aktion auf die *`my-example-widget`*-Ressource zugreifen kann.

Wenden Sie sich an Ihren AWS-Administrator, falls Sie weitere Unterstützung benötigen. Ihr Administrator hat Ihnen Ihre Anmeldeinformationen zur Verfügung gestellt.

Ich bin nicht autorisiert, iam durchzuführen:PassRole

Wenn Sie die Fehlermeldung erhalten, dass Sie nicht zur Ausführung der `iam:PassRole`-Aktion autorisiert sind, müssen Ihre Richtlinien aktualisiert werden, um eine Rolle an Amazon SNS übergeben zu können.

Einige AWS-Services erlauben die Übergabe einer vorhandenen Rolle an diesen Dienst, sodass keine neue Servicerolle oder serviceverknüpfte Rolle erstellt werden muss. Hierzu benötigen Sie Berechtigungen für die Übergabe der Rolle an den Dienst.

Der folgende Beispielfehler tritt auf, wenn ein IAM-Benutzer mit dem Namen `marymajor` versucht, die Konsole zu verwenden, um eine Aktion in Amazon SNS auszuführen. Die Aktion erfordert jedoch, dass der Service über Berechtigungen verfügt, die durch eine Servicerolle gewährt werden. Mary besitzt keine Berechtigungen für die Übergabe der Rolle an den Dienst.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

In diesem Fall müssen die Richtlinien von Mary aktualisiert werden, um die Aktion `iam:PassRole` ausführen zu können.

Wenden Sie sich an Ihren AWS-Administrator, falls Sie weitere Unterstützung benötigen. Ihr Administrator hat Ihnen Ihre Anmeldeinformationen zur Verfügung gestellt.

Ich möchte Personen außerhalb meines AWS-Konto Zugriff auf meine Amazon-SNS-Ressourcen gewähren

Sie können eine Rolle erstellen, die Benutzer in anderen Konten oder Personen außerhalb Ihrer Organisation für den Zugriff auf Ihre Ressourcen verwenden können. Sie können festlegen, wem die Übernahme der Rolle anvertraut wird. Im Fall von Diensten, die ressourcenbasierte Richtlinien oder Zugriffskontrolllisten (Access Control Lists, ACLs) verwenden, können Sie diese Richtlinien verwenden, um Personen Zugriff auf Ihre Ressourcen zu gewähren.

Weitere Informationen dazu finden Sie hier:

- Informationen dazu, ob Amazon SNS diese Funktionen unterstützt, finden Sie unter So funktioniert Amazon Simple Notification Service mit IAM.
- Informationen zum Gewähren des Zugriffs auf Ihre Ressourcen für alle Ihre AWS-Konten finden Sie unter Gewähren des Zugriffs für einen IAM-Benutzer in einem anderen Ihrer AWS-Konto im IAM-Benutzerhandbuch.
- Informationen dazu, wie Sie AWS-Konten-Drittanbieter Zugriff auf Ihre Ressourcen bereitstellen, finden Sie unter Gewähren des Zugriffs auf AWS-Konten von externen Benutzern im IAM-Benutzerhandbuch.
- Informationen dazu, wie Sie über einen Identitätsverbund Zugriff gewähren, finden Sie unter Gewähren von Zugriff für extern authentifizierte Benutzer (Identitätsverbund) im IAM-Benutzerhandbuch.
- Informationen zum Unterschied zwischen der Verwendung von Rollen und ressourcenbasierten Richtlinien für den kontoübergreifenden Zugriff finden Sie unter So unterscheiden sich IAM-Rollen von ressourcenbasierten Richtlinien im IAM-Benutzerhandbuch.

Protokollierung und Überwachung in Amazon SNS

Dieser Abschnitt enthält Informationen zur Protokollierung und Überwachung von Amazon-SNS-Themen.

Themen

Protokollierung von Amazon-SNS-API-Aufrufen mit CloudTrail

Amazon SNS ist in AWS CloudTrail integriert. Dieser Service zeichnet die Aktionen eines Benutzers, einer Rolle oder eines AWS-Service in Amazon SMS auf. CloudTrail erfasst alle API-Aufrufe für Amazon SNS als Ereignisse. Zu den erfassten Aufrufen gehören Aufrufe von der Amazon SNS-Konsole und Code-Aufrufe der Amazon SNS-API-Operationen. Wenn Sie einen Trail erstellen, können Sie die kontinuierliche Bereitstellung von CloudTrail-Ereignissen an einen Amazon-S3-Bucket, einschließlich Ereignissen für Amazon SNS, aktivieren. Wenn Sie keinen Trail konfigurieren, können Sie die neuesten Ereignisse in der CloudTrail-Konsole trotzdem im Ereignisverlauf anzeigen. Mit den von CloudTrail gesammelten Informationen können Sie die an Amazon SNS gestellte Anfrage, die IP-Adresse, von der die Anfrage gestellt wurde, den Initiator der Anfrage, den Zeitpunkt der Anfrage und zusätzliche Angaben bestimmen.

Weitere Informationen über CloudTrail, einschließlich Konfiguration und Aktivierung, finden Sie im AWS CloudTrail-Benutzerhandbuch.

Amazon-SNS-Informationen in CloudTrail

CloudTrail wird beim Erstellen Ihres AWS-Konto für Sie aktiviert. Wenn die unterstützte Ereignisaktivität in Amazon SNS eintritt, wird diese Aktivität in einem CloudTrail-Ereignis zusammen mit anderen AWS-Serviceereignissen im Ereignisverlauf aufgezeichnet. Sie können die neusten Ereignisse in Ihr(em) AWS-Konto anzeigen, suchen und herunterladen. Weitere Informationen finden Sie unter Anzeigen von Ereignissen mit dem CloudTrail-Ereignisverlauf.

Für eine fortlaufende Aufzeichnung der Ereignisse in Ihrem AWS-Konto, darunter Ereignisse für Amazon SNS, können Sie einen Pfad (Trail) erstellen. Ein Trail ermöglicht es CloudTrail, Protokolldateien in einem Amazon-S3-Bucket bereitzustellen. Wenn Sie einen Trail in der Konsole anlegen, gilt dieser für alle AWS-Regionen. Der Trail protokolliert Ereignisse aus allen Regionen in

der AWS-Partition und stellt die Protokolldateien in dem von Ihnen angegebenen Amazon S3 Bucket bereit. Darüber hinaus können Sie andere AWS-Services konfigurieren, um die in den CloudTrail-Protokollen erfassten Ereignisdaten weiter zu analysieren und entsprechend zu agieren. Weitere Informationen finden Sie unter:

- Übersicht zum Erstellen eines Trails
- Von CloudTrail unterstützte Dienste und Integrationen
- Konfigurieren von Amazon-SNS-Benachrichtigungen für CloudTrail
- Empfangen von CloudTrail-Protokolldateien aus mehreren Regionen und Empfangen von CloudTrail-Protokolldateien aus mehreren Konten

Ereignisse der Steuerebene in CloudTrail

Amazon SNS unterstützt die Protokollierung der folgenden Aktionen als Ereignisse in CloudTrail-Protokolldateien:

- `AddPermission`
- `CheckIfPhoneNumberIsOptedOut`
- `ConfirmSubscription`
- `CreatePlatformApplication`
- `CreatePlatformEndpoint`
- `CreateSMSSandboxPhoneNumber`
- `CreateTopic`
- `DeleteEndpoint`
- `DeletePlatformApplication`
- `DeleteSMSSandboxPhoneNumber`
- `DeleteTopic`
- `GetDataProtectionPolicy`
- `GetEndpointAttributes`
- `GetPlatformApplicationAttributes`
- `GetSMSAttributes`
- `GetSMSSandboxAccountStatus`

- GetSubscriptionAttributes
- GetTopicAttributes
- ListEndpointsByPlatformApplication
- ListOriginationNumbers
- ListPhoneNumbersOptedOut
- ListPlatformApplications
- ListSMSSandboxPhoneNumbers
- ListSubscriptions
- ListSubscriptionsByTopic
- ListTagsForResource
- ListTopics
- OptInPhoneNumber
- PutDataProtectionPolicy
- RemovePermission
- SetEndpointAttributes
- SetPlatformApplicationAttributes
- SetSMSAttributes
- SetSubscriptionAttributes
- SetTopicAttributes
- Subscribe
- TagResource
- Unsubscribe
- UntagResource
- VerifySMSSandboxPhoneNumber

> **Note**
>
> Wenn Sie nicht bei Amazon Web Services (nicht authentifizierter Modus) angemeldet sind und entweder die Aktion ConfirmSubscription oder Unsubscribe aufgerufen wird, werden diese nicht bei CloudTrail protokolliert. Ebenso wird, wenn Sie den angegebenen Link in der E-Mail-Benachrichtigung wählen, um ein ausstehendes Abonnement für ein Thema zu

bestätigen, die Aktion `ConfirmSubscription` im nicht authentifizierten Modus aufgerufen. In diesem Beispiel ist die Aktion `ConfirmSubscription` nicht bei CloudTrail protokolliert.

Jeder Ereignis- oder Protokolleintrag enthält Informationen zu dem Benutzer, der die Anforderung generiert hat. Anhand der Identitätsinformationen zur Benutzeridentität können Sie Folgendes bestimmen:

- Ob die Anfrage mit Stammbenutzer- oder AWS Identity and Access Management (IAM)-Anmeldeinformationen ausgeführt wurde.
- Ob die Anforderung mit temporären Sicherheitsanmeldeinformationen für eine Rolle oder einen Verbundbenutzer ausgeführt wurde.
- Gibt an, ob die Anforderung aus einem anderen AWS-Service gesendet wurde

Weitere Informationen finden Sie unter dem CloudTrail userIdentity-Element.

Datenebenenereignisse in CloudTrail

Um die Protokollierung der folgenden API-Aktionen in CloudTrail-Dateien zu aktivieren, müssen Sie die Protokollierung der API-Aktivität auf Datenebene in CloudTrail aktivieren. Weitere Informationen finden Sie unter Protokollieren von Datenereignissen im Benutzerhandbuch für AWS CloudTrail.

Die Ereignisse auf Datenebene können auch nach Ressourcentyp gefiltert werden, um genau zu steuern, welche Amazon SNS-API-Aufrufe Sie in CloudTrail selektiv protokollieren und bezahlen möchten. Wenn Sie beispielsweise `AWS::SNS::Topic` als Ressourcentyp angeben, können Sie Aufrufe an `Publish`- und `PublishBatch`-API-Aktionen für Themen protokollieren. Ebenso können Sie, indem Sie `AWS::SNS::PlatformEndpoint` als Ressourcentyp angeben, Aufrufe der API-Aktion „Veröffentlichen“ für Plattformendpunkte protokollieren. Weitere Informationen finden Sie unter `AdvancedEventSelector` in der AWS CloudTrail-API-Referenz.

> **Note**
>
> Der Amazon-SNS-Ressourcentyp `AWS::SNS::PhoneNumber` wird nicht von CloudTrail protokolliert.

Amazon SNS-Datenebenen-APIs

- Publish
- PublishBatch

Beispiel: Einträge in der Amazon-SNS-Protokolldatei

Ein Trail ist eine Konfiguration, durch die Ereignisse als Protokolldateien an den von Ihnen angegebenen Amazon-S3-Bucket übermittelt werden. CloudTrail-Protokolldateien können einen oder mehrere Einträge enthalten. Ein Ereignis stellt eine einzelne Anfrage aus einer beliebigen Quelle dar und enthält unter anderem Informationen über die angeforderte Aktion, das Datum und die Uhrzeit der Aktion sowie über die Anfrageparameter. CloudTrail-Protokolleinträge sind kein geordnetes Stack-Trace der öffentlichen API-Aufrufe und erscheinen daher in keiner bestimmten Reihenfolge.

Das folgende Beispiel zeigt einen CloudTrail-Protokolleintrag, der die Aktionen `ListTopics`, `CreateTopic` und `DeleteTopic` demonstriert.


```
{
  "Records": [
    {
      "eventVersion": "1.02",
      "userIdentity": {
       "type":"IAMUser",
       "userName":"Bob"
       "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn:aws:iam::123456789012:user/Bob",
        "accountId": "123456789012",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE"
      },
      "eventTime": "2014-09-30T00:00:00Z",
      "eventSource": "sns.amazonaws.com",
      "eventName": "ListTopics",
      "awsRegion": "us-west-2",
      "sourceIPAddress": "127.0.0.1",
      "userAgent": "aws-sdk-java/unknown-version",
      "requestParameters": {
        "nextToken": "ABCDEF1234567890EXAMPLE=="
      },
      "responseElements": null,
      "requestID": "example1-b9bb-50fa-abdb-80f274981d60",
      "eventID": "example0-09a3-47d6-a810-c5f9fd2534fe",
      "eventType": "AwsApiCall",
      "recipientAccountId": "123456789012"
```

```
    },
    {
      "eventVersion": "1.02",
      "userIdentity": {
       "type":"IAMUser",
       "userName":"Bob"
       "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn:aws:iam::123456789012:user/Bob",
        "accountId": "123456789012",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE"
      },
      "eventTime": "2014-09-30T00:00:00Z",
      "eventSource": "sns.amazonaws.com",
      "eventName": "CreateTopic",
      "awsRegion": "us-west-2",
      "sourceIPAddress": "127.0.0.1",
      "userAgent": "aws-sdk-java/unknown-version",
      "requestParameters": {
        "name": "hello"
      },
      "responseElements": {
        "topicArn": "arn:aws:sns:us-west-2:123456789012:hello-topic"
      },
      "requestID": "example7-5cd3-5323-8a00-f1889011fee9",
      "eventID": "examplec-4f2f-4625-8378-130ac89660b1",
      "eventType": "AwsApiCall",
      "recipientAccountId": "123456789012"
    },
    {
      "eventVersion": "1.02",
      "userIdentity": {
       "type":"IAMUser",
       "userName":"Bob"
       "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn:aws:iam::123456789012:user/Bob",
        "accountId": "123456789012",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE"
      },
      "eventTime": "2014-09-30T00:00:00Z",
      "eventSource": "sns.amazonaws.com",
      "eventName": "DeleteTopic",
      "awsRegion": "us-west-2",
      "sourceIPAddress": "127.0.0.1",
      "userAgent": "aws-sdk-java/unknown-version",
```

```
      "requestParameters": {
        "topicArn": "arn:aws:sns:us-west-2:123456789012:hello-topic"
      },
      "responseElements": null,
      "requestID": "example5-4faa-51d5-aab2-803a8294388d",
      "eventID": "example8-6443-4b4d-abfd-1b867280d964",
      "eventType": "AwsApiCall",
      "recipientAccountId": "123456789012"
    },
  ]
}
```

Die folgenden Beispiele zeigen CloudTrail-Protokolleinträge, die die Aktionen `Publish` und `PublishBatch` zeigen.

Veröffentlichen

```
{
 "eventVersion": "1.09",
 "userIdentity": {
  "type": "AssumedRole",
  "principalId": "EX_PRINCIPAL_ID",
  "arn": "arn:aws:iam::123456789012:user/Bob",
  "accountId": "123456789012",
  "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
  "sessionContext": {
   "sessionIssuer": {
    "type": "Role",
    "principalId": "AKIAIOSFODNN7EXAMPLE",
    "arn": "arn:aws:iam::123456789012:role/Admin",
    "accountId": "123456789012",
    "userName": "ExampleUser"
   },
   "attributes": {
    "creationDate": "2023-08-21T16:44:05Z",
    "mfaAuthenticated": "false"
   }
  }
 },
 "eventTime": "2023-08-21T16:48:37Z",
 "eventSource": "sns.amazonaws.com",
 "eventName": "Publish",
 "awsRegion": "us-east-1",
```

```
 "sourceIPAddress": "192.0.2.0",
 "userAgent": "aws-cli/1.29.16 md/Botocore#1.31.16 ua/2.0 os/
linux#5.4.250-173.369.amzn2int.x86_64 md/arch#x86_64 lang/python#3.8.17 md/
pyimpl#CPython cfg/retry-mode#legacy botocore/1.31.16",
 "requestParameters": {
  "topicArn": "arn:aws:sns:us-east-1:123456789012:ExampleSNSTopic",
  "message": "HIDDEN_DUE_TO_SECURITY_REASONS",
  "subject": "HIDDEN_DUE_TO_SECURITY_REASONS",
  "messageStructure": "json",
  "messageAttributes": "HIDDEN_DUE_TO_SECURITY_REASONS"
 },
 "responseElements": {
  "messageId": "0787cd1e-d92b-521c-a8b4-90434e8ef840"
 },
 "requestID": "0a8ab208-11bf-5e01-bd2d-ef55861b545d",
 "eventID": "bb3496d4-5252-4660-9c28-3c6aebdb21c0",
 "readOnly": false,
 "resources": [{
  "accountId": "123456789012",
  "type": "AWS::SNS::Topic",
  "ARN": "arn:aws:sns:us-east-1:123456789012:ExampleSNSTopic"
 }],
 "eventType": "AwsApiCall",
 "managementEvent": false,
 "recipientAccountId": "123456789012",
 "eventCategory": "Data",
 "tlsDetails": {
  "tlsVersion": "TLSv1.2",
  "cipherSuite": "ECDHE-RSA-AES128-GCM-SHA256",
  "clientProvidedHostHeader": "sns.us-east-1.amazonaws.com"
 }
}
```

PublishBatch

```
{
 "eventVersion": "1.09",
 "userIdentity": {
  "type": "AssumedRole",
  "principalId": "EX_PRINCIPAL_ID",
  "arn": "arn:aws:iam::123456789012:user/Bob",
  "accountId": "123456789012",
  "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
```

```
   "sessionContext": {
    "sessionIssuer": {
     "type": "Role",
     "principalId": "AKIAIOSFODNN7EXAMPLE",
     "arn": "arn:aws:iam::123456789012:role/Admin",
     "accountId": "123456789012",
     "userName": "ExampleUser"
    },
    "attributes": {
     "creationDate": "2023-08-21T19:20:49Z",
     "mfaAuthenticated": "false"
    }
   }
  },
  "eventTime": "2023-08-21T19:22:01Z",
  "eventSource": "sns.amazonaws.com",
  "eventName": "PublishBatch",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "192.0.2.0",
  "userAgent": "aws-cli/1.29.16 md/Botocore#1.31.16 ua/2.0 os/
linux#5.4.250-173.369.amzn2int.x86_64 md/arch#x86_64 lang/python#3.8.17 md/
pyimpl#CPython cfg/retry-mode#legacy botocore/1.31.16",
  "requestParameters": {
   "topicArn": "arn:aws:sns:us-east-1:123456789012:ExampleSNSTopic",
   "publishBatchRequestEntries": [{
     "id": "1",
     "message": "HIDDEN_DUE_TO_SECURITY_REASONS"
    },
    {
     "id": "2",
     "message": "HIDDEN_DUE_TO_SECURITY_REASONS"
    }
   ]
  },
  "responseElements": {
   "successful": [{
     "id": "1",
     "messageId": "30d68101-a64a-5573-9e10-dc5c1dd3af2f"
    },
    {
     "id": "2",
     "messageId": "c0aa0c5c-561d-5455-b6c4-5101ed84de09"
    }
   ],
```

```
  "failed": []
 },
 "requestID": "e2cdf7f3-1b35-58ad-ac9e-aaaea0ace2f1",
 "eventID": "10da9a14-0154-4ab6-b3a5-1825b229a7ed",
 "readOnly": false,
 "resources": [{
  "accountId": "123456789012",
  "type": "AWS::SNS::Topic",
  "ARN": "arn:aws:sns:us-east-1:123456789012:ExampleSNSTopic"
 }],
 "eventType": "AwsApiCall",
 "managementEvent": false,
 "recipientAccountId": "123456789012",
 "eventCategory": "Data",
 "tlsDetails": {
  "tlsVersion": "TLSv1.2",
  "cipherSuite": "ECDHE-RSA-AES128-GCM-SHA256",
  "clientProvidedHostHeader": "sns.us-east-1.amazonaws.com"
 }
}
```

Überwachung von Amazon-SNS-Themen mit CloudWatch

Amazon SNS und Amazon CloudWatch sind integriert, sodass Sie Metriken für alle aktiven Amazon SNS-Benachrichtigungen erfassen, anzeigen und analysieren können. Nachdem Sie CloudWatch für Amazon SNS konfiguriert haben, erhalten Sie bessere Einblicke in die Leistung Ihrer Amazon-SNS-Themen, Push-Benachrichtigungen und SMS-Zustellungen. Sie können z. B. einen Alarm festlegen, sodass Ihnen eine E-Mail-Benachrichtigung gesendet wird, sobald ein bestimmter Schwellenwert für eine Amazon-SNS-Metrik erreicht wird, wie z. B. `NumberOfNotificationsFailed`. Eine Liste aller Metriken, die von Amazon SNS an CloudWatch gesendet werden, finden Sie unter Amazon SNS-Metriken. Weitere Informationen über Amazon-SNS-Push-Benachrichtigungen finden Sie unter Mobile Push-Benachrichtigungen.

> **Note**
>
> Die Metriken, die Sie mit CloudWatch für Ihre Amazon SNS-Themen konfigurieren, werden automatisch gesammelt und an CloudWatch übergeben in 1 Minuten-Intervallen. Diese Metriken werden für alle Themen gesammelt, die im Sinne der CloudWatch-Richtlinien aktiv sind. Ein Thema wird von CloudWatch für bis zu sechs Stunden ab der letzten Aktivität (z. B. einem API-Aufruf) für das Thema als aktiv angesehen.

Es fallen keine Gebühren für die Amazon-SNS-Metriken an, die in CloudWatch gemeldet werden. Sie werden im Rahmen des Amazon-SNS-Service bereitgestellt.

Anzeigen von CloudWatch-Metriken für Amazon SNS

Sie können Metriken für Amazon SNS über die CloudWatch-Konsole, die Befehlszeilenschnittstelle (CLI) von CloudWatch oder programmgesteuert mithilfe der CloudWatch-API überwachen. Die folgenden Verfahren zeigen Ihnen, wie Sie mithilfe der AWS Management Console auf die Metriken zugreifen können.

So zeigen Sie Metriken mit der CloudWatch-Konsole an:

1. Melden Sie sich in der CloudWatch Konsole an.
2. Wählen Sie im Navigationsbereich Metriken aus.
3. Wählen Sie auf der Registerkarte All metrics (Alle Metriken) die Option SNS und danach eine der folgenden Dimensionen aus:
   - Country, SMS Type (Land, SMS-Typ)
   - Telefonnummer
   - Topics Metrics (Themen-Metriken)
   - Metrics with no dimensions (Metriken ohne Dimensionen)
4. Um weitere Details anzuzeigen, wählen Sie ein bestimmtes Element aus. Wenn Sie beispielsweise Themen-Metriken und dann NumberOfMessagesPublished auswählen, wird die durchschnittliche Anzahl der veröffentlichten Amazon-SNS-Nachrichten für einen Zeitraum von einer Minute innerhalb des 6-stündigen Zeitraums angezeigt.
5. Wenn Sie die Amazon-SNS-Nutzungsmetriken anzeigen möchten, wählen Sie auf der Registerkarte All metrics (Alle Metriken) die Option Usage (Nutzung) und die target Amazon SNS usage metric (Amazon-SNS-Zielmetrik) aus (z. B. `NumberOfMessagesPublishedPerAccount`).

Anzeigen von CloudWatch-Metriken für Amazon SNS

Mit CloudWatch können Sie auch Alarme für eine Metrik festlegen, die ausgelöst werden, wenn ein Schwellenwert erreicht wird. So können Sie beispielsweise einen Alarm für die Metrik NumberOfNotificationsFailed festlegen. Wenn Ihr angegebener Schwellenwert innerhalb des

Erfassungszeitraums erreicht wird, wird eine E-Mail-Benachrichtigung gesendet, um Sie über das Ereignis zu informieren.

Stellen Sie Alarme mit der CloudWatch-Konsole wie folgt ein:

1. Melden Sie sich bei AWS Management Console an und öffnen Sie die CloudWatch-Konsole unter https://console.aws.amazon.com/cloudwatch/.
2. Wählen Sie Alarme und dann die Schaltfläche Alarm erstellen aus. Dadurch wird der Assistent Alarm erstellen gestartet.
3. Scrollen Sie durch die Amazon-SNS-Metriken, um die Metrik zu finden, auf die Sie einen Alarm setzen möchten. Wählen Sie die Metrik aus, um einen Alarm zu erstellen, und wählen Sie dann Fortfahren.
4. Geben Sie für die Metrik die Werte Name, Beschreibung, Schwellenwert und Zeit an und wählen Sie dann Fortfahren.
5. Wählen Sie als Alarmzustand Alarm aus. Wenn von CloudWatch bei Erreichen des Alarmzustands eine E-Mail gesendet werden soll, wählen Sie entweder ein vorhandenes Amazon-SNS-Thema aus oder wählen Sie Neues E-Mail-Thema erstellen. Wenn Sie Neues E-Mail-Thema erstellen wählen, können Sie den Namen und die E-Mail Adressen für ein neues Thema einrichten. Diese Liste wird gespeichert und im Dropdown-Menü für künftige Alarme angezeigt. Klicken Sie auf Fortfahren.

   > **Note**
   >
   > Wenn Sie Neues E-Mail-Thema erstellen verwenden, um ein neues Amazon SNS-Thema zu erstellen, müssen die E-Mail Adressen überprüft werden, bevor die Empfänger Benachrichtigungen erhalten können. E-Mail Nachrichten werden nur gesendet, wenn der Alarm in einen Alarmzustand wechselt. Wenn es zu dieser Änderung des Alarmzustands kommt, bevor die E-Mail Adressen überprüft wurden, erhalten die Empfänger keine Benachrichtigung.

6. An diesem Punkt bietet Ihnen der Assistent Alarm erstellen die Möglichkeit, den Alarm, den Sie gerade erstellen, zu überprüfen. Wenn Sie Änderungen vornehmen müssen, können Sie die Links Bearbeiten auf der rechten Seite verwenden. Wenn Sie zufrieden sind, wählen Sie Alarm erstellen.

Weitere Informationen zur Verwendung von CloudWatch und Alarmen finden Sie in der CloudWatch-Dokumentation.

Amazon SNS-Metriken

Amazon SNS sendet die folgenden Metriken an CloudWatch.

Namespace	Metrik	Beschreibung
AWS/SNS	`NumberOfMessagesPublished`	Die Anzahl der zu Ihren Amazon SNS-Themen veröffentlichten Mitteilungen. Einheiten: Anzahl Gültige Dimensionen: Anwendung, Telefonnummer, Plattform und Themenname Gültige Statistiken: Summe
AWS/SNS	`NumberOfNotificationsDelivered`	Die Anzahl der Mitteilungen, die von Ihren Amazon SNS-Themen erfolgreich an Endpunkte übermittelt wurden. Damit ein Zustellungsversuch erfolgreich ausgeführt wird, muss das Abonnement des Endpunkts die Mitteilung entgegennehmen. Ein Abonnement akzeptiert eine Nachricht, wenn a.) eine Filterrichtlinie fehlt oder b.) die Filterrichtlinie Attribute enthält, die mit denjenigen übereinstimmen, die der Nachricht zugewiesen wurden. Falls das Abonnement die Mitteilung ablehnt, wird der Zustellun

Namespace	Metrik	Beschreibung
		gsversuch für diese Metrik nicht gezählt. Einheiten: Anzahl Gültige Dimensionen: Anwendung , Telefonnummer, Plattform und Themenname Gültige Statistiken: Summe

Namespace	Metrik	Beschreibung
AWS/SNS	`NumberOfNotificationsFailed`	Die Anzahl von Nachrichten, die Amazon SNS nicht zustellen konnte. Bei Amazon SQS-, E-Mail-, SMS- oder mobilen Push-Endpunkten steigt diese Metrik um 1, wenn Amazon SNS die Nachricht enzustellungsversuche einstellt. Bei HTTP- oder HTTPS-Endpunkten umfasst die Metrik jeden fehlgeschlagenen Zustellversuch, einschließlich Wiederholungen nach dem ursprünglichen Versuch. Für alle anderen Endpunkte steigt die Zahl um 1, wenn die Mitteilung nicht zugestellt wird (unabhängig von der Anzahl der Versuche). Diese Metrik enthält keine Nachrichten, die von Abonnementfilterrichtlinien abgelehnt wurden. Sie können die Anzahl der Wiederholungen für HTTP-Endpunkte kontrollieren. Weitere Informationen finden Sie unter Wiederholungsversuche bei der Nachrichtenzustellung Amazon SNS. Einheiten: Anzahl Gültige Dimensionen: Anwendung, Telefonnummer, Plattform und Themenname

Namespace	Metrik	Beschreibung
		Gültige Statistiken: Summe, Durchschnitt
AWS/SNS	`NumberOfNotificationsFilteredOut`	Die Anzahl der Nachrichten, die von Abonnementfilterrichtlinien abgelehnt wurden. Eine Filterrichtlinie lehnt eine Nachricht ab, wenn die Nachrichtenattribute nicht mit den Richtlinienattributen übereinstimmen. Einheiten: Anzahl Gültige Dimensionen: Anwendung, Telefonnummer, Plattform und Themenname Gültige Statistiken: Summe, Durchschnitt
AWS/SNS	`NumberOfNotificationsFilteredOut-MessageAttributes`	Die Anzahl der Nachrichten, die von Abonnementfilterrichtlinien für die attributbasierte Filterung abgelehnt wurden. Einheiten: CountValid Dimensionen: Anwendung, Telefonnummer, Plattform und Themenname Gültige Statistiken: Summe, Durchschnitt

Namespace	Metrik	Beschreibung
AWS/SNS	`NumberOfNotificationsFilteredOut-MessageBody`	Die Anzahl der Nachrichten, die von Abonnementfilterrichtlinien für die nutzlastbasierte Filterung abgelehnt wurden. Einheiten: Anzahl Gültige Dimensionen: Anwendung, Telefonnummer, Plattform und Themenname Gültige Statistiken: Summe, Durchschnitt
AWS/SNS	`NumberOfNotificationsFilteredOut-InvalidAttributes`	Die Anzahl der Nachrichten, die von Abonnementfilterrichtlinien abgelehnt wurden, weil die Nachrichtenattribute ungültig sind zum Beispiel, weil der JSON des Attributs falsch formatiert ist. Einheiten: Anzahl Gültige Dimensionen: Anwendung, Telefonnummer, Plattform und Themenname Gültige Statistiken: Summe, Durchschnitt

Namespace	Metrik	Beschreibung
AWS/SNS	`NumberOfNotificationsFilteredOut-NoMessageAttributes`	Die Anzahl der Nachrichten, die von Abonnementfilterrichtlinien abgelehnt wurden, weil die Nachrichten keine Attribute enthalten. Einheiten: Anzahl Gültige Dimensionen: Anwendung, Telefonnummer, Plattform und Themenname Gültige Statistiken: Summe, Durchschnitt
AWS/SNS	`NumberOfNotificationsFilteredOut-InvalidMessageBody`	Die Anzahl der Nachrichten, die von Abonnementfilterrichtlinien aufgrund eines für die Filterung ungültigen Nachrichtentexts abgelehnt wurden, zum Beispiel ein ungültiger JSON-Nachrichtentext. Einheiten: Anzahl Gültige Dimensionen: Anwendung, Telefonnummer, Plattform und Themenname Gültige Statistiken: Summe, Durchschnitt

Namespace	Metrik	Beschreibung
AWS/SNS	`NumberOfNotificationsRedrivenToDlq`	Die Anzahl der Nachrichten, die in eine Warteschlange für unzustellbare Nachrichten verschoben wurden. Einheiten: Anzahl Gültige Dimensionen: Anwendung, Telefonnummer, Plattform und Themenname Gültige Statistiken: Summe, Durchschnitt
AWS/SNS	`NumberOfNotificationsFailedToRedriveToDlq`	Die Anzahl der Nachrichten, die nicht in eine Warteschlange für unzustellbare Nachrichten verschoben werden konnten. Einheiten: Anzahl Gültige Dimensionen: Anwendung, Telefonnummer, Plattform und Themenname Gültige Statistiken: Summe, Durchschnitt

Namespace	Metrik	Beschreibung
AWS/SNS	`PublishSize`	Die Größe von veröffentlichten Nachrichten. Einheiten: Byte Gültige Dimensionen: Anwendung, Telefonnummer, Plattform und Themenname Gültige Statistiken: Minimum, Maximum, Durchschnitt, und Anzahl

Namespace	Metrik	Beschreibung
AWS/SNS	`SMSMonthToDateSpentUSD`	Ihre Gebühren für das Senden von SMS-Nachrichten seit Beginn des aktuellen Kalendermonats. Sie können für diese Metrik einen Alarm festlegen, um zu wissen, wann Ihre Gebühren für den aktuellen Monat sich dem monatlichen SMS-Ausgabenkontingent für Ihr Konto nähern. Wenn Amazon SNS feststellt, dass durch den Versand einer SMS-Nachricht Kosten entstehen, die dieses Kontingent überschreiten, wird die Veröffentlichung von SMS-Nachrichten innerhalb von Minuten beendet. Weitere Informationen zum Einrichten Ihres monatlichen Ausgabenkontingents für SMS oder zum Anfordern einer Erhöhung des Ausgabenkontingents bei AWS finden Sie unter Festlegen von SMS-Messaging-Einstellungen. Einheiten: USD Gültige Dimensionen: Telefonnummer Gültige Statistiken: Maximum

Namespace	Metrik	Beschreibung
AWS/SNS	`SMSSuccessRate`	Die Rate der erfolgreichen SMS-Nachrichtenzustellungen Einheiten: Anzahl Gültige Dimensionen: Telefonnummer Gültige Statistiken: Summe, Durchschnitt, Datenstichproben

Dimensionen für Amazon-SNS-Metriken

Amazon Simple Notification Service sendet die folgenden Dimensionen an CloudWatch.

Dimension	Beschreibung
`Application`	Filtert Anwendungsobjekte, die eine App und ein Gerät darstellen, die bei einem der unterstützten Push-Benachrichtigungs-Services wie APNs und FCM registriert sind.
`Application,Platform`	Filtert Anwendungs- und Plattformobjekte, wobei die Plattformobjekte für die unterstützten Push-Benachrichtigungs-Services wie APNs und FCM bestimmt sind.
`Country`	Filtert das Zielland oder die Zielregion einer SMS-Mitteilung. Das Land oder die Region werden durch den zugehörigen ISO 3166-1 Alpha-2-Code dargestellt.
`PhoneNumber`	Filtert die Telefonnummer, wenn Sie SMS direkt an eine Telefonnummer (ohne Thema) veröffentlichen.
`Platform`	Filtert Plattformobjekte für die Push-Benachrichtigungs-Services wie APNs und FCM.
`TopicName`	Filtert Amazon-SNS-Themennamen.

Dimension	Beschreibung
SMSType	Filtert den Nachrichtentyp einer SMS-Mitteilung. Mögliche Werte sind promotional oder transactional.

Amazon-SNS-Nutzungsmetriken

Amazon Simple Notification Service sendet die folgenden Nutzungsmetriken an CloudWatch.

Namespace	Service	Metrik	Ressource	Typ	Beschreibung
AWS/Verwendung	SNS	ResourceCount	NumberOfMessagesPublishedPerAccount	Ressource	• Die Anzahl der zu den Amazon-SNS-Themen in Ihrem AWS-Konto veröffentlichten Mitteilungen. • Einheiten: keine • Gültige Statistiken: Summe
AWS/Verwendung	SNS	ResourceCount	ApproximateNumberOfTopics	Ressource	• Die ungefähre Anzahl von Themen in Ihrem AWS-Konto.

Namespace	Service	Metrik	Ressource	Typ	Beschreibung
					• Einheiten: keine • Gültige Statistiken: Minimum, Maximum, Summe, Durchschnitt
AWS/Verwendung	SNS	`ResourceCount`	`ApproximateNumberOfFilterPolicies`	Ressource	• Die ungefähre Anzahl von Filterrichtlinien in Ihrem AWS-Konto. • Einheiten: keine • Gültige Statistiken: Minimum, Maximum, Summe, Durchschnitt

Namespace	Service	Metrik	Ressource	Typ	Beschreibung
AWS/Verwendung	SNS	`ResourceCount`	`ApproximateNumberOfPendingSubscriptions`	Ressource	• Die ungefähre Anzahl ausstehender Abonnements in Ihrem AWS-Konto. • Einheiten: keine • Gültige Statistiken: Minimum, Maximum, Summe, Durchschnitt

Namespace	Service	Metrik	Ressource	Typ	Beschreibung
AWS/Verwendung	SNS	`CallCount`	• `AddPermission` • `CheckIfPhoneNumberIsOptedOut` • `CreatePlatformApplication` • `CreatePlatformEndpoint` • `ConfirmSubscription` • `CreateSMSSandboxPhoneNumber` • `CreateTopic` • `DeleteEndpoint` • `DeletePlatformApplication` • `DeleteSMSSandboxPhoneNumber` • `DeleteTopic`	API	• Die Anzahl der API-Aufrufe für die ausgewählte Amazon-SNS-API in Ihrem AWS-Konto. • Einheiten: keine • Gültige Statistiken: Summe

Namespace	Service	Metrik	Ressource	Typ	Beschreibung
			• GetEndpointAttributes • GetPlatformApplicationAttributes • GetSMSAttributes • GetSMSSandboxAccountStatus • GetSubscriptionAttributes • GetTopicAttributes • ListEndpointsByPlatformApplication • ListOriginationNumbers • ListPhoneNumbersOptedOut • ListPlatformApplications		

Namespace	Service	Metrik	Ressource	Typ	Beschreibung
			• ListSMSSandboxPhoneNumbers • ListSubscriptions • ListSubscriptionsByTopic • ListTagsForResource • ListTopics • OptInPhoneNumber • RemovePermission • SetEndpointAttributes • SetPlatformApplicationAttributes • SetSMSAttributes • SetSubscriptionAttributes • SetTopicAttributes		

Namespace	Service	Metrik	Ressource	Typ	Beschreibung
			• `Subscribe` • `Unsubscribe` • `UntagResource` • `VerifySMSSandboxPhoneNumber`		

Compliance-Validierung für Amazon SNS

Externe Prüfer bewerten die Sicherheit und Compliance von Amazon SNS im Rahmen mehrerer AWS-Compliance-Programme, einschließlich des Health Insurance Portability and Accountability Act (HIPAA).

Eine Liste der AWS-Services im Bereich bestimmter Compliance-Programme finden Sie unter AWS-Services im Bereich nach Compliance-Programm. Allgemeine Informationen finden Sie unter AWS-Compliance-Programme.

Sie können Auditberichte von Drittanbietern unter AWS Artifact herunterladen. Weitere Informationen finden Sie unter Berichte herunterladen in AWS Artifact.

Ihre Compliance-Verantwortung bei der Verwendung von Amazon SNS ist von der Sensibilität Ihrer Daten, den Compliance-Zielen Ihres Unternehmens und den geltenden Gesetzen und Vorschriften abhängig. AWS stellt die folgenden Ressourcen zur Unterstützung der Compliance bereit:

- Kurzanleitungen für Sicherheit und Compliance– In diesen Bereitstellungsleitfäden finden Sie wichtige Überlegungen zur Architektur sowie die einzelnen Schritte zur Bereitstellung von sicherheits- und compliance-orientierten Basisumgebungen in AWS.
- Whitepaper zur Erstellung einer Architektur mit HIPAA-konformer Sicherheit und Compliance – In diesem Whitepaper wird beschrieben, wie Unternehmen mithilfe von AWS HIPAA-konforme Anwendungen erstellen können.

- AWS-Compliance-Ressourcen – Diese Arbeitsbücher und Leitfäden könnten für Ihre Branche und Ihren Standort relevant sein.
- Auswertung von Ressourcen mit Regeln im AWS ConfigEntwicklerhandbuch – Der AWS Config-Service bewertet, wie gut Ihre Ressourcenkonfigurationen mit internen Praktiken, Branchenrichtlinien und Vorschriften übereinstimmen.
- AWS Security Hub – Dieser AWS-Service liefert einen umfassenden Überblick über den Sicherheitsstatus in AWS. So können Sie die Compliance mit den Sicherheitsstandards in der Branche und den bewährten Methoden abgleichen.

Ausfallsicherheit bei Amazon SNS

Die Resilienz von Amazon SNS wird durch die Nutzung der AWS globalen Infrastruktur gewährleistet, die sich um Availability Zones AWS-Regionen dreht. AWS-Regionen bieten physisch getrennte und isolierte Availability Zones, die über Netzwerke mit niedriger Latenz, hohem Durchsatz und hoher Redundanz miteinander verbunden sind. Diese Architektur ermöglicht einen nahtlosen Failover zwischen Availability Zones ohne Unterbrechung, wodurch Anwendungen und Datenbanken im Vergleich zu herkömmlichen Rechenzentrumsinfrastrukturen von Natur aus fehlertoleranter und skalierbarer sind. Durch die Verwendung von Availability Zones profitieren Amazon SNS SNS-Abonnenten von verbesserter Verfügbarkeit und Zuverlässigkeit, sodass die Nachrichtenzustellung trotz potenzieller Unterbrechungen gewährleistet ist. Weitere Informationen zu Availability Zones AWS-Regionen und Availability Zones finden Sie unter AWS Globale Infrastruktur.

Darüber hinaus können Abonnements für Amazon SNS SNS-Themen mit Wiederholungsversuchen und Warteschlangen für unzustellbare Briefe konfiguriert werden, sodass vorübergehende Ausfälle automatisch behandelt werden und sichergestellt wird, dass Nachrichten zuverlässig ihren Bestimmungsort erreichen.

Amazon SNS unterstützt auch Nachrichtenfilterung und Nachrichtenattribute, sodass Sie Resilienzstrategien an ihre spezifischen Anwendungsfälle anpassen und so die allgemeine Robustheit Ihrer Anwendungen verbessern können.

Infrastruktursicherheit in Amazon SNS

Als verwalteter Service ist Amazon SNS durch die AWS globalen Netzwerksicherheitsverfahren geschützt, die in der Dokumentation Best Practices for Security, Identity and Compliance beschrieben sind.

Verwenden Sie AWS API-Aktionen, um über das Netzwerk auf Amazon SNS zuzugreifen. Clients müssen Transport Layer Security (TLS) 1.2 oder höher unterstützen. Clients müssen außerdem Cipher Suites mit PFS (Perfect Forward Secrecy) wie DHE (Ephemeral Diffie-Hellman) oder ECDHE (Elliptic Curve Ephemeral Diffie-Hellman) unterstützen.

Anforderungen müssen sowohl mit einer Zugriffsschlüssel-ID als auch mit einem geheimen Zugriffsschlüssel signiert sein, die mit einem IAM-Prinzipal verknüpft sind. Alternativ können Sie die AWS Security Token Service (AWS STS) verwenden, um temporäre Anmeldeinformationen für Signaturanforderungen zu generieren.

Sie können API-Aktionen von jedem Netzwerkstandort aus aufrufen. Amazon SNS unterstützt jedoch ressourcenbasierte Zugriffsrichtlinien, die Einschränkungen auf der Basis der Quell-IP-Adresse enthalten können. Sie können auch Amazon SNS-Richtlinien verwenden, um den Zugriff über bestimmte Amazon-VPC-Endpunkte oder bestimmte VPCs zu steuern. Dadurch wird der Netzwerkzugriff auf ein bestimmtes Amazon SNS SNS-Thema effektiv nur von der spezifischen VPC innerhalb des Netzwerks isoliert. AWS Weitere Informationen finden Sie unter Die Veröffentlichung von einem bestimmten VPC-Endpunkt aus auf nur ein Amazon SNS -Thema beschränken.

Bewährte Methoden für die Sicherheit in Amazon SNS

AWS bietet viele Sicherheitsfunktionen für Amazon SNS. Überprüfen Sie diese Sicherheitsfunktionen im Kontext Ihrer eigenen Sicherheitsrichtlinie.

> **Note**
>
> Der Leitlinien für diese Sicherheitsfunktionen gilt für allgemeine Anwendungsfälle und Implementierungen. Wir empfehlen Ihnen, diese bewährten Methoden im Kontext Ihres spezifischen Anwendungsfalls, der Architektur und des Bedrohungsmodells zu überprüfen.

Vorbeugende bewährte Methoden

Im Folgenden werden bewährte vorbeugende Sicherheitsmethoden für Amazon SNS beschrieben.

Themen

- Stellen Sie sicher, dass Themen nicht öffentlich zugänglich sind
- Implementieren der geringstmöglichen Zugriffsrechte

- IAM-Rollen für Anwendungen und AWS-Services verwenden, die Amazon SNS-Zugriff erfordern
- Implementieren serverseitiger Verschlüsselung
- Erzwingen der Verschlüsselung von Daten während der Übertragung
- Erwägen der Verwendung von VPC-Endpunkten für den Zugriff auf Amazon SNS
- Stellen Sie sicher, dass Abonnements nicht für die Bereitstellung an unformatierte HTTP-Endpunkte konfiguriert sind

Stellen Sie sicher, dass Themen nicht öffentlich zugänglich sind

Sofern es nicht ausdrücklich für Sie erforderlich ist, dass beliebiger Benutzer im Internet Ihr Amazon SNS-Thema lesen oder in es schreiben können, sollten Sie sicherstellen, dass Ihr Thema nicht öffentlich zugänglich ist (für jeden in aller Welt oder alle Personen auf der Welt oder für jeden authentifizierten AWS-Benutzer).

- Vermeiden Sie das Erstellen von Richtlinien mit auf `""` festgelegtem `Principal`.
- Vermeiden Sie die Verwendung eines Platzhalters (`*`). Benennen Sie stattdessen einen oder mehrere bestimmte Benutzer.

Implementieren der geringstmöglichen Zugriffsrechte

Wenn Sie Berechtigungen erteilen, entscheiden Sie, wer sie erhält, für welche Themen die Berechtigungen gelten und welche bestimmten API-Aktionen für diese Themen zugelassen werden. Die Umsetzung des Prinzips der geringsten Rechte ist für die Reduzierung von Sicherheitsrisiken ausschlaggebend. Es hilft auch, die negativen Auswirkungen von Fehlern oder böswilligen Absichten zu reduzieren.

Folgen Sie den standardmäßigen Sicherheitshinweisen zur Erteilung von geringsten Privilegien. Das heißt: erteilen Sie nur die Berechtigungen, die zum Ausführen einer bestimmten Aufgabe erforderlich sind. Sie können die geringsten Rechte implementieren, indem Sie eine Kombination von Sicherheitsrichtlinien für den Benutzerzugriff verwenden.

Amazon SNS verwendet das Herausgeber-Abonnenten-Modell, das drei Arten von Benutzerkontenzugriff erfordert:

- Administratoren – Zugriff auf das Erstellen, Ändern und Löschen von Themen. Administratoren steuern auch Themenrichtlinien.

- Herausgeber – Zugriff auf das Senden von Nachrichten an Themen.
- Abonnenten – Zugriff auf das Abonnieren von Themen.

Weitere Informationen finden Sie in den folgenden Abschnitten:

- Identity and Access Management in Amazon SNS
- Amazon SNS-API-Berechtigungen: Referenztabelle für Aktionen und Ressourcen

IAM-Rollen für Anwendungen und AWS-Services verwenden, die Amazon SNS-Zugriff erfordern

Damit Anwendungen oder AWS-Services, wie z. B. Amazon EC2, auf Amazon SNS-Themen zugreifen können, müssen sie in ihren AWS-API-Anforderungen gültige AWS-Anmeldeinformationen verwenden. Da diese Anmeldeinformationen nicht automatisch rotiert werden, sollten Sie AWS-Anmeldeinformationen nicht direkt in der Anwendung oder in der EC2-Instance speichern.

Sie sollten mithilfe einer IAM-Rolle temporäre Anmeldeinformationen für Anwendungen und Services verwalten, die Zugriff auf Amazon SNS benötigen. Wenn Sie eine Rolle verwenden, müssen Sie keine langfristigen Anmeldeinformationen (wie Benutzername, Passwort und Zugriffsschlüssel) an eine EC2-Instance und einen AWS-Service wie AWS Lambda ausgeben. Stattdessen stellt die Rolle temporäre Berechtigungen bereit, die von den Anwendungen beim Aufrufen von anderen AWS-Ressourcen genutzt werden können.

Weitere Informationen finden Sie unter IAM-Rollen und Gängige Szenarien für Rollen: Benutzer, Anwendungen und Services im IAM Benutzerhandbuch.

Implementieren serverseitiger Verschlüsselung

Probleme durch Datenlecks lassen sich verringern, indem Sie die Verschlüsselung im Ruhezustand verwenden. Dabei verschlüsseln Sie Ihre Nachrichten mithilfe eines Schlüssels, der an einem anderen Speicherort gespeichert ist als Ihre Nachrichten. Serverseitige Verschlüsselung (SSE) bietet Datenverschlüsselung im Ruhezustand. Amazon SNS verschlüsselt Ihre Daten auf Nachrichtenebene bei der Speicherung und entschlüsselt die Nachrichten für Sie bei Zugriff darauf. SSE verwendet Schlüssel, die in AWS Key Management Service verwaltet werden. Solange Sie Ihre Anfrage authentifizieren und über Zugriffsberechtigungen verfügen, besteht kein Unterschied zwischen dem Zugriff auf verschlüsselte und unverschlüsselte Themen.

Weitere Informationen finden Sie unter Verschlüsselung im Ruhezustand und Schlüsselverwaltung.

Erzwingen der Verschlüsselung von Daten während der Übertragung

Es ist möglich, aber nicht zu empfehlen, Nachrichten zu veröffentlichen, die während der Übertragung über HTTP nicht verschlüsselt sind. Sie können HTTP jedoch nicht verwenden, wenn Sie in einem verschlüsselten SNS-Thema veröffentlichen.

AWS empfiehlt, HTTPS anstelle von HTTP zu verwenden. Wenn Sie HTTPS verwenden, werden Nachrichten während des Transports automatisch verschlüsselt, selbst wenn das SNS-Thema selbst nicht verschlüsselt ist. Ohne HTTPS kann ein netzwerkbasierter Angreifer mithilfe eines Angriffs wie Man-in-the-Middle den Datenverkehr im Netzwerk abhören oder auf ihn einwirken.

Um nur verschlüsselte Verbindungen über HTTPS zu erzwingen, fügen Sie die`aws:SecureTransport`-Bedingung in der IAM-Richtlinie hinzu, die unverschlüsselten SNS-Themen zugeordnet ist. Dies zwingt Nachrichtenherausgeber, HTTPS anstelle von HTTP zu verwenden. Sie können folgendes Beispiel nutzen:

```
{
  "Id": "ExamplePolicy",
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowPublishThroughSSLOnly",
      "Action": "SNS:Publish",
      "Effect": "Deny",
      "Resource": [
        "arn:aws:sns:us-east-1:1234567890:test-topic"
      ],
      "Condition": {
        "Bool": {
          "aws:SecureTransport": "false"
        }
      },
      "Principal": "*"
    }
  ]
}
```

Erwägen der Verwendung von VPC-Endpunkten für den Zugriff auf Amazon SNS

Verwenden Sie bei Themen, mit denen eine Interaktion erforderlich ist, die jedoch dem Internet absolut nicht zugänglich gemacht werden dürfen, VPC-Endpunkte, um den Zugriff auf Themen

ausschließlich auf die Hosts innerhalb einer bestimmten VPC einzuschränken. Mit Themenrichtlinien können Sie den Zugriff auf Themen von bestimmten Amazon VPC-Endpunkten oder von bestimmten VPCs aus steuern.

Amazon SNS-VPC-Endpunkte bieten zwei Möglichkeiten zur Kontrolle des Zugriffs auf Ihre Nachrichten:

- Sie können steuern, welche Anfragen, Benutzer oder Gruppen durch einen spezifischen VPC-Endpunkt erlaubt sind.
- Mithilfe einer Themenrichtlinie können Sie steuern, welche VPCs oder VPC-Endpunkte Zugriff auf Ihre Themenrichtlinie haben.

Weitere Informationen finden Sie unter Erstellen des Endpunkts und Erstellen einer Amazon-VPC-Endpunktrichtlinie für Amazon SNS.

Stellen Sie sicher, dass Abonnements nicht für die Bereitstellung an unformatierte HTTP-Endpunkte konfiguriert sind

Vermeiden Sie das Konfigurieren von Abonnements für die Bereitstellung an unformatierte HTTP-Endpunkte. Stellen Sie immer Abonnements bereit, die an einen Endpunkt-Domänennamen gesendet werden. Beispiel: Ein Abonnement, das für die Bereitstellung an einen Endpunkt konfiguriert ist, `http://1.2.3.4/my-path`, sollte geändert werden in `http://my.domain.name/my-path`.

Fehlerbehebung Amazon-SNS-Themen

Dieser Abschnitt enthält Informationen zur Fehlerbehebung in Amazon-SNS-Themen.

Fehlerbehebung Amazon-SNS-Themen mit AWS X-Ray

AWS X-Ray erfasst Daten über Anforderungen, die von Ihrer Anwendung verarbeitet werden, und ermöglicht Ihnen, Daten anzuzeigen und zu filtern, um potenzielle Probleme und Möglichkeiten zur Optimierung aufzudecken. Für jede verfolgte Clientanforderung für Ihre Anwendung sehen Sie detaillierte Informationen über die Anforderung, die Antwort und die Aufrufe, die Ihre Anwendung an nachgelagerte AWSRessourcen, Mikroservices, Datenbanken und HTTP-Web-APIs sendet.

Sie können X-Ray mit Amazon SNS verwenden, um die in Ihrer Anwendung weitergeleiteten Nachrichten nachzuverfolgen und zu analysieren. Sie können mithilfe AWS Management Console der Karte die Zuordnung der Verbindungen zwischen Amazon SNS und anderen Services anzeigen, die von Ihrer Anwendung verwendet werden. Sie können mithilfe der Konsole auch Metriken, wie durchschnittliche Latenz- und Ausfallraten, anzuzeigen. Weitere Informationen finden Sie unter Amazon SNS und AWS X-Ray im AWS X-Ray Entwicklerhandbuch.

Aktives Tracing in Amazon SNS

Sie können verwendenAWS X-Ray, um Benutzeranfragen zu verfolgen und zu analysieren, während sie durch Ihre Amazon SNS-Themen zu Ihren Amazon-Data-Firehose-, -AWS Lambda, Amazon SQS- und HTTP/S-Endpunktabonnements gelangen. Da X-Ray Ihnen einen end-to-end Überblick über eine gesamte Anfrage gibt, können Sie sehen, was Ihr Amazon SNS-Thema aufruft und was nachgelagert zu den Abonnements Ihres Themas ist. Sie können die Latenzen in Ihren Nachrichten und deren Backend-Services analysieren (z. B., wie lange eine Anfrage in einem Thema verweilt, und wie lange es gedauert hat, bis die Nachricht an die einzelnen Abonnements des Themas zugestellt wurde).

> **Important**
>
> Amazon-SNS-Themen mit zahlreichen Abonnements erreichen möglicherweise eine Größenbeschränkung und können nicht vollständig verfolgt werden. Informationen zu den Größenbeschränkungen für die Nachverfolgung von Dokumenten finden Sie unter Servicekontingente für X-Ray in der Allgemeinen AWS-Referenz.

Wenn Sie eine Amazon-SNS-API von einem Service aufrufen, der bereits nachverfolgt wird, führt Amazon SNS die Nachverfolgung auch dann durch, wenn X-Ray-Tracing auf der API nicht aktiviert ist.

Amazon SNS unterstützt X-Ray-Tracing für Standard- und FIFO-Themen. Sie können X-Ray für ein Amazon-SNS-Thema aktivieren, indem Sie die Amazon-SNS-Konsole, die Amazon-SNS–API `SetTopicAttributes`, die CLI-Referenz zu Amazon Simple Notification Service oder AWS CloudFormation verwenden.

Weitere Informationen zur Verwendung von Amazon SNS mit X-Ray finden Sie unter Amazon SNS und AWS X-Ray im AWS X-Ray-Entwicklerhandbuch.

Themen

Berechtigungen für aktives Tracing

Wenn Sie die Amazon-SNS-Konsole verwenden, versucht Amazon SNS, die erforderlichen Berechtigungen für das Amazon-SNS-Thema zum Aufrufen von X-Ray zu erstellen. Der Versuch kann abgelehnt werden, wenn Sie nicht über ausreichende Berechtigungen verfügen, um die Amazon-SNS-Konsole zu verwenden. Weitere Informationen finden Sie unter Identity and Access Management in Amazon SNS und Beispiele für die Zugriffskontrolle in Amazon SNS.

Wenn Sie die CLI verwenden, müssen Sie die Berechtigungen manuell konfigurieren. Diese Berechtigungen werden mithilfe von Ressourcenrichtlinien konfiguriert. Weitere Informationen zur Verwendung der erforderlichen Berechtigungen in X-Ray finden Sie unter Amazon SNS und AWS X-Ray.

Aktivieren des aktiven Tracings für ein Amazon-SNS-Thema (Konsole)

Wenn das aktive Tracing für ein Amazon-SNS-Thema aktiviert ist, liest es die Trace-ID, sendet die Daten anhand der Trace-ID an den Kunden und gibt die Trace-ID an nachgelagerte Services weiter.

1. Melden Sie sich bei der Amazon-SNS-Konsole an.
2. Wählen Sie ein Thema aus oder erstellen Sie ein neues Thema. Weitere Informationen zum Erstellen von Themen finden Sie unter Erstellen eines Amazon-SNS-Themas.
3. Wählen Sie auf der Seite Thema erstellen im Abschnitt Details einen Thementyp aus: FIFO oder Standard.
   a. Geben Sie den Namen des neuen Themas ein.
   b. (Optional) Geben Sie einen Anzeigenamen für Ihr Thema ein.
4. Erweitern Sie Aktives Tracing und wählen Sie Aktives Tracing verwenden.

Sobald Sie X-Ray für Ihr Amazon SNS-Thema aktiviert haben, können Sie die X-Ray-Service-Übersicht verwenden, um die end-to-end Ablaufverfolgungen und Service-Übersichten für das Thema anzuzeigen.

Aktivieren des aktiven Tracings für ein Amazon-SNS-Thema (AWS-SDK)

Im folgenden Codebeispiel wird veranschaulicht, wie Sie mithilfe des AWS-SDK für Java das aktive Tracing für ein Amazon-SNS-Thema aktivieren.

```
public static void enableActiveTracing(SnsClient snsClient, String topicArn) {

        try {

            SetTopicAttributesRequest request = SetTopicAttributesRequest.builder()
                .attributeName("TracingConfig")
                .attributeValue("Active")
                .topicArn(topicArn)
                .build();

            SetTopicAttributesResponse result = snsClient.setTopicAttributes(request);
            System.out.println("\n\nStatus was " +
 result.sdkHttpResponse().statusCode() + "\n\nTopic " + request.topicArn()
                + " updated " + request.attributeName() + " to " +
 request.attributeValue());
```

```
        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }
    }
```

Aktivieren des aktiven Tracings für ein Amazon-SNS-Thema (AWS-CLI)

Im folgenden Codebeispiel wird veranschaulicht, wie Sie mit der AWS-CLI das aktive Tracing für ein Amazon-SNS-Thema aktivieren.

```
aws sns set-topic-attributes \
    --topic-arn arn:aws:sns:us-west-2:123456789012:MyTopic \
    --attribute-name TracingConfig \
    --attribute-value Active
```

Aktivieren des aktiven Tracings für ein Amazon-SNS-Thema (AWS CloudFormation)

Im folgenden AWS CloudFormation-Stack wird veranschaulicht, wie Sie das aktive Tracing für ein Amazon-SNS-Thema aktivieren.

```
AWSTemplateFormatVersion: 2010-09-09
Resources:
  MyTopicResource:
    Type: 'AWS::SNS::Topic'
    Properties:
      TopicName: 'MyTopic'
      TracingConfig: 'Active'
```

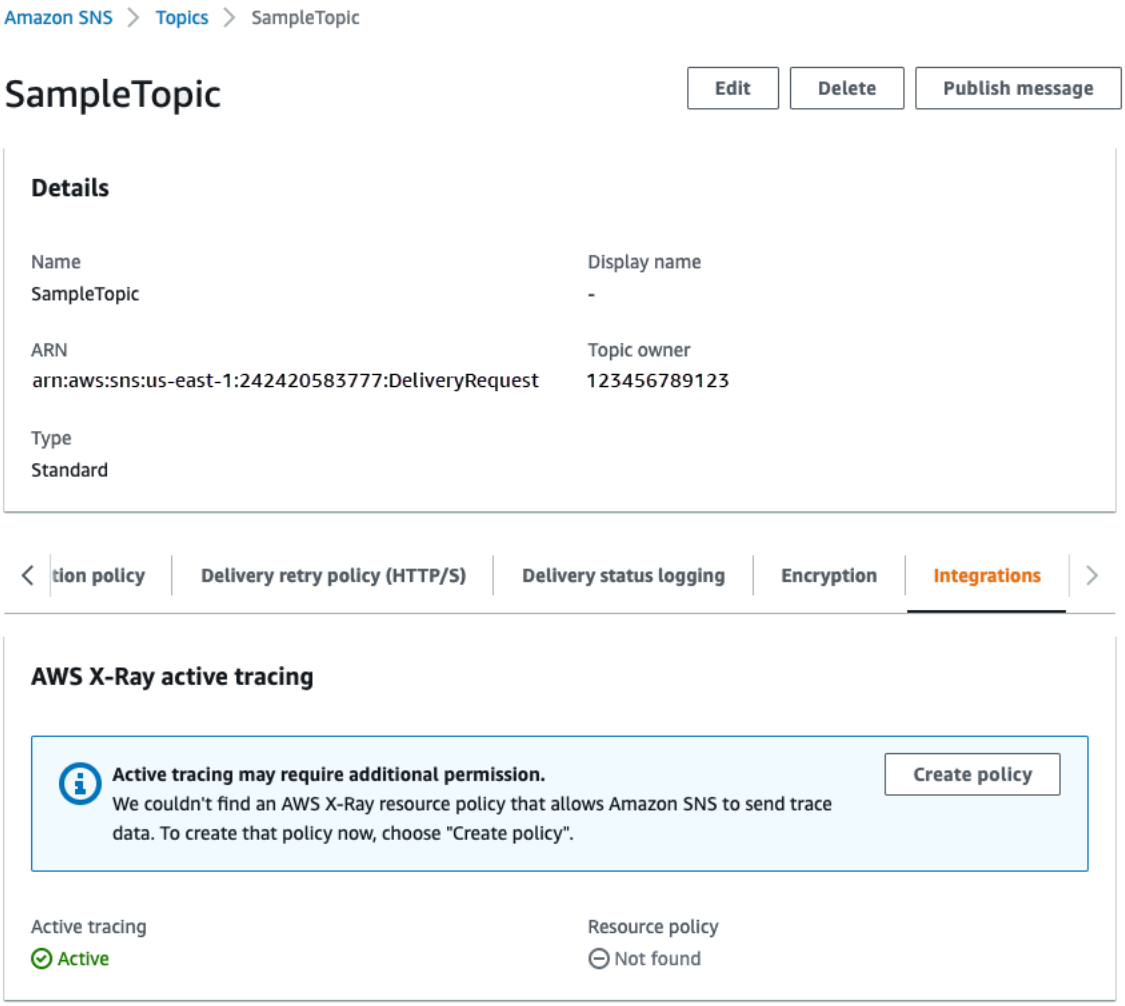
Überprüfen, ob das aktive Tracing für Ihr Thema aktiviert ist

Sie können die Amazon-SNS-Konsole verwenden, um zu überprüfen, ob das aktive Tracing für Ihr Thema aktiviert ist oder ob die Ressourcenrichtlinie nicht hinzugefügt werden konnte.

1. Melden Sie sich bei der Amazon-SNS-Konsole an.
2. Wählen Sie im linken Navigationsbereich Topics (Themen).
3. Wählen Sie auf der Seite Themen ein Thema aus.
4. Wählen Sie die Registerkarte Integrationen aus.

Wenn das aktive Tracing aktiviert ist, wird ein grünes Aktiv-Symbol angezeigt.

5. Wenn Sie das aktive Tracing aktiviert haben und nicht sehen, dass die Ressourcenrichtlinie hinzugefügt wurde, wählen Sie Richtlinie erstellen aus, um die zusätzlichen erforderlichen Berechtigungen hinzuzufügen.



Aktives Tracing testen

1. Melden Sie sich bei der Amazon-SNS-Konsole an.
2. Erstellen Sie ein Amazon-SNS-Thema. Weitere detaillierte Informationen hierzu finden Sie unter Um ein Thema mit dem zu erstellen AWS Management Console.
3. Erweitern Sie Aktives Tracing und wählen Sie Aktives Tracing verwenden.

4. Veröffentlichen Sie eine Nachricht im Amazon-SNS-Thema. Weitere detaillierte Informationen hierzu finden Sie unter Um Nachrichten in Amazon-SNS-Themen mithilfe von AWS Management Console zu veröffentlichen.
5. Verwenden Sie die X-Ray-Service-Übersicht, um die end-to-end Ablaufverfolgungen und Service-Übersichten für das Thema anzuzeigen.

Dokumentationsverlauf

In der folgenden Tabelle werden die letzten Änderungen am Amazon Simple Notification Service-Entwicklerhandbuch beschrieben.

Servicefunktionen werden manchmal schrittweise in den AWS Regionen eingeführt, in denen ein Dienst verfügbar ist. Wir aktualisieren diese Dokumentation nur für die erste Version. Wir stellen keine Informationen über die Verfügbarkeit von Regionen zur Verfügung und kündigen auch keine späteren Rollouts von Regionen an. Informationen zur regionalen Verfügbarkeit von Servicefunktionen und zum Abonnieren von Benachrichtigungen über Updates finden Sie unter Was gibt's Neues bei AWS? .

Änderung	Beschreibung	Datum
Unterstützung von Canada West (Calgary) für FIFO-Themen	Amazon SNS unterstützt das FIFO-Thema in Canada West (Calgary).	28. März 2024
Amazon SNS SMS-Unterstützung in fünf neuen Regionen	Amazon SNS hat SMS-Unterstützung für die folgenden Regionen hinzugefügt: Asien-Pazifik (Hyderabad), Asien-Pazifik (Melbourne), Naher Osten (VAE), Europa (Zürich). und Europa (Spanien).	8. Februar 2024
Unterstützung für Firebase Cloud Messaging (FCM) HTTP v1	Amazon SNS unterstützt FCM v1-Anmeldeinformationen.	18. Januar 2024
Amazon SNS-SMS werden in Asien-Pazifik (Jakarta) unterstützt	Amazon-SNS unterstützt SMS-Nachrichten in Asien-Pazifik (Jakarta)	14. Dezember 2023
AWS CloudFormation Unterstützung für die Konfiguration von `DeliveryS`	AWS CloudFormation Support steht für die Konfiguration `DeliveryStatusLogging` bei der Erstellung oder	07. Dezember 2023

tatusLogging Amazon SNS SNS-Themen	Aktualisierung von Amazon SNS SNS-Themen zur Verfügung.	
Neue Nachrichtenfilter-Operatoren hinzugefügt	Sie können jetzt beim Filtern von Amazon-SNS-Nachrichten Suffix-Abgleich, Gleich-Groß- und Kleinschreibung ignorieren und OR-Operatoren verwenden.	16. November 2023
Support für Nachrichtenarchivierung und -wiederholung hinzugefügt	Themeninhaber können Nachrichten bis zu 365 Tage lang zu einem Thema archivieren. Themen-Subscriber können die archivierten Nachrichten an einem abonnierten Endpunkt wiederholen, um Nachrichten wiederherzustellen, die auf einen Fehler in einer nachgelagerten Anwendung zurückzuführen sind, oder um den Status einer vorhandenen Anwendung zu replizieren.	26. Oktober 2023
Support für das Abonnieren einer Standard-Warteschlange für ein FIFO-Thema hinzugefügt	Sie können eine Amazon-SQS-FIFO-Warteschlange oder eine Standard-Warteschlange für ein Amazon-SNS-FIFO-Thema abonnieren. Nur Amazon-SQS-FIFO-Warteschlangen garantieren, dass Nachrichten in der richtigen Reihenfolge und ohne Duplikate empfangen werden.	14. September 2023

SMS-Unterstützung für Israel (Tel Aviv) hinzugefügt	Amazon-SNS-SMS wird jetzt in der Region Israel (Tel Aviv) unterstützt.	28. August 2023
Support für aktives X-Ray-Tracing zu FIFO-Themen hinzugefügt	Bisher nur mit Amazon SNS SNS-Standardthemen unterstützt, verfolgt und analysiert AWS X-Ray jetzt Benutzeranfragen auf ihrem Weg durch Ihre FIFO-Themen zu Ihren Amazon Data Firehose- AWS Lambda, Amazon SQS- und HTTP/S-Endpunktabonnements.	31. Mai 2023
Verbesserte Unterstützung für Content-Type-Header	Sie können in der Anforderungsrichtlinie den Content-Type-Header festlegen, um den Medientyp der Benachrichtigung anzugeben.	23. März 2023
Aktive Tracing-Unterstützung hinzugefügt	AWS X-Ray verfolgt und analysiert Benutzeranfragen auf ihrem Weg durch Ihre Amazon SNS SNS-Standardthemen zu Ihren Amazon Data Firehose- AWS Lambda, Amazon SQS- und HTTP/S-Endpunktabonnements.	8. Februar 2023
Registrierung einer Sender-ID für Singapur	Anweisungen für die Registrierung von Sender-IDs in Singapur wurden hinzugefügt.	10. Januar 2023

Nutzlastbasierte Nachricht enfilterung	Mit der nutzlastbasierten Filterung können Sie Nachricht en anhand der Nachricht ennutzlast filtern und so die Kosten vermeiden, die mit der Verarbeitung unerwünschter Daten verbunden sind.	22. November 2022
SHA256-Hash-Algorithmus für die Amazon-SNS-Nachric htensignatur hinzugefügt	Unterstützung für den SHA256-Hash-Algorithmus bei Verwendung der Amazon-SN S-Nachrichtensignatur wurde hinzugefügt.	15. September 2022
Zusätzliche Regionen zum SMS-Messaging hinzugefügt	Amazon SNS unterstützt SMS-Nachrichten in den folgenden Regionen: Afrika (Kapstadt), Asien-Pazifik (Osaka), Europa (Mailand) und AWS GovCloud (USA-Ost).	09. September 2022
Unterstützung für den Nachrichtendatenschutz hinzugefügt	Der Nachrichtendatensc hutz schützt die Daten, die in Ihren Amazon SNS SNS-Themen veröffentlicht werden, indem die vertraulichen Informationen, die zwischen Anwendungen oder AWS Diensten übertragen werden, mithilfe von Datenschutzrichtli nien geprüft und blockiert werden.	8. September 2022

Neues Registrierungsverfahren für gebührenfreie Nummern	Support für das Senden von Amazon-SNS-Nachrichten über gebührenfreie Telefonnummern (TFN) an Empfänger in den Vereinigten Staaten hinzugefügt.	1. August 2022
Unterstützung für attributbasierte Zugriffssteuerungen (ABAC)	Zusätzliche Unterstützung für attributbasierte Zugriffssteuerung (ABAC) für API-Aktionen, einschließlich `Publish` und `PublishBatch` . ABAC ist eine Autorisierungsstrategie, die Zugriffsberechtigungen auf der Grundlage von Tags definiert, die an IAM-Ressourcen wie IAM-Benutzer und -Rollen und an Ressourcen wie Amazon SNS AWS SNS-Themen angehängt werden können, um die Rechteverwaltung zu vereinfachen.	10. Januar 2022
Support für Apple-Token-basierte Authentifizierung für Push-Benachrichtigungen	Sie können Amazon SNS ermächtigen, Push-Benachrichtigungen an Ihre iOS- oder macOS-App zu senden, indem Sie Informationen bereitstellen, die Sie als Entwickler der App identifizieren.	28. Oktober 2021

Neue Absender von SMS-Nachrichten werden in der SMS-Sandbox platziert	Um Betrug und Missbrauch zu vermeiden und Ihren Ruf als Absender zu schützen, ist die SMS-Sandbox vorhanden . Solange sich Ihr AWS Konto in der SMS-Sandbox befindet, können Sie SMS-Nachrichten nur an verifizierte Zieltelef onnummern senden.	1. Juni 2021
Neue Absender von SMS-Nachrichten werden in der SMS-Sandbox platziert	Um Betrug und Missbrauch zu vermeiden und Ihren Ruf als Absender zu schützen, ist die SMS-Sandbox vorhanden . Solange sich Ihr AWS Konto in der SMS-Sandbox befindet, können Sie SMS-Nachrichten nur an verifizierte Zieltelef onnummern senden.	1. Juni 2021
Neue Attribute für das Senden von SMS-Nachrichten an Empfänger in Indien	Zwei neue Attribute Entity-ID und ID der Vorlage um SMS-Nachrichten an Empfänger in Indien zu senden, sind jetzt erforderlich.	22. April 2021
Aktualisierungen von Nachrichtenfilter-Operatoren	Ein neuer Operator, `cidr`, ist für die übereinstimmenden IP-Adressen und Subnetze der Nachrichtenquelle verfügbar . Sie können nun auch auf das Fehlen eines Attributs chlüssels prüfen und ein Präfix mit dem `anything-but` - Operator für Attribut-Zeichenfo lgenabgleich.	7. April 2021

Beenden der Unterstützung für P2P-LongCodes für US-Destinationen	Ab dem 1. Juni 2021 unterstützen die US-amerikanischen Telekommunikationsanbieter die Verwendung von person-to-person (P2P) -Langcodes für application-to-person (A2P) -Verbindungen zu Zielen in den USA nicht mehr. Stattdessen können Sie Kurzcodes, gebührenfreie Nummern oder eine neue Art von Ursprungsnummer, 10DLC genannt, benutzen.	16. Februar 2021
Support für einminütige Amazon-Metriken CloudWatch	Die CloudWatch 1-Minuten -Metrik für Amazon SNS ist jetzt in allen AWS Regionen verfügbar.	28. Januar 2021
Support für Amazon Data Firehose-Endpunkte	Sie können Firehose-Lieferstreams für SNS-Themen abonnieren. Auf diese Weise können Sie Benachrichtigungen an Archivierungs- und Analyseendpunkte wie Amazon Simple Storage Service (Amazon S3) -Buckets, Amazon Redshift Redshift-Tabellen, Amazon OpenSearch Service (OpenSearch Service) und mehr senden.	12. Januar 2021
Ursprungsnummern sind verfügbar	Beim Senden von Textnachrichten (SMS) können Sie Originalnummern verwenden.	23. Oktober 2020

Support für Amazon-SNS-FIFO-Themen	Um verteilte Anwendungen, die Datenkonsistenz in nahezu Echtzeit erfordern, zu integrieren, können Sie Amazon SNS First-In-First-Out-Themen (FIFO) in Amazon SQS FIFO-Queues verwenden.	22. Oktober 2020
Die Amazon SNS Extended Client Library für Java ist verfügbar	Sie können diese Bibliothek verwenden, um große Amazon SNS-Nachrichten zu veröffentlichen.	25. August 2020
SSE ist in den China-Regionen verfügbar	Serverseitige Verschlüsselung (SSE) für Amazon SNS ist in den chinesischen Regionen verfügbar.	20. Januar 2020
Support für die Verwendung von DLQs zum Erfassen nicht zustellbarer Nachrichten	Sie können eine Amazon SQS-Queue für unzustellbare Nachrichten mit einem Amazon SNS-Abonnement verwenden, um unzustellbare Nachrichten zu erfassen.	14. November 2019
Support für die Angabe benutzerdefinierter APNs-Header-Werte	Sie können einen benutzerdefinierten APN-Header-Wert angeben.	18. Oktober 2019
Support für das „`apns-push-type` “-Header-Feld für APNs	Sie können das `apns-push-type` -Header-Feld für mobile Benachrichtigungen verwenden, die über APNs gesendet werden.	10. September 2019

Support bei der Problembehandlung mit AWS X-Ray	Sie können mithilfe von X-Ray Fehler bei Nachrichten beheben, die über SNS-Themen weitergeleitet werden.	24. Juli 2019
Support für die Attribut-Schlüsselzuordnung mit dem „`exists`“-Operator	Sie können mithilfe des `exists`-Operators überprüfen, ob eine eingehende Nachricht über ein Attribut verfügt, dessen Schlüssel in der Filterrichtlinie aufgeführt ist.	5. Juli 2019
Unterstützung für den „Alles außer“-Abgleich mehrerer numerischer Werte	Zusätzlich zu mehreren Zeichenfolgen erlaubt Amazon SNS den „Alles außer“-Abgleich mehrerer numerischer Werte.	5. Juli 2019
Amazon-SNS-Versionshinweise sind als RSS-Feed verfügbar	Nach dem Titel auf dieser Seite (Dokumentationsverlauf), wählen Sie RSS-FEED aus.	22. Juni 2019

AWS-Glossar

Die neueste AWS-Terminologie finden Sie im AWS-Glossar in der AWS-Glossar-Referenz.

Die vorliegende Übersetzung wurde maschinell erstellt. Im Falle eines Konflikts oder eines Widerspruchs zwischen dieser übersetzten Fassung und der englischen Fassung (einschließlich infolge von Verzögerungen bei der Übersetzung) ist die englische Fassung maßgeblich.