aws

開發人員指南

Amazon Elastic Container Service

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

Amazon Elastic Container Service: 開發人員指南

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

Amazon 的商標和商業外觀不得用於任何非 Amazon 的產品或服務，也不能以任何可能造成客戶混淆、任何貶低或使 Amazon 名譽受損的方式使用 Amazon 的商標和商業外觀。所有其他非 Amazon 擁有的商標均為其各自擁有者的財產，這些擁有者可能附屬於 Amazon，或與 Amazon 有合作關係，亦或受到 Amazon 贊助。

Table of Contents

什麼是 Amazon ECS？ 1
Amazon ECS 術語和元件 1
Amazon ECS 容量 2
Amazon ECS 控制器 3
Amazon ECS 佈建 3
應用程式生命週期 3
相關資訊 5
開始使用 7
設定 7
註冊一個 AWS 帳戶 7
建立具有管理權限的使用者 8
建立 Virtual Private Cloud 9
建立安全群組 10
建立憑證來連線至 EC2 執行個體 12
安裝 AWS CLI 13
建立容器映像 13
必要條件 13
建立 Docker 映像 15
推送映像至 Amazon Elastic Container Registry 17
清除 18
後續步驟 19
瞭解如何為 Fargate 啟動類型建立 Linux 工作 19
必要條件 19
步驟 1：建立叢集 20
步驟 2：建立任務定義 21
步驟 3：建立服務 22
步驟 4：檢視服務 22
步驟 5：清除 23
了解如何為 Fargate 啟動類型建立 Windows 工作 23
必要條件 23
步驟 1：建立叢集 24
步驟 2：註冊 Windows 任務定義 25
步驟 3：以您的任務定義建立服務 26
步驟 4：檢視服務 26

步驟 5：清除 27
了解如何為 EC2 啟動類型建立視窗工作 28
必要條件 28
步驟 1：建立叢集 28
步驟 2：註冊任務定義 30
步驟 3：建立服務 31
步驟 4：檢視服務 32
步驟 5：清除 32
開發人員工具概觀 34
AWS Management Console 34
AWS Command Line Interface 35
AWS CloudFormation 35
AWS 副駕駛 CLI 35
AWS CDK 36
AWS 容器 36
Amazon ECS CLI 37
Docker Desktop 與 Amazon ECS 整合 37
AWS 開發套件 37
Summary 38
使用 AWS 副駕駛 CLI 建立資源 38
安裝 AWS 副駕駛 CLI 39
使用副駕駛 CLI 部署範例 Amazon ECS AWS 應用程式 47
使用 AWS CDK 48
步驟 1：設定您的 AWS CDK 專案 49
步驟 2：使用在 AWS CDK Fargate 上定義容器化網頁伺服器 52
步驟 3：測試 Web 伺服器 59
步驟 4：清理 59
後續步驟 59
使用建立資源 AWS CloudFormation 60
AWS CloudFormation 模板 60
範例範本 61
使用 AWS CLI 從範本建立資源 68
進一步了解 AWS CloudFormation 68
開始使用 Amazon ECS CLI 69
安裝 Amazon ECS CLI 69
設定 Amazon ECS CLI 77

AWS Fargate 80
逐步解說 80
容量提供者 80
任務定義 81
平台版本 81
服務負載平衡 81
用量指標 82
何時使用 Fargate 啟動類型的安全性暗示 82
Fargate 安全最佳實踐 82
用 AWS KMS 於加密 Fargate 的臨時存儲 82
SYS_PSTRACE 功能，用於使用 Fargate 進行內核系統調用跟踪 83
使用 Amazon GuardDuty 與 Fargate 運行時監控 83
Fargate 安全考量 84
適用於 Amazon ECS 的 Fargate Linux 平台版本 84
考量事項 85
1.4.0 85
1.3.0 87
移轉至平台版本 1.4.0 88
平台版本取代 88
Fargate 容器映像提取行為上的 Linux 容器 90
Amazon ECS 的 Fargate 視窗平台版本 91
平台版本考量事項 91
1.0.0 92
Amazon ECS Fargate 上的視窗容器注意事項 92
Fargate 容器映像提取行為上的 Windows 容器 93
適用於 Amazon ECS 的 Fargate 任務暫時儲存 94
Fargate Linux 容器平台版本 94
Fargate Windows 容器平台版本 95
用於 AWS Fargate 暫時儲存的客戶管理金鑰 95
AWS Amazon ECS 上的 Fargate 任務維護常見問題解答 107
什麼是 Fargate 任務維護和退休？ 107
什麼是在任務退休通知？ 108
我可以更改任務的退休等待時間嗎？ 111
我可以透過其他 AWS 服務取得任務退休通知嗎？ 111
我可以在排程之後變更任務退休嗎？ 111
我可以控制任務替換的時間嗎？ 111

Amazon ECS 如何處理屬於服務一部分的任務？ 112
Amazon ECS 能否自動處理獨立任務？ 112
AWS Fargate 地區 112
AWS Fargate 上的 Linux 容器 112
AWS Fargate 上的視窗容器 114
為 Amazon ECS 建構您的解決方案 117
容量 117
聯網 117
功能存取 118
IAM 角色 119
日誌 119
啟動類型 119
Fargate 119
EC2 122
外部 123
共用子網路、Local Zones 和 Wavelength 區域中的應用 123
共用子網路 124
本機區域 125
Wavelength 區域 125
Amazon 彈性容器服務 AWS Outposts 126
考量事項 126
必要條件 126
在上建立叢集 AWS Outposts 127
最佳化容量與可用性 129
最大化縮放速度 130
處理需求衝擊 131
將應用程序 Connect 到互聯網 132
公共子網和網際網路閘道 133
私有子網路和 NAT 閘道 135
從網際網路接收 Amazon ECS 輸入連線的最佳實務 136
Application Load Balancer 136
Network Load Balancer 137
Amazon API Gateway 139
透過帳戶設定存取功能 140
Amazon Resource Name (ARN) 和 ID 144
ARN 和資源 ID 格式時間表 146

AWS Fargate 符合聯邦資訊處理標準 (FIPS-140) 規範 146
標記授權 146
標記授權時間表 147
AWS Fargate 任務退休等待時間 148
運行時監控（Amazon GuardDuty 集成） 149
使用主控台檢視帳戶設定 150
修改帳戶設定 150
還原為預設的帳戶設定 151
使用管理帳戶設定 AWS CLI 151
Amazon ECS 的 IAM 角色 153
任務定義 156
任務定義狀態 157
可以封鎖刪除的 Amazon ECS 資源 158
架構您的應用 158
容器映像的最佳做法 159
工作大小的最佳做法 161
網路安全性最佳做法 162
EC2 啟動類型的任務聯網 166
適用於 Fargate 啟動類型的工作網路 175
工作的儲存選項 179
管理容器交換記憶體空間 252
Fargate 啟動類型的任務定義差異 253
執行 Windows 的 EC2 執行個體的任務定義差異 260
使用主控台建立任務定義 261
JSON 驗證 261
AWS CloudFormation 堆疊 261
程序 262
使用主控台更新任務定義 283
JSON 驗證 284
程序 284
使用主控台取消註冊任務定義修訂版 285
AWS CloudFormation 堆疊 261
程序 286
使用主控台刪除任務定義修訂版 286
可以封鎖刪除的 Amazon ECS 資源 158
程序 287

任務定義使用案例 287
GPU 工作負載的工作定義 288
視訊轉碼工作負載的任務定義 296
AWS 神經元機器學習工作負載的任務定義 308
深度學習執行個體的工作定義 317
64 位元 ARM 工作負載的工作定義 319
將記錄檔傳送至 CloudWatch 321
將日誌發送到 AWS 服務或 AWS Partner 324
使用非AWS 容器映像 336
將單個環境變量傳遞給容器 338
將環境變量傳遞給容器 339
將敏感數據傳遞到容器 342
任務定義參數 363
系列 363
啟動類型 364
任務角色 364
任務執行角色 364
網路模式 365
執行時間平台 366
任務大小 367
容器定義 371
Elastic Inference 加速器名稱 412
任務置放限制條件 413
代理組態 413
磁碟區 415
標籤 422
其他任務定義參數 422
任務定義範本 425
任務定義範例 436
网络服务器 437
splunk日誌驅動 439
fluentd日誌驅動 439
gelf日誌驅動 440
外部執行個體上的工 441
Amazon ECR 映像和任務定義 IAM 角色 442
具有指令的入口點 443

容器相依性 443
Windows 任務定義範例 445
叢集 447
針對 Fargate 啟動類型的叢集 448
Fargate 現貨終止通知 449
建立 Fargate 啟動類型的叢集 451
EC2 啟動類型的容量提供者 453
EC2 容器執行個體安全 454
為 Amazon EC2 啟動類型建立叢集 455
叢集自動擴展 459
Amazon EC2 容器實例 483
外部啟動類型的叢集 619
支援的作業系統和系統架構 619
考量事項 620
建立外部啟動類型的叢集 623
將外部執行個體註冊到 Amazon ECS 叢集 625
取消註冊外部執行個體 630
更新 AWS Systems Manager 代理程式和 Amazon ECS 容器代理程式 635
更新叢集 640
刪除叢集 641
建立容量提供者 642
更新容量提供者 642
刪除容量提供者 643
取消註冊容器執行個體 644
程序 645
容器執行個體耗盡 645
服務的耗盡行為 645
獨立任務的耗盡行為 646
程序 646
容器代理 647
生命週期 647
Amazon ECS 最佳化 AMI 648
其他資訊 648
容器代理程式組態 649
安裝 Amazon ECS 容器代理程式 651
容器代理程式記錄檔組態 657

為私有 Docker 映像檔設定容器執行個體 660
清理工作和影像 664
排程您的容器 667
運算選項 668
任務生命週期 669
生命週期狀態 670
Amazon ECS 如何在容器執行個體上放置任務 671
EC2 啟動類型 672
Fargate 啟動類型 673
使用策略定義任務放置 673
分組相關任務 677
定義用於工作的容器執行個體 678
獨立工作 687
工作流程 687
優化任務啟動時間 688
將應用程式作為工作執行 689
使用 Amazon EventBridge 排程器來排程任務 697
停止工作 702
服務 703
守護進程策 704
複本策略 705
服務參數的最佳做法 706
建立服務 709
更新服務 729
更新藍/綠部署 742
刪除服務 743
滾動更新部署 744
藍/綠部署 750
外部部署 769
使用負載平衡來分配服務流量 775
服務自動擴展 787
互連服務 796
任務縮減保護 842
服務調節邏輯 849
服務定義參數 850
標記資源 880

如何標記資源 880
在建立期間標記資源 883
限制 883
Amazon ECS 受管標籤 884
使用標籤進行帳單 885
將標籤新增至資源 885
將標籤新增至容器執行個體 887
外部容器實例 888
用量報告 889
任務層級成本和用量 890
監控 892
監控 Amazon ECS 的最佳實務 892
監控工具 893
自動化工具 893
手動工具 894
使用監控 Amazon ECS CloudWatch 895
考量事項 895
建議的指標 896
檢視 Amazon ECS 指標 896
Amazon ECS 指標 CloudWatch 898
AWS Fargate 使用量度 906
Amazon ECS 叢集保留指標 907
Amazon ECS 叢集使用率指標 908
Amazon ECS 服務使用率指標 910
使用以下方式自動回應 Amazon ECS 錯誤 EventBridge 912
Amazon ECS 事件 913
處理事件 930
使用容器洞察來監控 Amazon ECS 容器 933
考量事項 934
設定 Amazon ECS 的 CloudWatch 容器洞察 934
檢視 Amazon ECS 生命週期事件的 CloudWatch 容器洞察所需的許可 935
使用容器健康狀態檢查確定工作健 937
如何確定任務健康狀態 938
Health 狀態檢查和代理程式中斷 939
檢視容器健康狀 939
監控 Amazon ECS 容器執行個體健康 939

相關主題 940
使用應用程式追蹤資料識別 Amazon ECS 最佳化機會 940
AWS 發行版所需的 IAM 許可以 OpenTelemetry 與之整合 AWS X-Ray 941
指定要 AWS X-Ray 整合到您的任務定 AWS 義中的 OpenTelemetry 附屬發行版 942
使用應用程式指標關聯 Amazon ECS 應用程式效能 944
將應用程式指標匯出到 Amazon CloudWatch 944
將應用程式指標匯出至 Amazon Managed Service for Prometheus 948
使用記錄 Amazon ECS API 呼叫 AWS CloudTrail 951
Amazon ECS 信息 CloudTrail 952
了解 Amazon ECS 日誌檔案項目 952
使用中繼資料監控工 954
容器中繼資料檔案 955
適用於 EC2 上 Amazon ECS 任務的任務中繼資料 960
可用於 Fargate 上任務的任務中繼資料 1000
容器內省 1021
使用執行階段監控識別未經授 1024
運行時監控如何與 Amazon ECS 一起工作 1024
考量事項 1025
資源使用率 1026
Fargate 工作負載的執行階段監控 1026
EC2 工作負載的執行期監控 1030
疑難排解常見問答集 1035
使用 ECS 執行監控 Amazon ECS 容器 1039
考量事項 1039
必要條件 1041
架構 1042
使用 ECS Exec 1042
使用 ECS 執行程式進行記錄 1044
使用 IAM 政策限制 ECS Exec 的存取 1048
Compute Optimizer 建議 1051
針對 Fargate 任務大小提出的建議 1052
故障診斷 1053
解決停止的工作錯誤 1055
停止的工作錯誤訊息更新 1056
檢視停止的工作錯誤 1058
已停止任務的錯誤代碼 1059

驗證工作連線性 1078
檢視 IAM 角色請求 1082
檢視服務事件訊息 1083
Amazon ECS 服務事件消息 1084
疑難排解 Amazon ECS 中的服務負載平衡器 1092
疑難排解 Amazon ECS 中的服務 auto 擴展 1094
疑難排解工作定義無效 CPU 或記憶體錯誤 1094
檢視容器代理記錄檔 1096
使用 Amazon ECS 日誌收集器收集容器日誌 1097
代理程式內省 1099
Amazon ECS 中的碼頭診斷 1101
列出 Amazon ECS 中的碼頭集裝箱 1101
在 Amazon ECS 中查看碼頭日誌 1102
在 Amazon ECS 中檢查碼頭集裝箱 1103
在 Amazon ECS 中從 Docker 守護進程配置詳細輸出 1104
Amazon ECS 中的泊塢視API error (500): devmapper窗疑難排解 1105
疑難排解 ECS 執行程式問題 1107
使用執行檢查程式進行驗證 1107
呼叫 execute-command 時發生錯誤 1107
疑難排解 Amazon ECS Anywhere 可見 1107
外部執行個體註冊問題 1108
外部執行個體網路問題 1108
執行任務的問題 1108
AWS Fargate 節流配額 1109
在 Fargate 中節流 RunTask API 1109
調整 Fargate 中的費率配額 1110
處理節流問題 1110
同步節流 1110
Amazon ECS 中的異步節流 1110
監控節流 1111
用 CloudWatch 於監控節流 1112
API 故障原因 1112
安全 1119
身分和存取權管理 1119
物件 1120
使用身分驗證 1120

使用政策管理存取權 1123
Amazon Elastic Container Service 如何與 IAM 搭配使用 1125
身分型政策範例 1134
AWS Amazon ECS 的受管政策 1145
使用服務連結角色 1174
Amazon ECS 的 IAM 角色 1177
Amazon ECS 主控台必要的許可 1224
Amazon ECS 服務 auto 擴展所需的 IAM 許可 1230
在建立期間標記資源 1231
故障診斷 1235
IAM 最佳實務 1237
記錄和監控 1239
法規遵循驗證 1241
合規性與安全性最佳做法 1242
AWS Fargate 符合 FIPS-140 規範 1243
AWS Fargate FIPS-140 考量事項 1244
在 Fargate 上使用 FIPS 1244
用 CloudTrail 於 Fargate FIPS-140 稽核 1244
基礎設施安全性 1246
介面 VPC 端點 (AWS PrivateLink) 1246
工作和容器安全性最佳做法 1251
建立最小或使用 distroless 映像 1252
掃描您的映像是否有漏洞 1253
從映像移除特殊權限 1253
建立一組精選映像 1253
掃描應用程式套件和程式庫是否有漏洞 1253
執行靜態程式碼分析 1254
以非根使用者身分執行容器 1254
使用唯讀的根檔案系統 1254
使用 CPU 和記憶體限制來設定任務 (Amazon EC2) 1255
不可變標籤與 Amazon ECR 一起使用 1255
避免以特權方式執行容器 (Amazon EC2) 1255
從容器中移除不必要的 Linux 功能 1256
使用客戶自管金鑰 (CMK) 來加密推送至 Amazon ECR 的映像 1256
教學課程 1257
使用以下項目建立 Fargate 啟動類型的 Linux 工作 AWS CLI 1258

必要條件 1259
步驟 1：建立叢集 1260
步驟 2：註冊 Linux 任務定義 1261
步驟 3：列出任務定義 1262
步驟 4：建立服務 1262
步驟 5：列出服務 1263
步驟 6：描述執行中的服務 1263
步驟 7：測試 1266
步驟 8：清除 1269
Fargate 用 AWS CLI 1269
必要條件 1270
步驟 1：建立叢集 1270
步驟 2：註冊 Windows 任務定義 1271
步驟 3：列出任務定義 1272
步驟 4：建立服務 1273
步驟 5：列出服務 1273
步驟 6：描述執行中的服務 1274
步驟 7：清除 1276
建立 EC2 啟動類型的工作 AWS CLI 1276
必要條件 1277
步驟 1：建立叢集 1277
步驟 2：使用 Amazon ECS AMI 啟動執行個體 1278
步驟 3：列出容器執行個體 1278
步驟 4：描述您的容器執行個體 1278
步驟 5：註冊任務定義 1281
步驟 6：列出任務定義 1283
步驟 7：執行任務 1283
步驟 8：列出任務 1284
步驟 9：描述執行中的任務 1285
設定 Amazon ECS 以監聽 CloudWatch 事件事件 1286
先決條件：設定測試叢集 1286
步驟 1：建立 Lambda 函數 1286
步驟 2：註冊事件規則 1287
步驟 3：建立任務定義 1287
步驟 4：測試您的規則 1289
針對任務停止事件傳送 Amazon 簡單通知服務警示 1289

先決條件：設定測試叢集 1289
先決條件：設定 Amazon SNS 的許可 1289
步驟 1：建立並訂閱 Amazon SNS 主題 1290
步驟 2：註冊事件規則 1290
步驟 3：測試您的規則 1291
串連多行或堆疊追蹤記錄訊息 1293
所需的 IAM 許可 1293
確定何時使用多行日誌設定 1294
剖析並串連選項 1296
在視窗容器上部署流暢的位元 1314
必要條件 1316
步驟 1：建立 IAM 存取角色 1316
步驟 2：建立 Amazon ECS Windows 容器執行個體 1317
步驟 3：設定 Fluent Bit 1318
步驟 4：註冊將日誌路由到的 Windows 流利位任務定義 CloudWatch 1320
步驟 5：使用常駐程式排程策略以 Amazon ECS 服務形式執行 ecs-windows-fluent-bit 任務定義 1322
步驟 6：註冊會產生日誌的 Windows 任務定義 1323
步驟 7：執行 windows-app-task 任務定義 1325
步驟 8：驗證日誌 CloudWatch 1325
步驟 9：清除 1326
gMSA適用於 EC2 Linux 容器 1327
考量事項 1327
必要條件 1328
設定 1329
CredSpec file 1336
用gMSA於 Far Linux gate 上的容器 1337
考量事項 1337
必要條件 1337
設定 1338
CredSpec file 1340
將 Windows 容器與無網域搭配gMSA使用 AWS CLI 1342
必要條件 1342
步驟 1：在 Active Directory Domain Services (AD DS) 上建立和設定 gMSA 帳戶 1344
步驟 2：將憑證上傳至 Secrets Manager 1345
步驟 3：修改您的 CredSpec JSON 以包含無網域 gMSA 資訊 1346

步驟 4：將 CredSpec 上傳至 Amazon S3 1347
步驟 5：(選用) 建立 Amazon ECS 叢集 1348
步驟 6：針對容器執行個體建立 IAM 角色 1348
步驟 7：建立自訂任務執行角色 1348
步驟 8：為 Amazon ECS Exec 建立任務角色 1349
步驟 9：註冊任務定義 1351
步驟 10：註冊 Windows 容器執行個體 1352
步驟 11：驗證容器執行個體 1353
步驟 12：執行 Windows 任務 1354
步驟 13：驗證容器具有 gMSA 憑證 1355
步驟 14：清除 1355
除錯 1356
了解如何在 EC2 視窗容器上使用 gMSAs A 1357
考量事項 1358
必要條件 1358
設定 1359
使用 Image Builder 建置自訂的 Amazon ECS 最佳化 AMI 1365
將映像 ARN 與基礎結構即程式碼 (IaC) 搭配使用 1366
搭配使用圖像 ARN AWS CloudFormation 1368
將圖像 ARN 與地形一起使用 1369
使用 AWS Deep Learning Containers 1370
Deep Learning Containers 與 Amazon ECS 上的 Elastic Inference 1370
Service Quotas 1371
Amazon ECS 服務配額 1371
AWS Fargate 服務配額 1374
管理您的服務配額 AWS Management Console 1375
處理服務配額和 API 節流限制 1377
Elastic Load Balancing 1377
彈性網絡介面 1378
AWS Cloud Map 1380
Amazon ECS API 參考 1381
文件歷史紀錄 1382
..... mcdxii

什麼是 Amazon Elastic Container Service？

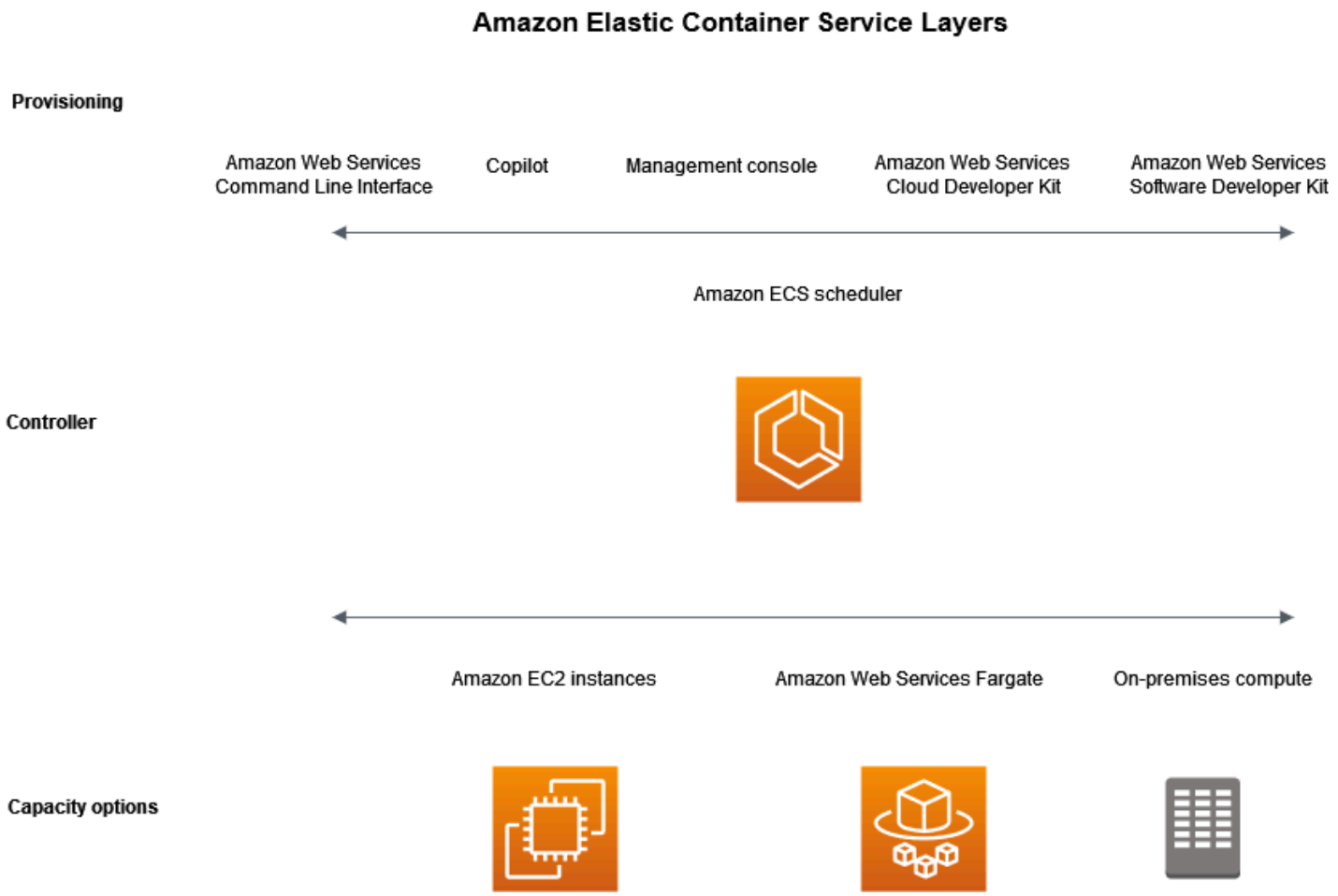
Amazon Elastic Container Service (Amazon ECS) 為全受管容器協同運作服務，可讓您輕鬆部署、管理和擴展容器化應用程式。Amazon ECS 是全受管服務，內建 AWS 組態和操作最佳實務。它與 AWS 和第三方工具集成，例如 Amazon 彈性容器註冊表和 Docker。這種整合可讓團隊能夠更輕鬆地專注於建置應用程式，而無需為環境分心。您可以在雲端和內部部署 AWS 區域 中執行和擴展容器工作負載，而不必擔心管理控制平面的複雜性。

Amazon ECS 術語和元件

Amazon ECS 有三層：

- 容量 - 您的容器執行所在的基礎設施
- 控制器 - 部署和管理在容器上執行的應用程式
- 佈建 - 可用於與排程器連接以部署和管理應用程式和容器的工具

下圖顯示 Amazon ECS 層。



Amazon ECS 容量

Amazon ECS 容量是您的容器執行所在的基礎設施。以下是容量選項概觀：

- AWS 雲端中的 Amazon EC2 執行個體

  您可以選擇執行個體類型、執行個體數量，以及管理容量。

- 雲端中的 AWS 無伺服器 (AWS Fargate (Fargate))

  Fargate 是無伺服器的 pay-as-you-go 運算引擎。搭配使用 Fargate，您無需管理伺服器、處理容量規劃，或出於安全性而隔離容器工作負載。

- 內部部署虛擬機器或伺服器

  Amazon ECS Anywhere 支援將外部執行個體 (例如內部部署伺服器或虛擬機器 (VM)) 註冊到 Amazon ECS 叢集。

容量可以位於下列任一資 AWS 源中：

- 可用區域
- 本機區域
- Wavelength 區域
- AWS 區域
- AWS Outposts

Amazon ECS 控制器

Amazon ECS 排程器是管理您應用程式的軟體。

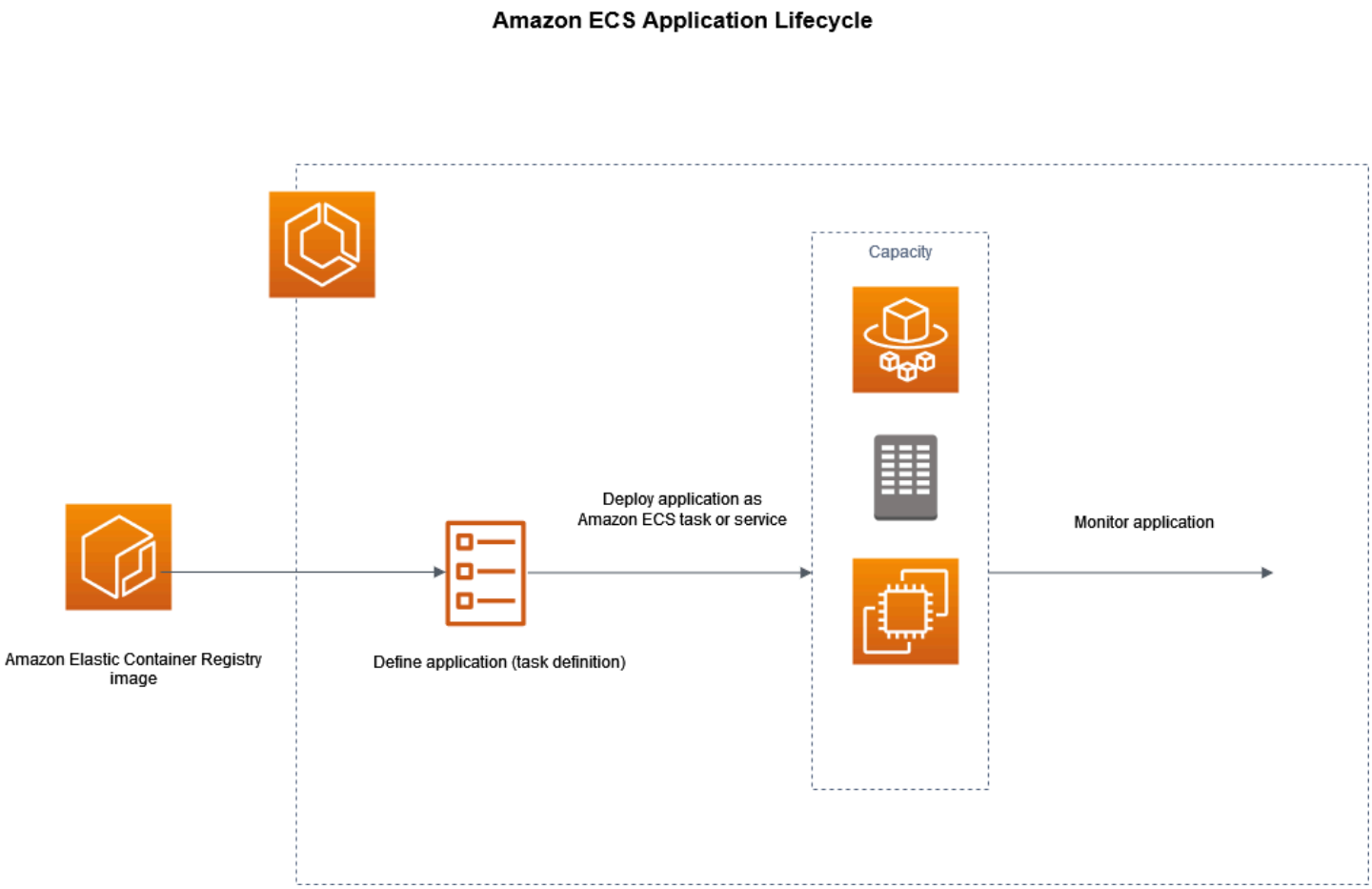
Amazon ECS 佈建

有多種佈建 Amazon ECS 的選項：

- AWS Management Console - 提供 Web 介面，您可使用此介面來存取 Amazon ECS 資源。
- AWS Command Line Interface (AWS CLI) — 為包括 Amazon ECS 在內的各種 AWS 服務提供命令。Windows、Mac 和 Linux 均支援此介面。如需詳細資訊，請參閱 AWS Command Line Interface。
- AWS SDK — 提供特定於語言的 API，並處理許多連接詳細信息。包括計算簽章、處理請求重試和錯誤處理。如需詳細資訊，請參閱 AWS 開發套件。
- Copilot - 提供開放原始碼工具，供開發人員在 Amazon ECS 上建置、發佈和操作可立即生產的容器化應用程式。如需詳細資訊，請參閱網站上的 GitHubCopilot。
- AWS CDK - 提供開放原始碼軟體開發架構，您可以使用該架構來建模，並使用熟悉的程式設計語言來佈建您的雲端應用程式資源。 AWS CDK 會透過 AWS CloudFormation以安全、可重複的方式佈建您的資源。

應用程式生命週期

下圖顯示應用程式生命週期及其如何與 Amazon ECS 元件搭配使用。



您必須架構應用程式，才能在容器上執行。容器是軟體開發的標準化單元，可容納軟體應用程式執行所需的所有一切。這包括相關代碼、執行時間、系統工具和系統庫。容器依據稱為映像的唯讀範本而建立。映像通常由 Dockerfile 建立。一個碼頭文件是一個明文文件，其中包含用於構建一個容器的說明。建置後，這些映像會存放在登錄檔(如 Amazon ECR) 中，可供下載。

建立並儲存映像後，您可以建立 Amazon ECS 任務定義。任務定義是您應用程式的藍圖。其是一種 JSON 格式的文字檔案，描述了構成應用程式的參數和一個或多個容器。例如，您可以使用其來指定作業系統的映像和參數、要使用的容器、要為您的應用程式開啟的連接埠，以及任務中的容器要使用的資料磁碟區。任務定義可用的特定參數取決於您特定應用程式的需求。

定義任務定義之後，您可以將其部署為叢集上的服務或任務。叢集是在註冊至叢集的容量基礎結構上執行的任務或服務的邏輯群組。

任務是在叢集內將任務定義執行個體化。您可以執行獨立任務，也可以將任務作為服務的一部分執行。您可以使用 Amazon ECS 服務在 Amazon ECS 叢集中同時執行和維護您所需的任務數量。其運作方式為，如果有任何任務因任何原因而出現故障或停止，Amazon ECS 服務排程器就會根據您的任務定義啟動另一個執行個體。這樣就可以取代該任務，從而在服務中保持所需的任務數量。

容器代理程式在 Amazon ECS 叢集內的每個容器執行個體上執行。代理程式會將目前正在執行的任務和您容器的資源使用率的相關資訊傳送至 Amazon ECS。只要接收到 Amazon ECS 的請求即開始和停止任務。

部署任務或服務後，您可以使用下列任何工具來監控部署和應用程式：

- CloudWatch
- 執行期監控

Amazon ECS 相關信息

以下相關資源可協助您使用此服務。

- AWS Fargate – Fargate 功能概觀。
- 視窗開啟 AWS — AWS 工作負載和服務的視窗概觀。
- Linux 來自 AWS — 基於 Linux 的現代操作系統產品組合。 AWS

開發人員教學課程

- AWS Compute Blogs – 有關新功能、深入探討功能、程式碼範例和最佳實務的資訊。

AWS re:Post

AWS re:Post— AWS 託管問答（問答）服務，為您的技術問題提供眾包，專家審查的答案。

定價

- Amazon ECS 定價 – Amazon ECS 定價資訊。
- AWS Fargate 定價 — Fargate 的定價信息。

一般 AWS 資源

下列一般資源可協助您使用 AWS。

- 課程和工作坊 — 除了可以幫助提高 AWS 技能並獲得實踐經驗的自定進度實驗室之外，還可以鏈接到基於角色和專業課程的鏈接。
- AWS 開發人員中心 — 探索教學課程、下載工具，以及瞭解 AWS 開發人員活動。

- AWS 開發人員工具 — 開發人員工具、SDK、IDE 工具組，以及用於開發和管理 AWS 應用程式的命令列工具的連結。
- 入門資源中心 — 瞭解如何設定 AWS 帳戶、加入 AWS 社群，以及啟動您的第一個應用程式。
- 實作教學課程 — 按照 step-by-step 教學課程啟動您的第一個應用程式 AWS。
- AWS 白皮書 — 完整的技術 AWS 白皮書清單連結，涵蓋架構、安全性和經濟等主題，並由 AWS 解決方案架構師或其他技術專家撰寫。
- AWS Support 中心 — 建立和管理 AWS Support 案例的中心。同時也包含其他實用資源的連結，例如論壇、技術常見問答集、服務健康狀態和 AWS Trusted Advisor.
- AWS Support— 有關資訊的主要網頁 AWS Support one-on-one, 快速回應的支援管道，可協助您在雲端中建置和執行應用程式.
- 聯絡我們 – 查詢有關 AWS 帳單、帳戶、事件、濫用與其他問題的聯絡中心。
- AWS 網站條款 — 有關我們的版權和商標的詳細資訊；您的帳戶、授權和網站存取權限；以及其他主題。

了解如何建立和使用 Amazon ECS 資源

以下指南介紹了可用於存取 Amazon ECS 的工具，以及執行容器的簡介程序。Docker 基本概念會逐步引導您透過基本步驟建立 Docker 容器映像並將其上傳到 Amazon ECR 私有儲存庫。入門指南會逐步引導您使用 C AWS opilot 命令列界面，以及完 AWS Management Console 成在 Amazon ECS 和上執行容器的常見任務。 AWS Fargate

目錄

- 設定以使用 Amazon ECS。
- 建立在 Amazon ECS 上使用的容器映像
- 了解如何為 Fargate 啟動類型建立 Amazon ECS Linux 任務
- 了解如何為 Fargate 啟動類型建立 Amazon ECS 視窗任務
- 了解如何為 EC2 啟動類型建立 Amazon ECS 視窗任務

設定以使用 Amazon ECS。

如果您已經註冊 Amazon Web Services (AWS) 且已在使用 Amazon Elastic Compute Cloud (Amazon EC2)，您也幾乎可使用 Amazon ECS 了。這兩項服務的設定程序很類似。下列指南將協助您為啟動您的首個 Amazon ECS 叢集做好準備。

完成下列任務，為 Amazon ECS 進行設定。

註冊一個 AWS 帳戶

如果您沒有 AWS 帳戶，請完成以下步驟來建立一個。

若要註冊成為 AWS 帳戶

1. 開啟 https://portal.aws.amazon.com/billing/signup。
2. 請遵循線上指示進行。

   部分註冊程序需接收來電，並在電話鍵盤輸入驗證碼。

   當您註冊時 AWS 帳戶，會建立AWS 帳戶根使用者一個。根使用者有權存取該帳戶中的所有 AWS 服務 和資源。安全性最佳做法是將管理存取權指派給使用者，並僅使用 root 使用者來執行需要 root 使用者存取權的工作。

AWS 註冊過程完成後，會向您發送確認電子郵件。您可以隨時登錄 https://aws.amazon.com/ 並選擇我的帳戶，以檢視您目前的帳戶活動並管理帳戶。

建立具有管理權限的使用者

註冊後，請保護您的 AWS 帳戶 AWS 帳戶根使用者 AWS IAM Identity Center、啟用和建立系統管理使用者，這樣您就不會將 root 使用者用於日常工作。

保護您的 AWS 帳戶根使用者

1. 選擇 Root 使用者並輸入您的 AWS 帳戶 電子郵件地址，以帳戶擁有者身分登入。AWS Management Console在下一頁中，輸入您的密碼。

   如需使用根使用者登入的說明，請參閱 AWS 登入 使用者指南中的以根使用者身分登入。
2. 若要在您的根使用者帳戶上啟用多重要素驗證 (MFA)。

   如需指示，請參閱《IAM 使用者指南》中的為 AWS 帳戶 根使用者啟用虛擬 MFA 裝置 (主控台)。

建立具有管理權限的使用者

1. 啟用 IAM Identity Center。

   如需指示，請參閱 AWS IAM Identity Center 使用者指南中的啟用 AWS IAM Identity Center。
2. 在 IAM 身分中心中，將管理存取權授予使用者。

   若要取得有關使用 IAM Identity Center 目錄 做為身分識別來源的自學課程，請參閱《使用指南》IAM Identity Center 目錄中的「以預設值設定使用AWS IAM Identity Center 者存取」。

以具有管理權限的使用者身分登入

- 若要使用您的 IAM Identity Center 使用者簽署，請使用建立 IAM Identity Center 使用者時傳送至您電子郵件地址的簽署 URL。

  如需使用 IAM 身分中心使用者登入的說明，請參閱使用AWS 登入 者指南中的登入 AWS 存取入口網站。

指派存取權給其他使用者

1. 在 IAM 身分中心中，建立遵循套用最低權限許可的最佳做法的權限集。

如需指示，請參閱《AWS IAM Identity Center 使用指南》中的「建立權限集」。

2. 將使用者指派給群組，然後將單一登入存取權指派給群組。

   如需指示，請參閱《AWS IAM Identity Center 使用指南》中的「新增群組」。

建立 Virtual Private Cloud

您可以使用 Amazon Virtual Private Cloud (Amazon VPC) 將 AWS 資源啟動到您已定義的虛擬網路中。強烈建議您在 VPC 中啟動您的容器執行個體。

如果您有預設 VPC，可以跳過本節，並進行下一個任務「建立安全群組」。若要判斷您是否擁有預設 VPC，請參閱 Amazon EC2 使用者指南中的 Amazon EC2 主控台中的支援平台。否則，您可以利用下面的步驟，在帳戶中建立非預設 VPC。

如需有關如何建立 VPC 的詳細資訊，請參閱《Amazon VPC 使用者指南》中的僅建立 VPC，並使用下表判斷要選取哪些選項。

選項	Value
要建立的資源	僅 VPC
名稱	可以選擇為 VPC 提供名稱。
IPv4 CIDR 區塊	IPv4 CIDR 手動輸入 CIDR 區塊大小必須為介於 /16 和 /28 之間的大小。
IPv6 CIDR 區塊	無 IPv6 CIDR 區塊
租用	預設

如需有關 Amazon VPC 的詳細資訊，請參閱《Amazon VPC 使用者指南》中的什麼是 Amazon VPC？。

建立安全群組

安全群組的功能為相關聯容器執行個體的防火牆，可在容器執行個體層級控制傳入和傳出流量。您可以新增規則至安全群組，讓您從您的 IP 地址使用 SSH 連接到您的容器執行個體。您也可以新增允許任何位置之傳入和傳出 HTTP 和 HTTPS 存取的規則。依您的任務所需在開放連接埠新增任何規則。容器執行個體需要外部網路存取，才可以與 Amazon ECS 服務端點通訊。

如果您打算在多個區域中啟動容器執行個體，則需要在每個區域中建立安全群組。如需詳細資訊，請參閱 Amazon EC2 使用者指南中的區域和可用區域。

> **Tip**
>
> 您需要本機電腦的公有 IP 地址 (可以使用服務來取得)。例如，我們提供下列服務：http://checkip.amazonaws.com/ 或 https://checkip.amazonaws.com/。若要尋找其他能夠提供您 IP 地址的服務，請使用搜尋片語 "what is my IP address" (我的 IP 地址為何)。如果您透過網際網路服務供應商 (ISP) 或是經由不具靜態 IP 地址的防火牆連線，則必須找出用戶端電腦使用的 IP 地址範圍。

如需如何建立安全群組的詳細資訊，請參閱 Amazon EC2 使用者指南中的建立安全群組，並使用下表決定要選取的選項。

選項	Value
區域	您建立金鑰對的同一個區域。
名稱	一個您易記的名稱 (例如 ecs-instances-default-cluster)。
VPC	預設 VPC (有「預設」標記)。 **Note** 如果您的帳戶支援 Amazon EC2 Classic，請選取您在先前任務中建立的 VPC。

如需要為使用案例新增的輸出規則的相關資訊，請參閱 Amazon EC2 使用者指南中針對不同使用案例的安全群組規則。

Amazon ECS 容器執行個體不需要開啟任何傳入連接埠。不過建議您新增 SSH 規則，以登入容器執行個體，並以 Docker 命令檢查任務。如果您希望您的容器執行個體託管執行 Web 伺服器的任務，則也可以新增 HTTP 和 HTTPS 的規則。而容器執行個體需要外部網路存取，才可以與 Amazon ECS 服務端點通訊。完成下列步驟，以新增這些選用的安全群組規則。

將以下三個輸入規則新增到您的安全群組。如需如何建立安全群組的詳細資訊，請參閱 Amazon EC2 使用者指南中的將規則新增至安全群組。

選項	Value	
HTTP 規則	類型：HTTP 來源：Anywhere (`0.0.0.0/0`) 此選項會自動新增 0.0.0.0/0 IPv4 CIDR 區塊作為來源。通常在測試環境中短暫進行此操作是沒有問題的，但用在生產環境則不安全。在生產環境中，建議您只授權特定 IP 地址或特定範圍的地址存取您的執行個體。	
HTTPS 規則	Type (類型)：HTTPS 來源：Anywhere (`0.0.0.0/0`) 通常在測試環境中短暫進行此操作是沒有問題的，但用在生產環境則不安全。在生產環境中，建議您只授權特定 IP 地址或特定範圍的地址存取您的執行個體。	

選項	Value
SSH 規則	Type (類型)：SSH 來源：Custom (自訂)，並以 CIDR 表示法來指定您的電腦或網路的公有 IP 位址。若要以 CIDR 表示法指定個別 IP 地址，請新增路由前綴 /32。例如，如果您的 IP 地址為 `203.0.113.25` ，請指定 `203.0.113.25/32` 。如果您的公司會分配某個範圍的地址，請指定整個範圍 (例如 `203.0.113.0/24`)。 **Important** 基於安全考量，不建議您允許從所有 IP 地址 (`0.0.0.0/0`) 對您執行個體進行 SSH 存取，除非僅為短時間測試。

建立憑證來連線至 EC2 執行個體

針對 Amazon ECS，只有當您想要使用 EC2 啟動類型時，才需要金鑰對。

AWS 使用公開金鑰加密技術來保護執行個體的登入資訊。Linux 執行個體 (例如 Amazon ECS 容器執行個體) 沒有密碼用於 SSH 存取。您需要使用金鑰對，以安全地登入執行個體。當您啟動容器執行個體時，要指定金鑰對名稱，再於使用 SSH 登入時，提供私有金鑰。

如果您尚未建立金鑰對，可以使用 Amazon EC2 主控台來建立。如果您打算在多個區域中啟動執行個體，則需要在每個區域中建立金鑰對。如需區域的詳細資訊，請參閱 Amazon EC2 使用者指南中的區域和可用區域。

建立一組金鑰對

- 使用 Amazon EC2 主控台來建立金鑰對。如需有關建立 key pair 的詳細資訊，請參閱 Amazon EC2 使用者指南中的建立 key pair。

有關如何 Connect 到執行個體的詳細資訊，請參閱 Amazon EC2 使用者指南中的連接到 Linux 執行個體。

安裝 AWS CLI

AWS Management Console 可以使用 Amazon ECS 手動管理所有操作。不過，您可以在本機桌面或開發人員方塊 AWS CLI 上安裝，以便建立可在 Amazon ECS 中自動執行常見管理任務的指令碼。

若要 AWS CLI 搭配 Amazon ECS 使用，請安裝最新 AWS CLI 版本。若要取得有關安裝 AWS CLI 或升級至最新版本的資訊，請參閱《AWS Command Line Interface 使用指南》中的〈安裝指 AWS 令行介面〉。

建立在 Amazon ECS 上使用的容器映像

Amazon ECS 使用任務定義中的 Docker 映像來啟動容器。Docker 是一種技術，可為您提供建置、執行、測試和部署分散式應用程式所需的工具。

此處概述的步驟旨在逐步引導您建立第一個 Docker 映像，並將該映像推送到 Amazon ECR (即容器登錄檔)，以便在 Amazon ECS 任務定義中使用。此引導過程假定您對 Docker 的含義和其運作方式有基本的了解。如需有關 Docker 的詳細資訊，請參閱什麼是 Docker？和 Docker 概觀。

必要條件

開始之前，請務必先達成以下先決條件。

- 確保您已完成 Amazon ECR 設定步驟。如需詳細資訊，請參閱《Amazon Elastic 容器登錄檔使用者指南》中的設定 Amazon ECR。
- 您的使用者已具備存取和使用 Amazon ECR 服務所需的 IAM 許可。如需詳細資訊，請參閱 Amazon ECR 受管政策。
- 您已安裝 Docker。如需有關 Amazon Linux 2 Docker 的安裝步驟，請參閱 在 AL2023 上安裝泊塢視窗。如需其他作業系統的相關資訊，請參閱 Docker Desktop 概觀中的 Docker 文件。
- 您已安 AWS CLI 裝並配置。如需詳細資訊，請參閱《AWS Command Line Interface 使用者指南》中的安裝 AWS Command Line Interface。

若您沒有或不需要本機開發環境，而且您偏好透過 Amazon EC2 執行個體使用 Docker，我們提供以下步驟來使用 Amazon Linux 2 啟動 Amazon EC2 執行個體並安裝 Docker Engine 和 Docker CLI。

在 AL2023 上安裝泊塢視窗

Docker 可在多個不同的作業系統上使用，包括大部分的現代 Linux 發行版本，例如 Ubuntu，甚至是 macOS 和 Windows。如需如何在特定作業系統上安裝 Docker 的詳細資訊，請前往「Docker 安裝指南」。

您不需要本機開發系統，就能使用 Docker。如果您已在使用 Amazon EC2，則可以啟動 Amazon Linux 2023 執行個體，並安裝 Docker 以開始使用。

如果您已經安裝 Docker，請跳到「建立 Docker 映像」。

使用 Amazon Linux 2023 AMI 在 Amazon EC2 執行個體上 安裝 Docker

1. 使用最新版 Amazon Linux 2023 AMI 啟動執行個體。如需詳細資訊，請參閱 Amazon EC2 使用者指南中的啟動執行個體。
2. 連線到您的執行個體。如需詳細資訊，請參閱 Amazon EC2 使用者指南中的 Connect 到 Linux 執行個體。
3. 更新已安裝的套裝服務，並在執行個體上封裝快取。

   ```
   sudo yum update -y
   ```

4. 安裝最新的 Docker Community Edition 套裝服務。

   ```
   sudo yum install docker
   ```

5. 啟動 Docker 服務。

   ```
   sudo service docker start
   ```

6. 將 `ec2-user` 新增至 `docker` 群組，讓您可以在不使用 `sudo` 的情況下執行 Docker 命令。

   ```
   sudo usermod -a -G docker ec2-user
   ```

7. 登出並重新登入，以取得新的 `docker` 群組許可。關閉目前的 SSH 終端機視窗，即可完成此操作，並在新的 SSH 終端機視窗中重新連接至執行個體。新的 SSH 工作階段將會有適當的 `docker` 群組許可。
8. 驗證 `ec2-user` 可以在不使用 `sudo` 的情況下執行 Docker 命令。

```
docker info
```

> **Note**
>
> 在某些情況下，您可能需要重新啟動執行個體，才能提供 `ec2-user` 存取 Docker 常駐程式的許可。如果您看到下列錯誤，請嘗試重新啟動執行個體：
>
> ```
> Cannot connect to the Docker daemon. Is the docker daemon running on this
>  host?
> ```

建立 Docker 映像

Amazon ECS 任務定義使用 Docker 映像，藉以啟動您叢集中之容器執行個體上的容器。在本節中，您將建立簡單 Web 應用程式的 Docker 映像，並在本機系統或 Amazon EC2 執行個體上進行測試，然後將映像推送至 Amazon ECR 容器登錄檔，讓您可以在 Amazon ECS 任務定義中使用。

建立簡單 Web 應用程式的 Docker 映像

1. 建立稱為 `Dockerfile` 的檔案。Dockerfile 是一種資訊清單，說明用於您 Docker 映像的基本映像，以及您要安裝並在其上執行的項目。如需 Dockerfile 的詳細資訊，請前往「Dockerfile 參考」。

   ```
   touch Dockerfile
   ```

2. 編輯您剛建立的 `Dockerfile`，並新增下列內容。

   ```
   FROM public.ecr.aws/amazonlinux/amazonlinux:latest

   # Install dependencies
   RUN yum update -y && \
    yum install -y httpd

   # Install apache and write hello world message
   RUN echo 'Hello World!' > /var/www/html/index.html

   # Configure apache
   RUN echo 'mkdir -p /var/run/httpd' >> /root/run_apache.sh && \
    echo 'mkdir -p /var/lock/httpd' >> /root/run_apache.sh && \
   ```

```
 echo '/usr/sbin/httpd -D FOREGROUND' >> /root/run_apache.sh && \
 chmod 755 /root/run_apache.sh

EXPOSE 80

CMD /root/run_apache.sh
```

該 Dockerfile 使用在 Amazon ECR 公共上託管的 Amazon Linux 2 映像。RUN 指令會更新套件快取，並安裝 Web 伺服器的一些軟體套件服務，然後寫入 "Hello World!" 內容至 Web 伺服器文件根目錄。該EXPOSE指令表示容器上的端口 80 是正在監聽的端口，並且指CMD令啟動 Web 服務器。

3. 從 Dockerfile 建置 Docker 映像。

> **Note**
>
> 在下列命令中，有些 Docker 版本可能需要 Dockerfile 的完整路徑，而不是下面所示的相對路徑。

```
docker build -t hello-world .
```

4. 列出您的容器映像。

```
docker images --filter reference=hello-world
```

輸出：

```
REPOSITORY          TAG                 IMAGE ID            CREATED
 SIZE
hello-world         latest              e9ffedc8c286        4 minutes ago
 194MB
```

5. 執行新建置的映像。-p 80:80 選項會將容器上的公開連接埠 80 映射至主機系統上的連接埠 80。如需 docker run 的詳細資訊，請前往「Docker run 參考」。

```
docker run -t -i -p 80:80 hello-world
```

> **Note**
>
> Apache Web 伺服器中的輸出會顯示在終端機視窗中。您可以忽略 "`Could not reliably determine the fully qualified domain name`" 訊息。

6. 開啟瀏覽器，然後指向執行 Docker 並託管容器的伺服器。

   - 如果您使用的是 EC2 執行個體，則這是伺服器的「公有 DNS」值，這是您使用 SSH 來連線至執行個體的同個地址。請確定您執行個體的安全群組允許連接埠 80 上的入站流量。
   - 如果您在本機執行 Docker，請將瀏覽器指向 http://localhost/。
   - *#### Windows # Mac ##**docker-machine**#######**docker-machine ip**###### docker # VirtualBox ##### IP ##########################*

     ```
     docker-machine ip machine-name
     ```

   您應該會看到網頁，內含您的 "Hello World!" 陳述式。

7. 輸入 Ctrl + c，以停止 Docker 容器。

推送映像至 Amazon Elastic Container Registry

Amazon ECR 是一個託管 AWS 碼頭註冊表服務。您可以使用 Docker CLI 在 Amazon ECR 儲存庫中推送、提取與管理映像。如需 Amazon ECR 產品詳細資訊、特色客戶案例研究和常見問答集，請參閱 Amazon Elastic Container Registry 產品詳細資訊頁面。

標記映像並將之推送至 Amazon ECR

1. 建立 Amazon ECR 儲存庫，以便存放 `hello-world` 映像。請記下輸出中的 `repositoryUri`。

   用您的替代品 `region` AWS 區域，例如，`us-east-1`.

   ```
   aws ecr create-repository --repository-name hello-repository --region region
   ```

   輸出：

   ```
   {
       "repository": {
   ```

```
        "registryId": "aws_account_id",
        "repositoryName": "hello-repository",
        "repositoryArn": "arn:aws:ecr:region:aws_account_id:repository/hello-
repository",
        "createdAt": 1505337806.0,
        "repositoryUri": "aws_account_id.dkr.ecr.region.amazonaws.com/hello-
repository"
    }
}
```

2. 為 hello-world 映像標記上一步中的 repositoryUri 值。

```
docker tag hello-world aws_account_id.dkr.ecr.region.amazonaws.com/hello-repository
```

3. 執行 aws ecr get-login-password 命令。指定您要驗證的登錄 URI。如需詳細資訊，請參閱《Amazon Elastic Container Registry 使用者指南》中的登錄檔身分驗證。

```
aws ecr get-login-password --region region | docker login --username AWS --
password-stdin aws_account_id.dkr.ecr.region.amazonaws.com
```

輸出：

```
Login Succeeded
```

> **Important**
>
> 若您收到錯誤，請安裝或升級至最新版本的 AWS CLI。如需詳細資訊，請參閱《AWS Command Line Interface 使用者指南》中的安裝 AWS Command Line Interface。

4. 使用先前步驟中的 repositoryUri 值，將映像推送至 Amazon ECR。

```
docker push aws_account_id.dkr.ecr.region.amazonaws.com/hello-repository
```

清除

若要繼續建立 Amazon ECS 任務定義並使用容器映像啟動任務，請跳到 後續步驟。在您試驗完 Amazon ECR 映像後，即可刪除儲存庫，這樣就不會向您收取映像儲存的費用。

```
aws ecr delete-repository --repository-name hello-repository --region region --force
```

後續步驟

任務定義需要任務執行角色。如需詳細資訊，請參閱 Amazon ECS 任務執行 IAM 角色。

建立容器映像並將其推送到 Amazon ECR 之後，您可以在任務定義中使用該映像。如需詳細資訊，請參閱下列其中一個項目：

- the section called “瞭解如何為 Fargate 啟動類型建立 Linux 工作”
- the section called “了解如何為 Fargate 啟動類型建立 Windows 工作”
- 使用以下方式為 Fargate 啟動類型建立 Amazon ECS Linux 任務 AWS CLI

了解如何為 Fargate 啟動類型建立 Amazon ECS Linux 任務

Amazon Elastic Container Service (Amazon ECS) 是具高可擴展性且快速的容器管理服務，可讓您輕鬆執行、停止和管理容器。您可以在 AWS Fargate上啟動服務或任務，以在受 Amazon ECS 管理的無伺服器基礎設施上託管容器。若要取得有關 Fargate 的更多資訊，請參閱AWS Fargate對於 Amazon ECS。

在 Amazon ECS 支援 Fargate 的 AWS Fargate 區域中，針對您的任務使用 Fargate 啟動類型，開始使用 Amazon ECS。 AWS

完成下列步驟，以在 AWS Fargate上開始使用 Amazon ECS。

必要條件

開始之前，請先完成中的步驟，設定以使用 Amazon ECS。 且您的 AWS 使用者具有 `AdministratorAccess` IAM 政策範例中指定的許可。

主控台會嘗試自動建立任務執行 IAM 角色，該角色是 Fargate 任務的必要項目。若要確保主控台可成功建立此 IAM 角色，下列其中一項必須為 True：

- 您的使用者具有管理員存取。如需詳細資訊，請參閱 設定以使用 Amazon ECS。 。
- 您的使用者具有建立服務角色的 IAM 許可。如需詳細資訊，請參閱建立角色以將權限委派給 AWS 服務。
- 具備管理員存取的使用者已手動建立任務執行角色，讓其可在帳戶上提供使用。如需詳細資訊，請參閱 Amazon ECS 任務執行 IAM 角色。

Important

使用任務定義建立服務時選取的安全群組，必須為入站流量開放連接埠 80。將下列傳入規則新增至安全群組。如需如何建立安全群組的詳細資訊，請參閱 Amazon EC2 使用者指南中的將規則新增至安全群組。

- Type (類型)：HTTP
- Protocol (通訊協定)：TCP
- 連接埠範圍：80
- 來源：Anywhere (`0.0.0.0/0`)

步驟 1：建立叢集

建立使用預設 VPC 的叢集。

在開始之前，請指派適當的 IAM 許可。如需詳細資訊，請參閱 the section called “Amazon ECS 集群示例”。

1. 開啟主控台，網址為 https://console.aws.amazon.com/ecs/v2。
2. 從導覽列中選取要使用的「區域」。
3. 在導覽窗格中，選擇叢集。
4. 在 Clusters (叢集) 頁面上，選擇 Create cluster (建立叢集)。
5. 在 Cluster configuration (叢集組態) 下的 Cluster name (叢集名稱) 中，輸入唯一的名稱。

   名稱可以包含最多 255 個字母 (大小寫)、數字與連字號。
6. (選用) 若要開啟 Container Insights，請展開 Monitoring (監控)，然後開啟 Use Container Insights (使用 Container Insights)。
7. (選用) 為協助識別您的叢集，請展開 Tags (標籤)，然後設定標籤。

   [新增標籤] 選擇新增標籤，並執行下列動作：

   - 對於 Key (金鑰)，輸入金鑰名稱。
   - 對於 Value (值)，進入金鑰值。

   [移除標籤] 選擇標籤「金鑰」和「值」右側的移除。

8. 選擇建立。

步驟 2：建立任務定義

任務定義就像您應用程式的藍圖。每次在 Amazon ECS 中啟動任務時，您都必須指定任務定義。服務接著會知道要為容器使用哪個 Docker 映像、要在任務中使用多少個容器，以及每個容器的資源配置。

1. 在導覽窗格中，選擇 Task Definitions (任務定義)。
2. 選擇 Create new Task Definition (建立新任務定義)，以及 Create new revision with JSON (使用 JSON 建立新修訂版)。
3. 複製下列任務定義範例並貼到方塊中，然後選擇 Save (儲存)。

```
{
    "family": "sample-fargate",
    "networkMode": "awsvpc",
    "containerDefinitions": [
        {
            "name": "fargate-app",
            "image": "public.ecr.aws/docker/library/httpd:latest",
            "portMappings": [
                {
                    "containerPort": 80,
                    "hostPort": 80,
                    "protocol": "tcp"
                }
            ],
            "essential": true,
            "entryPoint": [
                "sh",
  "-c"
            ],
            "command": [
                "/bin/sh -c \"echo '<html> <head> <title>Amazon ECS Sample
 App</title> <style>body {margin-top: 40px; background-color: #333;} </style> </
head><body> <div style=color:white;text-align:center> <h1>Amazon ECS Sample App</
h1> <h2>Congratulations!</h2> <p>Your application is now running on a container in
 Amazon ECS.</p> </div></body></html>' >  /usr/local/apache2/htdocs/index.html &&
 httpd-foreground\""
            ]
        }
    ],
```

```
    "requiresCompatibilities": [
        "FARGATE"
    ],
    "cpu": "256",
    "memory": "512"
}
```

4. 選擇建立。

步驟 3：建立服務

使用任務定義來建立服務。

1. 在導覽窗格中，選擇 Clusters (叢集)，然後選取您在 步驟 1：建立叢集 中建立的叢集。
2. 在 Services (服務) 索引標籤上，選擇 Create (建立)。
3. 在 Deployment configuration (部署組態)，指定應用程式的部署方式。
   a. 在 Task Definitions (任務定義) 中，選擇您在 步驟 2：建立任務定義 中建立的任務定義。
   b. 針對 Service name (服務名稱)，輸入服務的名稱。
   c. 在 Desired tasks (所需任務) 中，請輸入 1。
4. 在 Networking (聯網) 下，您可以為您的任務建立新的安全群組或選擇現有的安全群組。請確定您使用的安全群組已在 必要條件 下方列出傳入規則。
5. 選擇建立。

步驟 4：檢視服務

1. 開啟主控台，網址為 https://console.aws.amazon.com/ecs/v2。
2. 在導覽窗格中，選擇叢集。
3. 選擇您執行服務的叢集。
4. 在 Services (服務) 索引標籤中，Service name (服務名稱) 下，選擇您在 步驟 3：建立服務 中建立的服務。
5. 選擇 Tasks (任務) 索引標籤，然後選擇服務中的任務。
6. 在任務頁面 Configuration (組態) 區段的 Public IP (公有 IP) 中，選擇 Open address (開放地址)。

步驟 5：清除

完成使用 Amazon ECS 叢集時，您應該清除與其相關的資源，以免未使用的資源產生費用。

某些 Amazon ECS 資源 (例如任務、服務、叢集和容器執行個體) 可使用 Amazon ECS 主控台進行清除。其他資源 (例如 Amazon EC2 執行個體、Elastic Load Balancing 負載平衡器和 Auto Scaling 群組) 必須在 Amazon EC2 主控台中手動清理，或刪除建立這些執行個體的 AWS CloudFormation 堆疊。

1. 在導覽窗格中，選擇叢集。
2. 在 Clusters (叢集) 頁面，選取您為本教學課程建立的叢集。
3. 選擇 Services (服務) 索引標籤。
4. 選取服務，然後選擇 Delete (刪除)。
5. 在確認提示中，輸入 delete (刪除)，然後選擇 Delete (刪除)。或者，您可以使用`Force delete`選項讓 Amazon ECS 代表您縮減服務，然後再刪除服務。

   等到刪除該服務。
6. 選擇 Delete Cluster (刪除叢集)。在確認提示中，輸入 delete ***`cluster-name`***，然後選擇 Delete (刪除)。刪除叢集會清除與叢集一起建立的相關聯資源，包括 Auto Scaling 群組、VPC 或負載平衡器。

了解如何為 Fargate 啟動類型建立 Amazon ECS 視窗任務

在 Amazon ECS 支援 Fargate 的 AWS Fargate 區域中，針對您的任務使用 Fargate 啟動類型，開始使用 Amazon ECS。 AWS

完成下列步驟，以在 AWS Fargate上開始使用 Amazon ECS。

必要條件

開始之前，請先完成中的步驟，設定以使用 Amazon ECS。且您的 AWS 使用者具有`AdministratorAccess` IAM 政策範例中指定的許可。

主控台會嘗試自動建立任務執行 IAM 角色，該角色是 Fargate 任務的必要項目。若要確保主控台可成功建立此 IAM 角色，下列其中一項必須為 True：

- 您的使用者具有管理員存取。如需詳細資訊，請參閱 設定以使用 Amazon ECS。。
- 您的使用者具有建立服務角色的 IAM 許可。如需詳細資訊，請參閱建立角色以將權限委派給 AWS 服務。

- 具備管理員存取的使用者已手動建立任務執行角色，讓其可在帳戶上提供使用。如需詳細資訊，請參閱 Amazon ECS 任務執行 IAM 角色。

> **Important**
>
> 使用任務定義建立服務時選取的安全群組，必須為入站流量開放連接埠 80。將下列傳入規則新增至安全群組。如需如何建立安全群組的詳細資訊，請參閱 Amazon EC2 使用者指南中的將規則新增至安全群組。
>
> - Type (類型)：HTTP
> - Protocol (通訊協定)：TCP
> - 連接埠範圍：80
> - 來源：Anywhere (`0.0.0.0/0`)

步驟 1：建立叢集

您可以建立使用預設 VPC 且名為 windows 的全新叢集。

若要建立叢集 AWS Management Console

1. 開啟主控台，網址為 https://console.aws.amazon.com/ecs/v2。
2. 從導覽列中選取要使用的「區域」。
3. 在導覽窗格中，選擇叢集。
4. 在 Clusters (叢集) 頁面上，選擇 Create cluster (建立叢集)。
5. 在 Cluster configuration (叢集組態) 下的 Cluster name (叢集名稱) 中，輸入 windows。
6. (選用) 若要開啟 Container Insights，請展開 Monitoring (監控)，然後開啟 Use Container Insights (使用 Container Insights)。
7. (選用) 為協助識別您的叢集，請展開 Tags (標籤)，然後設定標籤。

   [新增標籤] 選擇新增標籤，並執行下列動作：

   - 對於 Key (金鑰)，輸入金鑰名稱。
   - 對於 Value (值)，進入金鑰值。

   [移除標籤] 選擇標籤「金鑰」和「值」右側的移除。

8. 選擇建立。

步驟 2：註冊 Windows 任務定義

您必須先註冊任務定義，才能在您的 Amazon ECS 叢集中執行 Windows 容器。以下任務定義範例會在具有 `mcr.microsoft.com/windows/servercore/iis` 容器映像之容器執行個體的連接埠 8080 顯示一個簡單的網頁。

若要註冊範例工作定義 AWS Management Console

1. 在導覽窗格中，選擇 Task Definitions (任務定義)。
2. 選擇 Create new task definitio (建立新任務定義)、Create new task definition with JSON (使用 JSON 建立新的任務定義)。
3. 複製下列任務定義範例並貼到方塊中，然後選擇 Save (儲存)。

```
{
    "containerDefinitions": [
        {
            "command": ["New-Item -Path C:\\inetpub\\wwwroot\\index.html
 -Type file -Value '<html> <head> <title>Amazon ECS Sample App</title>
 <style>body {margin-top: 40px; background-color: #333;} </style> </head><body>
 <div style=color:white;text-align:center> <h1>Amazon ECS Sample App</h1>
 <h2>Congratulations!</h2> <p>Your application is now running on a container in
 Amazon ECS.</p>'; C:\\ServiceMonitor.exe w3svc"],
            "entryPoint": [
                "powershell",
                "-Command"
            ],
            "essential": true,
            "cpu": 2048,
            "memory": 4096,
            "image": "mcr.microsoft.com/windows/servercore/iis:windowsservercore-
ltsc2019",
            "name": "sample_windows_app",
            "portMappings": [
                {
                    "hostPort": 80,
                    "containerPort": 80,
                    "protocol": "tcp"
                }
            ]
```

```
        }
    ],
    "memory": "4096",
    "cpu": "2048",
    "networkMode": "awsvpc",
    "family": "windows-simple-iis-2019-core",
    "executionRoleArn": "arn:aws:iam::012345678910:role/ecsTaskExecutionRole",
    "runtimePlatform": {"operatingSystemFamily": "WINDOWS_SERVER_2019_CORE"},
    "requiresCompatibilities": ["FARGATE"]
}
```

4. 驗證您的資訊，然後選擇 Create (建立)。

步驟 3：以您的任務定義建立服務

註冊您的任務定義之後，即可用以放置任務到您的叢集中。下列程序會以您的任務定義建立一項服務，並在您的叢集放置一項任務。

使用主控台從您的任務定義建立服務

1. 在導覽窗格中，選擇 Clusters (叢集)，然後選取您在 步驟 1：建立叢集 中建立的叢集。
2. 在 Services (服務) 索引標籤上，選擇 Create (建立)。
3. 在 Deployment configuration (部署組態)，指定應用程式的部署方式。
   a. 在 Task Definitions (任務定義) 中，選擇您在 步驟 2：註冊 Windows 任務定義 中建立的任務定義。
   b. 針對 Service name (服務名稱)，輸入服務的名稱。
   c. 在 Desired tasks (所需任務) 中，請輸入 1。
4. 在 Networking (聯網) 下，您可以建立新的安全群組，或選擇現有的安全群組。請確定您使用的安全群組已在 必要條件 下方列出傳入規則。
5. 選擇建立。

步驟 4：檢視服務

您的服務已於您的叢集中啟動任務後，您可以在瀏覽器中檢視服務並開啟 IIS 測試頁面，確認容器正在執行。

> **Note**
>
> 您的容器執行個體約需 15 分鐘來下載和解壓縮 Windows 容器基礎層。

檢視服務

1. 開啟主控台，網址為 https://console.aws.amazon.com/ecs/v2。
2. 在導覽窗格中，選擇叢集。
3. 選擇您執行服務的叢集。
4. 在 Services (服務) 索引標籤中，Service name (服務名稱) 下，選擇您在 步驟 3：以您的任務定義建立服務 中建立的服務。
5. 選擇 Tasks (任務) 索引標籤，然後選擇服務中的任務。
6. 在任務頁面 Configuration (組態) 區段的 Public IP (公有 IP) 中，選擇 Open address (開放地址)。

步驟 5：清除

完成使用 Amazon ECS 叢集時，您應該清除與其相關的資源，以免未使用的資源產生費用。

某些 Amazon ECS 資源 (例如任務、服務、叢集和容器執行個體) 可使用 Amazon ECS 主控台進行清除。其他資源 (例如 Amazon EC2 執行個體、Elastic Load Balancing 負載平衡器和 Auto Scaling 群組) 必須在 Amazon EC2 主控台中手動清理，或刪除建立這些執行個體的 AWS CloudFormation 堆疊。

1. 在導覽窗格中，選擇叢集。
2. 在 Clusters (叢集) 頁面，選取您為本教學課程建立的叢集。
3. 選擇 Services (服務) 索引標籤。
4. 選取服務，然後選擇 Delete (刪除)。
5. 在確認提示中，輸入 delete (刪除)，然後選擇 Delete (刪除)。

   等到刪除該服務。
6. 選擇 Delete Cluster (刪除叢集)。在確認提示中，輸入 delete ***cluster-name***，然後選擇 Delete (刪除)。刪除叢集會清除與叢集一起建立的相關聯資源，包括 Auto Scaling 群組、VPC 或負載平衡器。

了解如何為 EC2 啟動類型建立 Amazon ECS 視窗任務

使用 EC2 啟動類型開始使用 Amazon ECS，方法是註冊任務定義、建立叢集以及在主控台中建立服務。

完成下列步驟，以使用 EC2 啟動類型來開始使用 Amazon ECS。

必要條件

開始之前，請先完成中的步驟，設定以使用 Amazon ECS。且您的 AWS 使用者具有 `AdministratorAccess` IAM 政策範例中指定的許可。

主控台會嘗試自動建立任務執行 IAM 角色，該角色是 Fargate 任務的必要項目。若要確保主控台可成功建立此 IAM 角色，下列其中一項必須為 True：

- 您的使用者具有管理員存取。如需詳細資訊，請參閱 設定以使用 Amazon ECS。。
- 您的使用者具有建立服務角色的 IAM 許可。如需詳細資訊，請參閱建立角色以將權限委派給 AWS 服務。
- 具備管理員存取的使用者已手動建立任務執行角色，讓其可在帳戶上提供使用。如需詳細資訊，請參閱 Amazon ECS 任務執行 IAM 角色。

> **Important**
>
> 使用任務定義建立服務時選取的安全群組，必須為入站流量開放連接埠 80。將下列傳入規則新增至安全群組。如需如何建立安全群組的詳細資訊，請參閱 Amazon EC2 使用者指南中的將規則新增至安全群組。
>
> - Type (類型)：HTTP
> - Protocol (通訊協定)：TCP
> - 連接埠範圍：80
> - 來源：Anywhere (`0.0.0.0/0`)

步驟 1：建立叢集

Amazon ECS 叢集是任務、服務和容器執行個體的邏輯分組。

下列步驟將逐步引導您使用一個已註冊至叢集的 Amazon EC2 執行個體來建立叢集，以便讓我們在該叢集上執行任務。若未提及特定欄位，請保留預設主控台值。

建立新叢集 (Amazon ECS 主控台)

在開始之前，請指派適當的 IAM 許可。如需詳細資訊，請參閱 the section called “Amazon ECS 集群示例”。

1. 開啟主控台，網址為 https://console.aws.amazon.com/ecs/v2。
2. 從導覽列中選取要使用的「區域」。
3. 在導覽窗格中，選擇叢集。
4. 在 Clusters (叢集) 頁面上，選擇 Create cluster (建立叢集)。
5. 在 Cluster configuration (叢集組態) 下的 Cluster name (叢集名稱) 中，輸入唯一的名稱。

   名稱可以包含最多 255 個字母 (大小寫)、數字與連字號。
6. (選用) 若要變更任務和服務啟動所在的 VPC 和子網路，請在 Networking (聯網) 下，執行下列任一操作：

   - 若要移除子網路，請在 Subnets (子網路) 下，對您要移除之每一個子網路選擇 X。
   - 若要變更為非 default (預設) VPC，請在 VPC 下，選擇現有的 VPC，然後在 Subnets (子網路) 下選擇各個子網路。
7. 若要將 Amazon EC2 執行個體新增至叢集，展開 Infrastructure (基礎設施)，然後選取 Amazon EC2 執行個體。接下來，設定作為容量提供者的 Auto Scaling 群組：

   a. 要使用現有 Auto Scaling 群組，請從 Auto Scaling group (ASG) (Auto Scaling 群組 (ASG)) 中選取該群組。

   b. 若要建立 Auto Scaling 群組，請從 Auto Scaling group (ASG) (Auto Scaling 群組 (ASG)) 中選取 Create new group (建立新群組)，然後提供有關該群組的下列詳細資訊：

      - 針對 Operating system/Architecture (作業系統/架構)，為 Auto Scaling 群組執行個體選擇 Amazon ECS 最佳化 AMI。
      - 對於 EC2 instance type (EC2 執行個體類型)，選擇適合您工作負載的執行個體類型。如需不同執行個體類型的詳細資訊，請參閱 Amazon EC2 執行個體。

        如果 Auto Scaling 群組使用相同或類似的執行個體類型，則受管擴展效果最佳。
      - 對於 SSH key pair (SSH 金鑰對)，選擇在連線到執行個體時證明您身分的金鑰對。

- 對於 Capacity (容量)，輸入 Auto Scaling 群組中要啟動的最小執行個體數和最大執行個體數。Amazon EC2 執行個體存在於您的 AWS 資源中時會產生成本。如需詳細資訊，請參閱 Amazon EC2 定價。

8. (選用) 若要開啟 Container Insights，請展開 Monitoring (監控)，然後開啟 Use Container Insights (使用 Container Insights)。
9. (選用) 若要管理叢集標籤，請展開 Tags (標籤)，然後執行下列其中一項操作：

   [新增標籤] 選擇新增標籤，並執行下列動作：

   - 對於 Key (金鑰)，輸入金鑰名稱。
   - 對於 Value (值)，進入金鑰值。

   [移除標籤] 選擇標籤「金鑰」和「值」右側的移除。
10. 選擇建立。

步驟 2：註冊任務定義

若要註冊範例工作定義 AWS Management Console

1. 在導覽窗格中，選擇 Task Definitions (任務定義)。
2. 選擇 Create new task definitio (建立新任務定義)、Create new task definition with JSON (使用 JSON 建立新的任務定義)。
3. 複製下列任務定義範例並貼到方塊中，然後選擇 Save (儲存)。

```
{
    "containerDefinitions": [
        {
            "command": ["New-Item -Path C:\\inetpub\\wwwroot\\index.html
 -Type file -Value '<html> <head> <title>Amazon ECS Sample App</title>
 <style>body {margin-top: 40px; background-color: #333;} </style> </head><body>
 <div style=color:white;text-align:center> <h1>Amazon ECS Sample App</h1>
 <h2>Congratulations!</h2> <p>Your application is now running on a container in
 Amazon ECS.</p>'; C:\\ServiceMonitor.exe w3svc"],
            "entryPoint": [
                "powershell",
                "-Command"
            ],
            "essential": true,
```

```
            "cpu": 2048,
            "memory": 4096,
            "image": "mcr.microsoft.com/windows/servercore/iis:windowsservercore-
ltsc2019",
            "name": "sample_windows_app",
            "portMappings": [
                {
                    "hostPort": 443,
                    "containerPort": 80,
                    "protocol": "tcp"
                }
            ]
        }
    ],
    "memory": "4096",
    "cpu": "2048",
    "family": "windows-simple-iis-2019-core",
    "executionRoleArn": "arn:aws:iam::012345678910:role/ecsTaskExecutionRole",
    "runtimePlatform": {"operatingSystemFamily": "WINDOWS_SERVER_2019_CORE"},
    "requiresCompatibilities": ["EC2"]
}
```

4. 驗證您的資訊，然後選擇 Create (建立)。

步驟 3：建立服務

Amazon ECS 服務可協助您在 Amazon ECS 叢集中同時執行並維持指定數目的任務定義執行個體。如果您的有任務因為任何原因而故障或停止，Amazon ECS 服務排程器就會啟動任務定義的另一個執行個體取代之，以維護服務中所需的任務數量。如需服務的詳細資訊，請參閱 Amazon ECS 服務。

建立服務

1. 在導覽窗格中，選擇叢集。
2. 選取您在 步驟 1：建立叢集 中建立的叢集。
3. 在 Services (服務) 標籤上，選擇 Create (建立)。
4. 在 Environment (環境) 區段中，執行以下動作：
   a. 在 Compute options (運算選項) 中，選擇 Launch type (啟動類型)。
   b. 針對 Launch type (啟動類型)，選取 EC2
5. 在 Deployment configuration (部署組態) 區段中，執行以下操作：

a. 在 Family (系列) 中，選擇您在 步驟 2：註冊任務定義 中建立的任務定義。

b. 針對 Service name (服務名稱)，輸入服務的名稱。

c. 在 Desired tasks (所需任務) 中，請輸入 1。

6. 檢閱選項，然後選擇建立。

7. 選擇 View service (檢視服務) 以檢閱服務。

步驟 4：檢視服務

服務是一種 Web 應用程式，您可以使用 Web 瀏覽器檢視其容器。

1. 開啟主控台，網址為 https://console.aws.amazon.com/ecs/v2。

2. 在導覽窗格中，選擇叢集。

3. 選擇您執行服務的叢集。

4. 在 Services (服務) 索引標籤中，Service name (服務名稱) 下，選擇您在 步驟 3：建立服務 中建立的服務。

5. 選擇 Tasks (任務) 索引標籤，然後選擇服務中的任務。

6. 在任務頁面 Configuration (組態) 區段的 Public IP (公有 IP) 中，選擇 Open address (開放地址)。下面的屏幕截圖是預期的輸出。

步驟 5：清除

完成使用 Amazon ECS 叢集時，您應該清除與其相關的資源，以免未使用的資源產生費用。

某些 Amazon ECS 資源 (例如任務、服務、叢集和容器執行個體) 可使用 Amazon ECS 主控台進行清除。其他資源 (例如 Amazon EC2 執行個體、Elastic Load Balancing 負載平衡器和 Auto Scaling 群組) 必須在 Amazon EC2 主控台中手動清理，或刪除建立這些執行個體的 AWS CloudFormation 堆疊。

1. 在導覽窗格中，選擇叢集。
2. 在 Clusters (叢集) 頁面，選取您為本教學課程建立的叢集叢集。
3. 選擇 Services (服務) 索引標籤。
4. 選取服務，然後選擇 Delete (刪除)。
5. 在確認提示中，輸入 delete (刪除)，然後選擇 Delete (刪除)。

   等到刪除該服務。
6. 選擇 Delete Cluster (刪除叢集)。在確認提示中，輸入 delete ***cluster-name***，然後選擇 Delete (刪除)。刪除叢集會清除與叢集一起建立的相關聯資源，包括 Auto Scaling 群組、VPC 或負載平衡器。

Amazon ECS 開發人員工具概觀

無論您是大型企業還是新創公司的一員，Amazon ECS 都能提供各種工具，協助您快速啟動並執行容器，無論您的專業程度如何。您可以利用下列方式來使用 Amazon ECS：

- 使用 AWS Management Console 來了解、開發、管理及視覺化容器應用程式和服務。
- 使用 AWS Command Line Interface、AWS 開發套件 或 ECS API 透過程式設計或指令碼，利用自動化部署對 Amazon ECS 資源執行特定動作。
- 使用自動化部署來定義和管理環境中的所有 AWS 資源AWS CloudFormation。
- 使用完整的AWS 副駕駛 CLI end-to-end 開發人員工作流程來建立、發行和操作符合基礎結構 AWS 最佳實務的容器應用程式。
- 使用您偏好的程式設計語言，利用 AWS CDK 將基礎設施或架構定義為程式碼。
- 使用 AWS 容器 整合的可攜性和容器的工具生態系統，容器化託管在內部部署或 Amazon EC2 執行個體或兩者中的應用程式。
- 利用 Amazon ECS CLI，將應用程式部署到 Amazon ECS 或使用在 Amazon ECS 中執行的容器以 Docker Compose 檔案格式測試本機容器。
- 使用 Docker Desktop 中的 Amazon ECS，從 Docker Desktop 與 Amazon ECS 整合 中啟動容器。

AWS Management Console

AWS Management Console 這是一個基於瀏覽器的界面，用於管理 Amazon ECS 資源。主控台提供服務的可視概觀，讓您無需使用其他工具即可輕鬆探索 Amazon ECS 功能和函數。提供許多相關的教學課程和演練，可以引導您使用主控台。

如需主控台的指導教學，請參閱 了解如何建立和使用 Amazon ECS 資源。

開始使用時，許多客戶更喜歡使用主控台，因為主控台會提供即時的視覺回饋，告知他們採取的動作是否成功。 AWS 熟悉的客戶可以輕鬆管理相關資源 AWS Management Console，例如負載平衡器和 Amazon EC2 執行個體。

從開始 AWS Management Console。

AWS Command Line Interface

AWS Command Line Interface (AWS CLI) 是一個統一的工具，您可以用來管理您的 AWS 服務。僅使用這一個工具，您既可以控制多個 AWS 服務，也可以通過腳本自動化這些服務。中的 Amazon ECS 命令反映了 Amazon ECS API。 AWS CLI

AWS 提供兩組指令行工具：AWS Command Line Interface(AWS CLI) 和 AWS Tools for Windows PowerShell. 如需詳細資訊，請參閱 AWS Command Line Interface 使用者指南與 AWS Tools for Windows PowerShell 使用者指南。

AWS CLI 適合偏好並習慣於編寫指令碼和介接命令列工具的客戶，以及確切知道要在 Amazon ECS 資源上執行哪些動作的客戶。對於想要熟悉 Amazon ECS API 的客戶來說， AWS CLI 這也很有幫助。客戶可以直接從命令列界面使用 Amazon ECS 資源執行許多操作，包括建立、讀取、更新和刪除操作。 AWS CLI

AWS CLI 如果您正在或想要熟悉 Amazon ECS API 和對應的 CLI 命令，並且想要撰寫自動化指令碼並在 Amazon ECS 資源上執行特定動作，請使用。

AWS CloudFormation

Amazon ECS 的 AWS CloudFormation 和 Terraform 都能提供強大的方法，以定義您的 Infrastructure as Code。您可以輕鬆追蹤正在執行範本或 AWS CloudFormation 堆疊的哪個版本，並在需要時回復到以前的版本。可以相同的自動化方式執行基礎設施和應用程式部署。這種靈活性和自動化是使得 AWS CloudFormation 和 Terraform 從持續交付管道將工作負載部署到 Amazon ECS 的兩種流行格式。

如需有關的更多資訊 AWS CloudFormation，請參閱使用創建 Amazon ECS 資源 AWS CloudFormation。

如果您想要在 Amazon ECS 上自動化基礎設施部署和應用程式，並明確定義和管理環境中的所有 AWS 資源，請使用 AWS CloudFormation 或 Terraform。

AWS 副駕駛 CLI

C AWS opilot CLI (命令列界面) 是一種綜合性工具，可讓客戶直接從其原始程式碼在 Amazon ECS 上部署和操作封裝在容器和環境中的應用程式。使用 C AWS opilot 時，您可以在不了解 Amazon ECS 元素的情況下執行這些操作，例如應用程式負載平衡器、公有子網路、任務、服務 AWS 和叢集。AWS Copilot 代表您根據自以為是的服務模式（例如負載平衡的 Web 服務或後端服務）創建 AWS 資

源，為容器化應用程序提供立即的生產環境。您可以透過 AWS CodePipeline 管道跨多個環境、帳戶或區域進行部署，所有這些都可以在 CLI 中進行管理。透過使用 AWS Copilot，您也可以執行操作員工作，例如檢視記錄和服務的健康狀態。 AWS Copilot 是一種可協助您更輕鬆地管理雲端資源的 all-in-one 工具，讓您可以專注於開發和管理應用程式。

如需詳細資訊，請參閱 使用 AWS 副駕駛命令列界面建立 Amazon ECS 資源。

使用 AWS Copilot 完整的 end-to-end 開發人員工作流程，建立、發行和操作符合基礎架構 AWS 最佳實務的容器應用程式。

AWS CDK

這 AWS Cloud Development Kit (AWS CDK) 是一個開放原始碼軟體開發架構，您可以使用熟悉的程式設計語言建模和佈建雲端應用程式資源。 AWS CDK 通 AWS CloudFormation過以安全，可重複的方式佈建您的資源。使用 CDK，客戶可以使用與建置應用程式相同的語言，以較少的程式碼行產生他們的環境。Amazon ECS 在 CDK 中提供了一個名為 `ecs-patterns` 的模組，這可建立常見架構。可用的模式為 `ApplicationLoadBalancedFargateService()`。此模式會建立叢集、任務定義和其他資源，以便在 AWS Fargate上執行負載平衡的 Amazon ECS 服務。

如需詳細資訊，請參閱 使用建立 Amazon ECS 資源 AWS CDK。

AWS CDK 如果您要以偏好的程式設計語言將基礎結構或架構定義為程式碼，請使用。例如，您可以使用您用於撰寫應用程式的相同語言。

AWS 容器

有時企業客戶可能已經擁有託管於內部部署或 EC2 執行個體或兩者中的應用程式。他們對 Amazon ECS 上特別是容器的可移植性和工具生態系統感興趣，並且需要先進行容器化。 AWS App2Container 可幫助您做到這一點。App2Container (A2C) 是一個命令列工具，用於將 .NET 和 Java 應用程式更新到容器化應用程式內。A2C 會分析並建置在虛擬機器、內部部署或雲端中執行的所有應用程式的詳細目錄。選取要容器化的應用程式之後，A2C 會將應用程式成品和識別的相依性封裝到容器映像中。然後設定網路連接埠並產生 Amazon ECS 任務。最後，它會建立您可以視需要部署或修改的 CloudFormation 範本。

如需詳細資訊，請參閱 AWS App2Container 入門。

如果您的應用程式託管在內部部署或 Amazon EC2 執行個體或兩者中，請使用 App2Container。

Amazon ECS CLI

Amazon ECS CLI 允許您在 Amazon ECS 上運行應用程序，並 AWS Fargate 使用碼頭構成文件格式。您可以快速佈建資源、使用 Amazon ECR 推送及提取映像，以及監控在 Amazon ECS 或 AWS Fargate上執行的應用程式。您也可以在 CLI 內測試在本機執行的容器以及雲端中的容器。

如需詳細資訊，請參閱 開始使用 Amazon ECS 命令列界面。

如果您有 Compose 應用程式並希望將其部署到 Amazon ECS，或在雲端中使用 Amazon ECS 中執行的容器來測試本機容器，請使用 ECS CLI。

Docker Desktop 與 Amazon ECS 整合

AWS 而 Docker 已經協同合作，提供簡化的開發人員體驗，讓您可以使用 Docker 工具直接在 Amazon ECS 上部署和管理容器。您現在可以使用 Docker Desktop 和 Docker Compose 在本機建立與測試您的容器，然後將其部署到 Fargate 上的 Amazon ECS。若要開始使用 Amazon ECS 和 Docker 整合，請下載 Docker Desktop 並選擇性地註冊一個 Docker ID。如需詳細資訊，請參閱 Docker Desktop 和 Docker ID 註冊。

容器的初學者通常會使用 Docker 工具 (例如 Docker CLI 和 Docker Compose) 開始學習容器。這使得在本地測試 AWS 後，使用適用於 Amazon ECS 的 Docker 撰寫 CLI 外掛程式成為執行容器的自然下一步。Docker 提供在 Amazon ECS 上部署容器的演練。如需詳細資訊，請參閱碼頭工人撰寫 CLI-Amazon ECS。

您可以利用其他 Amazon ECS 功能，例如服務探索、負載平衡和其他 AWS 資源，以搭配 Docker 桌面應用程式使用。

您也可以直接從下載用於 Amazon ECS 的碼頭構成 CLI 插件。 GitHub如需詳細資訊，請參閱上的 Amazon ECS 的碼頭工人撰寫 CLI 外掛程式。 GitHub

AWS 開發套件

您也可以使用 AWS 開發套件，透過各種程式設計語言管理 Amazon ECS 資源和操作。開發套件提供模組來協助處理任務，包括下列清單中的任務。

- 加密簽署服務請求
- 重試請求
- 處理錯誤回應

如需可用開發套件的詳細資訊，請參閱 Amazon Web Services 適用工具。

Summary

有許多選項可供選擇，您可以選擇最適合自己的選項。請考量下列選項。

- 如果您喜歡可視化效果，可以使用 AWS Management Console可視化地建立和操作容器。
- 如果您喜歡 CLI，請考慮使用 AWS 副駕駛或. AWS CLI或者，如果您偏好 Docker 生態系統，可以利用 Docker CLI 內部的 ECS 功能來部署到 AWS。部署這些資源之後，您可以透過 CLI 繼續管理它們或透過主控台進行可視化管理。
- 如果您是開發人員，則可以使用與應用程式相同的語言 AWS CDK 來定義基礎結構。您可以使用 CDK 和 C AWS opilot 匯出至 CloudFormation 範本，您可以在其中變更精細設定、新增其他 AWS 資源，以及透過指令碼或 CI/CD 管線 (例如) 自動化部署。 AWS CodePipeline

AWS CLI、開發套件或 ECS API 是自動化 ECS 資源動作的實用工具，因此非常適合部署。若要使用部署應用程式， AWS CloudFormation 您可以使用各種程式設計語言或簡單的文字檔來建模和佈建應用程式所需的所有資源。然後，可以自動且安全的方式，在多個區域和帳戶中部署應用程式。例如，您可以將 ECS 叢集、服務、任務定義或容量提供者定義為檔案中的程式碼，並透過 AWS CLI CloudFormation 命令進行部署。

若要執行作業工作，您可以使用、SDK 或 ECS API 以程式設計方式檢視和管理資源。 AWS CLI`describe-tasks` 或 `list-services` 等命令會顯示最新的中繼資料或所有資源的清單。與部署類似，客戶可以撰寫包含 `update-service` 等命令的自動化，以便在偵測到意外停止的資源時提供糾正動作。您也可以使用 AWS Copilot 操作您的服務。`copilot svc logs` 或 `copilot app show` 等命令提供每個微服務的詳細資訊，或整個應用程式的詳細資訊。

客戶可以使用本文件中提到的任何可用工具，並以各種組合方式使用。ECS 工具提供各種途徑，掌握某些工具之後便可使用其他工具，以滿足您不斷變化的需求。例如，您可以根據需要選擇對資源進行更精細的控制或進行更多的自動化。ECS 也提供各式各樣的工具，以滿足各種需求和專業水平。

使用 AWS 副駕駛命令列界面建立 Amazon ECS 資源

C AWS opilot 命令列界面 (CLI) 命令可簡化從本機開發環境在 Amazon ECS 上生產就緒容器化應用程式的建置、釋放和操作。C AWS opilot CLI 與支援現代應用程式最佳實務的開發人員工作流程保持一致：從使用基礎結構即程式碼到建立代表使用者佈建的 CI/CD 管線。使用 AWS Copilot CLI 作為日常開發和測試週期的一部分，作為. AWS Management Console

AWS 副駕駛目前支援 Linux、macOS 和視窗系統。如需有關 AWS Copilot CLI 最新版本的詳細資訊，請參閱版本。

> **Note**
>
> AWS 副駕駛 CLI 的原始程式碼可在上取得。GitHub我們建議您提交問題，並為想要進行的變更提取請求。但是，Amazon Web Services 目前不支援執行 AWS Copilot 程式碼的已修改複本。通過在 Gitter 上與我們聯繫，或者您可以在GitHub其中打開問題，提供反饋和報告錯誤來報告 AWS Copilot 的問題。

如需有關安裝 AWS 副駕駛 CLI 的資訊，請參閱。安裝 AWS 副駕駛 CLI如需部署範例應用程式的詳細資訊，請參閱使用副駕駛 CLI 部署範例 Amazon ECS AWS 應用程式。副駕 AWS 駛 CLI 的其他文件可在 Copilot 網站上找到AWS 。

安裝 AWS 副駕駛 CLI

您可以使用 Homebrew 或按照以下步驟手動下載二進製文件來安裝 AWS Copilot CLI。

使用自製軟件

以下命令用於使用自 AWS 製軟件在您的 macOS 或 Linux 系統上安裝副駕駛員 CLI。在安裝之前，應先安裝 Homebrew。如需詳細資訊，請參閱 Homebrew。

```
brew install aws/tap/copilot-cli
```

下載二進制

作為自製軟件的替代方案，您可以在 macOS，Windows 或 Linux 系統上手動安裝 AWS 副駕駛員 CLI。對您的操作系統使用以下命令來下載二進製文件。macOS 和 Linux 範例也包括將執行權限套用至二進位檔的指令，並列出說明功能表以確認安裝是否正常運作。

macOS

macOS：

```
sudo curl -Lo /usr/local/bin/copilot https://github.com/aws/copilot-cli/releases/
latest/download/copilot-darwin \
   && sudo chmod +x /usr/local/bin/copilot \
```

```
   && copilot --help
```

macOS ARM 系統：

```
sudo curl -Lo /usr/local/bin/copilot https://github.com/aws/copilot-cli/releases/
latest/download/copilot-darwin-arm64 \
   && sudo chmod +x /usr/local/bin/copilot \
   && copilot --help
```

Linux

對於 Linux x86 (64 位元) 系統：

```
sudo curl -Lo /usr/local/bin/copilot https://github.com/aws/copilot-cli/releases/
latest/download/copilot-linux \
   && sudo chmod +x /usr/local/bin/copilot \
   && copilot --help
```

對於 Linux ARM 系統：

```
sudo curl -Lo /usr/local/bin/copilot https://github.com/aws/copilot-cli/releases/
latest/download/copilot-linux-arm64 \
   && sudo chmod +x /usr/local/bin/copilot \
   && copilot --help
```

Windows

使用 Powershell，執行下列命令：

```
New-Item -Path 'C:\copilot' -ItemType directory; `
  Invoke-WebRequest -OutFile 'C:\copilot\copilot.exe' https://github.com/aws/
copilot-cli/releases/latest/download/copilot-windows.exe
```

(選用) 使用 PGP 簽章驗證手動安裝的 AWS Copilot CLI

AWS 副駕駛 CLI 可執行檔是使用 PGP 簽章以密碼編譯方式簽署的。PGP 簽章可用於驗證 AWS 副駕駛 CLI 執行檔的有效性。使用 GnuPG 工具，透過下列步驟來驗證簽章。

1. 下載並安裝 GnuPG。如需詳細資訊，請參閱 GnuPG 網站。

macOS

建議您使用 Homebrew。參照 Homebrew 網站上的說明，以執行安裝程序。如需詳細資訊，請參閱 Homebrew。Homebrew 安裝完畢後，請從 macOS 終端機使用下列命令。

```
brew install gnupg
```

Linux

請使用套件軟體管理工具在 Linux 上安裝 gpg。

Windows

從 GnuPG 網站下載 Windows 簡易安裝程式並以管理員身分進行安裝。安裝 GnuPG 之後，請關閉並重新開啟管理員。 PowerShell

如需詳細資訊，請參閱「下載 GnuPG」。

2. 驗證 GnuPG 路徑是否已新增到您的環境路徑。

macOS

```
echo $PATH
```

如果未在輸出中發現 GnuPG 路徑，請執行以下命令將其新增到路徑中。

```
PATH=$PATH:<path to GnuPG executable files>
```

Linux

```
echo $PATH
```

如果未在輸出中發現 GnuPG 路徑，請執行以下命令將其新增到路徑中。

```
export PATH=$PATH:<path to GnuPG executable files>
```

Windows

```
Write-Output $Env:PATH
```

如果未在輸出中發現 GnuPG 路徑，請執行以下命令將其新增到路徑中。

```
$Env:PATH += ";<path to GnuPG executable files>"
```

3. 建立本機純文字檔案。

macOS

在終端機中，輸入：

```
touch <public_key_filename.txt>
```

使用開啟檔案 TextEdit。

Linux

在 gedit 等文字編輯器中建立文字檔案。另存為 `public_key_filename.txt`

Windows

在記事本等文字編輯器中建立文字檔案。另存為 `public_key_filename.txt`

4. 新增下列 Amazon ECS PGP 公有金鑰內容，並儲存檔案。

```
-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: GnuPG v2

mQINBFq1SasBEADliGcT1NVJ1ydfN8DqebYYe9ne3dt6jqKFmKowLmm6LLGJe7HU
jGtqhCWRDkN+qPpHqdArRgDZAtn2pXY5fEipHgar4CP8QgRnRMO2fl74lmavr4Vg
7K/KH8VHlq2uRw32/B94XLEgRbGTMdWFdKuxoPCttBQaMj3LGn6Pe+6xVWRkChQu
BoQAhjBQ+bEm0kNy0LjNgjNlnL3UMAG56t8E3LANIgGgEnpNsB1UwfWluPoGZoTx
N+6pHBJrKIL/1v/ETU4FXpYw2zvhWNahxeNRnoYj3uycHkeliCrw4kj0+skizBgO
2K7oVX8Oc3j5+ZilhL/qDLXmUCb2az5cMM1mOoF8EKX5HaNuq1KfwJxqXE6NNIcO
lFTrT7QwD5fMNld3FanLgv/ZnIrsSaqJOL6zRSq8O4LN1OWBVbndExk2Kr+5kFxn
5lBPgfPgRj5hQ+KTHMa9Y8Z7yUc64BJiN6F9Nl7FJuSsfqbdkvRLsQRbcBG9qxX3
rJAEhieJzVMEUNl+EgeCkxj5xuSkNU7zw2c3hQZqEcrADLV+hvFJktOz9Gm6xzbq
lTnWWCz4xrIWtuEBA2qE+MlDheVd78a3gIsEaSTfQq0osYXaQbvlnSWOoc1y/5Zb
zizHTJIhLtUyls9WisP2s0emeHZicVMfW61EgPrJAiupgc7kyZvFt4YwfwARAQAB
tCRBbWF6b24gRUNTIDxlY3Mtc2VjdXJpdHlAYW1hem9uLmNvbT6JAhwEEAECAAYF
AlrjL0YACgkQHivRXs0TaQrg1g/+JppwPqHnlVPmv7lessB8I5UqZeD6p6uVpHd7
Bs3pcPp8BV7BdRbs3sPLt5bV1+rkqOlw+0gZ4Q/ue/YbWtOAt4qY0OcEo0HgcnaX
lsB827QIfZIVtGWMhuh94xzm/SJkvngml6KB3YJNnWP61A9qJ37/VbVVLzvcmazA
McWB4HUMNrhd0JgBCo0gIpqCbpJEvUc02Bjn23eEJsS9kC7OUAHyQkVnx4d9UzXF
```

```
4OoISF6hmQKIBoLnRrAlj5Qvs3GhvHQ0ThYq0Grk/KMJJX2CSqt7tWJ8gk1n3H3Y
SReRXJRnv7DsDDBwFgT6r5Q2HW1TBUvaoZy5hF6maD09nHcNnvBjqADzeT8Tr/Qu
bBCLzkNSYqqkpgtwv7seoD2P4n1giRvDAOEfMZpVkUr+C252IaH1HZFEz+TvBVQM
Y8OWWxmIJW+J6evjo3N1eO19UHv71jvoF8zljbI4bsL2c+QTJmOv7nRqzDQgCWyp
Id/v2dUVVTk1j9omuLBBwNJzQCB+72LcIzJhYmaP1HC4LcKQG+/f41exuItenatK
lEJQhYtyVXcBlh6Yn/wzNg2NWOwb3vqY/F7m6u9ixAwgtIMgPCDE4aJ86zrrXYFz
N2HqkTSQh77Z8KPKmyGopsmN/reMuilPdINb249nA0dzoN+nj+tTFOYCIaLaFyjs
Z0r1QAOJAjkEEwECACMFAlq1SasCGwMHCwkIBwMCAQYVCAIJCgsEFgIDAQIeAQIX
gAAKCRC86dmkLVF4T9iFEACEnkm1dNXsWUx34R3c0vamHrPxvfkyI1FlEUen8D1h
uX9xy6jCEROHWEp0rjGK4QDPgM93sWJ+s1UAKg214QRVzft0y9/DdR+twApA0fzy
uavIthGd6+03jAAo6udYDE+cZC3P7XBbDiYEWk4XAF9I1JjB8hTZUgvXBL046JhG
eM17+crgUyQeetkiOQemLbsbXQ40Bd9V7zf7XJraFd8VrwNUwNb+9KFtgAsc9rk+
YIT/PEf+YOPysgcxI4sTWghtyCulVnuGoskgDv4v73PALU0ieUrvvQVqWMRvhVx1
0X90J7cC1KOyhlEQQ1aFTgmQjmXexVTwIBm8LvysFK6YXM41KjOrlz3+6xBIm/qe
bFyLUnf4WoiuOplAaJhK9pRY+XEnGNxdtN4D26Kd0F+PLkm3Tr3Hy3b1Ok34FlGr
KVHUq1TZD7cvMnnNKEELTUcKX+1mV3an16nmAg/my1JSUt6BNK2rJpY1s/kkSGSE
XQ4zuF2IGCpvBFhYAlt5Un5zwqkwwQR3/n2kwAoDzonJcehDw/C/cGos5D0aIU7I
K2X2aTD3+pA7Mx3IMe2hqmYqRt9X42yF1PIEVRneBRJ3HDezAgJrNh0GQWRQkhIx
gz6/cTR+ekr5TptVszS9few2GpI5bCgBKBisZIssT89aw7mAKWut0Gcm4qM9/yK6
1bkCDQRatUmrARAAxNPvVwreJ2yAiFcUpdRlVhsuOgnxvs1QgsIw3H7+Pacr9Hpe
8uftYZqdC82KeSKhpHq7c8gMTMucIINtH25x9BCc73E33EjCL9Lqov1TL7+QkgHe
T+JIhZwdD8Mx2K+LVVVu/aWkNrfMuNwyDUciSI4D5QHa8T+F8fgN4OTpwYjirzel
5yoICMr9hVcbzDNv/ozKCxjx+XKgnFc3wrnDfJfntfDAT7ecwbUTL+viQKJ646s+
psiqXRYtVvYInEhLVrJ0aV6zHFoigE/Bils6/g7ru1Q6CEHqEw++APs5CcE8VzJu
WAGSVHZgun5Y9N4quR/M9Vm+IPMhTxrAg7rOvyRN9cAXfeSMf77I+XTifigNna8x
t/MOdjXr1fjF4pThEi5u6WsuRdFwjY2azEv3vevodTi4HoJReH6dFRa6y8c+UDgl
2iHiOKIpQqLbHEfQmHcDd2fix+AaJKMnPGNku9qCFEMbgSRJpXz6BfwnY1QuKE+I
R6jA0frUNt2jhiGG/F8RceXzohaaC/Cx7LUCUFWc0n7z32C9/Dtj7I1PMOacdZzz
bjJzRKO/ZDv+UN/c9dwAkllzAyPMwGBkUaY68EBstnIliW34aWm6IiHhxioVPKSp
VJfyiXPO0EXqujtHLAeChfjcns3I12YshT1dv2PafG53fp33ZdzeUgsBo+EAEQEA
AYkCHwQYAQIACQUCWrVJqwIbDAAKCRC86dmkLVF4T+ZdD/9x/8APzgNJF3o3STrF
jvnV1ycyhWYGAeBJiu7wjsNWwzMFOv15tLjB7AqeVxZn+WKDD/mIOQ45OZvnYZuy
X7DR0JszaH9wrYTxZLVruAu+t6UL0y/XQ4L1GZ9QR6+r+7t1Mvbfy7BlHbvX/gYt
Rwe/uwdibI0CagEzyX+2D3kTOlHO5XThbXaNf8AN8zha91Jt2Q2UR2X5T6JcwtMz
FBvZnl3LSmZyE0EQehS2iUurU4uWOpGppuqVnbi0jbCvCHKgDGrqZ0smKNAQng54
F365W3g8AfY48s8XQwzmcliowYX9bT8PZiEi0J4QmQh0aXkpqZyFefuWeOL2R94S
XKzr+gRh3BAULoqF+qK+IUMxTip9KTPNvYDpiC66yBiT6gFDji5Ca9pGpJXrC3xe
TXiKQ8DBWDhBPVPrruLIaenTtZEOsPc4I85yt5U9RoPTStcOr34s3w5yEaJagt6S
Gc5r9ysjkfH6+6rbi1ujxMgROSqtqr+RyB+V9A5/OgtNZc8llK6u4UoOCde8jUUW
vqWKvjJB/Kz3u4zaeNu2ZyyHaOqOuH+TETcW+jsY9IhbEzqN5yQYGi4pVmDkY5vu
lXbJnbqPKpRXgM9BecV9AMbPgbDq/5LnHJJXg+G8YQOgp4lR/hC1TEFdIp5wM8AK
CWsENyt2o1rjgMXiZOMF8A5oBLkCDQRatUuSARAAr77kj7j2QR2SZeOSlFBvV7oS
mFeSNnz9xZssqrsm6bTwSHM6YLDwc7Sdf2esDdyzONETwqrVCg+FxgL8hmo9hS4c
rR6tmrP0mOmptr+xLLsKcaP7ogIXsyZnrEAEsvW8PnfayoiPCdc3cMCR/lTnHFGA
```

```
7EuR/XLBmi7Qg9tByVYQ5Yj5wB9V4B2yeCt3XtzPqeLKvaxl7PNelaHGJQY/xo+m
V0bndxf9IY+4oFJ4blD32WqvyxESo7vW6WBh7oqv3Zbm0yQrr8a6mDBpqLkvWwNI
3kpJR974tg5o5LfDu1BeeyHWPSGm4U/G4JB+JIG1ADy+RmoWEt4BqTCZ/knnoGvw
D5sTCxbKdmuOmhGyTssoG+3OOcGYHV7pWYPhazKHMPm201xKCjH1RfzRULzGKjD+
yMLT1I3AXFmLmZJXikAOlvE3/wgMqCXscbycbLjLD/bXIuFWo3rzoezeXjgi/DJx
jKBAyBTYO5nMcth1O9oaFd9d0HbsOUDkIMnsgGBE766Piro6MHo0T0rXl07Tp4pI
rwuSOsc6XzCzdImj0Wc6axS/HeUKRXWdXJwno5awTwXKRJMXGfhCvSvbcbc2Wx+L
IKvmB7EB4K3fmjFFE67yolmiw2qRcUBfygtH3eL5XZU28MiCpue8Y8GKJoBAUyvf
KeM1rO8Jm3iRAc5a/D0AEQEAAYkEPgQYAQIACQUCWrVLkgIbAgIpCRC86dmkLVF4
T8FdIAQZAQIABgUCWrVLkgAKCRDePL1hra+LjtHYD/9MucxdFe6bXO1dQR4tKhhQ
P0LRqy6zlBY9ILCLowNdGZdqorogUiUymgn3VhEhVtxTOoHcN7qOuM01PNsRnOeS
EYjf8Xrb1clzkD6xULwmOclTb9bBxnBc/4PFvHAbZW3QzusaZniNgkuxt6BTfloS
Of4inq71kjmGK+TlzQ6mUMQUg228NUQC+a84EPqYyAeY1sgvgB7hJBhYL0QAxhcW
6m20Rd8iEc6HyzJ3yCOCsKip/nRWAbf0OvfHfRBp0+m0ZwnJM8cPRFjOqqzFpKH9
HpDmTrC4wKP1+TL52LyEqNh4yZitXmZNV7giSRIkk0eDSko+bFy6VbMzKUMkUJK3
D3eHFAMkujmbfJmSMTJOPGn5SB1HyjCZNx6bhIIbQyEUB9gKCmUFaqXKwKpF6rj0
iQXAJxLR/shZ5Rk96VxzOphUl7T90m/PnUEEPwq8KsBhnMRgxa0RFidDP+n9fgtv
HLmrOqX9zBCVXh0mdWYLrWvmzQFWzG7AoE55fkf8nAEPsalrCdtaNUBHRXA0OQxG
AHMOdJQQvBsmqMvuAdjkDWpFu5y0My5ddU+hiUzUyQLjL5Hhd5LOUDdewlZgIw1j
xrEAUzDKetnemM8GkHxDgg8koev5frmShJuce7vSjKpCNg3EIJSgqMOPFjJuLWtZ
vjHeDNbJy6uNL65ckJy6WhGjEADS2WAW1D6Tfekkc21SsIXk/LqEpLMR/0g5OUif
wcEN1rS9IJXBwIy8MelN9qr5KcKQLmfdfBNEyyceBhyVl0MDyHOKC+7PofMtkGBq
13QieRHv5GJ8LB3fclqHV8pwTTo3Bc8z2g0TjmUYAN/ixETdReDoKavWJYSE9yoM
aaJu279ioVTrwpECse0XkiRyKToTjwOb73CGkBZZpJyqux/rmCV/fp4ALdSW8zbz
FJVORaivhoWwzjpfQKhwcU9lABXi2UvVm14v0AfeI7oiJPSU1zM4fEny4oiIBXlR
zhFNih1UjIu82X16mTm3BwbIga/s1fnQRGzyhqUIMii+mWra23EwjChaxpvjjcUH
5ilLc5Zq781aCYRygYQw+hu5nFkOH1R+Z50Ubxjd/aqUfnGIAX7kPMD3Lof4KldD
Q8ppQriUvxVo+4nPV6rpTy/PyqCLWDjkguHpJsEFsMkwajrAz0QNSAU5CJ0G2Zu4
yxvYlumHCEl7nbFrm0vIiA75Sa8KnywTDsyZsu3XcOcf3g+g1xWTpjJqy2bYXlqz
9uDOWtArWHOis6bq8l9RE6xr1RBVXS6uqgQIZFBGyq66b0dIq4D2JdsUvgEMaHbc
e7tBfeB1CMBdA64e9Rq7bFR7Tvt8gasCZYlNr3lydh+dFHIEkH53HzQe6l88HEic
+0jVnLkCDQRa55wJARAAyLya2Lx6gyoWoJN1a6740q3o8e9d4KggQOfGMTCflmeq
ivuzgN+3DZHN+9ty2KxXMtn0mhHBerZdbNJyjMNT1gAgrhPNB4HtXBXum2wS57WK
DNmade914L7FWTPAWBG2Wn448OEHTqsClICXXWy9IICgclAEyIq0Yq5mAdTEgRJS
Z8t4GpwtDL9gNQyFXaWQmDmkAsCygQMvhAlmu9xOIzQG5CxSnZFk7zcuL60k14Z3
Cmt49k4T/7ZU8goWi8tt+rU78/IL3J/fF9+1civ1OwuUidgfPCSvOUW1JojsdCQA
L+RZJcoXq7lfOFj/eNjeOSstCTDPfTCL+kThE6E5neDtbQHBYkEX1BRiTedsV4+M
ucgiTrdQFWKf89G72xdv8ut9AYYQ2BbEYU+JAYhUH8rYYui2dHKJIgjNvJscuUWb
+QEqJIRleJRhrO+/CHgMs4fZAkWF1VFhKBkcKmEjLn1f7EJJUUW84ZhKXjO/AUPX
1CHsNjziRceuJCJYox1cwsoq6jTE50GiNzcIxTn9xUc0UMKFeggNAFys1K+TDTm3
Bzo8H5ucjCUEmUm9lhkGwqTZgOlRX5eqPX+JBoSaObqhgqCa5IPinKRa6MgoFPHK
6sYKqroYwBGgZm6Js5chpNchvJMs/3WXNOEVg0J3z3vP0DMhxqWm+r+n9zlW8qsA
EQEAAYkEPgQYAQgACQUCWuecCQIbAgIpCRC86dmkLVF4T8FdIAQZAQgABgUCWuec
CQAKCRBQ3szEcQ5hr+ykD/4tOLRHFHXuKUcxgGaubUcVtsFrwBKma1cYjqaPms8u
```

```
6Sk0wfGRI32G/GhOrp0Ts/MOkbObq6VLTh8N5Yc/53MEl8zQFw9Y5AmRoW4PZXER
ujs5s7p4oR7xHMihMjCCBn1bvrR+34YPfgzTcgLiOEFHYT8UTxwnGmXOvNkMM7md
xD3CV5q6VAte8WKBo/220II3fcQlc9r/oWX4kXXkb0v9hoGwKbDJ1tzqTPrp/xFt
yohqnvImpnlz+Q9zXmbrWYL9/g8VCmW/NN2gju2G3Lu/TlFUWIT4v/5OPK6TdeNb
VKJO4+S8bTayqSG9CML1S57KSgCo5HUhQWeSNHI+fpe5oX6FALPT9JLDce8OZz1i
cZZ0MELP37mOOQun0AlmHm/hVzf0f311PtbzcqWaE51tJvgUR/nZFo6Ta3O5Ezhs
3VlEJNQ1Ijf/6DH87SxvAoRIARCuZd0qxBcDK0avpFzUtbJd24lRA3WJpkEiMqKv
RDVZkE4b6TW61f0o+LaVfK6E8oLpixegS4fiqC16mFrOdyRk+RJJfIUyz0WTDVmt
g0U1CO1ezokMSqkJ7724pyjr2xf/r9/sC6aOJwB/lKgZkJfC6NqL7TlxVA31dUga
LEOvEJTTE4gl+tYtfsCDvALCtqL0jduSkUo+RXcBItmXhA+tShW0pbS2Rtx/ixua
KohVD/0R4QxiSwQmICNtm9mw9ydIl1yjYXX5a9x4wMJracNY/LBybJPFnZnT4dYR
z4XjqysDwvvYZByaWoIe3QxjX84V6MlI2IdAT/xImu8gbaCI8tmyfpIrLnPKiR9D
VFYfGBXuAX7+HgPPSFtrHQONCALxxzlbNpS+zxt9r0MiLgcLyspWxSdmoYGZ6nQP
RO5Nm/ZVS+u2imPCRzNUZEMa+dlE6kHx0rS0dPiuJ4O7NtPeYDKkoQtNagspsDvh
cK7CSqAiKMq06UBTxqlTSRkm62eOCtcs3p3OeHu5GRZF1uzTET0ZxYkaPgdrQknx
ozjP5mC7X+45lcCfmcVt94TFNL5HwEUVJpmOgmzILCI8yoDTWzloo+i+fPFsXX4f
kynhE83mSEcr5VHFYrTY3mQXGmNJ3bCLuc/jq7ysGq69xiKmTlUeXFm+aojcRO5i
zyShIRJZ0GZfuzDYFDbMV9amA/YQGygLw//zP5ju5SW26dNxlf3MdFQE5JJ86rn9
MgZ4gcpazHEVUsbZsgkLizRp9imUiH8ymLqAXnfRGlU/LpNSefnvDFTtEIRcpOHc
bhayG0bk51Bd4mioOXnIsKy4j63nJXA27x5EVVHQ1sYRN8Ny4Fdr2tMAmj2O+X+J
qX2yy/UX5nSPU492e2CdZ1UhoU0SRFY3bxKHKB7SDbVeav+K5g==
=Gi5D
-----END PGP PUBLIC KEY BLOCK-----
```

以下為 Amazon ECS PGP 公有金鑰的詳細資訊，以供參考：

```
Key ID: BCE9D9A42D51784F
Type: RSA
Size: 4096/4096
Expires: Never
User ID: Amazon ECS
Key fingerprint: F34C 3DDA E729 26B0 79BE AEC6 BCE9 D9A4 2D51 784F
```

5. 在終端機中透過下列命令使用 Amazon ECS PGP 公有金鑰匯入檔案。

```
gpg --import <public_key_filename.txt>
```

6. 下載 AWS 副駕駛 CLI 簽章。簽章是 ASCII 分離的 PGP 簽章，存放於副檔名為 `.asc` 的檔案中。簽章檔案的名稱會與對應執行檔的名稱相同，並在名稱後加上 `.asc`。

   macOS

   對於 macOS 系統，請使用以下命令。

```
sudo curl -Lo copilot.asc https://github.com/aws/copilot-cli/releases/latest/
download/copilot-darwin.asc
```

Linux

對於 Linux x86 (64 位元) 系統，請執行以下命令。

```
sudo curl -Lo copilot.asc https://github.com/aws/copilot-cli/releases/latest/
download/copilot-linux.asc
```

對於 Linux ARM 系統，請執行以下命令。

```
sudo curl -Lo copilot.asc https://github.com/aws/copilot-cli/releases/latest/
download/copilot-linux-arm64.asc
```

Windows

使用 Powershell，執行下列命令。

```
Invoke-WebRequest -OutFile 'C:\copilot\copilot.asc' https://github.com/aws/
copilot-cli/releases/latest/download/copilot-windows.exe.asc
```

7. 使用以下命令驗證簽章。

- macOS 和 Linux 系統：

```
gpg --verify copilot.asc /usr/local/bin/copilot
```

- Windows 系統：

```
gpg --verify 'C:\copilot\copilot.asc' 'C:\copilot\copilot.exe'
```

預期的輸出結果：

```
gpg: Signature made Tue Apr  3 13:29:30 2018 PDT
gpg:                using RSA key DE3CBD61ADAF8B8E
gpg: Good signature from "Amazon ECS <ecs-security@amazon.com>" [unknown]
gpg: WARNING: This key is not certified with a trusted signature!
gpg:          There is no indication that the signature belongs to the owner.
```

```
Primary key fingerprint: F34C 3DDA E729 26B0 79BE  AEC6 BCE9 D9A4 2D51 784F
     Subkey fingerprint: EB3D F841 E2C9 212A 2BD4  2232 DE3C BD61 ADAF 8B8E
```

> **Important**
>
> 輸出中有警告很正常，不會有問題。這是因為個人 PGP 金鑰 (如果有) 和 Amazon ECS PGP 金鑰之間沒有信任鏈。如需詳細資訊，請參閱「信任網路」。

8. 對於 Windows 安裝，請在 PowerShell 上運行以下命令，以將 AWS 副駕駛目錄添加到路徑中。

```
$Env:PATH += ";<path to Copilot executable files>"
```

使用副駕駛 CLI 部署範例 Amazon ECS AWS 應用程式

安裝 AWS Copilot CLI 之後，您可以依照下列步驟部署範例應用程式、驗證部署，以及清理資源。

必要條件

在開始前，請確定您符合以下先決條件：

- 安裝及設定 AWS CLI。如需詳細資訊，請參閱 AWS 命令列介面。
- 執行`aws configure`以設定 AWS Copilot CLI 用來管理您的應用程式和服務的預設設定檔。
- 安裝和執行 Docker。如需詳細資訊，請參閱 Docker 入門。

使用單一命令部署範例 Amazon ECS 應用程式

1. 使用下列命令部署從 GitHub 儲存庫複製的範例 Web 應用程式。如需 AWS Copilot `init` 及其旗標的詳細資訊，請參閱 AWS Cop ilot 文件。

```
git clone https://github.com/aws-samples/aws-copilot-sample-service.git demo-app &&
 \
cd demo-app &&                                \
copilot init --app demo                       \
  --name api                                  \
  --type 'Load Balanced Web Service'          \
  --dockerfile './Dockerfile'                 \
  --port 80                                   \
  --deploy
```

2. 部署完成後， AWS Copilot CLI 將傳回一個 URL，您可以用來驗證部署。您還可以使用以下命令來驗證應用程序的狀態。

   - 列出您所有的 AWS 副駕駛應用程式。

     ```
     copilot app ls
     ```

   - 顯示有關應用程式中環境和服務的資訊。

     ```
     copilot app show
     ```

   - 顯示有關您的環境的資訊。

     ```
     copilot env ls
     ```

   - 顯示有關服務的資訊，包括端點、容量和相關資源。

     ```
     copilot svc show
     ```

   - 應用程式中所有服務的清單。

     ```
     copilot svc ls
     ```

   - 顯示已部署服務的日誌。

     ```
     copilot svc logs
     ```

   - 顯示服務狀態。

     ```
     copilot svc status
     ```

3. 完成此示範後，請執行下列命令以清除關聯的資源，並避免對未使用的資源產生費用。

   ```
   copilot app delete
   ```

使用建立 Amazon ECS 資源 AWS CDK

這 AWS Cloud Development Kit (AWS CDK) 是一個基礎架構即程式碼 (IAC) 架構，您可以使用您選擇的程式設計語言來定義 AWS 雲端基礎架構。若要定義您自己的雲端基礎設施，要先編寫包含一個或更多堆疊的應用程式 (使用 CDK 支援的其中一種語言)。然後，您將其合成到 AWS CloudFormation 模板

中，並將資源部署到您的 AWS 帳戶. 請遵循本主題中的步驟，透過 Amazon Elastic Container Service (Amazon ECS) 和 Fargate 部署容器化 Web 伺服器 AWS CDK 。

CDK 隨附的「 AWS 建構程式庫」提供了可用來建模提供資源的 AWS 服務 模組。針對熱門服務，程式庫會提供具有智能預設和最佳實務的彙整建構。其中的模組，特別是 `aws-ecs-patterns`，提供了高階抽象概念，讓您可以僅憑幾行程式碼行定義您的容器化服務和所有必要的支援資源。

本主題使用 `ApplicationLoadBalancedFargateService` 建構。此建構會在 Fargate 上一個 Application Load Balancer 的後方部署 Amazon ECS 服務。`aws-ecs-patterns` 模組還包含採用 Network Load Balancer 並在 Amazon EC2 上執行的建構。

在開始這項工作之前，請先設定您的開 AWS CDK 發環境，然後執 AWS CDK 行下列命令來安裝。如需如何設定開 AWS CDK 發環境的指示，請參閱 AWS CDK -先決條件入門。

```
npm install -g aws-cdk
```

> **Note**
>
> 這些指示假設您正在使用 AWS CDK v2。

主題

- 步驟 1：設定您的 AWS CDK 專案
- 步驟 2：使用在 AWS CDK Fargate 上定義容器化網頁伺服器
- 步驟 3：測試 Web 伺服器
- 步驟 4：清理
- 後續步驟

步驟 1：設定您的 AWS CDK 專案

為您的新 AWS CDK 應用程序創建一個目錄並初始化項目。

TypeScript

```
mkdir hello-ecs
cd hello-ecs
```

```
cdk init --language typescript
```

JavaScript

```
mkdir hello-ecs
cd hello-ecs
cdk init --language javascript
```

Python

```
mkdir hello-ecs
cd hello-ecs
cdk init --language python
```

專案啟動後，啟動專案的虛擬環境並安裝 AWS CDK的基準相依性。

```
source .venv/bin/activate
python -m pip install -r requirements.txt
```

Java

```
mkdir hello-ecs
cd hello-ecs
cdk init --language java
```

將此 Maven 專案匯入您的 Java IDE。例如，在 Eclipse 中，使用檔案 > 匯入 > Maven > 現有的 Maven 專案。

C#

```
mkdir hello-ecs
cd hello-ecs
cdk init --language csharp
```

Go

```
mkdir hello-ecs
cd hello-ecs
cdk init --language go
```

> **Note**
>
> AWS CDK 應用程式範本會使用專案目錄的名稱來產生來源檔案和類別的名稱。在此範例中，目錄名為 `hello-ecs`。若使用不同的專案目錄名稱，您的應用程式將與這些說明不相符。

AWS CDK v2 在所謂`aws-cdk-lib`的單個包 AWS 服務 中包含所有人的穩定構造。在初始化專案時，此套件會安裝為相依性套件。使用某些程式設計語言時，套件會在您第一次建置專案時安裝。本主題涵蓋了如何使用 Amazon ECS 模式建構，該建構提供了用於 Amazon ECS 的高階抽象概念。此模組依靠 Amazon ECS 建構和其他建構，來佈建您 Amazon ECS 應用程式所需的來源。

用於將這些程式庫匯入 CDK 應用程式的名稱可能會略有不同，具體取決於您使用的程式設計語言。以下是在每種支援的 CDK 程式設計語言中使用的名稱，供您參考。

TypeScript

```
aws-cdk-lib/aws-ecs
aws-cdk-lib/aws-ecs-patterns
```

JavaScript

```
aws-cdk-lib/aws-ecs
aws-cdk-lib/aws-ecs-patterns
```

Python

```
aws_cdk.aws_ecs
aws_cdk.aws_ecs_patterns
```

Java

```
software.amazon.awscdk.services.ecs
software.amazon.awscdk.services.ecs.patterns
```

C#

```
Amazon.CDK.AWS.ECS
Amazon.CDK.AWS.ECS.Patterns
```

Go

```
github.com/aws/aws-cdk-go/awscdk/v2/awsecs
github.com/aws/aws-cdk-go/awscdk/v2/awsecspatterns
```

步驟 2：使用在 AWS CDK Fargate 上定義容器化網頁伺服器

使用amazon-ecs-sample來自的容器映像檔 DockerHub。此映像包含在 Amazon Linux 2 上執行的 PHP Web 應用程式。

在您建立的 AWS CDK 專案中，編輯包含堆疊定義的檔案，使其類似於下列其中一個範例。

> **Note**
>
> 堆疊是部署的單位。所有資源必須在同一堆疊中，而一個堆疊中的所有資源都要部署在一起。若某資源無法部署，任何已部署的其他資源將會復原。一個 AWS CDK 應用程序可以包含多個堆棧，並且一個堆棧中的資源可以引用另一個堆棧中的資源。

TypeScript

更新 `lib/hello-ecs-stack.ts` 使其相似於以下內容。

```
import * as cdk from 'aws-cdk-lib';
import { Construct } from 'constructs';
import * as ecs from 'aws-cdk-lib/aws-ecs';
import * as ecsp from 'aws-cdk-lib/aws-ecs-patterns';

export class HelloEcsStack extends cdk.Stack {
  constructor(scope: Construct, id: string, props?: cdk.StackProps) {
    super(scope, id, props);

    new ecsp.ApplicationLoadBalancedFargateService(this, 'MyWebServer', {
      taskImageOptions: {
        image: ecs.ContainerImage.fromRegistry('amazon/amazon-ecs-sample'),
      },
      publicLoadBalancer: true
    });
  }
}
```

JavaScript

更新 lib/hello-ecs-stack.js 使其相似於以下內容。

```
const cdk = require('aws-cdk-lib');
const { Construct } = require('constructs');
const ecs = require('aws-cdk-lib/aws-ecs');
const ecsp = require('aws-cdk-lib/aws-ecs-patterns');

class HelloEcsStack extends cdk.Stack {
  constructor(scope = Construct, id = string, props = cdk.StackProps) {
    super(scope, id, props);

    new ecsp.ApplicationLoadBalancedFargateService(this, 'MyWebServer', {
      taskImageOptions: {
        image: ecs.ContainerImage.fromRegistry('amazon/amazon-ecs-sample'),
      },
      publicLoadBalancer: true
    });
  }
}

module.exports = { HelloEcsStack }
```

Python

更新 hello-ecs/hello_ecs_stack.py 使其相似於以下內容。

```
import aws_cdk as cdk
from constructs import Construct

import aws_cdk.aws_ecs as ecs
import aws_cdk.aws_ecs_patterns as ecsp

class HelloEcsStack(cdk.Stack):

    def __init__(self, scope: Construct, construct_id: str, **kwargs) -> None:
        super().__init__(scope, construct_id, **kwargs)

        ecsp.ApplicationLoadBalancedFargateService(self, "MyWebServer",
            task_image_options=ecsp.ApplicationLoadBalancedTaskImageOptions(
                image=ecs.ContainerImage.from_registry("amazon/amazon-ecs-sample")),
            public_load_balancer=True
```

```
        )
```

Java

更新 src/main/java/com.myorg/HelloEcsStack.java 使其相似於以下內容。

```
package com.myorg;

import software.constructs.Construct;
import software.amazon.awscdk.Stack;
import software.amazon.awscdk.StackProps;

import software.amazon.awscdk.services.ecs.ContainerImage;
import
 software.amazon.awscdk.services.ecs.patterns.ApplicationLoadBalancedFargateService;
import
 software.amazon.awscdk.services.ecs.patterns.ApplicationLoadBalancedTaskImageOptions;

public class HelloEcsStack extends Stack {
    public HelloEcsStack(final Construct scope, final String id) {
        this(scope, id, null);
    }

    public HelloEcsStack(final Construct scope, final String id, final StackProps
 props) {
        super(scope, id, props);

        ApplicationLoadBalancedFargateService.Builder.create(this, "MyWebServer")
         .taskImageOptions(ApplicationLoadBalancedTaskImageOptions.builder()
           .image(ContainerImage.fromRegistry("amazon/amazon-ecs-sample"))
           .build())
         .publicLoadBalancer(true)
         .build();
    }
}
```

C#

更新 src/HelloEcs/HelloEcsStack.cs 使其相似於以下內容。

```
using Amazon.CDK;
using Constructs;
using Amazon.CDK.AWS.ECS;
```

```
using Amazon.CDK.AWS.ECS.Patterns;
namespace HelloEcs
{
    public class HelloEcsStack : Stack
    {
        internal HelloEcsStack(Construct scope, string id, IStackProps props =
 null) : base(scope, id, props)
        {
            new ApplicationLoadBalancedFargateService(this, "MyWebServer",
                new ApplicationLoadBalancedFargateServiceProps
                {
                    TaskImageOptions = new ApplicationLoadBalancedTaskImageOptions
                    {
                        Image = ContainerImage.FromRegistry("amazon/amazon-ecs-
sample")
                    },
                    PublicLoadBalancer = true
                });
        }
    }
}
```

Go

更新 hello-ecs.go 使其相似於以下內容。

```
package main

import (
 "github.com/aws/aws-cdk-go/awscdk/v2"
 // "github.com/aws/aws-cdk-go/awscdk/v2/awssqs"
 "github.com/aws/aws-cdk-go/awscdk/v2/awsecs"
 "github.com/aws/aws-cdk-go/awscdk/v2/awsecspatterns"
 "github.com/aws/constructs-go/constructs/v10"
 "github.com/aws/jsii-runtime-go"
)

type HelloEcsStackProps struct {
 awscdk.StackProps
}

func NewHelloEcsStack(scope constructs.Construct, id string, props
 *HelloEcsStackProps) awscdk.Stack {
 var sprops awscdk.StackProps
```

```
 if props != nil {
  sprops = props.StackProps
 }
 stack := awscdk.NewStack(scope, &id, &sprops)

 // The code that defines your stack goes here

 // example resource
 // queue := awssqs.NewQueue(stack, jsii.String("HelloEcsQueue"),
 &awssqs.QueueProps{
 //  VisibilityTimeout: awscdk.Duration_Seconds(jsii.Number(300)),
 // })
 res := awsecspatterns.NewApplicationLoadBalancedFargateService(stack,
 jsii.String("MyWebServer"),
  &awsecspatterns.ApplicationLoadBalancedFargateServiceProps{
   TaskImageOptions: &awsecspatterns.ApplicationLoadBalancedTaskImageOptions{
    Image: awsecs.ContainerImage_FromRegistry(jsii.String("amazon/amazon-ecs-
sample"), &awsecs.RepositoryImageProps{}),
   },
  },
 )
 awscdk.NewCfnOutput(stack, jsii.String("LoadBalancerDNS"),
 &awscdk.CfnOutputProps{Value: res.LoadBalancer().LoadBalancerDnsName()})

 return stack
}

func main() {
 defer jsii.Close()

 app := awscdk.NewApp(nil)

 NewHelloEcsStack(app, "HelloEcsStack", &HelloEcsStackProps{
  awscdk.StackProps{
   Env: env(),
  },
 })

 app.Synth(nil)
}

// env determines the AWS environment (account+region) in which our stack is to
// be deployed. For more information see: https://docs.aws.amazon.com/cdk/latest/
guide/environments.html
```

```
func env() *awscdk.Environment {
 // If unspecified, this stack will be "environment-agnostic".
 // Account/Region-dependent features and context lookups will not work, but a
 // single synthesized template can be deployed anywhere.
 //---------------------------------------------------------------------------
 return nil

 // Uncomment if you know exactly what account and region you want to deploy
 // the stack to. This is the recommendation for production stacks.
 //---------------------------------------------------------------------------
 // return &awscdk.Environment{
 //  Account: jsii.String("123456789012"),
 //  Region:  jsii.String("us-east-1"),
 // }

 // Uncomment to specialize this stack for the AWS Account and Region that are
 // implied by the current CLI configuration. This is recommended for dev
 // stacks.
 //---------------------------------------------------------------------------
 // return &awscdk.Environment{
 //  Account: jsii.String(os.Getenv("CDK_DEFAULT_ACCOUNT")),
 //  Region:  jsii.String(os.Getenv("CDK_DEFAULT_REGION")),
 // }
}
```

前面的簡短程式碼片段包括以下內容：

- 服務的邏輯名稱：`MyWebServer`。
- 從以下位置取得的容器影像 DockerHub：`amazon/amazon-ecs-sample`。
- 其他相關資訊，例如負載平衡器具有公有地址，而且可以從網際網路存取的事實。

AWS CDK 將建立部署 Web 伺服器所需的所有資源，包括下列資源。此範例中省略了這些資源。

- Amazon ECS 叢集
- Amazon VPC 和 Amazon EC2 執行個體
- Auto Scaling 群組
- Application Load Balancer
- (IAM) 角色和政策

部分自動佈建的資源是由堆疊中定義的全部 Amazon ECS 服務進行共享。

保存來源檔案，然後在應用程式的主目錄中執行 `cdk synth` 指令。 AWS CDK 運行該應用程序並從中合成 AWS CloudFormation 模板，然後顯示模板。範本是大約 600 行的 YAML 檔案。檔案的開頭如此處所示。您的範本可能與此範例不同。

```
Resources:
  MyWebServerLB3B5FD3AB:
    Type: AWS::ElasticLoadBalancingV2::LoadBalancer
    Properties:
      LoadBalancerAttributes:
        - Key: deletion_protection.enabled
          Value: "false"
      Scheme: internet-facing
      SecurityGroups:
        - Fn::GetAtt:
            - MyWebServerLBSecurityGroup01B285AA
            - GroupId
      Subnets:
        - Ref: EcsDefaultClusterMnL3mNNYNVpcPublicSubnet1Subnet3C273B99
        - Ref: EcsDefaultClusterMnL3mNNYNVpcPublicSubnet2Subnet95FF715A
      Type: application
    DependsOn:
      - EcsDefaultClusterMnL3mNNYNVpcPublicSubnet1DefaultRouteFF4E2178
      - EcsDefaultClusterMnL3mNNYNVpcPublicSubnet2DefaultRouteB1375520
    Metadata:
      aws:cdk:path: HelloEcsStack/MyWebServer/LB/Resource
  MyWebServerLBSecurityGroup01B285AA:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupDescription: Automatically created Security Group for ELB
 HelloEcsStackMyWebServerLB06757F57
      SecurityGroupIngress:
        - CidrIp: 0.0.0.0/0
          Description: Allow from anyone on port 80
          FromPort: 80
          IpProtocol: tcp
          ToPort: 80
      VpcId:
        Ref: EcsDefaultClusterMnL3mNNYNVpc7788A521
    Metadata:
      aws:cdk:path: HelloEcsStack/MyWebServer/LB/SecurityGroup/Resource
```

```
# and so on for another few hundred lines
```

若要在您的中部署服務 AWS 帳戶，請在應用程式的主目錄中執行`cdk deploy`命令。系統會要求您核准 AWS CDK 產生的 IAM 政策。

部署需要幾分鐘的時間，在此期間 AWS CDK 建立數個資源。部署輸出的最後幾行包含負載平衡器的公開主機名稱，和您的新 Web 伺服器 URL。如下所示：

```
Outputs:
HelloEcsStack.MyWebServerLoadBalancerDNSXXXXXXX = Hello-MyWeb-ZZZZZZZZZZZZZ-
ZZZZZZZZZZ.us-west-2.elb.amazonaws.com
HelloEcsStack.MyWebServerServiceURLYYYYYYYY = http://Hello-MyWeb-ZZZZZZZZZZZZZ-
ZZZZZZZZZZ.us-west-2.elb.amazonaws.com
```

步驟 3：測試 Web 伺服器

從部署輸出複製 URL，並貼到您的 Web 瀏覽器。此時會顯示下列來自 Web 伺服器的歡迎使用訊息。

步驟 4：清理

在您完成使用 Web 伺服器之後，於應用程式的主目錄中執行 `cdk destroy` 指令來用 CDK 結束服務。這樣做可以防止您未來意外產生任何費用。

後續步驟

若要進一步了解如何使用開發 AWS 基礎結構 AWS CDK，請參閱開發AWS CDK 人員指南。

如需以您選擇的語言撰寫 AWS CDK 應用程式的相關資訊，請參閱下列內容：

TypeScript

- 使用 AWS CDK 中 TypeScript

JavaScript

使用 AWS CDK 中 JavaScript

Python

AWS CDK 在 Python 中使用

Java

AWS CDK 在 Java 中使用

C#

AWS CDK 在 C# 中使用

Go

AWS CDK 在 Go 中使用

如需本主題中使用之「 AWS 建構程式庫」模組的詳細資訊，請參閱下列 AWS CDK API 參考概觀。

- aws-ecs
- aws-ecs-patterns

使用創建 Amazon ECS 資源 AWS CloudFormation

Amazon ECS 與整合，這是一項服務 AWS CloudFormation，您可以使用該服務使用您定義的範本建立 AWS 資源模型和設定資源。如此一來，您可以花更少的時間建立並管理資源和基礎設施。您可以使用 AWS CloudFormation建立範本來描述所需的所有 AWS 資源，例如特定的 Amazon ECS 叢集。然後， AWS CloudFormation 會負責佈建和設定這些資源。

使用時 AWS CloudFormation，您可以重複使用範本，以一致且可重複的方式設定 Amazon ECS 資源。您一次描述您的資源，然後在多 AWS 帳戶 個和之間再次佈建相同的資源 AWS 區域。

AWS CloudFormation 模板

若要佈建和設定 Amazon ECS 和相關服務的資源，請確定您熟悉AWS CloudFormation 範本。 AWS CloudFormation 範本是 JSON 或 YAML 格式的文字檔案，用來描述您要在 AWS CloudFormation 堆疊中佈建的資源。如果您不熟悉 JSON 或 YAML 格式，或兩者都不熟悉，可以使用 AWS CloudFormation 設計師開始使 AWS CloudFormation 用範本。如需詳細資訊，請參閱什麼是 AWS CloudFormation 設計師？ 在《AWS CloudFormation 使用者指南》中。

Amazon ECS 支援在 AWS CloudFormation中建立叢集、任務定義、服務和任務集。下列範例示範如何使用 AWS CLI來以範本建立資源。您也可以使用 AWS CloudFormation 主控台建立這些資源。如欲進一步了解如何使用 AWS CloudFormation 主控台建立資源，請參閱 AWS CloudFormation 使用者指南。

範例範本

使用獨立堆疊建立 Amazon ECS 資源

下列範例示範如何為每個資源使用獨立堆疊來建立 Amazon ECS 資源。

任務定義

您可以使用下列範本來建立 Fargate Linux 任務。

JSON

```
  {
    "AWSTemplateFormatVersion": "2010-09-09",
    "Resources": {
         "ECSTaskDefinition": {
            "Type": "AWS::ECS::TaskDefinition",
            "Properties": {
                "ContainerDefinitions": [
                    {
                        "Command": [
                            "/bin/sh -c \"echo '<html> <head> <title>Amazon ECS
 Sample App</title> <style>body {margin-top: 40px; background-color: #333;} </style>
 </head><body> <div style=color:white;text-align:center> <h1>Amazon ECS Sample App</
h1> <h2>Congratulations!</h2> <p>Your application is now running on a container in
 Amazon ECS.</p> </div></body></html>' >  /usr/local/apache2/htdocs/index.html &&
 httpd-foreground\""
                        ],
                        "EntryPoint": [
                            "sh",
                            "-c"
                        ],
                        "Essential": true,
                        "Image": "httpd:2.4",
                        "LogConfiguration": {
                            "LogDriver": "awslogs",
                            "Options": {
                                "awslogs-group": "/ecs/fargate-task-definition",
```

```
                                "awslogs-region": "us-east-1",
                                "awslogs-stream-prefix": "ecs"
                            }
                        },
                        "Name": "sample-fargate-app",
                        "PortMappings": [
                            {
                                "ContainerPort": 80,
                                "HostPort": 80,
                                "Protocol": "tcp"
                            }
                        ]
                    }
                ],
                "Cpu": 256,
                "ExecutionRoleArn": "arn:aws:iam::aws_account_id:role/
ecsTaskExecutionRole",
                "Family": "task-definition-cfn",
                "Memory": 512,
                "NetworkMode": "awsvpc",
                "RequiresCompatibilities": [
                    "FARGATE"
                ],
                "RuntimePlatform": {
                    "OperatingSystemFamily": "LINUX"
                }
            }

        }
        }
    }
```

YAML

```
AWSTemplateFormatVersion: 2010-09-09
Resources:
  ECSTaskDefinition:
    Type: 'AWS::ECS::TaskDefinition'
    Properties:
      ContainerDefinitions:
        - Command:
            - >-
              /bin/sh -c "echo '<html> <head> <title>Amazon ECS Sample
```

```
            App</title> <style>body {margin-top: 40px; background-color:
            #333;} </style> </head><body> <div
            style=color:white;text-align:center> <h1>Amazon ECS Sample
            App</h1> <h2>Congratulations!</h2> <p>Your application is now
            running on a container in Amazon ECS.</p> </div></body></html>' >
            /usr/local/apache2/htdocs/index.html && httpd-foreground"
        EntryPoint:
          - sh
          - '-c'
        Essential: true
        Image: 'httpd:2.4'
        LogConfiguration:
          LogDriver: awslogs
          Options:
            awslogs-group: /ecs/fargate-task-definition
            awslogs-region: us-east-1
            awslogs-stream-prefix: ecs
        Name: sample-fargate-app
        PortMappings:
          - ContainerPort: 80
            HostPort: 80
            Protocol: tcp
    Cpu: 256
    ExecutionRoleArn: 'arn:aws:iam::aws_account_id:role/ecsTaskExecutionRole'
    Family: task-definition-cfn
    Memory: 512
    NetworkMode: awsvpc
    RequiresCompatibilities:
      - FARGATE
    RuntimePlatform:
      OperatingSystemFamily: LINUX
```

叢集

您可以使用下列範本來建立一個空叢集。

JSON

```
{
    "AWSTemplateFormatVersion": "2010-09-09",
    "Resources": {
        "ECSCluster": {
            "Type": "AWS::ECS::Cluster",
```

```
            "Properties": {
                "ClusterName": "MyEmptyCluster"
            }
        }
    }
}
```

YAML

```
AWSTemplateFormatVersion: 2010-09-09
Resources:
  ECSCluster:
    Type: 'AWS::ECS::Cluster'
    Properties:
      ClusterName: MyEmptyCluster
```

在同一堆疊中建立多個 Amazon ECS 資源

您可以使用下列範例範本，在同一堆疊中建立多個 Amazon ECS 資源。範本會建立一個名為 CFNCluster 的 Amazon ECS 叢集。此叢集包含一個用於設定 Web 伺服器的 Linux Fargate 任務定義。範本也會建立名為 cfn-service 的服務，該服務會啟動並維護由任務定義所定義的任務。在使用此範本之前，請確保服務的 NetworkConfiguration 中的子網路和安全群組 ID 皆屬於相同的 VPC，且安全群組具有必要的規則。如需安全群組規則的詳細資訊，請參閱《Amazon VPC 使用者指南》中的安全群組規則。

JSON

```
{
    "AWSTemplateFormatVersion": "2010-09-09",
    "Resources": {
        "ECSCluster": {
            "Type": "AWS::ECS::Cluster",
            "Properties": {
                "ClusterName": "CFNCluster"
            }
        },
        "ECSTaskDefinition": {
            "Type": "AWS::ECS::TaskDefinition",
            "Properties": {
                "ContainerDefinitions": [
                    {
```

```
                        "Command": [
                            "/bin/sh -c \"echo '<html> <head> <title>Amazon ECS
 Sample App</title> <style>body {margin-top: 40px; background-color: #333;} </style>
 </head><body> <div style=color:white;text-align:center> <h1>Amazon ECS Sample App</
h1> <h2>Congratulations!</h2> <p>Your application is now running on a container in
 Amazon ECS.</p> </div></body></html>' >  /usr/local/apache2/htdocs/index.html &&
 httpd-foreground\""
                        ],
                        "EntryPoint": [
                            "sh",
                            "-c"
                        ],
                        "Essential": true,
                        "Image": "httpd:2.4",
                        "LogConfiguration": {
                            "LogDriver": "awslogs",
                            "Options": {
                                "awslogs-group": "/ecs/fargate-task-definition",
                                "awslogs-region": "us-east-1",
                                "awslogs-stream-prefix": "ecs"
                            }
                        },
                        "Name": "sample-fargate-app",
                        "PortMappings": [
                            {
                                "ContainerPort": 80,
                                "HostPort": 80,
                                "Protocol": "tcp"
                            }
                        ]
                    }
                ],
                "Cpu": 256,
                "ExecutionRoleArn": "arn:aws:iam::aws_account_id::role/
ecsTaskExecutionRole",
                "Family": "task-definition-cfn",
                "Memory": 512,
                "NetworkMode": "awsvpc",
                "RequiresCompatibilities": [
                    "FARGATE"
                ],
                "RuntimePlatform": {
                    "OperatingSystemFamily": "LINUX"
                }
```

```
            }
        },
        "ECSService": {
            "Type": "AWS::ECS::Service",
            "Properties": {
                "ServiceName": "cfn-service",
                "Cluster": {
                    "Ref": "ECSCluster"
                },
                "DesiredCount": 1,
                "LaunchType": "FARGATE",
                "NetworkConfiguration": {
                    "AwsvpcConfiguration": {
                        "AssignPublicIp": "ENABLED",
                        "SecurityGroups": [
                            "sg-abcdef01234567890"
                        ],
                        "Subnets": [
                            "subnet-abcdef01234567890"
                        ]
                    }
                },
                "TaskDefinition": {
                    "Ref": "ECSTaskDefinition"
                }
            }
        }
    }
}
```

YAML

```
AWSTemplateFormatVersion: 2010-09-09
Resources:
  ECSCluster:
    Type: 'AWS::ECS::Cluster'
    Properties:
      ClusterName: CFNCluster
  ECSTaskDefinition:
    Type: 'AWS::ECS::TaskDefinition'
    Properties:
      ContainerDefinitions:
        - Command:
```

```
            - >-
              /bin/sh -c "echo '<html> <head> <title>Amazon ECS Sample
              App</title> <style>body {margin-top: 40px; background-color:
              #333;} </style> </head><body> <div
              style=color:white;text-align:center> <h1>Amazon ECS Sample
              App</h1> <h2>Congratulations!</h2> <p>Your application is now
              running on a container in Amazon ECS.</p> </div></body></html>' >
              /usr/local/apache2/htdocs/index.html && httpd-foreground"
          EntryPoint:
            - sh
            - '-c'
          Essential: true
          Image: 'httpd:2.4'
          LogConfiguration:
            LogDriver: awslogs
            Options:
              awslogs-group: /ecs/fargate-task-definition
              awslogs-region: us-east-1
              awslogs-stream-prefix: ecs
          Name: sample-fargate-app
          PortMappings:
            - ContainerPort: 80
              HostPort: 80
              Protocol: tcp
      Cpu: 256
      ExecutionRoleArn: 'arn:aws:iam::aws_account_id:role/ecsTaskExecutionRole'
      Family: task-definition-cfn
      Memory: 512
      NetworkMode: awsvpc
      RequiresCompatibilities:
        - FARGATE
      RuntimePlatform:
        OperatingSystemFamily: LINUX
  ECSService:
    Type: 'AWS::ECS::Service'
    Properties:
      ServiceName: cfn-service
      Cluster: !Ref ECSCluster
      DesiredCount: 1
      LaunchType: FARGATE
      NetworkConfiguration:
        AwsvpcConfiguration:
          AssignPublicIp: ENABLED
          SecurityGroups:
```

```
            - sg-abcdef01234567890
          Subnets:
            - subnet-abcdef01234567890
      TaskDefinition: !Ref ECSTaskDefinition
```

使用 AWS CLI 從範本建立資源

下列指令會使用名為 ecs-template-body.json 的範本內文檔，建立一個名為 ecs-stack 的堆疊。請確定範本內文檔為 JSON 或 YAML 格式。檔案的位置已指定於 --template-body 參數。在此情況下，範本內文檔位於目前的目錄。

```
aws cloudformation create-stack \
      --stack-name ecs-stack \
      --template-body file://ecs-template-body.json
```

若要確保正確建立資源，請檢查 Amazon ECS 主控台，或使用下列指令：

- 下列指令可列出所有任務定義。

  ```
  aws ecs list-task-definitions
  ```

- 下列指令可列出所有叢集。

  ```
  aws ecs list-clusters
  ```

- 下列指令可列出叢集 *CFNCluster* 中定義的所有服務。將 *CFNCluster* 替換為您要在其中建立服務的叢集名稱。

  ```
  aws ecs list-services \
        --cluster CFNCluster
  ```

進一步了解 AWS CloudFormation

若要進一步了解 AWS CloudFormation，請參閱下列資源：

- AWS CloudFormation
- AWS CloudFormation 使用者指南
- AWS CloudFormation 指令行介面使用者指南

開始使用 Amazon ECS 命令列界面

Amazon ECS 已發行 C AWS opilot，這是一種命令列界面 (CLI) 工具，可簡化從本機開發環境在 Amazon ECS 上建置、發行和操作生產就緒容器化應用程式。如需詳細資訊，請參閱 使用 AWS 副駕駛命令列界面建立 Amazon ECS 資源。

Amazon Elastic Container Service (Amazon ECS) 命令列介面 (CLI) 可提供高階命令，從本機開發環境中簡化叢集和任務的建立、更新以及監控。Amazon ECS CLI 支援 Docker Compose 檔案，它是一個常見的開源規格，用於定義和執行多容器應用程式。於每日開發和測試週期使用 ECS CLI，做為 AWS Management Console的替代方法。

最新版的 Amazon ECS CLI 僅支援 Docker Compose 檔案語法版本 1、2 和 3 的主要版本。Compose 檔案中指定的版本必須是 "1"、"1.0"、"2"、"2.0"、"3" 或 "3.0" 字串。不支援 Docker Compose 次要版本。

Amazon ECS CLI 的原始程式碼可在上 GitHub取得。不再積極開發此工具。

安裝 Amazon ECS CLI

Amazon ECS 已發行 C AWS opilot，這是一種命令列界面 (CLI) 工具，可簡化從本機開發環境在 Amazon ECS 上建置、發行和操作生產就緒容器化應用程式。如需詳細資訊，請參閱 使用 AWS 副駕駛命令列界面建立 Amazon ECS 資源。

下列步驟指出如何在您的 macOS、Linux 或 Windows 系統上安裝 Amazon ECS CLI。

安裝 Amazon ECS CLI

1. 下載 Amazon ECS CLI 二進位文件。

   macOS

   ```
   sudo curl -Lo /usr/local/bin/ecs-cli https://amazon-ecs-cli.s3.amazonaws.com/
   ecs-cli-darwin-amd64-latest
   ```

Linux

```
sudo curl -Lo /usr/local/bin/ecs-cli https://amazon-ecs-cli.s3.amazonaws.com/
ecs-cli-linux-amd64-latest
```

Windows

打開窗口， PowerShell 然後輸入以下命令。

> **Note**
>
> 如果您遇到權限問題，請確定您在 Windows 上擁有系統管理員存取權，而且您以系統管理員身分執行 PowerShell。

```
New-Item -Path 'C:\Program Files\Amazon\ECSCLI' -ItemType Directory
Invoke-WebRequest -OutFile 'C:\Program Files\Amazon\ECSCLI\ecs-cli.exe' https://
amazon-ecs-cli.s3.amazonaws.com/ecs-cli-windows-amd64-latest.exe
```

2. 使用 PGP 簽章來驗證 Amazon ECS CLI。Amazon ECS CLI 可執行檔會使用 PGP 簽章進行加密簽署。PGP 簽章可用來驗證 Amazon ECS CLI 可執行檔的有效性。使用 GnuPG 工具，透過下列步驟來驗證簽章。

   a. 下載並安裝 GnuPG。如需詳細資訊，請參閱 GnuPG 網站。

      macOS

      建議您使用 Homebrew。參照 Homebrew 網站上的說明，以執行安裝程序。如需詳細資訊，請參閱 Homebrew。Homebrew 安裝完畢後，請從 macOS 終端機使用下列命令。

      ```
      brew install gnupg
      ```

      Linux

      請使用套件軟體管理工具在 Linux 上安裝 gpg。

      Windows

      從 GnuPG 網站下載 Windows 簡易安裝程式並以管理員身分進行安裝。安裝 GnuPG 之後，請關閉並重新開啟管理員。 PowerShell

如需詳細資訊，請參閱「下載 GnuPG」。

b. 驗證 GnuPG 路徑是否已新增到您的環境路徑。

macOS

```
echo $PATH
```

如果未在輸出中發現 GnuPG 路徑，請執行以下命令將其新增到路徑中。

```
PATH=$PATH:<path to GnuPG executable files>
```

Linux

```
echo $PATH
```

如果未在輸出中發現 GnuPG 路徑，請執行以下命令將其新增到路徑中。

```
export PATH=$PATH:<path to GnuPG executable files>
```

Windows

```
Write-Output $Env:PATH
```

如果未在輸出中發現 GnuPG 路徑，請執行以下命令將其新增到路徑中。

```
$Env:PATH += ";<path to GnuPG executable files>"
```

c. 建立本機純文字檔案。

macOS

在終端機中，輸入：

```
touch <public_key_filename.txt>
```

使用開啟檔案 TextEdit。

Linux

在 gedit 等文字編輯器中建立文字檔案。另存為 public_key_filename.txt

Windows

在記事本等文字編輯器中建立文字檔案。另存為 public_key_filename.txt

d. 新增下列 Amazon ECS PGP 公有金鑰內容，並儲存檔案。

```
-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: GnuPG v2

mQINBFq1SasBEADliGcT1NVJ1ydfN8DqebYYe9ne3dt6jqKFmKowLmm6LLGJe7HU
jGtqhCWRDkN+qPpHqdArRgDZAtn2pXY5fEipHgar4CP8QgRnRMO2fl74lmavr4Vg
7K/KH8VHlq2uRw32/B94XLEgRbGTMdWFdKuxoPCttBQaMj3LGn6Pe+6xVWRkChQu
BoQAhjBQ+bEm0kNy0LjNgjNlnL3UMAG56t8E3LANIgGgEnpNsB1UwfWluPoGZoTx
N+6pHBJrKIL/1v/ETU4FXpYw2zvhWNahxeNRnoYj3uycHkeliCrw4kj0+skizBgO
2K7oVX8Oc3j5+ZilhL/qDLXmUCb2az5cMM1mOoF8EKX5HaNuq1KfwJxqXE6NNIcO
lFTrT7QwD5fMNld3FanLgv/ZnIrsSaqJOL6zRSq8O4LN1OWBVbndExk2Kr+5kFxn
5lBPgfPgRj5hQ+KTHMa9Y8Z7yUc64BJiN6F9Nl7FJuSsfqbdkvRLsQRbcBG9qxX3
rJAEhieJzVMEUNl+EgeCkxj5xuSkNU7zw2c3hQZqEcrADLV+hvFJktOz9Gm6xzbq
lTnWWCz4xrIWtuEBA2qE+MlDheVd78a3gIsEaSTfQq0osYXaQbvlnSWOoc1y/5Zb
zizHTJIhLtUyls9WisP2s0emeHZicVMfW61EgPrJAiupgc7kyZvFt4YwfwARAQAB
tCRBbWF6b24gRUNTIDxlY3Mtc2VjdXJpdHlAYW1hem9uLmNvbT6JAhwEEAECAAYF
AlrjL0YACgkQHivRXs0TaQrg1g/+JppwPqHnlVPmv7lessB8I5UqZeD6p6uVpHd7
Bs3pcPp8BV7BdRbs3sPLt5bV1+rkqOlw+0gZ4Q/ue/YbWtOAt4qY0OcEo0HgcnaX
lsB827QIfZIVtGWMhuh94xzm/SJkvngml6KB3YJNnWP61A9qJ37/VbVVLzvcmazA
McWB4HUMNrhd0JgBCo0gIpqCbpJEvUc02Bjn23eEJsS9kC7OUAHyQkVnx4d9UzXF
4OoISF6hmQKIBoLnRrAlj5Qvs3GhvHQ0ThYq0Grk/KMJJX2CSqt7tWJ8gk1n3H3Y
SReRXJRnv7DsDDBwFgT6r5Q2HW1TBUvaoZy5hF6maD09nHcNnvBjqADzeT8Tr/Qu
bBCLzkNSYqqkpgtwv7seoD2P4n1giRvDAOEfMZpVkUr+C252IaH1HZFEz+TvBVQM
Y8OWWxmIJW+J6evjo3N1eO19UHv71jvoF8zljbI4bsL2c+QTJmOv7nRqzDQgCWyp
Id/v2dUVVTk1j9omuLBBwNJzQCB+72LcIzJhYmaP1HC4LcKQG+/f41exuItenatK
lEJQhYtyVXcBlh6Yn/wzNg2NWOwb3vqY/F7m6u9ixAwgtIMgPCDE4aJ86zrrXYFz
N2HqkTSQh77Z8KPKmyGopsmN/reMuilPdINb249nA0dzoN+nj+tTFOYCIaLaFyjs
Z0r1QAOJAjkEEwECACMFAlq1SasCGwMHCwkIBwMCAQYVCAIJCgsEFgIDAQIeAQIX
gAAKCRC86dmkLVF4T9iFEACEnkm1dNXsWUx34R3c0vamHrPxvfkyI1FlEUen8D1h
uX9xy6jCEROHWEp0rjGK4QDPgM93sWJ+s1UAKg214QRVzft0y9/DdR+twApA0fzy
uavIthGd6+03jAAo6udYDE+cZC3P7XBbDiYEWk4XAF9I1JjB8hTZUgvXBL046JhG
eM17+crgUyQeetkiOQemLbsbXQ40Bd9V7zf7XJraFd8VrwNUwNb+9KFtgAsc9rk+
YIT/PEf+YOPysgcxI4sTWghtyCulVnuGoskgDv4v73PALU0ieUrvvQVqWMRvhVx1
0X90J7cC1KOyhlEQQ1aFTgmQjmXexVTwIBm8LvysFK6YXM41KjOrlz3+6xBIm/qe
bFyLUnf4WoiuOplAaJhK9pRY+XEnGNxdtN4D26Kd0F+PLkm3Tr3Hy3b1Ok34FlGr
```

```
KVHUq1TZD7cvMnnNKEELTUcKX+1mV3an16nmAg/my1JSUt6BNK2rJpY1s/kkSGSE
XQ4zuF2IGCpvBFhYAlt5Un5zwqkwwQR3/n2kwAoDzonJcehDw/C/cGos5D0aIU7I
K2X2aTD3+pA7Mx3IMe2hqmYqRt9X42yF1PIEVRneBRJ3HDezAgJrNh0GQWRQkhIx
gz6/cTR+ekr5TptVszS9few2GpI5bCgBKBisZIssT89aw7mAKWut0Gcm4qM9/yK6
1bkCDQRatUmrARAAxNPvVwreJ2yAiFcUpdRlVhsuOgnxvs1QgsIw3H7+Pacr9Hpe
8uftYZqdC82KeSKhpHq7c8gMTMucIINtH25x9BCc73E33EjCL9Lqov1TL7+QkgHe
T+JIhZwdD8Mx2K+LVVVu/aWkNrfMuNwyDUciSI4D5QHa8T+F8fgN4OTpwYjirzel
5yoICMr9hVcbzDNv/ozKCxjx+XKgnFc3wrnDfJfntfDAT7ecwbUTL+viQKJ646s+
psiqXRYtVvYInEhLVrJ0aV6zHFoigE/Bils6/g7ru1Q6CEHqEw++APs5CcE8VzJu
WAGSVHZgun5Y9N4quR/M9Vm+IPMhTxrAg7rOvyRN9cAXfeSMf77I+XTifigNna8x
t/MOdjXr1fjF4pThEi5u6WsuRdFwjY2azEv3vevodTi4HoJReH6dFRa6y8c+UDgl
2iHiOKIpQqLbHEfQmHcDd2fix+AaJKMnPGNku9qCFEMbgSRJpXz6BfwnY1QuKE+I
R6jA0frUNt2jhiGG/F8RceXzohaaC/Cx7LUCUFWc0n7z32C9/Dtj7I1PMOacdZzz
bjJzRKO/ZDv+UN/c9dwAkllzAyPMwGBkUaY68EBstnIliW34aWm6IiHhxioVPKSp
VJfyiXPO0EXqujtHLAeChfjcns3I12YshT1dv2PafG53fp33ZdzeUgsBo+EAEQEA
AYkCHwQYAQIACQUCWrVJqwIbDAAKCRC86dmkLVF4T+ZdD/9x/8APzgNJF3o3STrF
jvnV1ycyhWYGAeBJiu7wjsNWwzMFOv15tLjB7AqeVxZn+WKDD/mIOQ45OZvnYZuy
X7DR0JszaH9wrYTxZLVruAu+t6UL0y/XQ4L1GZ9QR6+r+7t1Mvbfy7BlHbvX/gYt
Rwe/uwdibI0CagEzyX+2D3kTOlHO5XThbXaNf8AN8zha91Jt2Q2UR2X5T6JcwtMz
FBvZnl3LSmZyE0EQehS2iUurU4uWOpGppuqVnbi0jbCvCHKgDGrqZ0smKNAQng54
F365W3g8AfY48s8XQwzmcliowYX9bT8PZiEi0J4QmQh0aXkpqZyFefuWeOL2R94S
XKzr+gRh3BAULoqF+qK+IUMxTip9KTPNvYDpiC66yBiT6gFDji5Ca9pGpJXrC3xe
TXiKQ8DBWDhBPVPrruLIaenTtZEOsPc4I85yt5U9RoPTStcOr34s3w5yEaJagt6S
Gc5r9ysjkfH6+6rbi1ujxMgROSqtqr+RyB+V9A5/OgtNZc8llK6u4UoOCde8jUUW
vqWKvjJB/Kz3u4zaeNu2ZyyHaOqOuH+TETcW+jsY9IhbEzqN5yQYGi4pVmDkY5vu
lXbJnbqPKpRXgM9BecV9AMbPgbDq/5LnHJJXg+G8YQOgp4lR/hC1TEFdIp5wM8AK
CWsENyt2o1rjgMXiZOMF8A5oBLkCDQRatUuSARAAr77kj7j2QR2SZeOSlFBvV7oS
mFeSNnz9xZssqrsm6bTwSHM6YLDwc7Sdf2esDdyzONETwqrVCg+FxgL8hmo9hS4c
rR6tmrP0mOmptr+xLLsKcaP7ogIXsyZnrEAEsvW8PnfayoiPCdc3cMCR/lTnHFGA
7EuR/XLBmi7Qg9tByVYQ5Yj5wB9V4B2yeCt3XtzPqeLKvaxl7PNelaHGJQY/xo+m
V0bndxf9IY+4oFJ4blD32WqvyxESo7vW6WBh7oqv3Zbm0yQrr8a6mDBpqLkvWwNI
3kpJR974tg5o5LfDu1BeeyHWPSGm4U/G4JB+JIG1ADy+RmoWEt4BqTCZ/knnoGvw
D5sTCxbKdmuOmhGyTssoG+3OOcGYHV7pWYPhazKHMPm201xKCjH1RfzRULzGKjD+
yMLT1I3AXFmLmZJXikAOlvE3/wgMqCXscbycbLjLD/bXIuFWo3rzoezeXjgi/DJx
jKBAyBTYO5nMcth1O9oaFd9d0HbsOUDkIMnsgGBE766Piro6MHo0T0rXl07Tp4pI
rwuSOsc6XzCzdImj0Wc6axS/HeUKRXWdXJwno5awTwXKRJMXGfhCvSvbcbc2Wx+L
IKvmB7EB4K3fmjFFE67yolmiw2qRcUBfygtH3eL5XZU28MiCpue8Y8GKJoBAUyvf
KeM1rO8Jm3iRAc5a/D0AEQEAAYkEPgQYAQIACQUCWrVLkgIbAgIpCRC86dmkLVF4
T8FdIAQZAQIABgUCWrVLkgAKCRDePL1hra+LjtHYD/9MucxdFe6bXO1dQR4tKhhQ
P0LRqy6zlBY9ILCLowNdGZdqorogUiUymgn3VhEhVtxTOoHcN7qOuM01PNsRnOeS
EYjf8Xrb1clzkD6xULwmOclTb9bBxnBc/4PFvHAbZW3QzusaZniNgkuxt6BTfloS
Of4inq71kjmGK+TlzQ6mUMQUg228NUQC+a84EPqYyAeY1sgvgB7hJBhYL0QAxhcW
6m20Rd8iEc6HyzJ3yCOCsKip/nRWAbf0OvfHfRBp0+m0ZwnJM8cPRFjOqqzFpKH9
HpDmTrC4wKP1+TL52LyEqNh4yZitXmZNV7giSRIkk0eDSko+bFy6VbMzKUMkUJK3
```

```
D3eHFAMkujmbfJmSMTJOPGn5SB1HyjCZNx6bhIIbQyEUB9gKCmUFaqXKwKpF6rj0
iQXAJxLR/shZ5Rk96VxzOphUl7T90m/PnUEEPwq8KsBhnMRgxa0RFidDP+n9fgtv
HLmrOqX9zBCVXh0mdWYLrWvmzQFWzG7AoE55fkf8nAEPsalrCdtaNUBHRXA0OQxG
AHMOdJQQvBsmqMvuAdjkDWpFu5y0My5ddU+hiUzUyQLjL5Hhd5LOUDdewlZgIw1j
xrEAUzDKetnemM8GkHxDgg8koev5frmShJuce7vSjKpCNg3EIJSgqMOPFjJuLWtZ
vjHeDNbJy6uNL65ckJy6WhGjEADS2WAW1D6Tfekkc21SsIXk/LqEpLMR/0g5OUif
wcEN1rS9IJXBwIy8MelN9qr5KcKQLmfdfBNEyyceBhyVl0MDyHOKC+7PofMtkGBq
13QieRHv5GJ8LB3fclqHV8pwTTo3Bc8z2g0TjmUYAN/ixETdReDoKavWJYSE9yoM
aaJu279ioVTrwpECse0XkiRyKToTjwOb73CGkBZZpJyqux/rmCV/fp4ALdSW8zbz
FJVORaivhoWwzjpfQKhwcU9lABXi2UvVm14v0AfeI7oiJPSU1zM4fEny4oiIBXlR
zhFNih1UjIu82X16mTm3BwbIga/s1fnQRGzyhqUIMii+mWra23EwjChaxpvjjcUH
5ilLc5Zq781aCYRygYQw+hu5nFkOH1R+Z50Ubxjd/aqUfnGIAX7kPMD3Lof4KldD
Q8ppQriUvxVo+4nPV6rpTy/PyqCLWDjkguHpJsEFsMkwajrAz0QNSAU5CJ0G2Zu4
yxvYlumHCEl7nbFrm0vIiA75Sa8KnywTDsyZsu3XcOcf3g+g1xWTpjJqy2bYXlqz
9uDOWtArWHOis6bq8l9RE6xr1RBVXS6uqgQIZFBGyq66b0dIq4D2JdsUvgEMaHbc
e7tBfeB1CMBdA64e9Rq7bFR7Tvt8gasCZYlNr3lydh+dFHIEkH53HzQe6l88HEic
+0jVnLkCDQRa55wJARAAyLya2Lx6gyoWoJN1a6740q3o8e9d4KggQOfGMTCflmeq
ivuzgN+3DZHN+9ty2KxXMtn0mhHBerZdbNJyjMNT1gAgrhPNB4HtXBXum2wS57WK
DNmade914L7FWTPAWBG2Wn448OEHTqsClICXXWy9IICgclAEyIq0Yq5mAdTEgRJS
Z8t4GpwtDL9gNQyFXaWQmDmkAsCygQMvhAlmu9xOIzQG5CxSnZFk7zcuL60k14Z3
Cmt49k4T/7ZU8goWi8tt+rU78/IL3J/fF9+1civ1OwuUidgfPCSvOUW1JojsdCQA
L+RZJcoXq7lfOFj/eNjeOSstCTDPfTCL+kThE6E5neDtbQHBYkEX1BRiTedsV4+M
ucgiTrdQFWKf89G72xdv8ut9AYYQ2BbEYU+JAYhUH8rYYui2dHKJIgjNvJscuUWb
+QEqJIRleJRhrO+/CHgMs4fZAkWF1VFhKBkcKmEjLn1f7EJJUUW84ZhKXjO/AUPX
1CHsNjziRceuJCJYox1cwsoq6jTE50GiNzcIxTn9xUc0UMKFeggNAFys1K+TDTm3
Bzo8H5ucjCUEmUm9lhkGwqTZgOlRX5eqPX+JBoSaObqhgqCa5IPinKRa6MgoFPHK
6sYKqroYwBGgZm6Js5chpNchvJMs/3WXNOEVg0J3z3vP0DMhxqWm+r+n9zlW8qsA
EQEAAYkEPgQYAQgACQUCWuecCQIbAgIpCRC86dmkLVF4T8FdIAQZAQgABgUCWuec
CQAKCRBQ3szEcQ5hr+ykD/4tOLRHFHXuKUcxgGaubUcVtsFrwBKma1cYjqaPms8u
6Sk0wfGRI32G/GhOrp0Ts/MOkbObq6VLTh8N5Yc/53MEl8zQFw9Y5AmRoW4PZXER
ujs5s7p4oR7xHMihMjCCBn1bvrR+34YPfgzTcgLiOEFHYT8UTxwnGmXOvNkMM7md
xD3CV5q6VAte8WKBo/220II3fcQlc9r/oWX4kXXkb0v9hoGwKbDJ1tzqTPrp/xFt
yohqnvImpnlz+Q9zXmbrWYL9/g8VCmW/NN2gju2G3Lu/TlFUWIT4v/5OPK6TdeNb
VKJO4+S8bTayqSG9CML1S57KSgCo5HUhQWeSNHI+fpe5oX6FALPT9JLDce8OZz1i
cZZ0MELP37mOOQun0AlmHm/hVzf0f311PtbzcqWaE51tJvgUR/nZFo6Ta3O5Ezhs
3VlEJNQ1Ijf/6DH87SxvAoRIARCuZd0qxBcDK0avpFzUtbJd24lRA3WJpkEiMqKv
RDVZkE4b6TW61f0o+LaVfK6E8oLpixegS4fiqC16mFrOdyRk+RJJfIUyz0WTDVmt
g0U1CO1ezokMSqkJ7724pyjr2xf/r9/sC6aOJwB/lKgZkJfC6NqL7TlxVA31dUga
LEOvEJTTE4gl+tYtfsCDvALCtqL0jduSkUo+RXcBItmXhA+tShW0pbS2Rtx/ixua
KohVD/0R4QxiSwQmICNtm9mw9ydIl1yjYXX5a9x4wMJracNY/LBybJPFnZnT4dYR
z4XjqysDwvvYZByaWoIe3QxjX84V6MlI2IdAT/xImu8gbaCI8tmyfpIrLnPKiR9D
VFYfGBXuAX7+HgPPSFtrHQONCALxxzlbNpS+zxt9r0MiLgcLyspWxSdmoYGZ6nQP
RO5Nm/ZVS+u2imPCRzNUZEMa+dlE6kHx0rS0dPiuJ4O7NtPeYDKkoQtNagspsDvh
cK7CSqAiKMq06UBTxqlTSRkm62eOCtcs3p3OeHu5GRZF1uzTET0ZxYkaPgdrQknx
```

```
ozjP5mC7X+45lcCfmcVt94TFNL5HwEUVJpmOgmzILCI8yoDTWzloo+i+fPFsXX4f
kynhE83mSEcr5VHFYrTY3mQXGmNJ3bCLuc/jq7ysGq69xiKmTlUeXFm+aojcRO5i
zyShIRJZ0GZfuzDYFDbMV9amA/YQGygLw//zP5ju5SW26dNxlf3MdFQE5JJ86rn9
MgZ4gcpazHEVUsbZsgkLizRp9imUiH8ymLqAXnfRGlU/LpNSefnvDFTtEIRcpOHc
bhayG0bk51Bd4mioOXnIsKy4j63nJXA27x5EVVHQ1sYRN8Ny4Fdr2tMAmj2O+X+J
qX2yy/UX5nSPU492e2CdZ1UhoU0SRFY3bxKHKB7SDbVeav+K5g==
=Gi5D
-----END PGP PUBLIC KEY BLOCK-----
```

以下為 Amazon ECS PGP 公有金鑰的詳細資訊，以供參考：

```
Key ID: BCE9D9A42D51784F
Type: RSA
Size: 4096/4096
Expires: Never
User ID: Amazon ECS
Key fingerprint: F34C 3DDA E729 26B0 79BE AEC6 BCE9 D9A4 2D51 784F
```

e. 在終端機中透過下列命令使用 Amazon ECS PGP 公有金鑰匯入檔案。

```
gpg --import <public_key_filename.txt>
```

f. 下載 Amazon ECS CLI 簽章。簽章是 ASCII 分離的 PGP 簽章，存放於副檔名為 .asc 的檔案中。簽章檔案的名稱會與對應執行檔的名稱相同，並在名稱後加上 .asc。

macOS

```
curl -Lo ecs-cli.asc https://amazon-ecs-cli.s3.amazonaws.com/ecs-cli-darwin-amd64-latest.asc
```

Linux

```
curl -Lo ecs-cli.asc https://amazon-ecs-cli.s3.amazonaws.com/ecs-cli-linux-amd64-latest.asc
```

Windows

```
Invoke-WebRequest -OutFile ecs-cli.asc https://amazon-ecs-cli.s3.amazonaws.com/ecs-cli-windows-amd64-latest.exe.asc
```

g. 驗證簽章。

macOS and Linux

```
gpg --verify ecs-cli.asc /usr/local/bin/ecs-cli
```

Windows

```
gpg --verify ecs-cli.asc 'C:\Program Files\Amazon\ECSCLI\ecs-cli.exe'
```

預期的輸出結果：

```
gpg: Signature made Tue Apr  3 13:29:30 2018 PDT
gpg:                using RSA key DE3CBD61ADAF8B8E
gpg: Good signature from "Amazon ECS <ecs-security@amazon.com>" [unknown]
gpg: WARNING: This key is not certified with a trusted signature!
gpg:          There is no indication that the signature belongs to the owner.
Primary key fingerprint: F34C 3DDA E729 26B0 79BE  AEC6 BCE9 D9A4 2D51 784F
     Subkey fingerprint: EB3D F841 E2C9 212A 2BD4  2232 DE3C BD61 ADAF 8B8E
```

> **Important**
>
> 輸出中有警告很正常，不會有問題。這是因為個人 PGP 金鑰 (如果有) 和 Amazon ECS PGP 金鑰之間沒有信任鏈。如需詳細資訊，請參閱「信任網路」。

3. 將執行許可套用至二進位檔。

macOS and Linux

```
sudo chmod +x /usr/local/bin/ecs-cli
```

Windows

編輯環境變數，將 `C:\Program Files\Amazon\ECSCLI` 新增到 PATH 變數欄位，使用分號與現有項目隔開。例如：

```
setx path "%path%;C:\Program Files\Amazon\ECSCLI"
```

重新啟動， PowerShell 以便更改生效。

> **Note**
>
> 設定PATH變數後，即可從視窗 PowerShell 或命令提示字元使用 Amazon ECS CLI。

4. 確認 CLI 是否正常運作。

```
ecs-cli --version
```

繼續執行「設定 Amazon ECS CLI」。

> **Important**
>
> 您必須先使用您的 AWS 登入資料、 AWS 區域和 Amazon ECS 叢集名稱來設定 Amazon ECS CLI，才能使用它。如需詳細資訊，請參閱 設定 Amazon ECS CLI。

設定 Amazon ECS CLI

Amazon ECS 已發行 C AWS opilot，這是一種命令列界面 (CLI) 工具，可簡化從本機開發環境在 Amazon ECS 上建置、發行和操作生產就緒容器化應用程式。如需詳細資訊，請參閱 使用 AWS 副駕駛命令列界面建立 Amazon ECS 資源。

Amazon ECS CLI 需要一些基本的組態資訊才能使用，例如您的 AWS 登入資料、建立叢集的 AWS 區域，以及要使用的 Amazon ECS 叢集名稱。組態資訊存放位置為：macOS 和 Linux 系統在 `~/.ecs` 目錄，Windows 系統在 `C:\Users\<username>\AppData\local\ecs`。

若要設定 Amazon ECS CLI

1. 使用下列命令設定 CLI 設定檔，並以您想要*profile_name*的設定檔名稱*$AWS_ACCESS_KEY_ID*和*$AWS_SECRET_ACCESS_KEY*環境變數替換為 AWS 憑證。

```
ecs-cli configure profile --profile-name profile_name --access-
key $AWS_ACCESS_KEY_ID --secret-key $AWS_SECRET_ACCESS_KEY
```

2. 使用下列命令完成組態，並將 *launch_type* 替換成您預設要使用的任務啟動類型，將 *region_name* 替換成您需要的 AWS 區域，將 *cluster_name* 替換成現有的 Amazon ECS 叢集或要使用的新叢集名稱，並將 *configuration_name* 替換成您想使用的組態名稱。

```
ecs-cli configure --cluster cluster_name --default-launch-type launch_type --
region region_name --config-name configuration_name
```

使用設定檔

Amazon ECS CLI 支援使用ecs-cli configure profile命令將多組 AWS 登入資料設定為具名設定檔。使用 ecs-cli configure profile default 命令可以設定預設描述檔。接著，當您執行需要使用 `--ecs-profile` 標記之憑證的 Amazon ECS CLI 命令時，即會參考這些描述檔，否則會使用預設的描述檔。

使用叢集組態

叢集組態是一組描述 Amazon ECS 叢集的欄位，包括叢集名稱和區域。使用 ecs-cli configure default 命令可以設定預設的叢集組態。Amazon ECS CLI 支援使用 `--config-name` 選項設定多個具名的叢集組態。

了解優先順序

使用 Amazon ECS CLI 命令有多種方法可傳送憑證和區域。以下是這些方法的優先順序。

憑證的優先順序：

1. Amazon ECS CLI 描述檔標記：
   a. Amazon ECS 描述檔 (`--ecs-profile`)
   b. AWS 配置文件 (`--aws-profile`)
2. 環境變數：
   a. `ECS_PROFILE`
   b. `AWS_PROFILE`
   c. `AWS_ACCESS_KEY_ID`、`AWS_SECRET_ACCESS_KEY` 與 `AWS_SESSION_TOKEN`
3. ECS Config - 嘗試從預設的 ECS 描述檔擷取憑證。
4. 預設設 AWS 定檔 — 嘗試從設定 AWS 檔名稱使用認證 `assume_role` (`aws_access_key_id`role_arn, `aws_secret_access_key`source_profile) 或 (,)。
   a. `AWS_DEFAULT_PROFILE` 環境變數 (預設為 `default`).

5. EC2 執行個體角色

區域的優先順序為：

1. Amazon ECS CLI 標記：
   a. 區域旗標 (`--region`)
   b. 叢集組態旗標 (`--cluster-config`)
2. ECS Config - 嘗試從預設的 ECS 描述檔擷取區域。
3. 環境變數 - 嘗試從以下環境變數擷取區域：
   a. `AWS_REGION`
   b. `AWS_DEFAULT_REGION`
4. AWS profile-嘗試使用配 AWS 置文件名稱中的區域：
   a. `AWS_PROFILE` 環境變數
   b. `AWS_DEFAULT_PROFILE` 環境變數 (預設為 `default`)

AWS Fargate對於 Amazon ECS

AWS Fargate 是一項技術，可以與 Amazon ECS 搭配使用以執行容器，而不需管理 Amazon EC2 執行個體的伺服器或叢集。有了 AWS Fargate，就不再需要佈建、設定或擴展虛擬機器的叢集來執行容器。這樣一來即無須選擇伺服器類型、決定何時擴展叢集，或最佳化叢集壓縮。

當您使用 Fargate 啟動類型執行任務和服務時，將會在容器中封裝應用程式、指定 CPU 和記憶體需求、定義聯網和 IAM 政策，並啟動應用程式。每個 Fargate 任務都有自己的隔離界限，並且不會與另一個任務共用基礎核心、CPU 資源、記憶體資源或彈性網路界面。您可以將 `requiresCompatibilities` 任務定義參數設定為 `FARGATE`，以設定 Fargate 的任務定義。如需詳細資訊，請參閱 啟動類型。

Fargate 提供了適用於 Amazon Linux 2 和 Microsoft Windows 2019 Server Full 和 Core 版本的平台版本。除非另有說明，否則本頁面上的資訊適用於所有 Fargate 平台。

本主題說明不同的 Fargate 任務和服務元件，並注意搭配使用 Fargate 與 Amazon ECS 的特殊考量。

如需在 Fargate 上支援 Linux 容器的區域的資訊，請參閱the section called “AWS Fargate 上的 Linux 容器”。

如需在 Fargate 上支援 Windows 容器的區域的資訊，請參閱the section called “AWS Fargate 上的視窗容器”。

逐步解說

如需如何開始使用主控台的詳細資訊，請參閱：

- 了解如何為 Fargate 啟動類型建立 Amazon ECS Linux 任務
- 了解如何為 Fargate 啟動類型建立 Amazon ECS 視窗任務

如需如何開始使用的相關資訊 AWS CLI，請參閱：

- 使用以下方式為 Fargate 啟動類型建立 Amazon ECS Linux 任務 AWS CLI
- 為 Fargate 啟動類型建立 Amazon ECS 視窗任務 AWS CLI

容量提供者

以下是可用的容量提供者：

- Fargate
- Fargate 現貨-與價格相比，以折扣率執行耐受中斷的 Amazon ECS 任務。AWS FargateFargate Spot 在備用運算容量上執行任務。當 AWS 需要恢復容量時，您的任務將被中斷並發出兩分鐘警告。如需詳細資訊，請參閱 適用於 Fargate 啟動類型的 Amazon ECS 叢集 。

  對於使用 X86 架構的 Linux 任務，您只能使用 Fargate 競價點。

任務定義

使用 Fargate 啟動類型的任務不支援所有可用的 Amazon ECS 任務定義參數。有些參數完全不予以支援，而其他參數對 Fargate 任務會有不同的行為。如需詳細資訊，請參閱 任務 CPU 和記憶體。

平台版本

AWS Fargate 平台版本用於指 Fargate 任務基礎設施的特定運行時環境。其結合了核心與容器執行時間版本。在執行任務或建立服務以維護許多相同的任務時，請選取平台版本。

為因應執行時間環境演進 (例如是否有核心或作業系統的更新、新功能、錯誤修正或安全性更新)，我們會不時發行新的平台版本修訂版。建立新的平台版本修訂版即可更新 Fargate 平台版本。每項任務在其生命週期期間都只會在一個平台版本修訂版上執行。如果想要使用最新的平台版本修訂版，必須啟動新任務。在 Fargate 上執行的新任務一律會在平台版本的最新修訂版上執行，以確保任務一律在已修補的安全基礎設施上啟動。

如果發現影響現有平台版本的安全性問題，請 AWS 建立平台版本的新修訂版本，並淘汰在易受攻擊修訂版本上執行的工作。在某些案例中，您可能會收到通知，告知您在 Fargate 上的任務已排程淘汰。如需詳細資訊，請參閱 AWS Amazon ECS 上的 Fargate 任務維護常見問題解答。

如需更多資訊，請參閱適用於 Amazon ECS 的 Fargate Linux 平台版本及Amazon ECS 的 Fargate 視窗平台版本。

服務負載平衡

AWS Fargate 上的 Amazon ECS 服務可以選擇性地設定為使用 Elastic Load Balancing，將您服務中任務的流量平均分配。

AWS Fargate 上的 Amazon ECS 服務支援 Application Load Balancer 和 Network Load Balancer 負載平衡器類型。Application Load Balancer 用於路由 HTTP/HTTPS (或 Layer 7) 流量。Network

Load Balancer 用於路由 TCP 或 UDP (或 Layer 4) 流量。如需詳細資訊，請參閱 使用負載平衡來分配 Amazon ECS 服務流量。

當您為這些服務建立目標群組時，必須選擇 `ip` 作為目標類型，而不是選擇 `instance`。因為採用 `awsvpc` 網路模式的任務是與彈性網路界面相關聯，而非與 Amazon EC2 執行個體相關聯。如需詳細資訊，請參閱 使用負載平衡來分配 Amazon ECS 服務流量。

當使用平台版本 1.4 或更高版本時，只支援使用 Network Load Balancer 將 UDP 流量路由到 AWS Fargate 上的 Amazon ECS 任務。

用量指標

您可以使用使 CloudWatch 用量度來提供您帳戶資源使用情況的可見性。使用這些指標，在 CloudWatch 圖形和儀表板上視覺化您目前的服務使用情況。

AWS Fargate 使用量度對應至 AWS 服務配額。您可以設定警示，在您的用量接近服務配額時發出警示。如需 AWS Fargate 的服務配額詳細資訊，請參閱 AWS Fargate 服務配額。

如需 AWS Fargate 使用量指標的詳細資訊，請參閱 Amazon 彈性容器服務AWS Fargate使用者指南中的使用量指標 AWS Fargate。

Amazon ECS 安全疑慮何時使用 Fargate 啟動類型

我們建議客戶希望對其任務進行強大隔離的客戶使用 Fargate。Fargate 會在硬體虛擬化環境中執行每項工作。這可確保這些容器化工作負載不會與其他工作共用網路介面、Fargate 暫時性儲存、CPU 或記憶體。如需詳細資訊，請參閱 AWS Fargate.

Amazon ECS 中的 Fargate 安全最佳實踐

建議您在使用 AWS Fargate時考量下列最佳實務。如需其他指引，請參閱的安全性概觀 AWS Fargate。

用 AWS KMS 於加密 Fargate 的臨時存儲

您應該使用加密的臨時存儲。 AWS KMS對於使用平台版本`1.4.0`或更高版本在 Fargate 上託管的任務，每個任務都會收到 20 GiB 的臨時存儲。您可以增加暫時性儲存的總量，最多可達 200 GiB，方法是在任務定義中指定 `ephemeralStorage` 參數。對於 2020 年 5 月 28 日或之後啟動的此類任務，臨時存儲使用由 Fargate 管理的加密密鑰使用 AES-256 加密算法進行加密。

如需詳細資訊，請參閱在任務中使用資料磁碟區。

示例：使用臨時存儲加密在 Fargate 平台 1.4.0 版上啟動任務

下面的命令將啟動 Fargate 平台 1.4 版本的任務。由於此工作是作為叢集的一部分啟動，因此它會使用自動加密的臨時儲存體的 20 GiB。

```
aws ecs run-task --cluster clustername \
  --task-definition taskdefinition:version \
  --count 1
  --launch-type "FARGATE" \
  --platform-version 1.4.0 \
  --network-configuration
 "awsvpcConfiguration={subnets=[subnetid],securityGroups=[securitygroupid]}" \
  --region region
```

SYS_PSTRACE 功能，用於使用 Fargate 進行內核系統調用跟踪

由 Docker 提供從容器中新增或刪除的 Linux 功能預設組態。如需可用功能的詳細資訊，請參閱 Docker run 文件中的執行期權限和 Linux 功能。

在 Fargate 上啟動的任務僅支援新增 SYS_PTRACE 核心功能。

下面的教程視頻展示瞭如何通過 Sysdig 法爾科項目使用此功能。

ContainersFromTheCouch -使用 SYS_PSTRACE 功能對 Fargate 任務進行故障排除

在上一個視頻中討論的代碼可以在 GitHub 這裡找到。

使用 Amazon GuardDuty 與 Fargate 運行時監控

Amazon GuardDuty 是一種威脅偵測服務，可協助保護您的帳戶、容器、工作負載和 AWS 環境中的資料。使用機器學習 (ML) 模型，以及異常和威脅偵測功能， GuardDuty持續監控不同的記錄檔來源和執行階段活動，以識別環境中潛在的安全風險和惡意活動並排定優先順序。

運行時監控通過持續監控 AWS 日誌和網絡活動以識別惡意或未經授權的行為來 GuardDuty 保護在 Fargate 上運行的工作負載。執行階段監控使用輕量型、完全受控的 GuardDuty 安全代理程式來分析主機上的行為，例如檔案存取、程序執行和網路連線。這涵蓋的問題包括權限提升、使用暴露的登入資料，或與惡意 IP 地址、網域通訊，以及 Amazon EC2 執行個體和容器工作負載上存在惡意軟體的情況。如需詳細資訊，請參閱GuardDuty 使用指南中的GuardDuty 執行階段監視。

Amazon ECS 的 Fargate 安全考量

每項任務都有專用的基礎架構容量，因為 Fargate 會在隔離的虛擬環境中執行每個工作負載。在 Fargate 上執行的工作負載不會與其他任務共用網路介面、暫時性儲存、CPU 或記憶體。您可以在任務中執行多個容器，包括應用程式容器和附屬容器，或者僅是附屬。附屬是在 Amazon ECS 任務中與應用程式容器一起執行的容器。當應用程式容器執行核心應用程式程式碼時，在附屬中執行的程序可以增強應用程式。附屬可協助您將應用程式功能隔離到專用容器中，讓您更輕鬆地更新應用程式的各個部分。

屬於相同任務的容器會共用 Fargate 啟動類型的資源，因為這些容器將始終會在相同的主機上執行並共用計算資源。這些容器也共用 Fargate 提供的暫時性儲存。任務中的 Linux 容器會共用網路命名空間，包括 IP 地址和網路連接埠。在任務內部，屬於任務的容器可以透過本機主機進行相互通訊。

Fargate 中的執行期環境可防止您使用 EC2 執行個體支援的某些控制器功能。架構在 Fargate 上執行的工作負載時，請考量下列事項：

- 沒有特殊權限的容器或存取 - Fargate 目前無法提供特殊權限的容器或存取等功能。這將影響使用案例，例如在 Docker 中執行 Docker。
- 對 Linux 功能的有限存取 - 在 Fargate 上執行容器的環境已被鎖定。其他 Linux 功能 (例如：CAP_SYS_ADMIN 和 CAP_NET_ADMIN) 受到限制，以防止權限提升。Fargate 支援將 CAP_SYS_PSTRACE Linux 功能新增至任務中，以允許在任務中部署的可觀測性和安全工具來監控容器化應用程式。
- 無法存取基礎主機-客戶和 AWS 操作員都無法連線到執行客戶工作負載的主機。您可以使用 ECS Exec 執行命令，或為在 Fargate 上執行的容器取得 Shell。您可以使用 ECS Exec 來協助收集診斷資訊以進行偵錯。Fargate 也可防止容器存取基礎主機的資源，例如檔案系統、裝置、網路和容器執行期。
- 網路 - 您可以使用安全群組和網路 ACL 來控制傳入和傳出流量。Fargate 任務會從 VPC 中設定的子網路接收 IP 地址。

適用於 Amazon ECS 的 Fargate Linux 平台版本

AWS Fargate 平台版本用於指 Fargate 任務基礎設施的特定運行時環境。其結合了核心與容器執行時間版本。在執行任務或建立服務以維護許多相同的任務時，請選取平台版本。

為因應執行時間環境演進 (例如是否有核心或作業系統的更新、新功能、錯誤修正或安全性更新)，我們會不時發行新的平台版本修訂版。建立新的平台版本修訂版即可更新 Fargate 平台版本。每項任務在其生命週期期間都只會在一個平台版本修訂版上執行。如果想要使用最新的平台版本修訂版，必須啟動新

任務。在 Fargate 上執行的新任務一律會在平台版本的最新修訂版上執行，以確保任務一律在已修補的安全基礎設施上啟動。

如果發現影響現有平台版本的安全性問題，請 AWS 建立平台版本的新修訂版本，並淘汰在易受攻擊修訂版本上執行的工作。在某些案例中，您可能會收到通知，告知您在 Fargate 上的任務已排程淘汰。如需詳細資訊，請參閱 AWS Amazon ECS 上的 Fargate 任務維護常見問題解答。

考量事項

指定平台版本時，應考慮以下項目：

- 指定平台版本時，您可以使用特定版本編號 (如 `1.4.0`) 或 LATEST。

  如果選取 LATEST (最新) 平台版本，則會使用 `1.4.0` 平台版本。
- 如果要更新服務的平台版本，請建立部署。例如，假設您有在 Linux 平台版本 `1.3.0` 上執行任務的服務。若要變更服務以在 Linux 平台版本 `1.4.0` 上執行任務，您可以更新服務並指定新的平台版本。您的任務會透過最新的平台版本和最新的平台版本修訂版重新部署。如需有關部署的詳細資訊，請參閱 Amazon ECS 服務。
- 當您沒有更新平台版本便擴展服務規模時，這些任務所獲得的平台版本，即為服務目前部署上指定的版本。例如，假設您有在 Linux 平台版本 `1.3.0` 上執行任務的服務。如果增加所需的服務數量，服務排程器會使用平台版本 `1.3.0` 的最新平台版本修訂版來啟動新任務。
- 新任務一律會在平台版本的最新修訂版上執行，確保任務一律在已修補的安全基礎設施上啟動。
- Fargate 上 Linux 容器和 Windows 容器的平台版本編號各自獨立。例如，Fargate 上 Windows 容器的平台版本 `1.0.0` 中使用的行為、功能和軟體與 Fargate 上 Linux 容器的平台版本 `1.0.0` 無法相比。

以下是可用的 Linux 平台版本。如需平台版本取代的相關資訊，請參閱AWS Fargate Linux 平台版本棄用。

1.4.0

以下是平台版本 `1.4.0` 的變更日誌。

- 從 2020 年 11 月 5 日開始，在 Fargate 上啟動的使用平台版本 `1.4.0` 的任何新 Amazon ECS 任務都將能夠使用以下功能：
  - 當您使用 Secrets Manager 來存放敏感資料時，您可以將特定的 JSON 金鑰或特定版本的秘密作為環境變數插入在日誌組態中。如需詳細資訊，請參閱 將敏感資料傳遞至 Amazon ECS 容器。

  - 使用 `environmentFiles` 容器定義參數，指定大量環境變數。如需詳細資訊，請參閱 將個別環境變數傳遞至 Amazon ECS 容器。
  - 在 VPC 中執行的任務以及為 IPv6 啟用的子網路將被指派一個私有 IPv4 地址和一個 IPv6 地址。如需詳細資訊，請參閱 AWS Fargate的 Amazon Elastic Container Service 使用者指南中的 Fargate 任務聯網。
  - 任務中繼資料端點版本 4 提供有關任務和容器的其他中繼資料，包括任務啟動類型、容器的 Amazon Resource Name (ARN)，以及使用的日誌驅動程式和日誌驅動程式選項。查詢 `/stats` 端點時，您還會收到容器的網路速度統計資訊。如需詳細資訊，請參閱工作中繼資料端點版本 4。
- 從 2020 年 7 月 30 日開始，在 Fargate 上啟動的使用平台版本 `1.4.0` 的任何新 Amazon ECS 任務將能夠使用 Network Load Balancer 將 UDP 流量路由到 Fargate 任務上的 Amazon ECS。如需詳細資訊，請參閱 使用負載平衡來分配 Amazon ECS 服務流量。
- 從 2020 年 5 月 28 日開始，任何使用平台版本在 Fargate 上啟動的新 Amazon ECS 任務，都`1.4.0`將使用擁有的加密金鑰以 AES-256 加密演算法加密其臨時儲存。 AWS 如需詳細資訊，請參閱 適用於 Amazon ECS 的 Fargate 任務暫時儲存 及 Amazon ECS 任務的儲存選項。
- 新增了對使用 Amazon EFS 檔案系統磁碟區的支援，以便實現持久性任務儲存。如需詳細資訊，請參閱 搭配 Amazon ECS 使用 Amazon EFS 磁碟區。
- 暫時任務儲存已增加到每個任務至少 20 GB。如需詳細資訊，請參閱 適用於 Amazon ECS 的 Fargate 任務暫時儲存。
- 進出任務的網路流量行為已更新。從平台版本 1.4.0 開始，所有 Fargate 任務會收到單一彈性網路界面 (也稱為任務 ENI)，所有網路流量都會流經 VPC 中的 ENI，而且您可以透過 VPC 流程日誌查看。如需有關 Amazon EC2 啟動類型聯網的詳細資訊，請參閱 Fargate 任務聯網。如需有關 Fargate 啟動類型之網路的詳細資訊，請參閱適用於 Fargate 啟動類型的 Amazon ECS 任務聯網選項。
- 任務 ENI 新增 Jumbo Frame 的支援。網路界面皆以最大傳輸單位 (MTU) 來設定，這是適合單一框架的最大酬載大小。MTU 越大，單一框架能容納的應用程式酬載越多，可降低每個框架的額外負荷並提高效率。當任務和目的地之間的網路路徑支援 Jumbo Frame (例如，VPC 中剩餘的所有流量) 時，支援 Jumbo Frame 會降低額外負荷。
- CloudWatch 容器洞見將包括 Fargate 工作的網路效能指標。如需詳細資訊，請參閱 使用容器洞察來監控 Amazon ECS 容器。
- 已新增任務中繼資料端點第 4 版的支援，該端點為您的 Fargate 任務提供額外的資訊，包括任務的網路統計資料，以及執行中任務所在的可用區域。如需詳細資訊，請參閱 > Amazon ECS 任務中繼資料端點版本 4 和適用於 Fargate 上任務的 Amazon ECS 任務中繼資料端點第 4 版。
- 在容器定義中，新增對 SYS_PTRACE Linux 參數的支援。如需詳細資訊，請參閱 Linux 參數。

- Fargate 容器代理會取代所有 Fargate 任務的 Amazon ECS 容器代理的使用。此變更通常不會對您的任務執行方式產生影響。
- 容器執行時間目前使用 Containerd，而非 Docker。此變更很可能不會對您的任務執行方式產生影響。您會注意到，容器執行時間產生的一些錯誤訊息會從提及 Docker 變成為較為一般的錯誤。如需詳細資訊，請參閱 AWS Fargate的 Amazon Elastic Container Service 使用者指南中的已停止的任務錯誤代碼。
- 以 Amazon Linux 2 為基礎。

1.3.0

以下是平台版本 `1.3.0` 的變更日誌。

- 從 2019 年 9 月 30 日開始，啟動的任何新 Fargate 任務都支援 `awsfirelens` 日誌驅動程式。將 Amazon ECS 設定 FireLens 為使用任務定義參數將日誌路由到 AWS 服務或 AWS 合作夥伴網路 (APN) 目的地以進行日誌儲存和分析。如需詳細資訊，請參閱 將 Amazon ECS 日誌傳送到 AWS 服務或 AWS Partner。
- 增加了對 Fargate 任務的任務回收機制，此過程將重新整理屬於 Amazon ECS 服務的各項任務。如需詳細資訊，請參閱 AWS Fargate的 Amazon Elastic Container Service 使用者指南中的任務維護。
- 從 2019 年 3 月 27 日開始，啟動的任何新的 Fargate 任務可以使用額外的任務定義參數，來定義代理組態、容器啟動和關機的相依性，以及每一容器的啟動和停止逾時值。如需詳細資訊，請參閱代理組態、容器相依性及容器逾時。
- 從 2019 年 4 月 2 日開始，任何啟動的新 Fargate 工作都支援將敏感資料注入容器，方法是將敏感資料儲存在機 AWS Secrets Manager 密或參數存放區 AWS Systems Manager 參數中，然後在容器定義中參照這些參數。如需詳細資訊，請參閱 將敏感資料傳遞至 Amazon ECS 容器。
- 從 2019 年 5 月 1 日開始，啟動的任何新 Fargate 任務支援使用 `secretOptions` 容器定義參數，以參考容器日誌組態中的敏感資料。如需詳細資訊，請參閱 將敏感資料傳遞至 Amazon ECS 容器。
- 從 2019 年 5 月 1 日開始，啟動的任何新 Fargate 任務除了支援 `awslogs` 日誌驅動程式外，還支援 `splunk` 日誌驅動程式。如需詳細資訊，請參閱 儲存與記錄。
- 從 2019 年 7 月 9 日開始，任何啟動的新 Fargate 任務都支持 CloudWatch 容器洞察。如需詳細資訊，請參閱 使用容器洞察來監控 Amazon ECS 容器。
- 從 2019 年 12 月 3 日開始，支援 Fargate Spot 容量提供者。如需詳細資訊，請參閱 適用於 Fargate 啟動類型的 Amazon ECS 叢集 。
- 以 Amazon Linux 2 為基礎。

移轉至平台版本 1.4.0

將 Fargate 任務上的 Amazon ECS 從平台版本 `1.0.0`、`1.1.0`、`1.2.0` 或 `1.3.0` 遷移至平台版本 `1.4.0` 時，應考慮以下項目。這被認為是最佳實務，在遷移任務之前確認您的任務在平台版本 `1.4.0` 上正常運作。

- 進出任務的網路流量行為已更新。從平台版本 1.4.0 開始，在 Fargate 任務上的所有 Amazon ECS 都會收到單一彈性網路界面 (也稱為任務 ENI)，所有網路流量都會流經 VPC 中的 ENI，而且您可以透過 VPC 流程日誌查看。如需更多資訊，請參閱適用於 Fargate 啟動類型的 Amazon ECS 任務聯網選項。
- 如果您正在使用界面 VPC 端點，應考慮以下項目。
  - 使用 Amazon ECR 託管的容器映像時，需要 com.amazonaws.region.ecr.dkr 和 com.amazonaws.region.ecr.api Amazon ECR VPC 端點，以及 Amazon S3 閘道端點。如需詳細資訊，請參閱 Amazon Elastic Container Registry 使用者指南中的 Amazon ECR 介面 VPC 端點 (AWS PrivateLink)。
  - 使用參考 Secrets Manager 秘密的任務定義來擷取容器的敏感資料時，您必須建立 Secrets Manager 的介面 VPC 端點。如需詳細資訊，請參閱 AWS Secrets Manager 使用者指南中的搭配使用 Secrets Manager 與 VPC 端點。
  - 使用參考 Systems Manager 參數存放區參數的任務定義來擷取容器的敏感資料時，您必須建立 Systems Manager 的介面 VPC 端點。如需詳細資訊，請參閱 AWS Systems Manager 使用者指南中的搭配使用 Systems Manager 與 VPC 端點。
  - 確保與任務相關聯的彈性網路界面 (ENI) 中的安全群組具有已建立的安全群組規則，以允許任務與您使用的 VPC 端點之間的流量。

AWS Fargate Linux 平台版本棄用

此頁面列出了 AWS Fargate 已棄用或已計劃棄用的 Linux 平台版本。這些平台版本在其公佈的取代日期前都會保持可用。

對已排定取代的每個平台版本提供強制更新日期。在強制更新日期當天，任何使用 `LATEST` 平台版本 (已排定取代的平台版本) 的服務都將使用強制新部署選項進行更新。使用強制新部署選項更新服務時，在排定取代的平台版本上執行的所有任務都會停止，並且會使用 `LATEST` 標籤當時指示的平台版本啟動新任務。已設定明確平台版本的獨立任務或服務不受強制更新日期的影響。

我們建議您更新您的服務和獨立任務，以使用最新的平台版本。如需遷移至最新平台版本的詳細資訊，請參閱移轉至平台版本 1.4.0。

平台版本到達取代日期後，平台版本將不再適用於新任務或服務。任何明確使用過時平台版本的獨立任務或服務，都會繼續使用該平台版本，直到任務停止為止。在取代日期之後，已被取代的平台版本將不再收到任何安全性更新或錯誤修正。

平台版本	強制更新日期	取代日期
1.0.0	2020 年 10 月 26 日	2020 年 12 月 14 日
1.1.0	2020 年 10 月 26 日	2020 年 12 月 14 日
1.2.0	2020 年 10 月 26 日	2020 年 12 月 14 日

如需目前平台版本的相關詳細資訊，請參閱適用於 Amazon ECS 的 Fargate Linux 平台版本。

不推薦使用的 AWS Fargate Linux 版本的更新日誌

1.2.0

以下是平台版本 `1.2.0` 的變更日誌。

> **Note**
>
> 平台版本 `1.2.0` 已無法使用。如需平台版本取代的相關資訊，請參閱AWS Fargate Linux 平台版本棄用。

- 增加了對私人註冊表身份驗證的支持 AWS Secrets Manager。如需詳細資訊，請參閱 在 Amazon ECS 中使用非AWS 容器映像。

1.1.0

以下是平台版本 `1.1.0` 的變更日誌。

> **Note**
>
> 平台版本 `1.1.0` 已無法使用。如需平台版本取代的相關資訊，請參閱AWS Fargate Linux 平台版本棄用。

- 新增對 Amazon ECS 任務中繼資料端點的支援。如需詳細資訊，請參閱 Amazon ECS 任務中繼資料可用於 Fargate 上的任務。
- 在容器定義中，新增對 Docker 運作狀態檢查的支援。如需詳細資訊，請參閱 運作狀態檢查。
- 新增對 Amazon ECS 服務探索的支援。如需詳細資訊，請參閱 使用服務探索將 Amazon ECS 服務與 DNS 名稱連接。

1.0.0

以下是平台版本 `1.0.0` 的變更日誌。

> **Note**
>
> 平台版本 `1.0.0` 已無法使用。如需平台版本取代的相關資訊，請參閱AWS Fargate Linux 平台版本棄用。

- 以 Amazon Linux 2017.09 為基礎。
- 初始版本。

Amazon ECS Fargate 容器映像提取行為上的 Linux 容器

每個 Fargate 任務都在其自己的單一使用單一租用戶執行個體上執行。當您在 Fargate 上執行 Linux 容器時，不會在執行個體上快取容器映像檔或容器映像層。因此，對於任務中定義的每個容器映像，需要從每個 Fargate 任務的容器映像註冊表中提取整個容器映像。提取圖像所花費的時間與開始 Fargate 任務所花費的時間直接相關。

請考慮以下幾點，以最佳化影像提取時間。

容器映像接近

若要減少下載容器映像所需的時間，請盡可能將資料定位在靠近運算的位置。透過網際網路或跨越擷取容器映像檔 AWS 區域 可能會影響下載時間。建議您將容器映像儲存在執行工作的相同區域中。如果您將容器映像存放在 Amazon ECR 中，請使用 VPC 界面端點進一步縮短映像提取時間。如需詳細資訊，請參閱 Amazon ECR 使用者指南中的 Amazon ECR 介面 VPC 人雲端節點 (AWS PrivateLink)。

容器圖像尺寸縮小

容器映像檔的大小會直接影響下載時間。減少容器映像的大小或容器圖像層的數量，可以減少下載圖像所需的時間。輕量型基本映像 (例如最小的 Amazon Linux 2023 容器映像) 可能比以傳統作業系統基礎映像為基礎的映像檔要小得多。如需有關最小映像檔的詳細資訊，請參閱 Amazon 2023 使用者指南中的 AL2023 最小容器映像檔。

替代壓縮演算法

容器映像層通常會在推送至容器映像登錄時進行壓縮。壓縮容器映像層可減少必須透過網路傳輸並儲存在容器映像登錄中的資料量。在容器執行階段將容器影像圖層下載至實體之後，該圖層就會解壓縮。使用的壓縮演算法和執行階段可用的 vCPUs 數量會影響解壓縮容器映像所需的時間。在 Fargate 上，您可以增加任務的大小或利用更高效的 zstd 壓縮算法來減少解壓縮所花費的時間。如需詳細資訊，請參閱中的 ztsd。 GitHub如需如何實作 Fargate 映像的相關資訊，請參閱使用 zstd 壓縮容器映像縮短 AWS Fargate 啟動時間。

延遲加載容器圖像

對於大型容器映像（ > 250mb ），延遲加載容器映像而不是下載所有容器映像可能是最佳選擇。在 Fargate 上，您可以使用可搜索的 OCI（SOCI）從容器映像註冊表延遲加載容器映像。有關更多信息，請參閱社交快照 GitHub 和使用可搜索 OCI（SOCI）延遲加載容器映像。

Amazon ECS 的 Fargate 視窗平台版本

AWS Fargate 平台版本用於指 Fargate 任務基礎設施的特定運行時環境。其結合了核心與容器執行時間版本。在執行任務或建立服務以維護許多相同的任務時，請選取平台版本。

為因應執行時間環境演進 (例如是否有核心或作業系統的更新、新功能、錯誤修正或安全性更新)，我們會不時發行新的平台版本修訂版。建立新的平台版本修訂版即可更新 Fargate 平台版本。每項任務在其生命週期期間都只會在一個平台版本修訂版上執行。如果想要使用最新的平台版本修訂版，必須啟動新任務。在 Fargate 上執行的新任務一律會在平台版本的最新修訂版上執行，以確保任務一律在已修補的安全基礎設施上啟動。

如果發現影響現有平台版本的安全性問題，請 AWS 建立平台版本的新修訂版本，並淘汰在易受攻擊修訂版本上執行的工作。在某些案例中，您可能會收到通知，告知您在 Fargate 上的任務已排程淘汰。如需詳細資訊，請參閱 AWS Amazon ECS 上的 Fargate 任務維護常見問題解答。

平台版本考量事項

指定平台版本時，應考慮以下項目：

- 指定平台版本時，您可以使用特定版本編號 (如 `1.0.0`) 或 `LATEST`。

  如果選取 LATEST (最新) 平台版本，則會使用 `1.0.0` 平台版本。
- 新任務一律會在平台版本的最新修訂版上執行，確保任務一律在已修補的安全基礎設施上啟動。
- Microsoft Windows Server 容器映像必須在特定版本的 Windows Server 中建立。執行任務或建立符合 Windows Server 容器映像的服務時，必須在 `platformFamily` 中選取相同版本的 Windows Server。此外，您可以在任務定義中提供相符的 `operatingSystemFamily`，以防止任務在錯誤的 Windows 版本上執行。如需詳細資訊，請參閱 Microsoft Learn 網站上的比對容器主機版本與容器映像版本。
- Fargate 上 Linux 容器和 Windows 容器的平台版本編號各自獨立。例如，Fargate 上 Windows 容器的平台版本 `1.0.0` 中使用的行為、功能和軟體與 Fargate 上 Linux 容器的平台版本 `1.0.0` 無法相比。

以下是可用於 Windows 容器的平台版本。

1.0.0

以下是平台版本 `1.0.0` 的變更日誌。

- 下列 Microsoft Windows Server 作業系統上支援的初始版本：
  - Windows Server 2019 Full
  - Windows Server 2019 Core
  - Windows Server 2022 Full
  - Windows Server 2022 Core

Amazon ECS Fargate 上的視窗容器注意事項

以下是在 AWS Fargate 上執行 Windows 容器時需要瞭解的差異和注意事項。

如果您需要在 Linux 和 Windows 容器上執行工作，則需要為每個作業系統建立個別的工作定義。

AWS 處理操作系統許可證管理，所以你不需要任何額外的 Microsoft Windows 服務器許可證。

AWS Fargate 上的 Windows 容器支持以下操作系統：

- Windows Server 2019 Full
- Windows Server 2019 Core

- Windows Server 2022 Full
- Windows Server 2022 Core

AWS Fargate 上的視窗容器支援 awslog 驅動程式。如需詳細資訊，請參閱 the section called “將記錄檔傳送至 CloudWatch ”。

Fargate 上的 Windows 容器不支援下列功能：

- 群組受管服務帳戶 (gMSA)
- Amazon FSx
- ENI 中繼
- 適用於任務的 App Mesh 服務和代理整合
- 適用於任務的 Firelens 日誌路由器整合
- EFS 磁碟區
- 下列任務定義參數：
  - `maxSwap`
  - `swappiness`
  - `environmentFiles`
- Fargate Spot 容量提供者
- 映像磁碟區

  Dockerfile `volume` 選項會遭到忽略。而是在任務定義中使用綁定掛載。如需詳細資訊，請參閱 搭配 Amazon ECS 使用綁定掛載。

Windows 停止了 Amazon ECS 的 Fargate 容器映像提取行為

Fargate Windows 緩存最近一個月的, 和上個月的, 服務器核心基礎映像由 Microsoft 提供. 這些映像符合每個週二修補程式更新的 KB/ 組建編號修補程式。例如，在 2024 年 9 月 4 日，Microsoft 發布了適用於視窗服務器 2019 的 KB5036896（17763.5696）。上個月在二零二四年三月十二日的知識庫為 KB5035849。因此，對於平台`WINDOWS_SERVER_2019_CORE`和以`WINDOWS_SERVER_2019_FULL`下容器映像被緩存：

- `mcr.microsoft.com/windows/servercore:ltsc2019`
- `mcr.microsoft.com/windows/servercore:10.0.17763.5696`

- mcr.microsoft.com/windows/servercore:10.0.17763.5576

此外，Microsoft 在 2021 年 9 月 4 日發布了適用於視窗服務器的 KB5036909（20348.2402）。前幾個月在二零二四年三月十二日的知識庫是 KB5035857。因此，對於平台WINDOWS_SERVER_2022_CORE和以WINDOWS_SERVER_2022_FULL下容器映像被緩存：

- mcr.microsoft.com/windows/servercore:ltsc2022
- mcr.microsoft.com/windows/servercore:10.0.20348.2402
- mcr.microsoft.com/windows/servercore:10.0.20348.2340

適用於 Amazon ECS 的 Fargate 任務暫時儲存

佈建後，在 Linux 容器上託管的每個 Amazon ECS 任務都 AWS Fargate 會收到以下用於繫結掛載的暫時儲存。這可以在任務定義中使用 volumes、mountPoints 和 volumesFrom 參數的容器之間進行掛載和共用。在開啟的視窗容器中，不支援此功能 AWS Fargate。

Fargate Linux 容器平台版本

版本 1.4.0 或更新版本

根據預設，使用平台 1.4.0 版或更新版本的託管於 Fargate 上的所有 Amazon ECS 任務都會收到至少 20 GB 的暫時性儲存。暫時性儲存的總量可以增加，最多可達 200 GiB。您可在任務定義中指定 ephemeralStorage 參數來實現這一操作。

任務的提取、壓縮和未壓縮的容器映像都會存放在暫時性儲存中。若要判斷您的任務必須使用的暫時性儲存總量，您必須從為任務配置的暫時性儲存總量中減去容器映像使用的儲存量。

對於 2020 年 5 月 28 日或之後啟動並使用平台版本 1.4.0 或更新版本的任務，會使用 AES-256 加密演算法來加密暫時性儲存。此演算法使用 AWS 擁有的加密金鑰，或者您可以建立自己的客戶管理金鑰。如需詳細資訊，請參閱AWS Fargate 暫時儲存的客戶管理金鑰。

若任務使用 2022 年 11 月 18 日或之後啟動的平台版本 1.4.0 或更高版本，暫時性儲存使用量會透過任務中繼資料端點回報。任務中的應用程式可以查詢任務中繼資料端點版本 4，以取得其暫時性儲存保留大小和使用量。

此外，如果您開啟容器深入解析，暫時儲存保留大小和使用量會傳送至 Amazon CloudWatch 容器深入解析。

> **Note**
>
> Fargate 會在磁盤上保留空間。此空間僅由 Fargate 使用。我們不會向您收費。它不會顯示在這些指標中。但是，您可以在其他工具 (例如 df) 中看到此額外儲存空間。

版本 1.3.0 或更早版本

對於使用平台 `1.3.0` 版或更早版本的 Fargate 任務上的 Amazon ECS，每個任務都會收到下列暫時性儲存。

- 10 GB 的 Docker 層儲存體

  > **Note**
  >
  > 此數量包括壓縮和未壓縮的容器映像成品。

- 額外的 4 GB 磁碟區掛載。這可以在任務定義中使用 `volumes`、`mountPoints` 和 `volumesFrom` 參數的容器之間進行掛載和共用。

Fargate Windows 容器平台版本

版本 1.0.0 或更新版本

根據預設，使用平台 `1.0.0` 版或更新版本的託管於 Fargate 上的所有 Amazon ECS 任務都會收到至少 20 GB 的暫時性儲存。暫時性儲存的總量可以增加，最多可達 200 GiB。您可在任務定義中指定 `ephemeralStorage` 參數來實現這一操作。

任務的提取、壓縮和未壓縮的容器映像都會存放在暫時性儲存中。若要判斷您的任務必須使用的暫時性儲存總量，您必須從為任務配置的暫時性儲存總量中減去容器映像使用的儲存量。

如需詳細資訊，請參閱 搭配 Amazon ECS 使用綁定掛載。

用於 AWS Fargate 暫時儲存的客戶管理金鑰

AWS Fargate 支援客戶受管金鑰，針對存放在暫時儲存的 Amazon ECS 任務加密資料，以協助對法規敏感的客戶滿足其內部安全政策。客戶仍可享有 Fargate 的無伺服器優勢，同時為合規稽核人員提供更強大的自我管理儲存加密能見度。儘管 Fargate 默認情況下具有 Fargate 管理的臨時存儲加密，但客戶還可以在加密敏感數據（例如財務或健康相關信息）時使用自己的自我管理密鑰。

您可以將自己的金鑰匯入 AWS KMS 或在中建立金鑰 AWS KMS。這些自我管理的金鑰會儲存在中，AWS KMS 並執行標準 AWS KMS 生命週期動作，例如輪換、停用和刪除。您可以在 CloudTrail 日誌中審核密鑰訪問和使用情況。

根據預設，KMS 金鑰支援每個金鑰 50,000 筆授權。Fargate 為每個客戶管理的關鍵任務使用單一 AWS KMS 授權，因此一個密鑰最多支持 50,000 個並發任務。如果你想增加這個數字，你可以要求增加限制，這是 case-by-case 基礎上批准的。

Fargate 不收取任何額外費用使用客戶託管的金鑰。您只需支付使用 AWS KMS 金鑰進行儲存和 API 要求的標準價格。

主題

- 為 Fargate 臨時存儲創建加密密鑰
- 管理 Fargate 暫時儲存的 AWS KMS 金鑰

為 Fargate 臨時存儲創建加密密鑰

> **Note**
>
> 使用客戶管理的金鑰進行 Fargate 端暫時性儲存加密不適用於 Windows 工作叢集。
> 使用客戶管理金鑰進行的 Fargate 臨時儲存加密功能不早於. `platformVersions 1.4.0` Fargate 會在僅由 Fargate 使用的暫時儲存空間上保留空間，而您不需為空間付費。配置可能與非客戶管理的金鑰工作不同，但總空間保持不變。您可以在工具中查看此更改`df`。

要在 AWS KMS中創建客戶管理密鑰（CMK）以加密 Fargate 的臨時存儲，請按照下列步驟操作。

1. 瀏覽至 https://console.aws.amazon.com/kms。
2. 按照AWS Key Management Service 開發人員指南中有關創建密鑰的說明進行操作。
3. 創建密 AWS KMS 鑰時，請確保在關鍵策略中提供 Fargate 服務相關的 AWS KMS 操作權限。政策中必須允許下列 API 操作，才能將客戶受管金鑰與 Amazon ECS 叢集資源搭配使用。

   - `kms:GenerateDataKeyWithoutPlainText`-呼叫`GenerateDataKeyWithoutPlainText`從提供的密鑰生成加密的數據密 AWS KMS 鑰。
   - `kms:CreateGrant`-向客戶管理的金鑰新增授權。授予對指定 AWS KMS 金鑰的控制存取權，以允許存取授與 Amazon ECS Fargate 所需的操作。如需有關使用授權的詳細資訊，請參閱開AWS Key Management Service 發人員指南。這允許 Amazon ECS Fargate 執行以下操作：

- 呼叫Decrypt以 AWS KMS 獲取加密密鑰以解密臨時存儲數據。
- 設定退休本金以允許服務。RetireGrant

- kms:DescribeKey-提供客戶受管金鑰詳細資料，讓 Amazon ECS 在金鑰對稱且已啟用時驗證金鑰。

下列範例顯示您要套用至目標 AWS KMS 金鑰以進行加密的金鑰原則。若要使用範例原則陳述式，請以您自己的資訊取代使用*#######*。與往常一樣，只設定您需要的權限。

```
{
      "Sid": "Allow generate data key access for Fargate tasks.",
      "Effect": "Allow",
      "Principal": { "Service":"fargate.amazonaws.com" },
      "Action": [
        "kms:GenerateDataKeyWithoutPlaintext"
      ],
      "Condition": {
        "StringEquals": {
          "kms:EncryptionContext:aws:ecs:clusterAccount": [
            "customerAccountId"
          ],
          "kms:EncryptionContext:aws:ecs:clusterName": [
             "clusterName"
          ]
        }
      },
      "Resource": "*"
    },
    {
      "Sid": "Allow grant creation permission for Fargate tasks.",
      "Effect": "Allow",
      "Principal": { "Service":"fargate.amazonaws.com" },
      "Action": [
        "kms:CreateGrant"
      ],
      "Condition": {
        "StringEquals": {
          "kms:EncryptionContext:aws:ecs:clusterAccount": [
            "customerAccountId"
          ],
          "kms:EncryptionContext:aws:ecs:clusterName": [
             "clusterName"
```

```
            ]
          },
          "ForAllValues:StringEquals": {
            "kms:GrantOperations": [
              "Decrypt"
            ]
          }
        },
        "Resource": "*"
      },
      {
        "Sid": "Allow describe key permission for cluster operator - CreateCluster
 and UpdateCluster.",
        "Effect": "Allow",
        "Principal": { "AWS":"arn:aws:iam::customerAccountId:role/
ClusterOperatorRole" },
        "Action": [
          "kms:DescribeKey"
        ],
        "Resource": "*"
      }
```

Fargate 工作會使用aws:ecs:clusterAccount和aws:ecs:clusterName加密內容金鑰，以金鑰進行加密編譯作業。客戶應新增這些權限，以限制對特定帳戶和/或叢集的存取。

如需詳細資訊，請參閱 AWS KMS 開發人員指南中的加密內容。

建立或更新叢集時，您可以選擇使用條件索引鍵fargateEphemeralStorageKmsKeyId。此條件金鑰可讓客戶更精細地控制 IAM 政策。fargateEphemeralStorageKmsKeyId組態的更新只會在新服務部署上生效。

以下是允許客戶僅授與一組特定核准 AWS KMS 金鑰的權限的範例。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ecs:CreateCluster",
        "ecs:UpdateCluster"
      ],
```

```
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "ecs:fargate-ephemeral-storage-kms-key": "arn:aws:kms:us-
west-2:111122223333:key/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"
        }
      }
    }
  ]
}
```

接下來是拒絕嘗試移除已與叢集相關聯之 AWS KMS 金鑰的範例。

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Deny",
    "Action": [
        "ecs:CreateCluster",
        "ecs:UpdateCluster"
    ],
    "Resource": "*",
    "Condition": {
      "Null": {
        "ecs:fargate-ephemeral-storage-kms-key": "true"
      }
    }
  }
}
```

客戶可以使用 AWS CLI `describe-tasks`、`describe-cluster`或`describe-services`命令，查看其未受管理的工作或服務工作是否使用金鑰加密。

如需詳細資訊，請參閱AWS KMS 開發人員指南 AWS KMS中的條件金鑰。

AWS Management Console

1. 開啟主控台，網址為 https://console.aws.amazon.com/ecs/v2。
2. 在左側導覽中選擇 [叢集]，然後選擇右上角的 [建立叢集]，或選擇現有叢集。對於現有叢集，請選擇右上角的 [更新叢集]。

3. 在工作流程的加密部分下，您可以選擇在託管存儲和 Fargate 臨時存儲下選擇密 AWS KMS 鑰。您也可以選擇從這裡建立 AWS KMS 金鑰。
4. 完成建立新叢集後，請選擇 [建立]，或選擇 [更新] (如果您要更新現有叢集)。

AWS CLI

以下是建立叢集和使用 (將*##*值取代為您自己的值 AWS CLI) 設定 Fargate 暫時儲存的範例：

```
aws ecs create-cluster --cluster clusterName \
--configuration '{"managedStorageConfiguration":
{"fargateEphemeralStorageKmsKeyId":"arn:aws:kms:us-
west-2:012345678901:key/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"}}'
{
    "cluster": {
        "clusterArn": "arn:aws:ecs:us-west-2:012345678901:cluster/clusterName",
        "clusterName": "clusterName",
        "configuration": {
            "managedStorageConfiguration": {
                "fargateEphemeralStorageKmsKeyId": "arn:aws:kms:us-
west-2:012345678901:key/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"
            }
        },
        "status": "ACTIVE",
        "registeredContainerInstancesCount": 0,
        "runningTasksCount": 0,
        "pendingTasksCount": 0,
        "activeServicesCount": 0,
        "statistics": [],
        "tags": [],
        "settings": [],
        "capacityProviders": [],
        "defaultCapacityProviderStrategy": []
    },
    "clusterCount": 5
}
```

AWS CloudFormation

以下範例範本可用來建立叢集並使用 (將*##*值取代為您自己的值 AWS CloudFormation) 來設定 Fargate 暫時儲存：

```
AWSTemplateFormatVersion: 2010-09-09
```

```
Resources:
  MyCluster:
    Type: AWS::ECS::Cluster
    Properties:
      ClusterName: "clusterName"
      Configuration:
        ManagedStorageConfiguration:
          FargateEphemeralStorageKmsKeyId: "arn:aws:kms:us-
west-2:012345678901:key/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"
```

管理 Fargate 暫時儲存的 AWS KMS 金鑰

創建或導入密 AWS KMS 鑰以加密 Fargate 臨時存儲後，您可以像對其他密鑰一樣進行管理。 AWS KMS

自動旋轉 AWS KMS 按鍵

您可以啟用自動按鍵旋轉或手動旋轉它們。自動金鑰輪換會為金鑰產生新的加密資料，每年為您旋轉金鑰。 AWS KMS 還可以保存所有先前版本的加密材料，因此您將能夠解密使用舊金鑰版本的任何資料。除 AWS KMS 非您刪除鍵，否則任何旋轉的材料都不會被刪除。

自動按鍵旋轉是可選的，可以隨時啟用或停用。

停用或撤銷金鑰 AWS KMS

如果您停用客戶管理的金鑰 AWS KMS，則不會對執行中的工作產生任何影響，而且它們會在整個生命週期中繼續運作。如果新工作使用停用或已撤銷的金鑰，則工作會因為無法存取金鑰而失敗。您應該設置 CloudWatch 警報或類似警報，以確保從不需要禁用的密鑰來解密已加密的數據。

刪除 AWS KMS 金鑰

刪除密鑰應該永遠是最後的手段，只有在確定不再需要刪除的密鑰時才應該這樣做。嘗試使用已刪除金鑰的新工作將會失敗，因為無法存取該金鑰。 AWS KMS 建議禁用密鑰而不是刪除它。如果您覺得有必要刪除金鑰，我們建議您先將其停用，並設定 CloudWatch 鬧鐘以確保不需要。如果您確實刪除了金鑰，請至少提 AWS KMS 供七天的時間來改變主意。

稽核 AWS KMS 金鑰存取

您可以使用 CloudTrail 記錄來稽核 AWS KMS 金鑰的存取權。您可以檢查作 AWS KMS 業`CreateGrant`、`GenerateDataKeyWithoutPlaintext`、和`Decrypt`。這些作業也會顯

示aws:ecs:clusterAccount和aws:ecs:clusterName作為EncryptionContext登入的一部分CloudTrail。

以下是GenerateDataKeyWithoutPlaintext、、GenerateDataKeyWithoutPlaintext (DryRun)CreateGrant、CreateGrant (DryRun)和的範例 CloudTrail 事件 RetireGrant (用您自己的值取代##值)。

GenerateDataKeyWithoutPlaintext


```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AWSService",
        "invokedBy": "ec2-frontend-api.amazonaws.com"
    },
    "eventTime": "2024-04-23T18:08:13Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "GenerateDataKeyWithoutPlaintext",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "ec2-frontend-api.amazonaws.com",
    "userAgent": "ec2-frontend-api.amazonaws.com",
    "requestParameters": {
        "numberOfBytes": 64,
        "keyId": "arn:aws:kms:us-west-2:account-id:key/a1b2c3d4-5678-90ab-cdef-
EXAMPLE11111",
        "encryptionContext": {
            "aws:ecs:clusterAccount": "account-id",
            "aws:ebs:id": "vol-xxxxxxx",
            "aws:ecs:clusterName": "cluster-name"
        }
    },
    "responseElements": null,
    "requestID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
    "eventID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE33333",
    "readOnly": true,
    "resources": [
        {
            "accountId": "AWS Internal",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-west-2:account-id:key/a1b2c3d4-5678-90ab-cdef-
EXAMPLE11111"
        }
    ],
```

```
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "account-id",
    "sharedEventID": "a1b2c3d4-5678-90ab-cdef-EXAMPLEaaaaa",
    "eventCategory": "Management"
}
```

GenerateDataKeyWithoutPlaintext (DryRun)

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AWSService",
        "invokedBy": "fargate.amazonaws.com"
    },
    "eventTime": "2024-04-23T18:08:11Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "GenerateDataKeyWithoutPlaintext",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "fargate.amazonaws.com",
    "userAgent": "fargate.amazonaws.com",
    "errorCode": "DryRunOperationException",
    "errorMessage": "The request would have succeeded, but the DryRun option is
 set.",
    "requestParameters": {
        "keyId": "arn:aws:kms:us-west-2:account-id:key/a1b2c3d4-5678-90ab-cdef-
EXAMPLE11111",
        "dryRun": true,
        "numberOfBytes": 64,
        "encryptionContext": {
            "aws:ecs:clusterAccount": "account-id",
            "aws:ecs:clusterName": "cluster-name"
        }
    },
    "responseElements": null,
    "requestID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
    "eventID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE33333",
    "readOnly": true,
    "resources": [
        {
            "accountId": "AWS Internal",
            "type": "AWS::KMS::Key",
```

```
            "ARN": "arn:aws:kms:us-west-2:account-id:key/a1b2c3d4-5678-90ab-cdef-
EXAMPLE11111"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "account-id",
    "sharedEventID": "a1b2c3d4-5678-90ab-cdef-EXAMPLEaaaaa",
    "eventCategory": "Management"
}
```

CreateGrant

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AWSService",
        "invokedBy": "ec2-frontend-api.amazonaws.com"
    },
    "eventTime": "2024-04-23T18:08:13Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "CreateGrant",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "ec2-frontend-api.amazonaws.com",
    "userAgent": "ec2-frontend-api.amazonaws.com",
    "requestParameters": {
        "keyId": "arn:aws:kms:us-west-2:account-id:key/a1b2c3d4-5678-90ab-cdef-
EXAMPLE11111",
        "granteePrincipal": "fargate.us-west-2.amazonaws.com",
        "operations": [
            "Decrypt"
        ],
        "constraints": {
            "encryptionContextSubset": {
                "aws:ecs:clusterAccount": "account-id",
                "aws:ebs:id": "vol-xxxx",
                "aws:ecs:clusterName": "cluster-name"
            }
        },
        "retiringPrincipal": "ec2.us-west-2.amazonaws.com"
    },
    "responseElements": {
```

```
        "grantId":
 "e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855",
        "keyId": "arn:aws:kms:us-west-2:account-id:key/a1b2c3d4-5678-90ab-cdef-
EXAMPLE11111"
    },
    "requestID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
    "eventID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE33333",
    "readOnly": false,
    "resources": [
        {
            "accountId": "AWS Internal",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-west-2:account-id:key/a1b2c3d4-5678-90ab-cdef-
EXAMPLE11111"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "account-id",
    "sharedEventID": "a1b2c3d4-5678-90ab-cdef-EXAMPLEaaaaa",
    "eventCategory": "Management"
}
```

CreateGrant (DryRun)

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AWSService",
        "invokedBy": "fargate.amazonaws.com"
    },
    "eventTime": "2024-04-23T18:08:11Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "CreateGrant",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "fargate.amazonaws.com",
    "userAgent": "fargate.amazonaws.com",
    "errorCode": "DryRunOperationException",
    "errorMessage": "The request would have succeeded, but the DryRun option is
 set.",
    "requestParameters": {
        "keyId": "arn:aws:kms:us-west-2:account-id:key/a1b2c3d4-5678-90ab-cdef-
EXAMPLE11111",
```

```
        "granteePrincipal": "fargate.us-west-2.amazonaws.com",
        "dryRun": true,
        "operations": [
            "Decrypt"
        ],
        "constraints": {
            "encryptionContextSubset": {
                "aws:ecs:clusterAccount": "account-id",
                "aws:ecs:clusterName": "cluster-name"
            }
        }
    },
    "responseElements": null,
    "requestID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
    "eventID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE33333",
    "readOnly": false,
    "resources": [
        {
            "accountId": "AWS Internal",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-west-2:account-id:key/a1b2c3d4-5678-90ab-cdef-
EXAMPLE11111"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "account-id",
    "sharedEventID": "a1b2c3d4-5678-90ab-cdef-EXAMPLEaaaaa",
    "eventCategory": "Management"
}
```

RetireGrant

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AWSService",
        "invokedBy": "AWS Internal"
    },
    "eventTime": "2024-04-20T18:37:38Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "RetireGrant",
    "awsRegion": "us-west-2",
```

```
    "sourceIPAddress": "AWS Internal",
    "userAgent": "AWS Internal",
    "requestParameters": null,
    "responseElements": {
        "keyId": "arn:aws:kms:us-west-2:account-id:key/a1b2c3d4-5678-90ab-cdef-
EXAMPLE11111"
    },
    "additionalEventData": {
        "grantId":
 "e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855"
    },
    "requestID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
    "eventID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE33333",
    "readOnly": false,
    "resources": [
        {
            "accountId": "AWS Internal",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-west-2:account-id:key/a1b2c3d4-5678-90ab-cdef-
EXAMPLE11111"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "account-id",
    "sharedEventID": "a1b2c3d4-5678-90ab-cdef-EXAMPLEaaaaa",
    "eventCategory": "Management"
}
```

AWS Amazon ECS 上的 Fargate 任務維護常見問題解答

什麼是 Fargate 任務維護和退休？

AWS 負責維護 AWS Fargate 的基礎設施。 AWS 決定何時需要將平台版本修訂版取代為基礎結構的新修訂版本。這就是所謂的任務退休。 AWS 當平台版本修訂已淘汰時，傳送作業淘汰通知。我們會定期更新支援的平台版本，以引入包含 Fargate 執行階段軟體的更新以及基礎相依性 (例如作業系統和容器執行階段) 的新版本。一旦提供了更新的修訂版本，我們將淘汰舊版本，以確保所有客戶的工作負載都在 Fargate 平台版本的最新版本上運行。當修訂版本淘汰時，在該修訂版本上執行的所有任務都會停止。

Amazon ECS 任務可以分類為服務任務和獨立任務。服務任務是作為服務的一部分進行部署，並由 Amazon ECS 排程控制。如需詳細資訊，請參閱 Amazon ECS 服務。獨立任務是由 Amazon ECS RunTask API 直接或由外部排程器啟動的任務，例如排程任務 (由 Amazon 啟動 EventBridge) AWS Batch、或 AWS Step Functions。

對於服務工作，除非您想要先 AWS 取代這些工作，否則您不需要採取任何動作。當 Amazon ECS 排程器停止任務時，它會使用最低運作狀態百分比，並啟動新任務以嘗試維持所需的服務計數。根據預設，服務的運作狀態百分比下限為 100%，因此在任務停止前會先啟動新任務。當您擴展服務、部署組態變更或部署工作定義修訂時，會以相同的方式來取代服務工作。若要為任務淘汰程序做準備，建議您透過模擬此情境來測試應用程式的行為。方式如下：停止您服務中的個別任務，藉以測試復原能力。

對於獨立的任務處分，在任務處分日期當天或之後停止任務。當任務停止時，我們不 AWS 會啟動替代任務。如果您需要繼續執行這些工作，您必須停止執行中的工作，並在通知中指定的時間之前啟動取代工作。因此，我們建議客戶監控獨立任務的狀態，並視需要實作邏輯以取代已停止的任務。

當任務在任何場景下停止時，您都可以執行 `describe-tasks`。回應中的 `stoppedReason` 是 `ECS is performing maintenance on the underlying infrastructure hosting the task`。

當有新的平台版本修訂需要以新的修訂版本取代時，會套用任務維護。如果基礎 Fargate 主機發生問題，Amazon ECS 會在沒有任務淘汰通知的情況下取代主機。

什麼是在任務退休通知？

工作淘汰通知會透過 AWS Health 儀表板傳送，也會透過電子郵件傳送至註冊的電子郵件地址，並包含下列資訊：

- 任務淘汰日期 - 任務將在此日期或之後停止。
- 若為獨立任務，則為任務的 ID。
- 若為服務任務，則為執行服務所在之叢集的 ID 以及服務的 ID。
- 您需要採取的後續步驟。

通常，我們會針對每個服務和獨立任務傳送一個通知 AWS 區域。但是，在某些情況下，您可能會針對每個任務類型收到多個事件，例如，當有太多要淘汰的任務而超過我們通知機制中的限制時。

您可以透過以下方法確定已排程要淘汰的任務：

- 該 AWS Health Dashboard

AWS Health 通知可透過 Amazon 傳送 EventBridge 至存檔儲存 (例如 Amazon 簡單儲存服務)、採取自動化動作 (例如執行 AWS Lambda 功能) 或其他通知系統 (例如 Amazon 簡單通知服務)。如需詳細資訊，請參閱使用 Amazon 監控 AWS Health 事件 EventBridge。如需將通知傳送至 Amazon Chime、Slack 或 Microsoft 團隊的範例組態，請參閱上AWS Health 的。 GitHub

以下是範例 EventBridge 事件。

```
{
    "version": "0",
    "id": "3c268027-f43c-0171-7425-1d799EXAMPLE",
    "detail-type": "AWS Health Event",
    "source": "aws.health",
    "account": "123456789012",
    "time": "2023-08-16T23:18:51Z",
    "region": "us-east-1",
    "resources": [
        "cluster/service",
        "cluster/service"
    ],
    "detail": {
        "eventArn": "arn:aws:health:us-east-1::event/ECS/
AWS_ECS_TASK_PATCHING_RETIREMENT/AWS_ECS_TASK_PATCHING_RETIREMENT_test1",
        "service": "ECS",
        "eventScopeCode": "ACCOUNT_SPECIFIC",
        "communicationId":
 "7988399e2e6fb0b905ddc88e0e2de1fd17e4c9fa60349577446d95a18EXAMPLE",
        "lastUpdatedTime": "Wed, 16 Aug 2023 23:18:52 GMT",
        "eventRegion": "us-east-1",
        "eventTypeCode": "AWS_ECS_TASK_PATCHING_RETIREMENT",
        "eventTypeCategory": "scheduledChange",
        "startTime": "Wed, 16 Aug 2023 23:18:51 GMT",
        "endTime": "Fri, 18 Aug 2023 23:18:51 GMT",
        "eventDescription": [
            {
                "language": "en_US",
                "latestDescription": "\\nA software update has been deployed to
 Fargate which includes CVE patches or other critical patches. No action is required
 on your part. All new tasks launched automatically uses the latest software
 version. For existing tasks, your tasks need to be restarted in order for these
 updates to apply. Your tasks running as part of the following ECS Services will
 be automatically updated beginning Wed, 16 Aug 2023 23:18:51 GMT.\\n\\nAfter Wed,
 16 Aug 2023 23:18:51 GMT, the ECS scheduler will gradually replace these tasks,
```

```
 respecting the deployment settings for your service. Typically, services should
 see little to no interruption during the update and no action is required. When AWS
 stops tasks, AWS uses the minimum healthy percent (1) and launches a new task in
 an attempt to maintain the desired count for the service. By default, the minimum
 healthy percent of a service is 100 percent, so a new task is started first before
 a task is stopped. Service tasks are routinely replaced in the same way when
 you scale the service or deploy configuration changes or deploy task definition
 revisions. If you would like to control the timing of this restart you can update
 the service before Wed, 16 Aug 2023 23:18:51 GMT, by running the update-service
 command from the ECS command-line interface specifying force-new-deployment for
 services using Rolling update deployment type. For example:\\n\\n$ aws ecs update-
service -service service_name \\\n--cluster cluster_name -force-new-deployment\
\n\\nFor services using Blue/Green deployment type with AWS CodeDeploy:\\nPlease
 refer to create-deployment document (2) and create new deployment using same task
 definition revision.\\n\\nFor further details on ECS deployment types, please
 refer to ECS Deployment Developer Guide (1).\\nFor further details on Fargate's
 update process, please refer to the AWS Fargate User Guide (3).\\nIf you have
 any questions or concerns, please contact AWS Support (4).\\n\\n(1) https://
docs.aws.amazon.com/AmazonECS/latest/developerguide/deployment-types.html\\n(2)
 https://docs.aws.amazon.com/cli/latest/reference/deploy/create-deployment.html\\n(3)
 https://docs.aws.amazon.com/AmazonECS/latest/userguide/task-maintenance.html\\n(4)
 https://aws.amazon.com/support\\n\\nA list of your affected resources(s) can be
 found in the 'Affected resources' tab in the 'Cluster/ Service' format in the AWS
 Health Dashboard. \\n\\n"
            }
        ],
        "affectedEntities": [
            {
                "entityValue": "cluster/service"
            },
            {
                "entityValue": "cluster/service"
            }
        ]
    }
}
```

- 電子郵件

  系統會將一封電子郵件傳送至 AWS 帳戶 ID 的註冊電子郵件。

我可以更改任務的退休等待時間嗎？

您可以設定 Fargate 開始任務淘汰的時間。對於需要立即套用更新的工作負載，請選擇立即設定 (0)。當您需要更多控制項時，例如當任務只能在特定時段中停止時，請設定 7 日 (7) 或 14 日 (14) 選項。

我們建議您選擇較短的等待期，以便更快獲得更新的平台版本修訂版。

透過執行`put-account-setting-default`或`put-account-setting`以 root 使用者或系統管理使用者身分設定等待期間。對 `name` 使用 `fargateTaskRetirementWaitPeriod` 選項，並將 `value` 選項設定為以下值之一：

- `0`- AWS 傳送通知，並立即開始淘汰受影響的工作。
- 7- AWS 傳送通知，並在開始淘汰受影響的工作前等待 7 個行事曆天。
- 14 - AWS 傳送通知，並等待 14 個日曆日，然後再開始淘汰受影響的任務。

預設值是 7 天。

如需詳細資訊，請參閱《Amazon Elastic Container Service API 參考》中的 put-account-setting-default 和 put-account-setting。

如需詳細資訊，請參閱 AWS Fargate 任務退休等待時間。

我可以透過其他 AWS 服務取得任務退休通知嗎？

AWS 會將作業淘汰通知傳送給 AWS Health Dashboard 與上的主要電子郵件連絡人 AWS 帳戶。提 AWS Health Dashboard 供了許多與其他 AWS 服務的集成，包括 EventBridge. 您可以使 EventBridge 用自動化通知的可見性（例如，將消息轉發到 ChatOps 工具）。如需詳細資訊，請參閱解決方案概觀：擷取作業處分通知。

我可以在排程之後變更任務退休嗎？

沒有 排程是以預設值為 7 天的作業處分等待時間為基礎。如果您需要更多時間，可以選擇將等待期設定為 14 天。如需詳細資訊，請參閱 我可以更改任務的退休等待時間嗎？。此組態中的變更適用於將 future 排程的淘汰。目前排定的處分不會受到影響。如果您有任何進一步的疑慮，請聯繫 AWS Support。

我可以控制任務替換的時間嗎？

對於使用滾動式部署的服務，您可以在淘汰開始時間之前使`update-service`用該`force-deployment`選項來更新服務。

下列update-service範例會使用force-deployment選項。

```
aws ecs update-service –-service service_name \
    --cluster cluster_name \
     --force-new-deployment
```

對於使用藍/綠部署的服務，您需要在中 AWS CodeDeploy建立新部署。如需如何建立部署的相關資訊，請參閱〈參考〉中的〈建立部署〉。AWS Command Line Interface

Amazon ECS 如何處理屬於服務一部分的任務？

當 Fargate 退休期開始時，Amazon ECS 會逐漸取代您服務中受影響的任務。當 Amazon ECS 停止任務時，它會使用服務的最低運作狀態百分比，並啟動新任務來維護服務所需的任務計數。因為預設的最低健全狀況百分比為 100，因此在工作停止之前就會啟動新工作。當您擴展服務、部署組態變更或部署工作定義修訂時，會以相同的方式來取代服務工作。如需有關最小健康百分比的詳細資訊，請參閱部署組態。

Amazon ECS 能否自動處理獨立任務？

沒有 AWS 無法為啟動的獨立工作RunTask、排程工作 (例如透過 EventBridge Scheduler) 或建立取代工作 AWS Step Functions。 AWS Batch Amazon ECS 只會管理屬於服務一部分的任務。

支援 AWS Fargate 上 Amazon ECS 的區域

您可以使用下表來驗證 Fargate 上 Linux 容器的區域支援，以及 AWS Far AWS gate 上的 Windows 容器。

AWS Fargate 上的 Linux 容器

以 AWS 區域下 AWS Fargate 是支援上的 Amazon ECS Linux 容器。適用時，支援的可用區域 ID 會註明。

區域名稱	區域
美國東部 (俄亥俄)	us-east-2
美國東部 (維吉尼亞北部)	us-east-1
美國西部 (加利佛尼亞北部)	us-west-1 (僅 usw1-az1 和 usw1-az3)

區域名稱	區域
美國西部 (奧勒岡)	us-west-2
非洲 (開普敦)	af-south-1
亞太區域 (香港)	ap-east-1
亞太區域 (孟買)	ap-south-1
亞太區域 (東京)	ap-northeast-1 (僅 `apne1-az1` 、`apne1-az2` 和 `apne1-az4`)
亞太區域 (首爾)	ap-northeast-2
亞太區域 (大阪)	ap-northeast-3
亞太區域 (海德拉巴)	ap-south-2
亞太區域 (新加坡)	ap-southeast-1
亞太區域 (雪梨)	ap-southeast-2
亞太區域 (雅加達)	ap-southeast-3
亞太區域 (墨爾本)	ap-southeast-4
加拿大 (中部)	ca-central-1
加拿大西部 (卡加利)	ca-west-1
中國 (北京)	cn-north-1 (僅 `cnn1-az1` 和 `cnn1-az2`)
中國 (寧夏)	cn-northwest-1
歐洲 (法蘭克福)	eu-central-1
歐洲 (蘇黎世)	eu-central-2
歐洲 (愛爾蘭)	eu-west-1
歐洲 (倫敦)	eu-west-2

區域名稱	區域
歐洲 (巴黎)	eu-west-3
歐洲 (米蘭)	eu-south-1
歐洲 (西班牙)	eu-south-2
歐洲 (斯德哥爾摩)	eu-north-1
南美洲 (聖保羅)	sa-east-1
以色列 (特拉維夫)	il-central-1
中東 (巴林)	me-south-1
中東 (阿拉伯聯合大公國)	me-central-1
AWS GovCloud (美國東部)	us-gov-east-1
AWS GovCloud (美國西部)	us-gov-west-1

AWS Fargate 上的視窗容器

以 AWS 區域下支援 Amazon ECS 視窗容器。 AWS Fargate 適用時，支援的可用區域 ID 會註明。

區域名稱	區域
美國東部 (俄亥俄)	us-east-2
美國東部 (維吉尼亞北部)	us-east-1 (只有 `use1-az1`、`use1-az2`、`use1-az4`、`use1-az5` 和 `use1-az6`)
美國西部 (加利佛尼亞北部)	us-west-1 (僅 `usw1-az1` 和 `usw1-az3`)
美國西部 (奧勒岡)	us-west-2
非洲 (開普敦)	af-south-1
亞太區域 (香港)	ap-east-1

區域名稱	區域
亞太區域 (孟買)	ap-south-1
亞太區域 (海德拉巴)	ap-south-2
亞太區域 (大阪)	ap-northeast-3
亞太區域 (首爾)	ap-northeast-2
亞太區域 (新加坡)	ap-southeast-1
亞太區域 (雪梨)	ap-southeast-2
亞太區域 (墨爾本)	ap-southeast-4
亞太區域 (東京)	ap-northeast-1 (僅 apne1-az1 、apne1-az2 和 apne1-az4)
加拿大 (中部)	ca-central-1 (僅 cac1-az1 和 cac1-az2)
加拿大西部 (卡加利)	ca-west-1
中國 (北京)	cn-north-1 (僅 cnn1-az1 和 cnn1-az2)
中國 (寧夏)	cn-northwest-1
歐洲 (法蘭克福)	eu-central-1
歐洲 (蘇黎世)	eu-central-2
歐洲 (愛爾蘭)	eu-west-1
歐洲 (倫敦)	eu-west-2
歐洲 (巴黎)	eu-west-3
歐洲 (米蘭)	eu-south-1
歐洲 (西班牙)	eu-south-2
歐洲 (斯德哥爾摩)	eu-north-1

區域名稱	區域
南美洲 (聖保羅)	sa-east-1
以色列 (特拉維夫)	il-central-1
中東 (阿拉伯聯合大公國)	me-central-1
中東 (巴林)	me-south-1

為 Amazon ECS 建構您的解決方案

在使用 Amazon ECS 之前，您需要做出有關容量、聯網、帳戶設定和日誌記錄的決策，以便能夠正確設定 Amazon ECS 資源。

容量

容量是容器執行的基礎結構。以下是選項：

- Amazon EC2 執行個體
- 無伺服器 ()AWS Fargate (Fargate)
- 內部部署虛擬機器或伺服器

您可以在建立叢集時指定基礎結構。您也可以在註冊工作定義時指定基礎結構類型。任務定義將基礎結構稱為「啟動類型」。當您執行獨立工作或部署服務時，也會使用啟動類型。如需啟動類型選項的詳細資訊，請參閱Amazon ECS 啟動類型。

聯網

AWS 資源是在子網路中建立的。當您使用 EC2 執行個體時，Amazon ECS 會在您建立叢集時指定的子網路中啟動執行個體。您的工作會在執行個體子網路中執行。對於 Fargate 或內部部署虛擬機器，您可以在執行工作或建立服務時指定子網路。

視您的應用程式而定，子網路可以是私有子網路或公用子網路，而子網路可以位於下列任一 AWS 資源中：

- 可用區域
- 本機區域
- Wavelength 區域
- AWS 區域
- AWS Outposts

如需詳細資訊，請參閱 共用子網路、Local Zones 和 Wavelength 區域中的 Amazon ECS 應用程式 或 Amazon 彈性容器服務 AWS Outposts 。

您可以使用下列其中一種方法，讓應用程式連線到網際網路：

- 具有網際網路閘道的公用子網路

  當您擁有需要大量頻寬或最小延遲的公用應用程式時，請使用公用子網路。適用場景包括視頻流和遊戲服務。

- 具有 NAT 閘道的私有子網路

  當您想要保護容器免於直接外部存取時，請使用私有子網路。適用案例包括付款處理系統或儲存使用者資料和密碼的容器。

功能存取

您可以使用 Amazon ECS 帳戶設定存取下列功能：

- Container Insights

  CloudWatch Container Insights 會從您的容器化應用程式和微服務收集、彙總和摘要指標和記錄。指標包含 CPU、記憶體、磁碟和網路這類資源的使用率。

- awsvpc集中管道

  對於某些 EC2 執行個體類型，您可以在新啟動的容器執行個體上使用其他網路界面 (ENI)。

- 標記授權

  使用者必須具有建立資源之動作的權限，例如ecsCreateCluster。如果在資源建立動作中指定了標籤，請對動作 AWS 執行其他授權，以確認使用者或角色是否具有建立標籤的權限。ecs:TagResource

- Fargate FIPS-140 法規遵循

  Fargate 支援聯邦資訊處理標準 (FIPS-140)，該標準具體說明了保護敏感資訊的加密模組的安全要求。這是目前的美國和加拿大政府標準，且適用於需要符合聯邦資訊安全管理法案 (FISMA) 或聯邦風險和授權管理計劃 (FedRAMP) 的系統。

- Fargate 任務退休時間變化

  您可以設定停用 Fargate 工作以進行修補之前的等待期間。

- 雙堆疊 VPC

  允許工作透過 IPv4、IPv6 或兩者進行通訊。

- Amazon Resource Name (ARN) 格式

  某些功能 (例如標記授權) 需要新的 Amazon 資源名稱 (ARN) 格式。

如需詳細資訊，請參閱 透過帳戶設定存取 Amazon ECS 功能。

IAM 角色

IAM 角色是您可以在帳戶中建立的另一種 IAM 身分，具有特定的許可。在 Amazon ECS 中，您可以建立角色，將許可授與 Amazon ECS 資源 (例如容器或服務)。

某些 Amazon ECS 功能需要角色。如需詳細資訊，請參閱 Amazon ECS 的 IAM 角色。

日誌

記錄和監控是維護 Amazon ECS 工作負載的可靠性、可用性和效能的重要方面。以下是可用的選項：

- Amazon CloudWatch 日誌-路由日誌 Amazon CloudWatch
- FireLens適用於 Amazon ECS-將日誌路由到 AWS 服務或 AWS Partner Network 目的地以進行日誌儲存和分析。這 AWS Partner Network 是一個由合作夥伴組成的全球社群，利用計畫、專業知識和資源來建置、行銷和銷售客戶產品。

Amazon ECS 啟動類型

例如，作業定義啟動類型會定義工作可以執行的容量 AWS Fargate。

選擇啟動類型後，Amazon ECS 會驗證您設定的任務定義參數是否與啟動類型一起使用。

Fargate

Fargate 是無伺服器的 pay-as-you-go 運算引擎，可讓您專注於建置應用程式，而無需管理伺服器。當您選擇 Fargate 時，您不需要管理 EC2 基礎設施。您只需要建置容器映像檔，並定義要在哪個叢集上執行應用程式。Fargate 與 AWS 服務的原生集成，包括：

- Amazon VPC
- Auto Scaling

- Elastic Load Balancing
- IAM
- Secrets Manager

與 EC2 相比，Fargate 擁有更多的控制權，因為您可以選擇應用程式所需的確切 CPU 和記憶體。Fargate 可處理您的容量擴展，因此您無需擔心流量尖峰。這意味著使用 Fargate 的操作工作量較少。

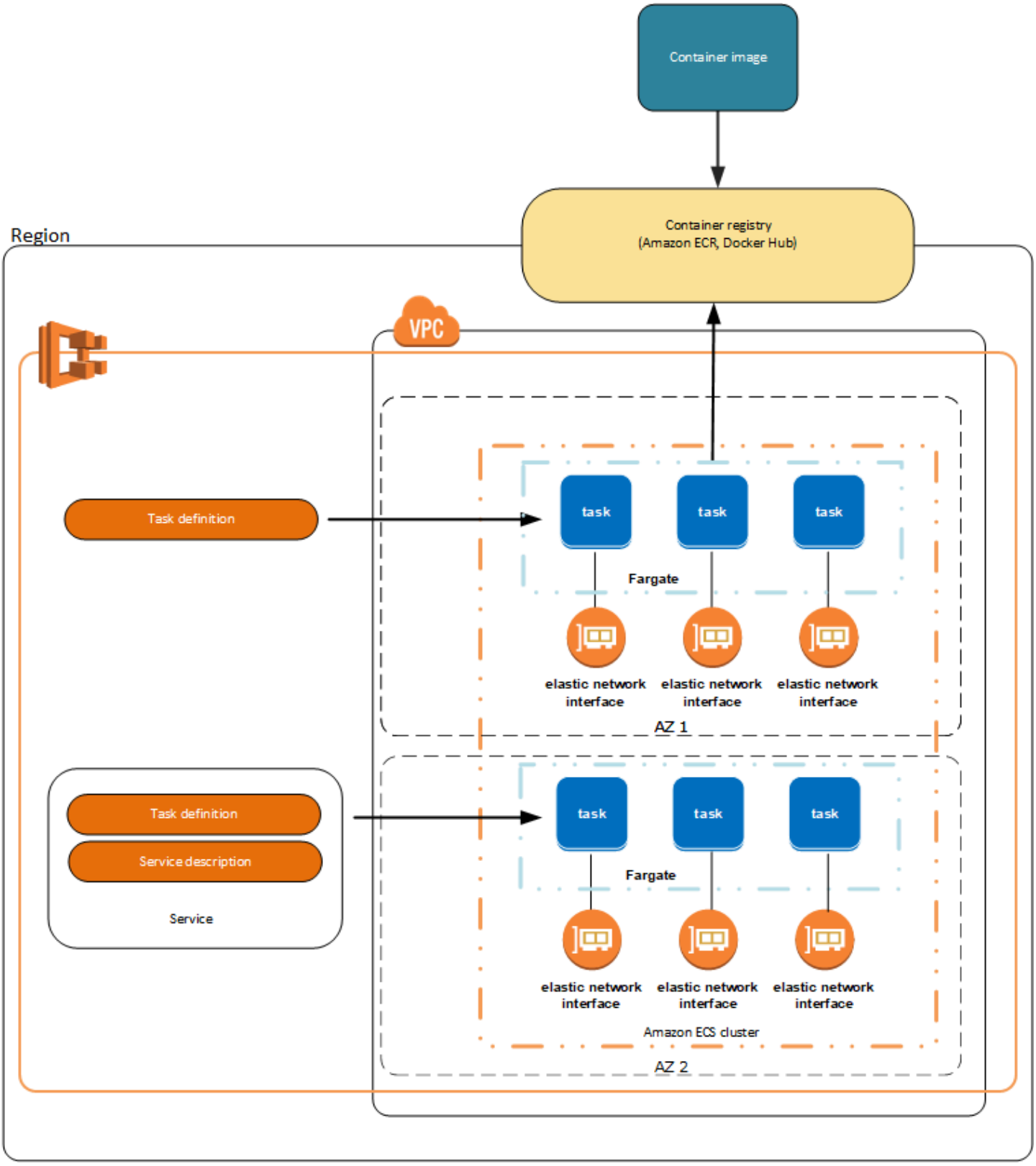
Fargate 符合合規性計劃的標準，包括 PCI，FIPS 140-2，FedRAMP 和 HIPAA。如需詳細資訊，請參閱合規計劃範圍內的AWS 服務。

Fargate 適用於下列工作負載：

- 需要低廉營運開銷的大型工作負載
- 偶爾會爆量的小型工作負載
- 微小工作負載
- 批次工作負載

如需支援 Fargate 的區域的資訊，請參閱the section called “AWS Fargate 地區”。

下圖顯示一般架構。



如需 Fargate 上 Amazon ECS 的詳細資訊，請參閱 AWS Fargate對於 Amazon ECS。

EC2

EC2 啟動類型適用於必須為價格最佳化的大型工作負載。

當考慮如何使用 EC2 啟動類型塑造任務定義和服務時，建議您思考哪些程序需要一起執行，以及如何擴展每個元件。

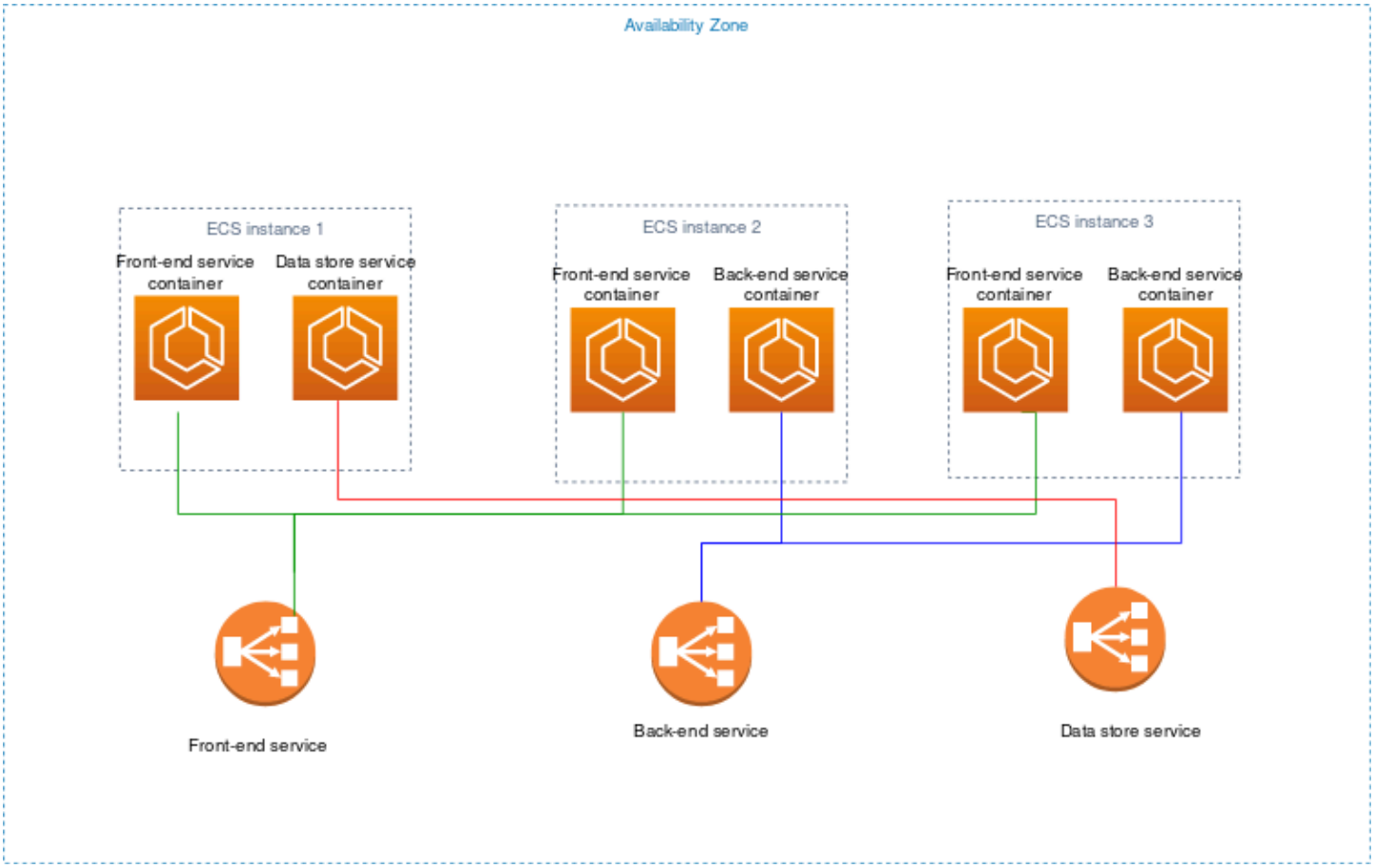
舉例來說，假設由下列元件組成的應用程式：

- 在網頁顯示資訊的前端服務
- 為前端服務提供 API 的後端服務
- 資料存放區

在此範例中，請建立將用於常見用途的容器分為同一群組的任務定義。將不同的元件分隔為多個單獨的任務定義。下列的範例叢集具有在三個前端服務容器上執行的三個容器執行個體、兩個後端服務容器，以及一個資料存放區服務容器。

您可以將任務定義中的相關容器分組，例如必須一起執行的連結容器。例如，將日誌串流容器新增到您的前端服務，並將此服務包含在相同的任務定義中。

在您擁有任務定義之後，您可以從中建立服務，以維護您所需任務的可用性。如需詳細資訊，請參閱使用主控台建立 Amazon ECS 服務。在您的服務中，您可以建立容器與 Elastic Load Balancing 負載平衡器的關聯。如需詳細資訊，請參閱 使用負載平衡來分配 Amazon ECS 服務流量。當您的應用程式需求變更時，您可以更新您的服務以調整所需任務的數量。或者，您也可以更新您的服務以部署任務中的新版容器。如需詳細資訊，請參閱 使用主控台更新 Amazon ECS 服務。



外部

外部啟動類型用於在內部部署伺服器或虛擬機器 (VM) 上執行容器化的應用程式，您將它們註冊到 Amazon ECS 叢集並進行遠端管理。如需詳細資訊，請參閱 適用於外部啟動類型的 Amazon ECS 叢集。

共用子網路、Local Zones 和 Wavelength 區域中的 Amazon ECS 應用程式

Amazon ECS 支援使用 Local Zones、Wavelength 區域以及 AWS Outposts 需要低延遲或本機資料處理的工作負載。

- 您可以使用「Local Zones」做為延伸， AWS 區域 將資源放置在距離最終使用者更近的多個位置。
- 您可以使用 Wavelength Zone 建置可為 5G 裝置與最終使用者提供極低延遲的應用程式。Wavelength 將標準 AWS 運算和儲存服務部署到電信業者的 5G 網路邊緣。

- AWS Outposts 將原生 AWS 服務、基礎架構和作業模型帶入幾乎任何資料中心、主機代管空間或內部部署設施。

> **Important**
>
> 本機區域、Wavelength 區域或目 AWS Outposts 前不支援 AWS Fargate 工作負載上的 Amazon ECS。

如需本機區域、Wavelength 區域之間差異的詳細資訊 AWS Outposts ，請參閱常見問題集中如何考慮何時使用 AWS Wavelength、L AWS ocal Zones 或 AWS Outposts 需要低延遲或本機資料處理的應用程式。 AWS Wavelength

共用子網路

您可以使用 VPC 共用來與相同 AWS Organizations內的其他 AWS 帳戶共用子網路。

您可以將共用 VPC 用於 EC2 啟動類型，並考量下列事項：

- VPC 子網路的擁有者必須與參與者帳戶共用子網路，該帳戶才能將其用於 Amazon ECS 資源。
- 您無法針對容器執行個體使用 VPC 預設安全性群組，因為它屬於擁有者。此外，參與者無法使用其他參與者或擁有者所擁有的安全性群組來啟動執行個體。
- 在共用子網路中，參與者和擁有者會分別控制每個各別帳戶內的安全群組。子網路擁有者可以查看由參與者建立的安全群組，但無法執行任何動作。如果子網路擁有者想要移除或修改這些安全群組，建立安全群組的參與者必須採取動作。
- 共用 VPC 擁有者無法檢視、更新或刪除參與者在共用子網路中建立的叢集。此為 VPC 資源 (每個帳戶具有不同存取權) 以外。如需更多資訊，請參閱《Amazon VPC 使用者指南》中的擁有者和參與者的責任與許可。

您可以將共用 VPC 用於 Fargate 啟動類型，並考量下列事項：

- VPC 子網路的擁有者必須與參與者帳戶共用子網路，該帳戶才能將其用於 Amazon ECS 資源。
- 您無法使用 VPC 的預設安全性群組建立服務或執行工作，因為它屬於擁有者。此外，參與者無法使用其他參與者或擁有者所擁有的安全性群組來建立服務或執行工作。
- 在共用子網路中，參與者和擁有者會分別控制每個各別帳戶內的安全群組。子網路擁有者可以查看由參與者建立的安全群組，但無法執行任何動作。如果子網路擁有者想要移除或修改這些安全群組，建立安全群組的參與者必須採取動作。

- 共用 VPC 擁有者無法檢視、更新或刪除參與者在共用子網路中建立的叢集。此為 VPC 資源 (每個帳戶具有不同存取權) 以外。如需更多資訊，請參閱《Amazon VPC 使用者指南》中的擁有者和參與者的責任與許可。

如需有關 VPC 子網路共用的詳細資訊，請參閱《Amazon VPC 使用者指南》中的與其他帳戶共用 VPC。

本機區域

「本機區域」是您使用者附近地理位置的延伸。 AWS 區域 Local Zones 有自己的網際網路連線，並支援 AWS Direct Connect。在 Local Zone 中建立的資源可以為本機使用者提供低延遲的通訊服務。如需詳細資訊，請參閱 AWS Local Zones。

Local Zone 由一個區域代碼加上一個代表位置的識別符來表示 (例如 `us-west-2-lax-1a`)。

若要使用 Local Zone，您必須先選擇加入區域。選擇加入後，您必須在 Local Zone 中建立 Amazon VPC 與子網路。

您可以啟動 Amazon EC2 執行個體、Amazon FSx 檔案伺服器和 Application Load Balancer，以用於 Amazon ECS 叢集和任務。

如需詳細資訊，請參閱什麼是 L AWS ocal Zones？ 在「L AWS ocal Zones 使用者指南」中。

Wavelength 區域

您可使用 AWS Wavelength 建置可為行動裝置與最終使用者提供極低延遲的應用程式。Wavelength 將標準 AWS 運算和儲存服務部署到電信業者的 5G 網路邊緣。您可將 Amazon Virtual Private Cloud 延伸至一或多個 Wavelength Zone。然後，您可以使用 Amazon EC2 執行個體之類的 AWS 資源執行需要超低延遲和在該區域連線 AWS 服務 的應用程式。

Wavelength Zone 是部署 Wavelength 基礎設施之電信業者位置的隔離區域。Wavelength 區被綁定到一個 AWS 區域. Wavelength 區域是區域的邏輯延伸，並且由區域中的控制平面管理。

Wavelength Zone 由一個區域代碼加上一個代表 Wavelength Zone 的識別符來表示 (例如 `us-east-1-wl1-bos-wlz-1`)。

若要使用 Wavelength Zone，您必須先選擇加入區域。選擇加入後，您必須在 Wavelength Zone 中建立 Amazon VPC 與子網路。然後，您就能在區域中啟動 Amazon EC2 執行個體，以用於 Amazon ECS 叢集和任務。

如需詳細資訊，請參閱 AWS Wavelength 開發人員指南中的 AWS Wavelength入門。

並非所有 Wavelength 區都 AWS 區域可用。如需支援 Wavelength 區域的區域相關資訊，請參閱 AWS Wavelength 開發人員指南中的可用 Wavelength 區域。

Amazon 彈性容器服務 AWS Outposts

AWS Outposts 在內部部署設施中啟用原生 AWS 服務、基礎架構和作業模型。在 AWS Outposts 環境中 AWS ，您可以使用與 AWS 雲端.

Amazon ECS on 非 AWS Outposts 常適合需要在鄰近現場部署資料和應用程式執行的低延遲工作負載。

若要取得有關的更多資訊 AWS Outposts，請參閱AWS Outposts 使用者指南。

考量事項

以下是在 AWS Outposts上使用 Amazon ECS 的注意事項：

- Amazon 彈性容器登錄檔和 Network Load Balancer 在該 AWS 區域中執行，而不是在上執行 AWS Outposts。 AWS Identity and Access Management這將提高這些服務和容器之間的延遲。
- AWS Fargate 不適用於 AWS Outposts。

以下是用於的網路連線考量 AWS Outposts：

- 如果您 AWS Outposts 與其 AWS 區域之間的網路連線中斷，您的叢集將繼續執行。不過，在連線恢復之前，您將無法建立新叢集或對現有叢集採取新動作。在執行個體失敗的情況下，執行個體將不會被自動替換。記 CloudWatch 錄檔代理程式將無法更新記錄檔和事件資料。
- 我們建議您在您與其 AWS 區域之間提供可靠、高可用性 AWS Outposts 和低延遲的連線能力。

必要條件

以下是在 AWS Outposts上使用 Amazon ECS 的先決條件：

- 內部部署資料中心必須已安裝和設定 Outpost。
- Outpost 與其 AWS 區域之間必須有可靠的網路連線。

在上建立叢集 AWS Outposts

若要在上建立 Amazon ECS 叢集 AWS CLI，請指定要 AWS Outposts 與您 AWS Outposts的產生關聯的安全群組和子網路。

若要建立與您關聯的子網路 AWS Outposts。

```
aws ec2 create-subnet \
    --cidr-block 10.0.3.0/24 \
    --vpc-id vpc-xxxxxxxx \
    --outpost-arn arn:aws:outposts:us-west-2:123456789012:outpost/op-xxxxxxxxxxxxxxxxx \
    --availability-zone-id usw2-az1
```

以下範例會在 AWS Outposts上建立 Amazon ECS 叢集。

1. 建立具有上權限的角色和策略 AWS Outposts。

   `role-policy.json` 檔案是政策文件，其中包含資源的效果和動作。如需檔案格式的相關資訊，請參閱 IAM API 參考中的PutRole政策

   ```
   aws iam create-role --role-name ecsRole \
       --assume-role-policy-document file://ecs-policy.json
   aws iam put-role-policy --role-name ecsRole --policy-name ecsRolePolicy \
       --policy-document file://role-policy.json
   ```

2. 在 AWS Outposts上建立具有權限的 IAM 執行個體描述檔。

   ```
   aws iam create-instance-profile --instance-profile-name outpost
   aws iam add-role-to-instance-profile --instance-profile-name outpost \
       --role-name ecsRole
   ```

3. 建立 VPC。

   ```
   aws ec2 create-vpc --cidr-block 10.0.0.0/16
   ```

4. 為容器執行個體建立安全群組，指定 AWS Outposts適當的 CIDR 範圍。（此步驟是不同的 AWS Outposts。）

   ```
   aws ec2 create-security-group --group-name MyOutpostSG
   aws ec2 authorize-security-group-ingress --group-name MyOutpostSG --protocol tcp \
       --port 22 --cidr 10.0.3.0/24
   ```

```
aws ec2 authorize-security-group-ingress --group-name MyOutpostSG --protocol tcp \
    --port 80 --cidr 10.0.3.0/24
```

5. 建立叢集。
6. 定義 Amazon ECS 容器代理程式環境變數，將執行個體啟動至上一個步驟中建立的叢集，並定義任何您要新增的標籤，以幫助識別該叢集 (例如 Outpost 可指示叢集的用途是用於 Outpost)。

```
#! /bin/bash
cat << 'EOF' >> /etc/ecs/ecs.config
ECS_CLUSTER=MyCluster
ECS_IMAGE_PULL_BEHAVIOR=prefer-cached
ECS_CONTAINER_INSTANCE_TAGS={"environment": "Outpost"}
EOF
```

> **Note**
>
> 為了避免從區域 Amazon ECR 拉提取容器映像造成的延遲，請使用映像快取。若要進行此操作，請在每次執行工作時將 ECS_IMAGE_PULL_BEHAVIOR 設為 prefer-cached，以便將 Amazon ECS 代理程式設定為預設，在執行個體本身上使用快取的映像。

7. 建立容器執行個體，指定 AWS Outposts 應執行此執行個體的 VPC 和子網路，以及 AWS Outposts上可用的執行個體類型。（此步驟是不同的 AWS Outposts。）

   userdata.txt 檔案中包含使用者資料，執行個體可以使用這些使用者資料來執行常見的自動化組態任務，甚至在執行個體啟動之後執行指令碼。如需 API 呼叫檔案的相關資訊，請參閱 Amazon EC2 使用者指南中的啟動時在 Linux 執行個體上執行命令。

```
aws ec2 run-instances --count 1 --image-id ami-xxxxxxxx --instance-type c5.large \
    --key-name aws-outpost-key --subnet-id subnet-xxxxxxxxxxxxxxxxx \
    --iam-instance-profile Name outpost --security-group-id sg-xxxxxx \
    --associate-public-ip-address --user-data file://userdata.txt
```

> **Note**
>
> 將其他執行個體新增至叢集時，也會使用此指令。在叢集中部署的任何容器都會放置在該特定 AWS Outposts上。

8. 註冊您的任務定義。使用下列命令並將 ecs-task.json 替換為任務定義的名稱。

```
aws ecs register-task-definition --cli-input-json file://ecs-task.json
```

9. 執行工作或建立服務。

   Run the task

   ```
   aws ecs run-task --cluster mycluster --count 1 --task-definition outpost-app:1
   ```

   Create the service

   ```
   aws ecs create-service --cluster mycluster --service-name outpost-service \
       --task-definition outpost-app:1 --desired-count 1
   ```

優化 Amazon ECS 容量和可用性

應用程式可用性對於提供無錯誤的體驗，以及將應用程式延遲降至最低至關 可用性取決於具有可存取的資源，並具有足夠的容量來滿足需求。 AWS 提供數種機制來管理可用性。對於 Amazon ECS 上託管的應用程式，這些應用程式包括自動調度資源和可用區域 (AZ)。自動調度資源會根據您定義的指標來管理工作或執行個體的數量，而可用區域則可讓您將應用程式託管在隔離但靠近地理位置的位置。

與任務規模一樣，容量和可用性存在您必須考慮的某些權衡。理想情況下，容量將與需求完全一致。總是有足夠的容量來處理請求和處理任務，以滿足服務等級目標 (SLO)，包括低延遲和錯誤率。容量永遠不會太高，導致成本過高；也不會太低，從而導致高延遲和錯誤率。

自動調度資源是一個潛在的過程。首先，必須將即時指標傳遞給 CloudWatch。然後，需要將它們彙總以進行分析，這可能需要長達幾分鐘的時間，具體取決於指標的粒度。 CloudWatch 將測量結果與警示臨界值進行比較，以識別資源短缺或過多。為了防止不穩定，請將警報配置為要求在警報關閉之前超過設定的閾值幾分鐘。佈建新任務和終止不再需要的任務也需要時間。

由於系統上所描述的這些可能延遲，因此請務必透過過度佈建來維持一些預留空間。這樣做可以幫助滿足短期的需求爆發。這也有助於您的應用程序在不飽和的情況下服務其他請求。最好的做法是，您可以將擴展目標設定在使用率的 60-80% 之間。這有助於您的應用程式更有效地處理突發的額外需求，而額外的容量仍在佈建的過程中。

我們建議您過度佈建的另一個原因是，您可以快速回應可用區域故障。 AWS 建議從多個可用區域提供生產工作負載。這是因為，如果發生可用區域失敗，在剩餘可用區域中執行的工作仍然可以滿足需求。如果您的應用程式在兩個可用區域中執行，則需要將正常工作計數加倍。這樣您就可以在任何潛在的故

障期間立即提供容量。如果您的應用程式在三個可用區域中執行，建議您執行正常工作計數的 1.5 倍。也就是說，為普通服務所需的每兩個運行三個任務。

最大化縮放速度

自動調度資源是一種反應過程，需要時間才能生效。但是，有一些方法可以幫助最大程度地減少擴展所需的時間。

最小化圖像大小。較大的影像需要較長的時間才能從映像儲存庫下載並解壓縮。因此，保持較小的影像大小可減少容器啟動所需的時間量。要減小圖像大小，您可以遵循以下具體建議：

- 如果您可以構建靜態二進製文件或使用 Golang，請構建映像FROM暫存並在生成的映像中僅包含二進製應用程序。
- 使用來自上游發行版供應商（例如 Amazon Linux 或 Ubuntu）的最小化基本映像。
- 不要在最終映像中包含任何構建工件。使用多階段構建可以幫助解決此問題。
- 盡可能緊湊的RUN階段。每個RUN階段都會創建一個新的圖像圖層，從而導致額外的往返下載圖層。結合多個指令的單一RUN階段，所&&擁有的圖層少於具有多個RUN階段的圖層。
- 如果您想要在最終映像中包含 ML 推論資料之類的資料，請僅包含啟動和開始提供流量所需的資料。如果您在不影響服務的情況下從 Amazon S3 或其他儲存視需求擷取資料，請改為將資料存放在這些位置。

保持您的圖像靠近。網路延遲時間越高，下載映像檔所需的時間越長。將映像託管在工作負載所在的相同 AWS 區域中的儲存庫中。Amazon ECR 是一個高效能映像儲存庫，可在 Amazon ECS 提供的所有區域中使用。避免穿越網際網路或 VPN 連結以下載容器映像檔。在同一區域託管圖像可提高整體可靠性。它降低了不同區域中網絡連接問題和可用性問題的風險。或者，您也可以實作 Amazon ECR 跨區域複寫，以協助進行此操作。

降低負載平衡器健康狀態檢查閾值 負載平衡器會在傳送流量至應用程式之前執行健康狀態檢查。目標群組的預設健全狀況檢查組態可能需要 90 秒或更長時間。在此期間，負載平衡器會檢查健康狀態並接收要求。降低健全狀況檢查間隔和臨界值計數可讓您的應用程式更快接受流量，並減少其他工作的負載。

考慮冷啟動效能。一些應用程序使用運行時，如 Java 執行即時（JIT）編譯。編譯過程至少在啟動時可以顯示應用程序性能。解決方法是使用不會造成冷啟動效能損失的語言來重新撰寫工作負載的延遲關鍵部分。

使用步驟調整，而不是目標追蹤擴展政策。您有多個適用於 Amazon ECS 任務的 Application Auto Scaling 選項。目標追蹤是最容易使用的模式。這樣一來，您僅需要為指標設定目標值，例如 CPU 平

均使用率。然後，自動縮放器會自動管理獲得該值所需的任務數量。使用步驟擴展，您可以更快對需求變化做出反應，因為您定義了擴展指標的特定閾值，以及超越閾值時要新增或刪除的任務數量。而且，更重要的是，您可以透過最大限度減少閾值警報違反的時間來對需求變化做出非常快速的反應。如需詳細資訊，請參閱《Amazon Elastic Container Service 開發人員指南》中的服務自動擴展。

如果您使用 Amazon EC2 執行個體提供叢集容量，請考慮下列建議：

使用更大的 Amazon EC2 實例和更快的 Amazon EBS 卷。您可以使用更大的 Amazon EC2 執行個體和更快的 Amazon EBS 磁碟區來改善映像下載和準備速度。在指定的 Amazon EC2 執行個體系列中，網路和 Amazon EBS 最大輸送量會隨著執行個體大小的增加而增加 (例如，從`m5.xlarge`到`m5.2xlarge`)。此外，您也可以自訂 Amazon EBS 磁碟區，以增加其輸送量和 IOPS。例如，如果您使用`gp2`磁碟區，請使用可提供更多基準輸送量的較大磁碟區。如果您使用`gp3`磁碟區，請在建立磁碟區時指定輸送量和 IOPS。

對在 Amazon EC2 執行個體上執行的任務使用橋接網路模式。Amazon EC2 上使用`bridge`網路模式的任務開始速度比使用`awsvpc`網路模式的任務快。使用`awsvpc`網路模式時，Amazon ECS 會在啟動任務之前將 elastic network interface (ENI) 附加至執行個體。這會引入額外的延遲。儘管如此，使用橋接網絡有幾種權衡。這些工作不會取得自己的安全性群組，而且負載平衡有一些含意。如需詳細資訊，請參閱《E lastic Load Balancing 使用者指南》中的負載平衡器目標群組。

處理需求衝擊

某些應用程序遇到突然的大衝擊需求。發生這種情況的原因有多種：新聞事件，大型銷售，媒體活動或其他一些事件，會傳播病毒並導致流量在很短的時間內快速顯著增加。如果未計劃，這可能會導致需求快速超出可用資源。

處理需求衝擊的最佳方法是預測它們並進行相應的計劃。由於自動調度資源可能需要一些時間，因此建議您在需求衝擊開始之前向外擴充應用程式。為了獲得最佳效果，我們建議制定一個商業計劃，該計劃涉及使用共享日曆的團隊之間的緊密協作。籌劃活動的團隊應提前與負責申請的團隊緊密合作。這為該團隊提供了足夠的時間來制定明確的計劃。他們可以排程容量以在活動開始前向外擴展，並在活動結束後擴展。如需詳細資訊，請參閱《Application Auto Scaling 使用者指南》中的排程擴展。

如果您有企業 Support 方案，請務必與您的技術客戶經理 (TAM) 合作。您的 TAM 可以驗證您的服務配額，並確保在活動開始之前提出任何必要的配額。如此一來，您就不會意外碰到任何服務配額。他們還可以通過預熱負載平衡器等服務來幫助您，以確保您的活動順利進行。

處理不定期的需求衝擊是一個更困難的問題。不定期的衝擊，如果振幅足夠大，可能會迅速導致需求超出容量。它也可能超過自動調度資源的反應能力。為非計劃衝擊做好準備的最佳方法是過度佈建資源。您必須擁有足夠的資源，以便隨時處理最大的預期流量需求。

在預期意外需求衝擊的情況下維持最大容量可能非常昂貴。為了減輕成本影響，請找到一個領先的指標指標或預測大量需求衝擊的事件即將到來。如果量度或事件可靠地提供明顯的預先通知，請在事件發生或量度超過您設定的特定臨界值時立即開始向外延展程序。

如果您的應用程式容易遭受突然的非排程需求衝擊，請考慮在應用程式中加入高效能模式，以犧牲非關鍵功能，但為客戶保留重要功能。例如，假設您的應用程式可以從產生昂貴的自訂回應切換到提供靜態回應頁面。在這個案例中，您可以大幅增加輸送量，完全不需要擴展應用程式。

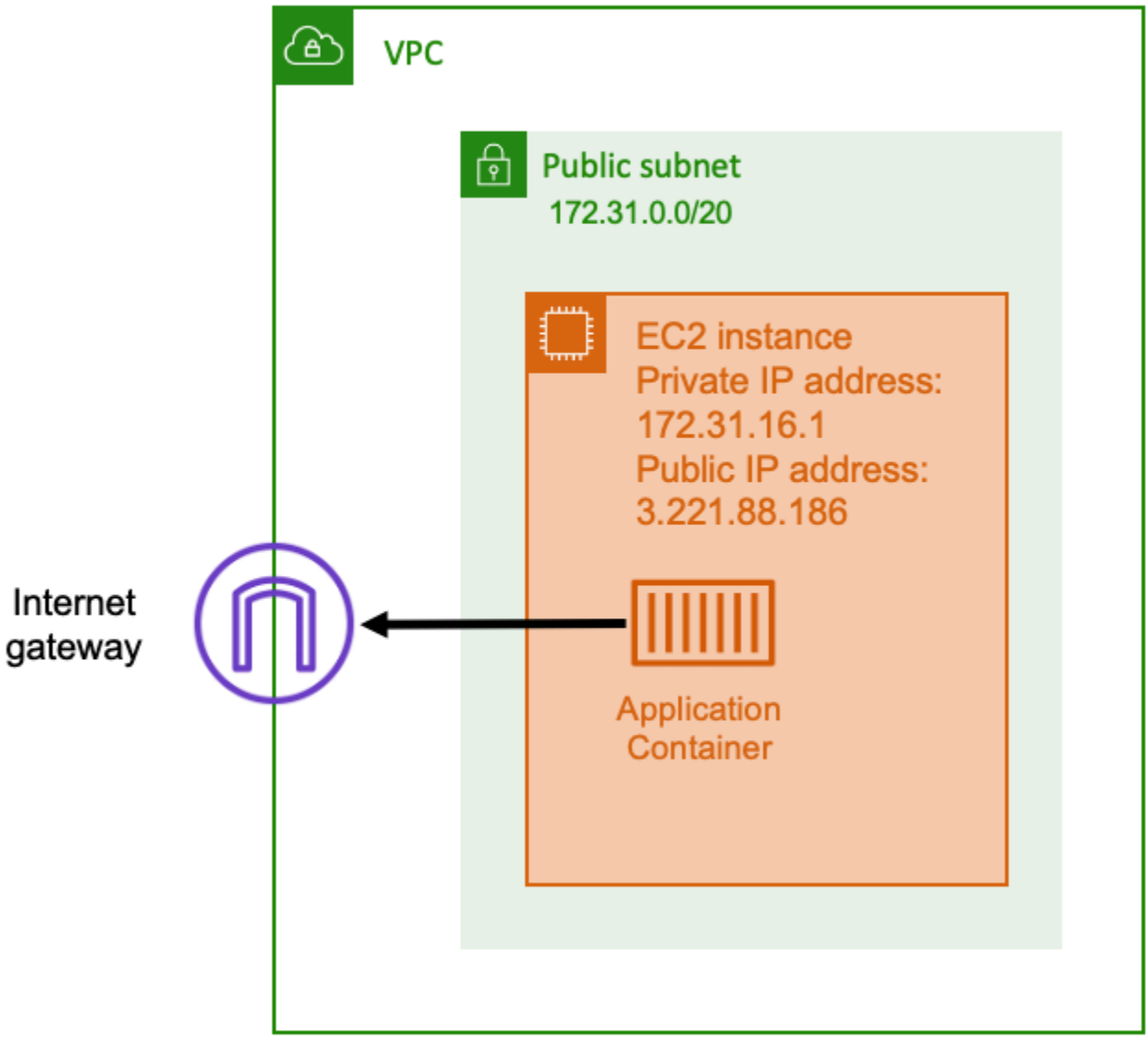
最後，您可以考慮打破整體服務，以更好地應對需求衝擊。如果您的應用程式是執行成本昂貴且擴充速度緩慢的整合式服務，您可能可以擷取或重寫效能關鍵部分，並以個別服務的形式執行它們。然後，這些新服務可以獨立於不太重要的組件進行擴展。與應用程式的其他部分分開擴充關鍵效能功能的彈性，既能縮短增加容量所需的時間，也能協助節省成本。

將 Amazon ECS 應用程式 Connect 到網際網路

大多數容器化應用程式都至少有一些需要對外存取網際網路的元件。例如，行動應用程式的後端需要對外存取推送通知。

Amazon 虛擬私有雲有兩種主要方法可促進 VPC 和網際網路之間的通訊。

公共子網和網際網路閘道



當您使用具有網際網路閘道路由的公用子網路時，您的容器化應用程式可以在公用子網路上 VPC 內的主機上執行。執行您容器的主機會指派公用 IP 位址。此公用 IP 位址可從網際網路路由。如需詳細資訊，請參閱《Amazon VPC 使用者指南》中的網際網路閘道。

這種網絡架構有助於運行應用程序的主機和 Internet 上的其他主機之間的直接通信。通信是雙向的。這表示您不僅可以建立與網際網路上任何其他主機的輸出連線，而且網際網路上的其他主機也可能會嘗試連線到您的主機。因此，您應該密切注意您的安全組和防火牆規則。這樣可以確保 Internet 上的其他主機無法打開您不想打開的任何連接。

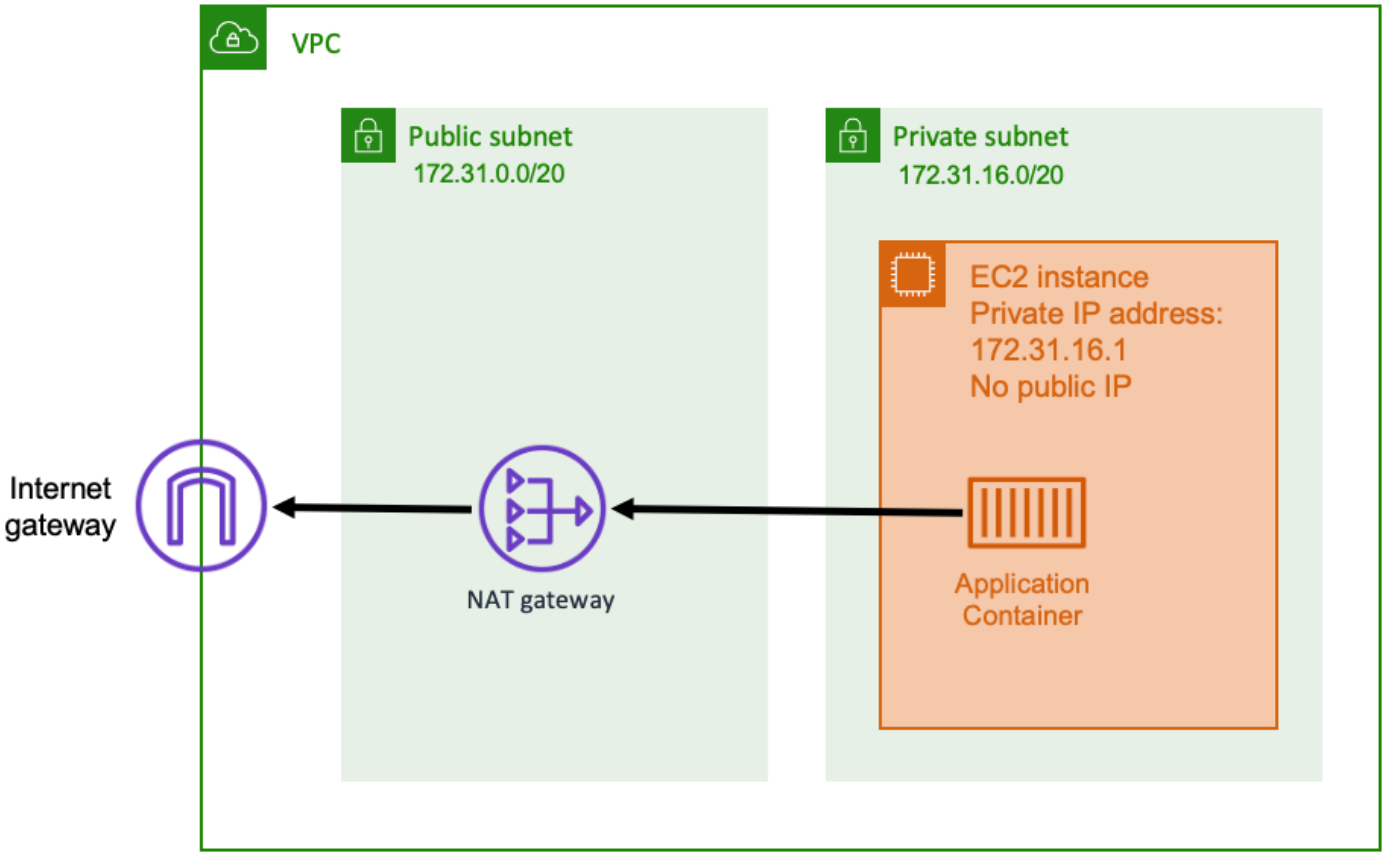
例如，如果您的應用程式在 Amazon EC2 上執行，請確定未開啟用於 SSH 存取的連接埠 22。否則，您的執行個體可能會持續接收來自網際網路上惡意機器人的 SSH 連線嘗試。這些機器人通過公共 IP 地址進行拖網。找到開放的 SSH 連接埠後，他們會嘗試使用暴力密碼來嘗試存取您的執行個體。因此，許多組織會限制公用子網路的使用，而且偏好在私有子網路中擁有大部分 (如果不是全部) 資源。

使用公用子網路進行網路適用於需要大量頻寬或最小延遲的公用應用程式。適用的使用案例包括視訊串流和遊戲服務。

當您在 Amazon Amazon EC2 上使用 Amazon ECS 以及在上使用時，都支援此聯網方法。 AWS Fargate

- Amazon EC2 — 您可以在公有子網路上啟動 EC2 執行個體。Amazon ECS 使用這些 EC2 執行個體做為叢集容量，執行個體上執行的任何容器都可以使用主機的基礎公有 IP 地址進行輸出聯網。這適用於`host`和`bridge`網路模式。但是，`awsvpc`網絡模式不提供具有公共 IP 地址的任務 ENI。因此，他們不能直接使用互聯網網關。
- Fargate — 建立 Amazon ECS 服務時，請為服務的聯網組態指定公用子網路，並使用「指派公用 IP 位址」選項。每個 Fargate 任務都在公共子網中聯網，並擁有自己的公共 IP 地址，用於與互聯網直接通信。

私有子網路和 NAT 閘道



使用私有子網路和 NAT 閘道時，您可以在私有子網路中的主機上執行容器化應用程式。因此，此主機具有可在您的 VPC 中路由的私有 IP 位址，但無法從網際網路路由。這表示 VPC 內的其他主機可以使用其私有 IP 位址連線到主機，但網際網路上的其他主機無法對主機進行任何輸入通訊。

使用私有子網路時，您可以使用網路位址轉譯 (NAT) 閘道，允許私有子網路內的主機連線到網際網路。網際網路上的主機會收到輸入連線，該連線似乎來自位於公有子網路內之 NAT 閘道的公用 IP 位址。NAT 閘道負責做為網際網路與私有 VPC 之間的橋樑。出於安全原因，通常首選此配置，因為這意味著您的 VPC 受到 Internet 上的攻擊者直接訪問的保護。如需詳細資訊，請參閱 《Amazon VPC 使用者指南》中的 NAT 閘道。

這種私有網路方法適用於您想要保護容器免於直接外部存取的案例。適用案例包括付款處理系統或儲存使用者資料和密碼的容器。您將需要為您帳戶中建立及使用的 NAT 閘道支付費用。NAT 閘道的每小時使用量和資料處理速率也適用。基於備援目的，您應該在每個可用區域中都有一個 NAT 閘道。如此一來，單一可用區域的可用性損失就不會影響您的輸出連線。因此，如果您的工作負載很小，使用私有子網路和 NAT 閘道可能會更符合成本效益。

在 Amazon Amazon EC2 上使用 Amazon ECS 和在上使用時，都支援此聯網方法。 AWS Fargate

- Amazon EC2 — 您可以在私有子網路上啟動 EC2 執行個體。在這些 EC2 主機上執行的容器會使用基礎主機聯網，而輸出請求則透過 NAT 閘道。
- Fargate — 建立 Amazon ECS 服務時，請為服務的聯網組態指定私有子網路，並且不要使用「指派公用 IP 位址」選項。每個 Fargate 任務都託管在一個私有子網中。其輸出流量會透過您與該私有子網路相關聯的任何 NAT 閘道路由傳送。

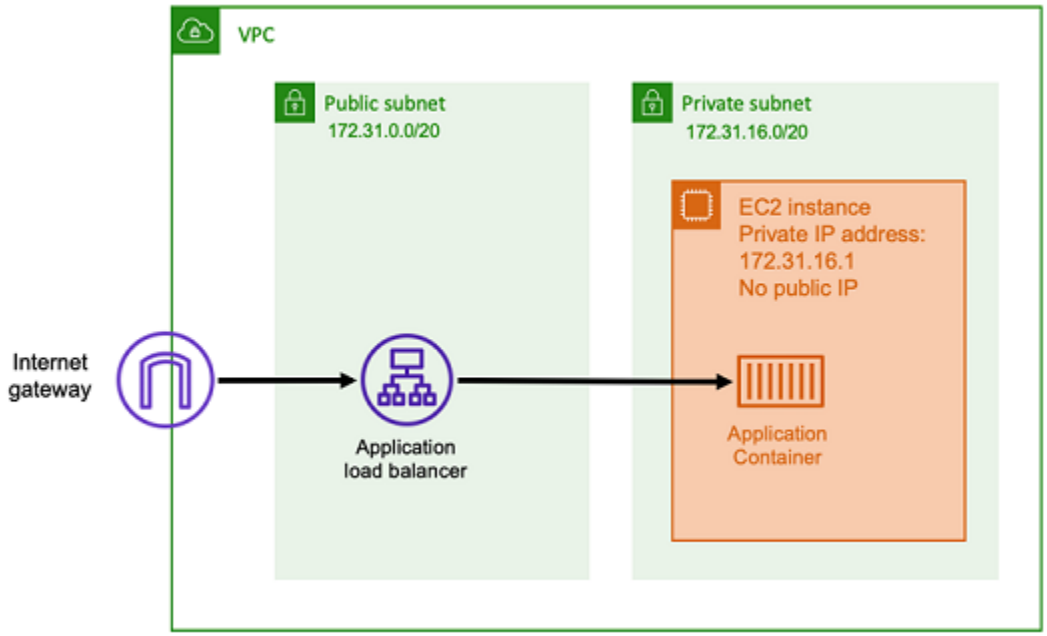
從網際網路接收 Amazon ECS 輸入連線的最佳實務

如果您執行公共服務，則必須接受來自網際網路的輸入流量。例如，您的公開網站必須接受來自瀏覽器的輸入 HTTP 要求。在這種情況下，網際網路上的其他主機也必須啟動到應用程式主機的輸入連線。

解決此問題的一種方法是在具有公用 IP 位址的公用子網路中的主機上啟動容器。但是，我們不建議大規模應用程序使用此功能。對於這些，更好的方法是有一個位於互聯網和應用程序之間的可擴展輸入層。對於這種方法，您可以使用本節中列出的任何 AWS 服務作為輸入。

Application Load Balancer

應用程式負載平衡器會在應用程式層運作。這是開放系統互連（OSI）模型的第七層。這使得應用程序負載平衡器適用於公共 HTTP 服務。如果您有網站或 HTTP REST API，則 Application Load Balancer 就是適用於此工作負載的負載平衡器。如需詳細資訊，請參閱什麼是應用程式負載平衡器？ 在《應用程式負載平衡器使用者指南》中。



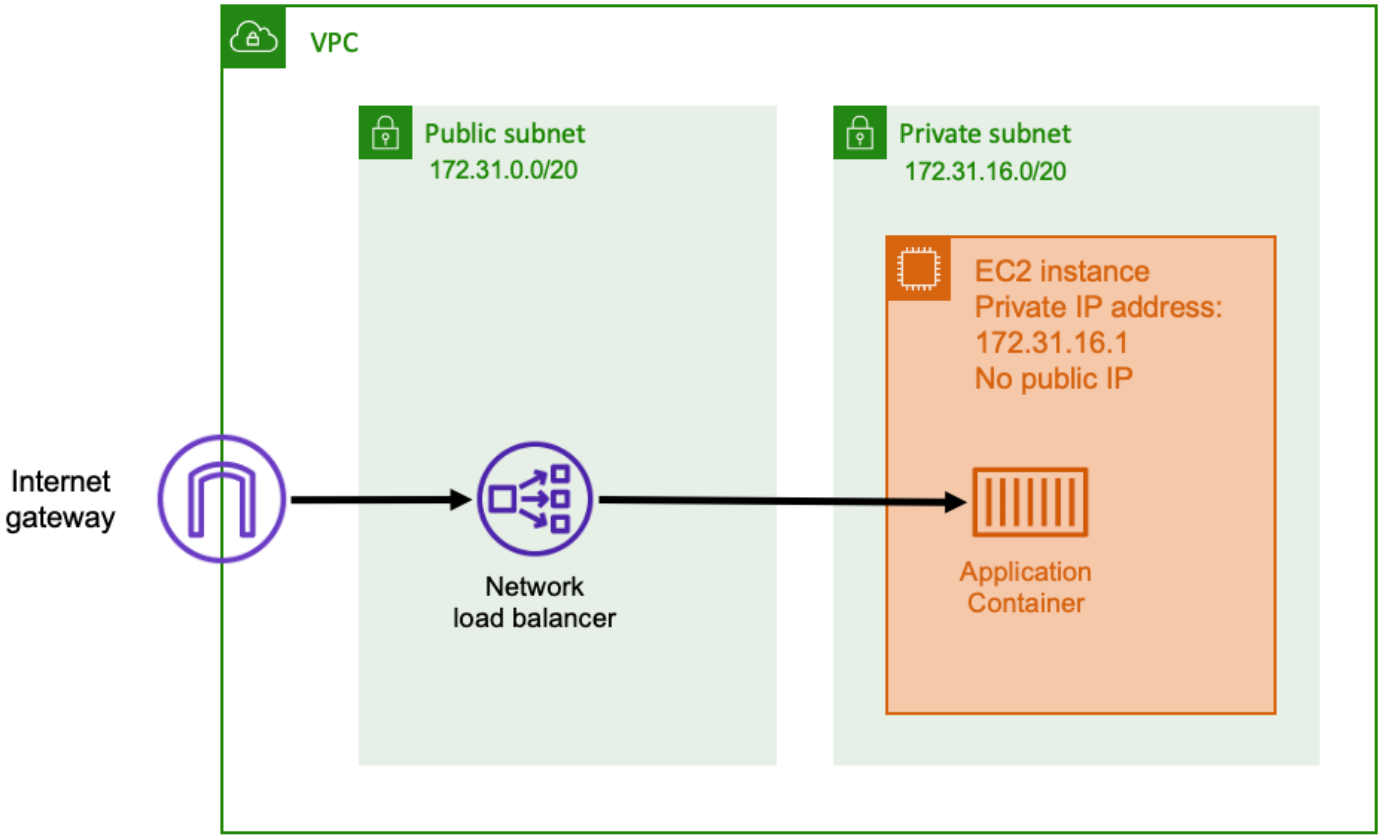
透過此架構，您可以在公用子網路中建立 Application Load Balancer，以便其具有公用 IP 位址，並可從網際網路接收輸入連線。當應用程式負載平衡器收到輸入連線，或更具體地說是 HTTP 要求時，它會使用其私有 IP 位址開啟與應用程式的連線。然後，它通過內部連接轉發請求。

應用程式負載平衡器具有下列優點。

- SSL/TLS 終止 — Application Load Balancer 可以維持與用戶端通訊的安全 HTTPS 通訊和憑證。它可以選擇性地在負載平衡器層級終止 SSL 連接，這樣您就不必在自己的應用程序中處理證書。
- 進階路由 — 應用程式負載平衡器可以有多個 DNS 主機名稱。它還具有進階路由功能，可根據主機名稱或要求路徑等指標，將傳入的 HTTP 要求傳送至不同的目的地。這表示您可以使用單一 Application Load Balancer 做為許多不同內部服務的輸入，甚至可以使用 REST API 不同路徑上的微服務。
- gRPC 支援和網路通訊端 — Application Load Balancer 可以處理的不僅僅是 HTTP。它還可以負載平衡 gRPC 和基於網絡通訊端的服務，並支持 HTTP/2。
- 安全性 — 應用程式負載平衡器可協助保護您的應用程式免受惡意流 它包含 HTTP de 同步緩和措施等功能，並與 AWS Web 應用程式防火牆整合 (AWS WAF)。 AWS WAF 可進一步過濾掉可能包含攻擊模式的惡意流量，例如 SQL 插入或跨網站指令碼。

Network Load Balancer

Network Load Balancer 是在開放系統互連 (OSI) 模型的第四層運作。它適用於非 HTTP 通訊協定或需要 end-to-end 加密的案例，但不具有 Application Load Balancer 相同的 HTTP 特定功能。因此，Network Load Balancer 最適合不使用 HTTP 的應用程式。如需詳細資訊，請參閱什麼是 Network Load Balancer？ 在《網路負載平衡器使用者指南》中。



將 Network Load Balancer 用作輸入時，其功能類似於 Application Load Balancer。這是因為它是在公共子網中創建的，並且具有可以在 Internet 上訪問的公共 IP 地址。然後，Network Load Balancer 會開啟與執行容器之主機私有 IP 位址的連線，並將封包從公用端傳送到私有端。

Network Load Balancer 功能

由於 Network Load Balancer 在較低層級的網路堆疊上運作，因此它沒有與 Application Load Balancer 相同的功能集。但是，它確實具有以下重要功能。

- E nd-to-end 加密 — 由於 Network Load Balancer 在 OSI 模型的第四層運作，因此不會讀取封包的內容。這使得它適用於需要 end-to-end加密的負載平衡通信。
- TLS 加密 — 除了加 end-to-end 密之外，Network Load Balancer 也可以終止 TLS 連線。如此一來，您的後端應用程式就不必實作自己的 TLS。
- UDP 支援 — 由於 Network Load Balancer 是在 OSI 模型的第四層運作，因此適用於非 HTTP 工作負載和 TCP 以外的通訊協定。

關閉連線

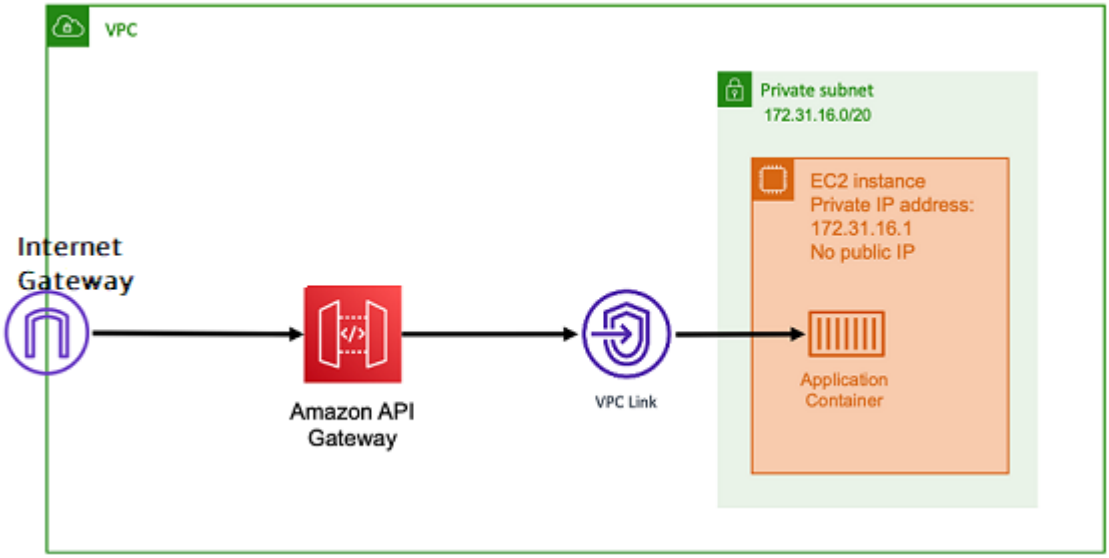
由於 Network Load Balancer 不會在 OSI 模型的較高層觀察應用程式通訊協定，因此無法傳送關閉訊息給這些通訊協定中的用戶端。與應用程式負載平衡器不同，這些連線必須由應用程式關閉，或者您可以設定 Network Load Balancer，以在任務停止或取代時關閉第四層連線。請參閱 Network Load Balancer 說明文件中的 Network Load Balancer 目標群組的連線終止設定。

如果用戶端未處理這些錯誤訊息，則讓 Network Load Balancer 關閉第四層的連線可能會導致用戶端顯示不需要的錯誤訊息。有關建議用戶端組態的詳細資訊，請參閱這裡的建置程式庫。

關閉連線的方法會因應用程式而異，但其中一種方法是確保 Network Load Balancer 目標取消註冊延遲時間長於用戶端連線逾時。用戶端會先逾時，然後透過 Network Load Balancer 正常重新連線至下一個工作，而舊工作會慢慢排空其所有用戶端。如需有關 Network Load Balancer 目標取消註冊延遲的詳細資訊，請參閱 Network Load Balancer 說明文件。

Amazon API Gateway

Amazon API Gateway 適用於在請求磁碟區突然爆發的 HTTP 應用程式或請求量低。如需詳細資訊，請參閱什麼是 Amazon API Gateway？ 在 API Gateway 開發人員指南中。



應用程式負載平衡器和 Network Load Balancer 的定價模式均包含每小時價格，讓負載平衡器隨時可用於接受傳入連線。相反地，API Gateway 會分別針對每個要求收取費用。這樣做的效果是，如果沒有請求進入，則不收取任何費用。在高流量負載下，應用程式負載平衡器或 Network Load Balancer 可以以比 API Gateway 更便宜的每個請求價格處理更大量的請求。不過，如果您的整體要求數量較少或流量較低，則使用 API Gateway 的累計價格應該比支付每小時費用來維護未充分利用的負載平衡器費用更具成本效益。API Gateway 也可以快取 API 回應，這可能會導致後端要求率較低。

API Gateway 功能使用 VPC 連結，該連結允許 AWS 受管理服務使用其私有 IP 位址連線至 VPC 私有子網路內的主機。它可以透過查看 Amazon ECS 服務探索管理的 AWS Cloud Map 服務探索記錄來偵測這些私有 IP 地址。

API Gateway 支援下列功能。

- API Gateway 作業類似於負載平衡器，但具有 API 管理專屬的其他功能
- API Gateway 提供有關用戶端授權、用量層和請求/回應修改的其他功能。如需詳細資訊，請參閱 Amazon API Gateway 功能。
- API Gateway 可支援邊緣、地區和私有 API 閘道端點。Edge 端點可透過受管理 CloudFront 發佈取得。區域和私有端點都是區域的本地端點。
- SSL/TLS 終止
- 將不同的 HTTP 路徑路由到不同的後端微服務

除了上述功能之外，API Gateway 還支援使用自訂 Lambda 授權器，您可以使用這些授權程式來保護您的 API 不受未經授權的使用。如需詳細資訊，請參閱欄位備註：搭配 Amazon ECS 和 Amazon API 閘道的無伺服器容器型 API。

透過帳戶設定存取 Amazon ECS 功能

您可以前往 Amazon ECS 帳戶設定以選擇加入或退出特定功能。針對每個 AWS 區域，您可以在帳戶層級或針對特定的使用者或角色選擇加入或退出每個帳戶設定。

如果以下任何一項與您相關，則您可能會想要選擇加入或退出特定功能：

- 使用者或角色可以為其個別帳戶選擇加入或退出特定帳戶設定。
- 使用者或角色可以為帳戶上的所有使用者設定預設選擇加入或退出設定。
- root 使用者或具有管理員權限的使用者可以選擇加入或退出帳戶上的任何特定角色或使用者。如果根使用者的帳戶設定發生變更，對於未選取個別帳戶設定的所有使用者和角色，將會設為預設值。

> **Note**
>
> 聯合身分使用者會採用根使用者的帳戶設定，而且無法另外為他們設定明確的帳戶設定。

下列帳戶設定可供使用。您必須分別選擇加入和選擇退出每個帳戶設定。

Amazon Resource Name (ARN) 和 ID

資源名稱：`serviceLongArnFormat`、`taskLongArnFormat` 和 `containerInstanceLongArnFormat`

Amazon ECS 正在為 Amazon Resource Name (ARN) 和 Amazon ECS 服務、任務和容器執行個體的資源 ID 引入新格式。每個資源類型的選擇加入狀態會決定資源使用的 Amazon Resource Name (ARN) 格式。您必須選擇加入新的 ARN 格式，以便使用適合該資源類型的功能，例如資源標記。如需詳細資訊，請參閱 Amazon Resource Name (ARN) 和 ID。

預設值為 `enabled`。

只有在加入後啟動的資源才會收到新的 ARN 和資源 ID 格式。所有現有的資源不會受到影響。若要讓 Amazon ECS 服務和任務轉換至新的 ARN 和資源 ID 格式，您必須重新建立服務或任務。若要將容器執行個體轉換至新 ARN 和資源 ID 格式，必須排空容器執行個體，且必須啟動新的容器執行個體，並使其向叢集註冊。

> **Note**
>
> 只有在 Amazon ECS 服務建立時間為或晚於 2018 年 11 月 16 日，而且建立該服務的使用者已決定為任務加入新格式時，該服務所啟動的任務才會收到新的 ARN 和資源 ID 格式。

AWSVPC 集中管道

資源名稱：`awsvpcTrunking`

Amazon ECS 支援使用受支援的 Amazon EC2 執行個體類型，以更高的彈性網路介面 (ENI) 密度來啟動容器執行個體。當您使用這些執行個體類型並選擇使用 `awsvpcTrunking` 帳戶設定時，新啟動的容器執行個體會提供額外的 ENI。您可以使用此組態，使用 `awsvpc` 網路模式在每個容器執行個體中安排更多任務。使用此功能，已啟用 `awsvpcTrunking` 的 `c5.large` 執行個體的 ENI 配額增加到 10。容器執行個體會有主要網路介面，而 Amazon ECS 會建立「幹線」網路介面，將其連接到容器執行個體。主要網路介面和幹線網路介面不會計入 ENI 配額。因此，您可以使用此組態在容器執行個體上啟動十項任務，而不是目前的兩項任務。如需詳細資訊，請參閱 增加 Amazon ECS Linux 容器實例網絡界面。

預設值為 `disabled`。

只有在加入後啟動的資源才會收到增加的 ENI 限制。所有現有的資源都不會受到影響。若要將容器執行個體轉換至增加的 ENI 配額，必須排空容器執行個體，且必須向叢集註冊新的容器執行個體。

CloudWatch 容器洞察

資源名稱：`containerInsights`

CloudWatch Container Insights 會從您的容器化應用程式和微服務收集、彙總和摘要指標和記錄。指標包含 CPU、記憶體、磁碟和網路這類資源的使用率。Container Insights 還提供診斷資訊，例如容器重新啟動故障，協助您快速隔離和解決這些故障。您也可以針對容器深入解析收集的指標設定 CloudWatch 警示。如需詳細資訊，請參閱 使用容器洞察來監控 Amazon ECS 容器。

當您選擇加入 `containerInsights` 帳戶設定時，所有新叢集都會預設啟用 Container Insights。您可以在建立特定叢集時停用此設定。您也可以使用 UpdateClusterSettings API 變更此設定。

對於內含任務的叢集或使用 EC2 啟動類型的服務，您的容器執行個體必須執行版本 1.29.0 或更新版本的 Amazon ECS 代理程式，才能使用 Container Insights。如需詳細資訊，請參閱 Amazon ECS Linux 容器執行個體管理。

預設值為 `disabled`。

雙堆疊 VPC IPv6

資源名稱：`dualStackIPv6`

除了主要私有 IPv4 地址之外，Amazon ECS 還支援提供具有 IPv6 地址的任務。

若要讓任務接收 IPv6 地址，任務必須使用 `awsvpc` 網路模式，必須在針對雙堆疊模式設定的 VPC 中啟動，且必須啟用 `dualStackIPv6` 帳戶設定。如需其他需求的詳細資訊，在雙堆疊模式下使用 VPC請參閱 EC2 啟動類型和 在雙堆疊模式下使用 VPC Fargate 啟動類型。

> **Important**
>
> 只能使用 Amazon ECS API 或 AWS CLI變更 `dualStackIPv6` 帳戶設定。如需詳細資訊，請參閱 修改 Amazon ECS 帳戶設置。

如果您有一個正在執行的任務，它在 2020 年 10 月 1 日和 2020 年 11 月 2 日之間的已啟用 IPv6 的子網路中使用 `awsvpc` 網路模式，則正執行該任務的區域中的預設 `dualStackIPv6` 帳戶設定為 `disabled`。如果不符合該條件，則該區域中的預設 `dualStackIPv6` 設定為 `enabled`。

預設值為 `disabled`。

Fargate FIPS-140 法規遵循

資源名稱：`fargateFIPSMode`

Fargate 支援聯邦資訊處理標準 (FIPS-140)，該標準具體說明了保護敏感資訊的加密模組的安全要求。這是目前的美國和加拿大政府標準，且適用於需要符合聯邦資訊安全管理法案 (FISMA) 或聯邦風險和授權管理計劃 (FedRAMP) 的系統。

預設值為 `disabled`。

您必須開啟 FIPS-140 法規遵循。如需詳細資訊，請參閱 the section called “AWS Fargate 符合 FIPS-140 規範”。

> **Important**
>
> 只能使用 Amazon ECS API 或 AWS CLI變更 `fargateFIPSMode` 帳戶設定。如需詳細資訊，請參閱 修改 Amazon ECS 帳戶設置。

標籤資源授權

資源名稱：`tagResourceAuthorization`

有些 Amazon ECS API 動作可讓您在建立資源時指定標籤。

Amazon ECS 正在為資源建立引入標記授權。使用者必須具有建立資源之動作的權限，例如`ecsCreateCluster`。如果在資源建立動作中指定了標籤，請對動作 AWS 執行其他授權，以確認使用者或角色是否具有建立標籤的權限。`ecs:TagResource`因此，您必須授予使用 `ecs:TagResource` 動作的明確許可。如需詳細資訊，請參閱 the section called “在建立期間標記資源”。

Fargate 任務淘汰等待期

資源名稱：`fargateTaskRetirementWaitPeriod`

AWS 負責修補和維護 AWS Fargate 的基礎設施。當 AWS 確定在 Fargate 上託管的 Amazon ECS 任務需要安全或基礎設施更新時，需要停止任務並啟動新任務以取代它們。您可以設定任務淘汰以進行修補之前的等待期。您可以選擇立即淘汰任務，等待 7 個日曆天，或等待 14 個日曆天。

此設定位於帳戶層級。

運行時監視激活

資源名稱：`guardDutyActivate`

此`guardDutyActivate`參數在 Amazon ECS 中是唯讀的，並指出 Amazon ECS 帳戶中的安全管理員是否開啟或關閉執行階段監控。 GuardDuty 代表您控制此帳戶設定。如需詳細資訊，請參閱使用執行時期監控保護 Amazon ECS 工作負載。

主題

- Amazon Resource Name (ARN) 和 ID
- ARN 和資源 ID 格式時間表
- AWS Fargate 符合聯邦資訊處理標準 (FIPS-140) 規範
- 標記授權
- 標記授權時間表
- AWS Fargate 任務退休等待時間
- 運行時監控（Amazon GuardDuty 集成）
- 使用主控台檢視 Amazon ECS 帳戶設定
- 修改 Amazon ECS 帳戶設置
- 還原為預設的 Amazon ECS 帳戶設定
- 使用管理 Amazon ECS 帳戶設定 AWS CLI

Amazon Resource Name (ARN) 和 ID

建立 Amazon ECS 資源時，會將唯一的 Amazon Resource Name (ARN) 和資源識別符 (ID) 指派給每個資源。若您透過命令列工具或 Amazon ECS API 來使用 Amazon ECS，則特定命令會需要資源 ARN 或 ID。例如，如果您使用停止工作 AWS CLI 命令停止工作，則必須在命令中指定工作 ARN 或 ID。

您可以在每個區域的基礎上，選擇加入和退出新的 Amazon Resource Name (ARN) 和資源 ID 格式。目前，預設會選擇加入建立的所有新帳戶。

您可以隨時選擇加入或退出新的 Amazon Resource Name (ARN) 和資源 ID 格式。選擇加入之後，您建立的任何新資源都會使用新格式。

> **Note**
>
> 資源 ID 在建立之後即不會變更。因此，選擇加入或退出新格式並不會影響現有的資源 ID。

下列章節說明 ARN 和資源 ID 格式如何改變。有關轉換至新格式的詳細資訊，請參閱 Amazon Elastic Container Service 常見問答集。

Amazon Resource Name (ARN) 格式

部分資源具有使用者易記名稱 (例如，名為 `production` 的服務)。在其他情況下，您必須使用 Amazon Resource Name (ARN) 格式來指定資源。Amazon ECS 任務、服務和容器執行個體的新 ARN 格式包含叢集名稱。如需選擇加入新 ARN 格式的資訊，請參閱 修改 Amazon ECS 帳戶設置。

下表顯示每個資源類型的當前格式和新格式。

資源類型	ARN
容器執行個體	當前：arn:aws:ecs: *region*:*aws_account_id* :container-instance/ *container-instance-id* 新：arn:aws:ecs: *region*:*aws_account_id* :container-instance/ *cluster-name* /*container-instance-id*
Amazon ECS 服務	當前：arn:aws:ecs: *region*:*aws_account_id* :service/ *service-name* 新：arn:aws:ecs: *region*:*aws_account_id* :service/ *cluster-name* /*service-name*
Amazon ECS 任務	當前：arn:aws:ecs: *region*:*aws_account_id* :task/*task-id* 新：arn:aws:ecs: *region*:*aws_account_id* :task/*cluster-name* /*task-id*

資源 ID 長度

資源 ID 的形式採用字母和數字的唯一組合。新資源 ID 格式包括 Amazon ECS 任務和容器執行個體的較短 ID。當前資源 ID 格式長度為 36 個字元。新 ID 採用 32 個字元的格式，且不包含任何連字號。如需選擇加入新資源 ID 格式的資訊，請參閱 修改 Amazon ECS 帳戶設置。

ARN 和資源 ID 格式時間表

Amazon ECS 資源的新 Amazon Resource Name (ARN) 和資源 ID 格式的選擇加入和退出期間截止時間為 2021 年 4 月 1 日。根據預設，所有帳戶都會選擇加入新格式。所有建立的新資源都會收到新格式，且您無法再選擇退出。

AWS Fargate 符合聯邦資訊處理標準 (FIPS-140) 規範

您必須在 Fargate 上開啟聯邦資訊處理標準 (FIPS-140) 法規遵循。如需詳細資訊，請參閱 the section called “AWS Fargate 符合 FIPS-140 規範”。

執行 `put-account-setting-default` 並將 `fargateFIPSMode` 選項設定為 `enabled`。如需詳細資訊，請參閱 Amazon Elastic Container Service API 參考中的 《put-account-setting-default》。

- 您可以使用下列命令來開啟 FIPS-140 相容性。

```
aws ecs put-account-setting-default --name fargateFIPSMode --value enabled
```

範例輸出

```
{
    "setting": {
        "name": "fargateFIPSMode",
        "value": "enabled",
        "principalArn": "arn:aws:iam::123456789012:root",
         "type": user
    }
}
```

您可以執行 `list-account-settings` 以檢視目前的 FIPS-140 法規遵循狀態。使用 `effective-settings` 選項來檢視帳戶層級設定。

```
aws ecs list-account-settings --effective-settings
```

標記授權

Amazon ECS 正在為資源建立引入標記授權。使用者必須具有建立資源之動作的標記權限，例如`ecsCreateCluster`。當您建立資源並指定該資源的標籤時，請 AWS 執行其他授權以確認是否具

有建立標籤的權限。因此，您必須授予使用 ecs:TagResource 動作的明確許可。如需詳細資訊，請參閱 the section called “在建立期間標記資源”。

若要選擇加入標記授權，請執行 put-account-setting-default，並將 tagResourceAuthorization 選項設為 enable。如需詳細資訊，請參閱 Amazon Elastic Container Service API 參考中的 《put-account-setting-default》。您可以執行 list-account-settings 以檢視目前的標記授權狀態。

- 您可以使用以下命令來啟用標記授權。

```
aws ecs put-account-setting-default --name tagResourceAuthorization --value on --
region region
```

範例輸出

```
{
    "setting": {
        "name": "tagResourceAuthorization",
        "value": "on",
        "principalArn": "arn:aws:iam::123456789012:root",
        "type": user
    }
}
```

啟用標記授權後，您必須配置適當的權限，以允許使用者在建立時標記資源。如需詳細資訊，請參閱 the section called “在建立期間標記資源”。

您可以執行 list-account-settings 以檢視目前的標記授權狀態。使用 effective-settings 選項來檢視帳戶層級設定。

```
aws ecs list-account-settings --effective-settings
```

標記授權時間表

您可以透過執行 list-account-settings 檢視 tagResourceAuthorization 值來確認標記授權是否處於作用中狀態。當值為 on 時，表示正在使用標記授權。如需詳細資訊，請參閱 Amazon Elastic Container Service API 參考中的 《list-account-settings》。

以下是與標記授權相關的重要日期。

- 2023 年 4 月 18 日 – 引入標記授權。所有新帳戶和現有帳戶均須選擇加入使用該功能。您可以選擇加入以開始使用標記授權。選擇加入後，您必須授予適當的許可。
- 2024 年 2 月 9 日至 2024 年 3 月 6 日 — 預設情況下，所有新帳戶和未受影響的現有帳戶均啟用標記授權。您可以啟用或停用`tagResourceAuthorization`帳戶設定以驗證 IAM 政策。

  AWS 已通知受影響的帳戶。

  若要停用此功能，請在將`tagResourceAuthorization`選項設定為的情況下執行`put-account-setting-default`。`off`
- 2024 年 3 月 7 日 — 如果您已啟用標記授權，則無法再停用帳戶設定。

  建議您在此日期之前完成 IAM 政策測試。
- 2024 年 3 月 29 日 — 所有帳戶均使用標記授權。帳戶層級設定將不再可用於 Amazon ECS 主控台或。 AWS CLI

AWS Fargate 任務退休等待時間

AWS 當您在標記為淘汰的平台版本修訂上執行 Fargate 任務時，會傳送通知。如需詳細資訊，請參閱 AWS Amazon ECS 上的 Fargate 任務維護常見問題解答。

您可以設定 Fargate 開始任務淘汰的時間。對於需要立即套用更新的工作負載，請選擇立即設定 (0)。當您需要更多控制項時，例如當任務只能在特定時段中停止時，請設定 7 日 (7) 或 14 日 (14) 選項。

我們建議您選擇較短的等待期，以便更快獲得更新的平台版本修訂版。

`put-account-setting`以 root 使用者`put-account-setting-default`或系統管理使用者身分執行來設定等待期間。對 `name` 使用 `fargateTaskRetirementWaitPeriod` 選項，並將 `value` 選項設定為以下值之一：

- `0`- AWS 傳送通知，並立即開始淘汰受影響的工作。
- 7- AWS 傳送通知，並在開始淘汰受影響的工作之前等待 7 個行事曆天。
- 14 - AWS 傳送通知，並等待 14 個日曆日，然後再開始淘汰受影響的任務。

預設值是 7 天。

如需詳細資訊，請參閱《Amazon Elastic Container Service API 參考》中的 put-account-setting-default 和 put-account-setting。

您可以執行下列命令，將等待期限設定為 14 天。

```
aws ecs put-account-setting-default --name fargateTaskRetirementWaitPeriod --value 14
```

範例輸出

```
{
    "setting": {
        "name": "fargateTaskRetirementWaitPeriod",
        "value": "14",
        "principalArn": "arn:aws:iam::123456789012:root",
        "type: user"
    }
}
```

您可以執行 `list-account-settings` 以檢視目前的 Fargate 任務淘汰等待時間。設定 `effective-settings` 選項。

```
aws ecs list-account-settings --effective-settings
```

運行時監控（Amazon GuardDuty 集成）

執行階段監控是一種智慧型威脅偵測服務，可持續監控 AWS 日誌和聯網活動以識別惡意或未經授權的行為，保護在 Fargate 和 EC2 容器執行個體上執行的工作負載

此`guardDutyActivate`參數在 Amazon ECS 中是唯讀的，並指出 Amazon ECS 帳戶中的安全管理員是否開啟或關閉執行階段監控。 GuardDuty 代表您控制此帳戶設定。如需詳細資訊，請參閱使用執行時期監控保護 Amazon ECS 工作負載。

您可以執行`list-account-settings`以檢視目前的 GuardDuty 整合設定。

```
aws ecs list-account-settings
```

範例輸出

```
{
    "setting": {
        "name": "guardDutyActivate",
        "value": "on",
        "principalArn": "arn:aws:iam::123456789012:doej",
        "type": aws-managed"
```

```
    }
}
```

使用主控台檢視 Amazon ECS 帳戶設定

您可以使用 AWS Management Console 檢視您的帳戶設定。

> **Important**
>
> 只能使用 AWS CLI來檢視或變更 `dualStackIPv6`、`fargateFIPSMode` 和 `fargateTaskRetirementWaitPeriod` 帳戶設定。

1. 開啟主控台，網址為 https://console.aws.amazon.com/ecs/v2。
2. 在頂部導覽列中，選取要檢視帳戶設定的區域。
3. 在導覽頁面中，選擇帳戶設定。

修改 Amazon ECS 帳戶設置

您可以使用 AWS Management Console 來修改您的帳戶設定。

此`guardDutyActivate`參數在 Amazon ECS 中是唯讀的，並指出 Amazon ECS 帳戶中的安全管理員是否開啟或關閉執行階段監控。 GuardDuty 代表您控制此帳戶設定。如需詳細資訊，請參閱使用執行時期監控保護 Amazon ECS 工作負載。

> **Important**
>
> 只能使用 AWS CLI來檢視或變更 `dualStackIPv6`、`fargateFIPSMode` 和 `fargateTaskRetirementWaitPeriod` 帳戶設定。

1. 開啟主控台，網址為 https://console.aws.amazon.com/ecs/v2。
2. 在頂部導覽列中，選取要檢視帳戶設定的區域。
3. 在導覽頁面中，選擇 Account Settings (帳戶設定)。
4. 選擇更新。
5. 若要增加或減少可在 awsvpc 網路模式中為每個 EC2 執行個體執行的工作數目，請在「AWSVPC 主幹連線」下選取「中繼」。AWSVPC

6. 若要針對叢集預設使用或停止使用 CloudWatch 容器深入解析，請在CloudWatch 容器深入解析下選取或清除CloudWatch容器深入解析。
7. 若要啟用或停用標籤授權，請在資源標記授權下，選取或清除資源標籤授權。
8. 選擇儲存變更。
9. 在確認畫面，選擇確認以儲存選項。

還原為預設的 Amazon ECS 帳戶設定

您可以使用 AWS Management Console 將 Amazon ECS 帳戶設定還原為預設值。

只有當帳戶設定不再是預設設定時，才能使用還原為帳戶預設值選項。

1. 開啟主控台，網址為 https://console.aws.amazon.com/ecs/v2。
2. 在頂部導覽列中，選取要檢視帳戶設定的區域。
3. 在導覽頁面中，選擇 Account Settings (帳戶設定)。
4. 選擇更新。
5. 選擇還原為帳戶預設值。
6. 在確認畫面，選擇確認以儲存選項。

使用管理 Amazon ECS 帳戶設定 AWS CLI

您可以使用 Amazon ECS API AWS CLI 或開發套件來管理您的帳戶設定。`fargateFIPSMode`和`fargateTaskRetirementWaitPeriod`帳戶設定只能使用這些工具檢視或變更。`dualStackIPv6`

如需有關任務定義可用 API 動作的資訊，請參閱《Amazon Elastic Container Service API 參考》中的帳戶設定動作。

使用以下其中一個命令來為帳戶中所有使用者或角色修改預設帳戶設定。除非使用者或角色明確覆寫自己的這些設定，否則這些變更會套用至整個 AWS 帳戶。

- put-account-setting-default (AWS CLI)

```
aws ecs put-account-setting-default --name serviceLongArnFormat --value enabled --
region us-east-2
```

您也可以使用此命令來修改其他帳戶設定。若要執行此操作，請使用對應的帳戶設定來取代 name 參數。

- 寫入 ECS () AccountSettingAWS Tools for Windows PowerShell

```
Write-ECSAccountSettingDefault -Name serviceLongArnFormat -Value enabled -Region us-east-1 -Force
```

修改使用者帳戶的帳戶設定 (AWS CLI)

使用以下其中一個命令來修改使用者的帳戶設定。若您以根使用者身分使用這些命令，變更會套用到整個 AWS 帳戶，除非使用者或角色自行明確覆寫這些設定。

- put-account-setting (AWS CLI)

```
aws ecs put-account-setting --name serviceLongArnFormat --value enabled --region us-east-1
```

您也可以使用此命令來修改其他帳戶設定。若要執行此操作，請使用對應的帳戶設定來取代 name 參數。

- 寫入 ECS () AccountSettingAWS Tools for Windows PowerShell

```
Write-ECSAccountSetting -Name serviceLongArnFormat -Value enabled -Force
```

修改特定使用者或角色的帳戶設定 (AWS CLI)

使用下列其中一個命令並在請求中指定使用者、角色或根使用者的 ARN，即可修改特定使用者或角色的帳戶設定。

- put-account-setting (AWS CLI)

```
aws ecs put-account-setting --name serviceLongArnFormat --value enabled --principal-arn arn:aws:iam::aws_account_id:user/principalName --region us-east-1
```

您也可以使用此命令來修改其他帳戶設定。若要執行此操作，請使用對應的帳戶設定來取代 name 參數。

- 寫入 ECS () AccountSettingAWS Tools for Windows PowerShell

```
Write-ECSAccountSetting -Name serviceLongArnFormat -Value enabled -PrincipalArn
 arn:aws:iam::aws_account_id:user/principalName -Region us-east-1 -Force
```

Amazon ECS 的 IAM 角色

IAM 角色是您可以在帳戶中建立的另一種 IAM 身分，具有特定的許可。在 Amazon ECS 中，您可以建立角色，將許可授與 Amazon ECS 資源 (例如容器或服務)。

Amazon ECS 所需的角色取決於任務定義啟動類型和您使用的功能。使用下表判斷 Amazon ECS 所需的 IAM 角色。

角色	定義	需要時	其他資訊
任務執行角色	此角色可讓 Amazon ECS 代表您使用其他 AWS 服務。	您的任務託管在外部實例上 AWS Fargate 或在外部實例上，並且： • 從 Amazon ECR 私有存儲庫中提取容器映像。 • 從執行任務的帳戶不同的帳戶中的 Amazon ECR 私有儲存庫中提取容器映像。 • 使用記錄驅動程式將容器 CloudWatch 記錄檔傳送至記`awslogs`錄檔。 您的任務託管在任一個 AWS Fargate 或 Amazon EC2 執行個體上，並且：	Amazon ECS 任務執行 IAM 角色

角色	定義	需要時	其他資訊
		• 使用私人登錄驗證。 • 使用執行階段監視。 • 工作定義會使用 Secret Manager 密碼或 AWS Systems Manager 管理員參數存放區參數來參照敏感資料。	
任務角色	此角色可讓您的應用程式程式碼 (在容器上) 使用其他 AWS 服務。	您的應用程式會存取其他 AWS 服務，例如 Amazon S3。	Amazon ECS 任務 IAM 角色
容器執行個體角色	此角色可讓您的 EC2 執行個體或外部執行個體向叢集註冊。	您的任務託管在 Amazon EC2 執行個體或外部執行個體上。	Amazon ECS 容器執行個體 IAM 角色
Amazon ECS Anywhere 角色	此角色可讓您的外部執行個體存取 AWS API。	您的任務託管在外部實例上。	Amazon ECS Anywhere IAM 角色
Amazon ECS 角色 CodeDeploy	此角色允許 CodeDeploy 對您的服務進行更新。	您可以使用 CodeDeploy 藍/綠部署類型來部署服務。	Amazon ECS CodeDeploy IAM 角色
Amazon ECS 角色 EventBridge	此角色允許 EventBridge 對您的服務進行更新。	您可以使用 EventBridge 規則和目標來排程工作。	Amazon ECS EventBridge IAM 角色

角色	定義	需要時	其他資訊
Amazon ECS 基礎設施角色	此角色可讓 Amazon ECS 管理叢集中的基礎設施資源。	• 您想要將 Amazon EBS 磁碟區連接到您的 Fargate 或 EC2 啟動類型的 Amazon ECS 任務。基礎設施角色可讓 Amazon ECS 為您的任務管理 Amazon EBS 磁碟區。 • 您想要使用傳輸層安全性 (TLS) 來加密 Amazon ECS 服務 Connect 服務之間的流量。	Amazon ECS 基礎設施 IAM 角色

Amazon ECS 任務定義

任務定義是您應用程式的藍圖。其是一種 JSON 格式的文字檔案，描述了構成應用程式的參數和一個或多個容器。

以下是您可在任務定義中指定的一些參數：

- 要使用的啟動類型，決定您任務託管所在的基礎設施
- 和您任務中每個容器一起使用的 Docker 映像
- 每個任務或任務中每個容器使用多少 CPU 和記憶體
- 記憶體和 CPU 需求
- 工作執行所在之容器的作業系統
- 您任務中的容器所使用的 Docker 聯網模式
- 用於任務的記錄組態
- 如果容器完成或失敗，任務是否繼續執行
- 容器啟動時執行的命令
- 任務中的容器使用的任何資料磁碟區
- 任務使用的 IAM 角色

如需有關任務定義參數的完整清單，請參閱 Amazon ECS 任務定義參數。

建立任務定義之後，可以將任務定義當做任務或服務來執行。

- 任務是在叢集內將任務定義執行個體化。您在 Amazon ECS 內為應用程式建立任務定義後，可以指定要在您叢集上執行的任務數量。
- Amazon ECS 服務在 Amazon ECS 叢集中同時執行和維護您所需的任務數量。其運作方式為，如果有任何任務因任何原因而出現故障或停止，Amazon ECS 服務排程器就會根據您的任務定義啟動另一個執行個體。這樣就可以取代該任務，從而在服務中保持所需的任務數量。

主題

- Amazon ECS 任務定義狀態
- 架構您的 Amazon ECS 應用程式
- 使用主控台建立 Amazon ECS 任務定義
- 使用主控台更新 Amazon ECS 任務定義

- 使用主控台取消註冊 Amazon ECS 任務定義修訂
- 使用主控台刪除 Amazon ECS 任務定義修訂
- Amazon ECS 任務定義使用案例
- Amazon ECS 任務定義參數
- Amazon ECS 任務定義模板
- Amazon ECS 任務定義範例

Amazon ECS 任務定義狀態

當您建立、取消註冊或刪除任務定義時，任務定義會變更狀態。您可以在主控台中或使用來檢視工作定義狀態DescribeTaskDefinition。

以下是可能出現的任務定義狀態：

ACTIVE

任務定義在 Amazon ECS 上註冊之後就會是 ACTIVE 狀態。您可以使用狀態為 ACTIVE 的任務定義來執行任務或建立服務。

非作用中

取消註冊任務定義後，任務定義會從 ACTIVE 狀態轉變為 INACTIVE 狀態。您可以透過呼叫 DescribeTaskDefinition 來擷取處於 INACTIVE 狀態的任務定義。您不能使用狀態為 INACTIVE 的任務定義，來執行新任務或建立新服務。這不會對現有服務或任務造成影響。

DELETE_IN_PROGRESS

提交要刪除的任務定義後，這些任務定義會從 INACTIVE 狀態轉換為 DELETE_IN_PROGRESS 狀態。任務定義處於 DELETE_IN_PROGRESS 狀態後，Amazon ECS 會定期驗證目標任務定義未被任何作用中的任務或部署引用，然後永久刪除該任務定義。您不能使用狀態為 DELETE_IN_PROGRESS 的任務定義，來執行新任務或建立新服務。您可以隨時提交作業定義以進行刪除，而不會影響現有的工作和服務。

您可以在主控台中檢視狀態為 DELETE_IN_PROGRESS 的任務定義，且可以透過呼叫 DescribeTaskDefinition 來擷取這些任務定義。

當您刪除所有 INACTIVE 任務定義修訂時，任務定義名稱不會顯示在主控台中，也不會在 API 中傳回。如果任務定義修訂版本DELETE_IN_PROGRESS處於狀態，則任務定義名稱會顯示在主控台中，並在 API 中傳回。Amazon ECS 會保留任務定義名稱，下次您使用該名稱建立任務定義時，修訂版本將會遞增。

如果您使用 AWS Config 來管理作業定義，則會向您 AWS Config 收取所有作業定義註冊的費用。您只需支付將最新處於 ACTIVE 狀態的任務定義取消註冊的費用。刪除任務定義無須付費。如需定價的詳細資訊，請參閱AWS Config 定價。

可以封鎖刪除的 Amazon ECS 資源

如果有任何 Amazon ECS 資源依賴於任務定義修訂版本，則任務定義刪除請求將不會完成。以下資源可能會阻止任務定義遭刪除：

- Amazon ECS 任務 - 需要任務定義才能讓任務保持良好的運作狀態。
- Amazon ECS 部署和任務集 - 當為 Amazon ECS 部署或任務集啟動擴展事件時，需要任務定義。

如果您的任務定義保持在DELETE_IN_PROGRESS狀態，您可以使用控制台或 AWS CLI 來識別，然後停止阻止任務定義刪除的資源。

移除封鎖的資源後刪除任務定義

移除封鎖任務定義刪除的資源後，以下規則適用：

- Amazon ECS 任務 - 任務停止後，任務定義刪除最多可能需要 1 小時才能完成。
- Amazon ECS 部署和任務集 - 刪除部署或任務集後，刪除任務定義最多可能需要 24 小時才能完成。

架構您的 Amazon ECS 應用程式

您可以建立應用程式的工作定義來架構應用程式。作業定義包含定義應用程式相關資訊的參數，包括：

- 要使用的啟動類型，這會決定您的工作託管在哪個基礎結構上。

  使用 EC2 啟動類型時，您也可以選擇執行個體類型。對於某些執行個體類型 (例如 GPU)，您需要設定其他參數。如需詳細資訊，請參閱 Amazon ECS 任務定義使用案例。
- 容器映像檔，其中包含您的應用程式程式碼，以及應用程式程式碼執行所需的所有相依性。
- 用於工作中容器的網路模式

  網路模式會決定您的工作如何透過網路進行通訊。

  對於在 EC2 執行個體上執行的任務，有多個選項，但我們建議您使用awsvpc網路模式。awsvpc網路模式可簡化容器網路，因為您可以更好地控制應用程式彼此之間的通訊方式，以及 VPC 中的其他服務。

對於在 Fargate 上執行的工作，您只能使用awsvpc網路模式。

- 用於工作的記錄組態。
- 任何與工作中容器搭配使用的資料磁碟區。

如需有關任務定義參數的完整清單，請參閱 Amazon ECS 任務定義參數。

建立任務定義時，請遵循下列準則：

- 將每個任務定義系列僅用於一個業務目的。

  如果您將多種類型的應用程式容器分組在相同的任務定義中，則無法獨立擴展這些容器。例如，網站和 API 不太可能都需要以相同的速率進行橫向擴展。隨著流量增加，所需的 Web 容器數量將與 API 容器不同。如果這兩個容器部署在相同的任務定義中，則每個任務都會執行相同數量的 Web 容器和 API 容器。
- 將每個應用程式版本與任務定義系列中的任務定義修訂版進行比對。

  在任務定義系列中，將每個任務定義修訂視為特定容器映像設定的時間點快照集。這類似於容器是執行特定應用程式程式碼版本所需之所有內容的快照。

  請確定應用程式程式碼版本、容器 one-to-one 映像標記和工作定義修訂版本之間有對應。典型的發佈過程涉及 git commit，該提交會轉換為使用 git commit SHA 標記的容器映像。然後，該容器映像標籤會取得自己的 Amazon ECS 任務定義修訂版。最後，Amazon ECS 服務已更新，以指示其部署新的任務定義修訂版本。
- 針對每個任務定義系列使用不同的 IAM 角色。

  使用自己的 IAM 角色定義每個任務定義。此建議應與我們為每個業務組件提供自己的任務定義系列的建議一起完成。透過實作這兩種最佳做法，您可以限制每個服務對 AWS 帳戶中資源的存取權限。例如，您可以授予身分驗證服務存取權限以連線到您的密碼資料庫。同時，您還可以確保只有您的訂單服務才能存取信用卡付款資訊。

Amazon ECS 容器映像的最佳實務

容器映像是一組關於如何建置容器的指示。容器映像會包含您應用程式的程式碼，以及應用程式的程式碼執行所需的所有相依性。應用程式相依性包括應用程式程式碼所依賴的原始程式碼套件、解譯語言的語言執行期，以及動態連結程式碼所依賴的二進位套件。

設計和建置容器映像時，請遵循下列準則：

- 將所有應用程式相依性儲存為容器映像內的靜態檔案，使容器映像完整。

  如果您變更容器映像檔中的某些項目，請使用變更建立新的容器映像檔。

- 在容器內執行單一應用程式處理程序。

  容器生命週期是應用程式處理程序執行的長度。Amazon ECS 會取代損毀的程序，並決定要在何處啟動更換程序。完整的映像可讓整體部署更具彈性。

- 讓你的應用程序處理SIGTERM。

  當 Amazon ECS 停止任務時，它會先向任務傳送 SIGTERM 訊號，以通知應用程式需要完成並關閉。Amazon ECS 然後發送一SIGKILL條消息。當應用程式忽略時SIGTERM，Amazon ECS 服務必須等待傳送SIGKILL訊號才能終止程序。

  您需要確定應用程序完成其工作所需的時間，並確保您的應用程序處理SIGTERM信號。應用程序的信號處理需要阻止應用程序採取新的工作並完成正在進行的工作，或者當工作花費太長時間才能完成時，將未完成的工作保存到任務以外的存儲中。

- 設定容器化應用程式以將日誌寫入 `stdout` 和 `stderr`。

  將記錄處理與應用程式程式碼解耦，可讓您彈性調整基礎結構層級的記錄處理。其中一個例子是更改您的日誌系統。您可以調整設定，而不是修改服務以及建置和部署新的容器映像檔。

- 使用標籤對容器映像進行版本控制。

  容器映像儲存在容器登錄中。登錄中的每個映像皆以標籤識別。有一個標籤名為 `latest`。此標籤的功能是指向最新版應用程式容器映像的指標，類似於 git 儲存庫中的 HEAD。建議您僅將 `latest` 標記用於測試目的。最佳實務是使用每個建置的唯一標籤來標記容器映像。我們建議您使用用於建置映像的 git commit 的 git SHA 來標記映像。

  您不需要為每次提交建置容器映像。不過，我們建議您在每次將特定程式碼提交發佈至生產環境時，都建置新的容器映像。我們還建議您使用與映像內部程式碼的 git commit 相對應的標籤來標記映像。如果您使用 git commit 標記映像，則可以更快找到映像正在執行的程式碼版本。

  我們也建議您在 Amazon Elastic Container Registry 中開啟不可變的映像標籤。使用此設定時，您無法變更標記指向的容器映像。相反，Amazon ECR 強制執行必須將新映像上傳到新標籤。如需詳細資訊，請參閱《Amazon ECR 使用者指南》中的映像標籤可變性。

當您設計要執行的應用程式時 AWS Fargate，您必須決定將多個容器部署到相同的工作定義，以及在多個工作定義中分別部署容器。如需要下列條件，我們建議您在相同的任務定義中部署多個容器：

- 容器共用相同的生命週期 (亦即，它們一起啟動和終止)。
- 容器必須在相同的基礎主機上執行 (亦即，一個容器參考 localhost 連接埠上的另一個容器)。
- 您容器共享資源。
- 您的容器共用資料磁碟區。

如果不需要這些條件，我們建議在多個任務定義中分別部署容器。這可讓您分別擴展、佈建和取消佈建容器。

Amazon ECS 任務規模的最佳實務

在 Amazon ECS 上部署容器時，最重要的選擇之一就是容器和任務大小。您的容器和任務大小對於擴展和容量規劃都是必不可少的。在 Amazon ECS 中，有兩種用於容量的資源指標：CPU 和記憶體。CPU 的測量單位為完整 vCPU 的 1/1024 個單位 (其中 1024 個單位等於 1 個整個 vCPU)。記憶體以 MB 為單位。在您的任務定義中，您可以宣告資源保留和限制。

宣告保留項目時，您正在宣告工作所需的最小資源量。您的任務至少會收到請求的資源數量。您的應用程式可能能夠使用比宣告的保留區更多的 CPU 或記憶體。但是，這受到您還聲明的任何限制的約束。使用超過預留金額的使用稱為成組分解。在 Amazon ECS 中，保證預訂。例如，如果您使用 Amazon EC2 執行個體提供容量，Amazon ECS 不會在無法完成保留的執行個體上放置任務。

限制是容器或工作可以使用的 CPU 單元或記憶體的最大數量。任何嘗試使用超過此限制的 CPU 都會導致節流。任何嘗試使用更多內存的嘗試都會導致您的容器被停止。

選擇這些值可能具有挑戰性。這是因為最適合您應用程式的值在很大程度上取決於應用程式的資源需求。負載測試您的應用程式是成功規劃資源需求的關鍵，以及更好地瞭解應用程式的需求。

無狀態應用程式

對於水平擴充的無狀態應用程式 (例如負載平衡器後方的應用程式)，建議您先判斷應用程式在提供要求時所耗用的記憶體量。為此，您可以使用傳統工具，例如`ps`或`top`，或監視解決方案，例如 CloudWatch 容器見解。

決定 CPU 保留時，請考慮如何擴展應用程式以符合您的業務需求。您可以使用較小的 CPU 保留區 (例如 256 個 CPU 單位 (或 1/4 vCPU)，以精細的方式向外擴充，將成本降至最低。但是，它們可能無法快速擴展以滿足需求的顯著峰值。您可以使用較大的 CPU 保留來更快速地擴展和擴展，從而更快地匹配需求峰值。不過，較大的 CPU 保留成本較高。

其他應用

對於無法水平擴充的應用程式 (例如單一背景工作者或資料庫伺服器)，可用的容量和成本代表您最重要的考量。您應該根據負載測試指出您需要為流量提供服務以符合服務等級目標的記憶體和 CPU 數量。Amazon ECS 可確保應用程式放置在容量足夠的主機上。

Amazon ECS 的網路安全最佳實務

網路安全性是包含數個子主題的廣泛主題。其中包括 encryption-in-transit網路區段和隔離、防火牆、流量路由和可觀察性。

傳輸中加密

加密網路流量可防止未經授權的使用者在經由網路傳輸資料時攔截和讀取資料。使用 Amazon ECS，網路加密可以透過以下任何方式來實作。

- 使用服務網格 (TLS)：

  您可以使用 AWS App Mesh，在使用網狀端點部署的 Envoy 代理伺服器之間設定 TLS 連線。兩個範例為虛擬節點和虛擬閘道。TLS 憑證可以來自 AWS Certificate Manager (ACM)。或者，此憑證可以來自您自己的私人憑證授權機構。

  - 啟用的 Transport Layer Security (TLS)
  - 在 AWS App Mesh 使用 ACM 憑證或客戶提供的憑證時，啟用服務之間的流量加密
  - TLS ACM 演練
  - TLS 檔案演練
  - Envoy
- 使用 Nitro 執行個體：

  根據預設，會自動加密下列 Nitro 執行個體類型之間的流量：
  C5n、G4、I3en、M5dn、M5n、P3dn、R5dn 和 R5n。當流量透過傳輸閘道、負載平衡器或類似媒介進行路由時，不會加密流量。

  - 傳輸中加密
  - 自 2019 年起的新公告
  - 這場演講來自 re:Inforce 2019
- 將伺服器名稱指示 (SNI) 與 Application Load Balancer 搭配使用：

應用 Application Load Balancer (ALB) 和 Network Load Balancer (NLB) 支援伺服器名稱指示 (SNI)。透過使用 SNI，您可以將多個安全應用程式置於單一接聽程式之後。為此，每個接聽程式都具有自己的 TLS 憑證。建議您使用 AWS Certificate Manager (ACM) 為負載平衡器佈建憑證，然後將其新增至接聽程式的憑證清單。 AWS 負載平衡器會搭配 SNI 使用智慧型憑證選取演算法。如果用戶端提供的主機名稱符合憑證清單中的單一憑證，負載平衡器會選擇該憑證。如果用戶端提供的主機名稱符合清單中的多個憑證，負載平衡器會選取用戶端可支援的憑證。範例包括自我簽署憑證或透過 ACM 產生的憑證。

- 使用 Application Load Balancer 的 SNI
- 使用 Network Load Balancer 的 SNI

- E 使用 nd-to-end TLS 憑證加密：

  這涉及部署具有任務的 TLS 憑證。這可以是自我簽署的憑證，也可以是受信任憑證授權機構的憑證。您可以透過參考憑證的秘密來取得憑證。如果不是，您可以選擇執行向 ACM 發出憑證簽署請求 (CSR) 的容器，然後將產生的秘密掛載至共用磁碟區。

  - 使用具有 Amazon ECS 第 1 部分的 Network Load Balancer，一路維護傳輸層安全至您容器
  - 將傳輸層安全性 (TLS) 一路維護到您的容器第 2 部分：使用 AWS Private Certificate Authority

任務聯網

以下建議是考量 Amazon ECS 的運作方式。Amazon ECS 不使用覆蓋網路。反之，任務會設定為在不同的網路模式下運作。例如，設定為使用 `bridge` 模式的任務會從每部主機上執行的 Docker 網路取得不可路由的 IP 地址。設定為使用 `awsvpc` 網路模式的任務會從主機的子網路取得 IP 地址。設定為使用 `host` 網路的任務會使用主機的網路介面。`awsvpc` 是偏好的網路模式。這是因為此模式為唯一可用來為任務指派安全群組的模式。這也是 Amazon ECS 上唯一可用於 AWS Fargate 任務的模式。

任務的安全群組

建議您將任務設定為使用 `awsvpc` 網路模式。將任務設定為使用此模式後，Amazon ECS 代理程式會自動佈建彈性網絡介面 (ENI) 並將其附加至任務。佈建 ENI 時，工作會在 AWS 安全性群組中註冊。安全群組做為虛擬防火牆，可用於控制傳入及傳出的流量。

AWS PrivateLink 和 Amazon ECS

AWS PrivateLink 是一種網路技術，可讓您為包括 Amazon ECS 在內的不同 AWS 服務建立私有端點。在沒有網際網路閘道 (IGW) 連接至 Amazon VPC 且沒有其他路由到網際網路的沙盒環境中，則需要端點。使用 AWS PrivateLink 可確保對 Amazon ECS 服務的呼叫保留在 Amazon VPC 內，而且不

會周遊網際網路。如需如何為 Amazon ECS 和其他相關服務建立 AWS PrivateLink 端點的指示，請參閱 Amazon ECS 介面 Amazon V PC 端點。

> **Important**
>
> AWS Fargate 任務不需要 Amazon ECS 的 AWS PrivateLink 端點。

Amazon ECR 和 Amazon ECS 都支援端點政策。這些政策可讓您完善服務的 API 存取。例如，您可以為 Amazon ECR 建立端點政策，該政策僅允許將映像推送到特定 AWS 帳戶的登錄檔。這類政策可用於防止透過容器映像洩漏資料，同時仍允許使用者推送至授權的 Amazon ECR 登錄檔。如需詳細資訊，請參閱使用 VPC 端點政策。

下列政策允許帳戶中 AWS 的所有主體僅對 Amazon ECR 儲存庫執行所有動作：

```
{
  "Statement": [
    {
      "Sid": "LimitECRAccess",
      "Principal": "*",
      "Action": "*",
      "Effect": "Allow",
      "Resource": "arn:aws:ecr:region:account_id:repository/*"
    },
  ]
}
```

您可以透過設定使用新 `PrincipalOrgID` 屬性的條件來進一步增強此功能。這樣可以防止不屬於您的 IAM 主體推送和提取映像檔 AWS Organizations。如需詳細資訊，請參閱 aws：PrincipalOrgID。

建議對 `com.amazonaws.region.ecr.dkr` 和 `com.amazonaws.region.ecr.api` 端點套用相同的政策。

容器用戶端設定

Amazon ECS 容器代理程式組態檔案包含數個與網路安全相關的環境變數。`ECS_AWSVPC_BLOCK_IMDS` 和 `ECS_ENABLE_TASK_IAM_ROLE_NETWORK_HOST` 用於封鎖任務對 Amazon EC2 中繼資料的存取。`HTTP_PROXY` 用於設定代理程式，以透過 HTTP Proxy 路由連線至網際網路。如需設定代理程式和 Docker 執行期以透過 Proxy 路由的指示，請參閱 HTTP Proxy 組態。

> **Important**
>
> 這些設定在您使用 AWS Fargate時無法使用。

網路安全性建議

建議您在設定 Amazon VPC、負載平衡器和網路時執行下列動作。

在適用於 Amazon ECS 的情況下使用網路加密

您應該在適用的情況下使用網路加密。如果資料包含持卡人資料，某些合規計劃 (例如 PCI DSS) 會要求您加密傳輸中的資料。如果您的工作負載有類似的需求，請設定網路加密。

現代瀏覽器在連接至不安全的站點時警告使用者。如果面向大眾的負載平衡器前端是您的伺服器，請使用 TLS/SSL 加密從用戶端瀏覽器到負載平衡器的流量，並在授權時重新加密至後端。

使用**awsvpc**網路模式和安全群組控制 Amazon ECS 中任務與其他資源之間的流量

當您需要控制任務之間以及任務與其他網路資源之間的流量時，您應該使用 awsvpc 網路模式和安全群組。如果您的服務位於 ALB 之後，請使用安全性群組，以僅允許來自其他使用與 ALB 相同安全群組之網路資源的輸入流量。如果您的應用程式位於 NLB 之後，請將任務的安全群組設定為僅允許來自 Amazon VPC CIDR 範圍以及指派給 NLB 的靜態 IP 地址的輸入流量。

安全群組也應該用來控制 Amazon VPC 內任務與其他資源 (例如 Amazon RDS 資料庫) 之間的流量。

當網路流量需要嚴格隔離時，在不同的 Amazon VPC 中建立 Amazon ECS 叢集

當網路流量需要嚴格隔離時，您應該在獨立的 Amazon VPC 中建立叢集。避免在叢集上執行具有嚴格安全要求的工作負載，而工作負載不必遵循這些要求。當強制執行嚴格的網路隔離時，請在獨立的 Amazon VPC 中建立叢集，並使用 Amazon VPC 端點將服務選擇性公開給其他 Amazon VPC。如需詳細資訊，請參閱 Amazon VPC 端點。

在保固 Amazon ECS 的情況下設定 AWS PrivateLink 端點

您應該在保固時設定 AWS PrivateLink 端點端點。如果您的安全政策阻止您將 Internet Gateway (IGW) 連接到 Amazon VPC，請為 Amazon ECS 和其他服務 (例如 Amazon ECR 和 Amazon) 設定 AWS PrivateLink 端點。 AWS Secrets Manager CloudWatch

使用 Amazon VPC 流程日誌來分析 Amazon ECS 中長時間執行任務的流量

您應該使用 Amazon VPC 流程日誌分析流入及流出長時間執行任務的流量。使用 awsvpc 網路模式的任務會取得自己的 ENI。執行此操作，您可以使用 Amazon VPC 流程日誌監控進出個別任務的流量。Amazon VPC 流程日誌 (v3) 的近期更新使用流量中繼資料 (包括 VPC ID、子網路 ID 和執行個體 ID) 來豐富日誌。此中繼資料可用於協助縮小調查範圍。如需詳細資訊，請參閱 Amazon VPC 流程日誌。

> **Note**
>
> 由於容器的暫時性本質，流程日誌不一定是分析不同容器或容器以及其他網路資源之間流量模式的有效方法。

適用於 EC2 啟動類型的 Amazon ECS 任務聯網選項

在 Amazon EC2 執行個體上託管的 Amazon ECS 任務聯網行為取決於任務定義中定義的網路模式。建議使用 awsvpc 網路模式，除非您有使用不同網路模式的特定需要。

以下是可用的網路模式。

網路模式	EC2 上的 Linux 容器	EC2 上的 Windows 容器	描述
awsvpc	是	是	任務會配置自己的彈性網路介面 (ENI) 和主要私有 IPv4 地址。這會為任務提供與 Amazon EC2 執行個體相同的聯網屬性。
bridge	是	否	任務會使用 Docker 的 Linux 內建虛擬網路，該虛擬網路在託管任務的每個 Amazon EC2 執行個體內執行。Linux 上的內建虛擬網路會使用 bridge Docker 網路驅動程式。如果未在任務定義中指定網路模式，則此為 Linux 上的預設網路模式。
host	是	否	任務會使用主機網路，並直接將容器連接埠映射至託管任務的 Amazon EC2 執行個體 ENI，以略過 Docker 的內建虛擬網路。動態連接埠映射

網路模式	EC2 上的 Linux 容器	EC2 上的 Windows 容器	描述
			無法在此網路模式下使用。任務定義中使用此模式的容器必須指定特定 hostPort 數字。主機上的連接埠號碼無法由多個任務使用。因此，您無法在單一 Amazon EC2 執行個體上執行同一個任務定義的多個任務。
none	是	否	任務沒有外部網路連線。
default	否	是	任務會使用 Docker 的 Windows 內建虛擬網路，該虛擬網路在託管任務的每個 Amazon EC2 執行個體內執行。Windows 上的內建虛擬網路會使用 nat Docker 網路驅動程式。如果未在任務定義中指定網路模式，則此為 Windows 上的預設網路模式。

如需有關 Docker 的 Linux 聯網詳細資訊，請參閱 Docker 文件中的 Networking overview (聯網概觀)。

如需 Windows 上的 Docker 聯網功能詳細資訊，請參閱 Microsoft Containers on Windows Documentation (Windows 上的容器文件) 中的 Windows container networking (Windows 容器聯網功能)。

為 Amazon ECS 任務分配網絡界面

awsvpc 網路模式提供的任務聯網功能可為 Amazon ECS 任務提供與 Amazon EC2 執行個體相同的聯網屬性。使用awsvpc網路模式可簡化容器網路，因為您可以更好地控制應用程式與 VPC 中的其他服務之間的通訊方式。awsvpc網路模式還允許您在工作中更精細地使用安全群組和網路監視工具，為您的容器提供更高的安全性。您也可以使用其他 Amazon EC2 聯網功能 (例如 VPC 流程日誌) 來監控進出任務的流量。此外，屬於同一個任務的容器可以透過 localhost 界面進行通訊。

任務 elastic network interface (ENI) 是 Amazon ECS 的全受管功能。Amazon ECS 會建立 ENI 並將其連接到具有指定安全群組的主機 Amazon EC2 執行個體。任務在 ENI 上傳送和接收網路流量的方式，與 Amazon EC2 執行個體處理其主要網路介面的方式相同。根據預設，會對每個任務 ENI 指派一個私有 IPv4 地址。如果已對雙堆疊模式啟用 VPC，並且您使用具有 IPv6 CIDR 區塊的子網路，則任務 ENI 也會收到 IPv6 地址。每個任務只能有一個 ENI。

這些 ENI 會顯示在您帳戶的 Amazon EC2 主控台中。您的帳戶無法分離或修改 ENI。這是為了防止意外刪除與正在執行之任務相關聯的 ENI。您可以在 Amazon ECS 主控台或使用 DescribeTasksAPI 操作來檢視任務的 ENI 附件資訊。當任務停止或服務縮小規模時，任務 ENI 即予以分離和刪除。

當您需要增加 ENI 密度時，請使用awsvpcTrunking帳戶設定。Amazon ECS 也會為您的容器執行個體建立和附加「中繼」網路界面。幹線網路由 Amazon ECS 全受管。當您在 Amazon ECS 叢集中終止或取消註冊您的容器執行個體時，即會刪除幹線 ENI。如需有關awsvpcTrunking帳戶設定的更多資訊，請參閱必要條件。

您可以awsvpc在任務定義的networkMode參數中指定。如需詳細資訊，請參閱 網路模式。

然後，當您執行工作或建立服務時，請使用包含一或多個子網路的networkConfiguration參數，將您的工作放在一或多個安全群組中，以附加至 ENI。如需詳細資訊，請參閱 網路組態。任務放置在與這些子網相同之可用區域中的有效 Amazon EC2 執行個體上，而指定之安全群組則與針對任務所佈建的 ENI 建立關聯。

Linux 考量事項

使用 Linux 作業系統時，請考量下列事項。

- 如果您在awsvpc模式中使用 p5.48xlarge 執行個體，則無法在執行個體上執行超過 1 個工作。
- 使用awsvpc網路模式的任務和服務需要 Amazon ECS 服務連結角色，才能為 Amazon ECS 提供代表您撥打其他 AWS 服務電話的許可。這個角色會在您建立叢集，或在 AWS Management Console 中建立或更新服務時，自動為您建立。如需詳細資訊，請參閱 使用 Amazon ECS 的服務連結角色。您也可以使用下列 AWS CLI 命令建立服務連結角色：

```
aws iam create-service-linked-role --aws-service-name ecs.amazonaws.com
```

- 您的 Amazon EC2 Linux 執行個體需要容器代理程式的版本 `1.15.0` 或更新版本，以執行使用 `awsvpc` 網路模式的任務。如果您使用的是 Amazon ECS 最佳化 AMI，您的執行個體至少需要 `1.15.0-4` 版的 `ecs-init` 套件。
- 在 VPC 上同時啟用 `enableDnsHostnames` 和 `enableDnsSupport` 選項時，Amazon ECS 會使用 Amazon 提供的 (內部) DNS 主機名稱填入任務的主機名稱。如果未啟用這些選項，任務的 DNS 主機名稱會設定為隨機的主機名稱。如需 VPC DNS 設定的詳細資訊，請參閱《Amazon VPC 使用者指南》中的搭配使用 DNS 與 VPC。
- 使用 `awsvpc` 網路模式的每個 Amazon ECS 任務都會收到自己的彈性網路介面 (ENI)，它連接到託管該任務的 Amazon EC2 執行個體。Amazon EC2 Linux 執行個體可連接的網路介面數量有預設配額。主要網路介面視為一個配額。例如，根據預設，`c5.large` 執行個體最多只有三個可與其連接的 ENI。執行個體的主要網路介面視為一個配額。您可以將額外兩個 ENI 連接到執行個體。因為每

項使用 awsvpc 網路模式的任務都需要 ENI，所以通常只能對此執行個體類型執行兩個這類任務。如需有關每種執行個體類型預設 ENI 限制的詳細資訊，請參閱 Amazon EC2 使用者指南中每個執行個體類型每個網路界面的 IP 地址。

- Amazon ECS 支援使用受支援的 Amazon EC2 Linux 執行個體類型啟動加強 ENI 密度的容器執行個體。當您選擇加入 awsvpcTrunking 帳戶設定並使用這些執行個體類型向叢集註冊 Amazon EC2 Linux 執行個體，則這些執行個體的 ENI 配額較高。使用這些具有更高配額的執行個體代表您可以在每個 Amazon EC2 Linux 執行個體中安排更多任務。若要使用具有中繼功能的增強 ENI 密度，您的 Amazon EC2 執行個體必須使用 1.28.1 版或更新版本的容器代理程式。如果您使用的是 Amazon ECS 最佳化 Linux AMI，您的執行個體也至少需要 1.28.1-2 版的 ecs-init 套裝服務。如需選擇使用的 awsvpcTrunking 帳戶設定詳細資訊，請參閱 透過帳戶設定存取 Amazon ECS 功能。如需 ENI 中繼的詳細資訊，請參閱增加 Amazon ECS Linux 容器實例網絡界面。
- 當在 Amazon EC2 Linux 執行個體上託管使用 awsvpc 網路模式的任務時，您的任務 ENI 不會提供公有 IP 位址。若要存取網際網路，必須在設定為使用 NAT 閘道的私有子網路中啟動任務。如需詳細資訊，請參閱 《Amazon VPC 使用者指南》中的 NAT 閘道。入站網路存取必須出自使用私有 IP 位址的 VPC，或自 VPC 透過負載平衡器路由。從公有子網路內啟動的任務無法存取網際網路。
- Amazon ECS 只會識別連接到您的 Amazon EC2 Linux 執行個體的 ENI。如果您將 ENI 手動連接到執行個體，Amazon ECS 可能會嘗試向沒有足夠網路介面卡的執行個體新增任務。這可能導致任務逾時並進入解除佈建狀態，然後進入已停止狀態。建議您不要將 ENI 手動連接到容器執行個體。
- Amazon EC2 Linux 執行個體必須使用 ecs.capability.task-eni 註冊，用以考量來放置具 awsvpc 網路模式的任務。執行 1.15.0-4 版或更新版本 ecs-init 的容器執行個體使用此屬性註冊。
- 您的帳戶無法手動分離或修改由 Amazon EC2 Linux 執行個體建立並連接的 ENI。這是為了防止意外刪除與正在執行之任務相關聯的 ENI。若要釋出任務的 ENI，請停止該任務。
- 當執行任務或建立使用 awsvpc 網路模式的服務時，限制只能在 awsVpcConfiguration 中指定 16 個子網路和 5 個安全群組。如需詳細資訊，請參閱 Amazon 彈性容器服務 API 參考中的AwsVpc組態。
- 當任務在 awsvpc 網路模式中啟動時，Amazon ECS 容器代理程式會先為每項任務建立額外的 pause 容器，然後在任務定義中啟動容器。然後，它通過運行 amazon-ecs-cni-plugins CNI 插件配置pause容器的網絡命名空間。然後代理會啟動任務中的其他容器，讓它們共用 pause 容器的網路堆疊。這表示任務中的所有容器都可由 ENI 的 IP 地址定址，而且它們彼此之間可以透過 localhost 界面通訊。
- 服務的任務使用 awsvpc 網路模式僅支援 Application Load Balancer 和 Network Load Balancer。當您為這些服務建立任何目標群組時，必須選擇 ip 做為目標類型。請勿選擇 instance。這是因為使用 awsvpc 網路模式的任務與 ENI 相關聯，不與 Amazon EC2 Linux 執行個體相關聯。如需詳細資訊，請參閱 使用負載平衡來分配 Amazon ECS 服務流量。

- 如果您的 VPC 已更新以變更其使用的 DHCP 選項集，則無法將這些變更套用到現有的任務。啟動套用這些變更的新任務，驗證這些任務是否正確運作，接著停止現有的任務，以便安全地變更這些網路組態。

Windows 考量

當您使用 Windows 作業系統時，請考慮下列事項：

- 使用 Amazon ECS 最佳化 Windows Server 2016 AMI 的容器執行個體無法託管使用 `awsvpc` 網路模式的任務。如果您的叢集包含 Amazon ECS 最佳化且支援 `awsvpc` 網路模式的 Windows Server 2016 AMI 和 Windows AMI，使用 `awsvpc` 網路模式的任務不會在 Windows 2016 Server 執行個體上啟動。但會在支援 `awsvpc` 網路模式的執行個體上啟動。
- 您的 Amazon EC2 Windows 執行個體需要容器代理程式的版本`1.57.1`或更新版本，才能針對使用該`awsvpc`網路模式的 Windows 容器使用 CloudWatch 指標。
- 使用`awsvpc`網路模式的任務和服務需要 Amazon ECS 服務連結角色，才能為 Amazon ECS 提供代表您撥打其他 AWS 服務電話的許可。這個角色會在您建立叢集，或在 AWS Management Console 中建立或更新服務時，自動為您建立。如需詳細資訊，請參閱 使用 Amazon ECS 的服務連結角色。您也可以使用下列 AWS CLI 命令建立服務連結角色。

```
aws iam create-service-linked-role --aws-service-name ecs.amazonaws.com
```

- 您的 Amazon EC2 Windows 執行個體需要容器代理程式的版本 `1.54.0` 或更新版本，以執行使用 `awsvpc` 網路模式的任務。啟動執行個體時，必須設定 `awsvpc` 網路模式所需的選項。如需詳細資訊，請參閱 the section called “引導容器實例”。
- 在 VPC 上同時啟用 `enableDnsHostnames` 和 `enableDnsSupport` 選項時，Amazon ECS 會使用 Amazon 提供的 (內部) DNS 主機名稱填入任務的主機名稱。如果未啟用這些選項，任務的 DNS 主機名稱會是隨機的主機名稱。如需 VPC DNS 設定的詳細資訊，請參閱《Amazon VPC 使用者指南》中的搭配使用 DNS 與 VPC。
- 每項使用 `awsvpc` 網路模式的 Amazon ECS 任務都會收到自己的彈性網路介面 (ENI)，連接到裝載該任務的 Amazon EC2 Windows 執行個體。Amazon EC2 Windows 執行個體可連接的網路介面數量有預設配額。主要網路介面視為一個配額。例如，根據預設，`c5.large` 執行個體最多只有三個可與其連接的 ENI。執行個體的主要網路介面視為一個配額。您可以將額外兩個 ENI 連接到執行個體。因為每項使用 `awsvpc` 網路模式的任務都需要 ENI，所以通常只能對此執行個體類型執行兩個這類任務。如需有關每種執行個體類型預設 ENI 限制的詳細資訊，請參閱 Amazon EC2 使用者指南中每個執行個體類型每個網路界面的 IP 地址。

- 當在 Amazon EC2 Windows 執行個體上託管使用 awsvpc 網路模式的任務時，您的任務 ENI 不會提供公有 IP 位址。若要存取網際網路，在設定為使用 NAT 閘道的私有子網路中啟動任務。如需詳細資訊，請參閱 《Amazon VPC 使用者指南》中的 NAT 閘道。入站網路存取必須出自使用私有 IP 位址的 VPC，或自 VPC 透過負載平衡器路由。從公有子網路內啟動的任務無法存取網際網路。
- Amazon ECS 只會識別連接到您的 Amazon EC2 Windows 執行個體的 ENI。如果您將 ENI 手動連接到執行個體，Amazon ECS 可能會嘗試向沒有足夠網路介面卡的執行個體新增任務。這可能導致任務逾時並進入解除佈建狀態，然後進入已停止狀態。建議您不要將 ENI 手動連接到容器執行個體。
- Amazon EC2 Windows 執行個體必須使用 ecs.capability.task-eni 註冊，用以考量來放置具 awsvpc 網路模式的任務。
- 您無法手動修改或分離由 Amazon EC2 Windows 執行個體建立並連接的 ENI。這是為了防止您意外刪除與正在執行之任務相關聯的 ENI。若要釋出任務的 ENI，請停止該任務。
- 您在執行任務或建立使用 awsvpc 網路模式的服務時，您最多只能在 awsVpcConfiguration 指定 16 個子網和五個安全群組。如需詳細資訊，請參閱 Amazon 彈性容器服務 API 參考中的AwsVpc組態。
- 當任務在 awsvpc 網路模式中啟動時，Amazon ECS 容器代理程式會先為每項任務建立額外的 pause 容器，然後在任務定義中啟動容器。然後，它通過運行 amazon-ecs-cni-plugins CNI 插件配置pause容器的網絡命名空間。然後代理會啟動任務中的其他容器，讓它們共用 pause 容器的網路堆疊。這表示任務中的所有容器都可由 ENI 的 IP 地址定址，而且它們彼此之間可以透過 localhost 界面通訊。
- 服務的任務使用 awsvpc 網路模式僅支援 Application Load Balancer 和 Network Load Balancer。當您為這些服務建立任何目標群組時，必須選擇 ip 做為目標類型，而不是選擇 instance。這是因為使用 awsvpc 網路模式的任務與 ENI 相關聯，不與 Amazon EC2 Windows 執行個體相關聯。如需詳細資訊，請參閱 使用負載平衡來分配 Amazon ECS 服務流量。
- 如果您的 VPC 已更新以變更其使用的 DHCP 選項集，則無法將這些變更套用到現有的任務。啟動套用這些變更的新任務，驗證這些任務是否正確運作，接著停止現有的任務，以便安全地變更這些網路組態。
- 若您在 EC2 Windows 組態中使用 awsvpc 網路模式，則系統不支援下列項目：
  - 雙堆疊組態
  - IPv6
  - ENI 中繼

在雙堆疊模式下使用 VPC

在雙堆疊模式下使用 VPC 時，您的工作可以透過 IPv4 或 IPv6 或兩者進行通訊。IPv4 和 IPv6 位址是彼此獨立的。因此，您必須在 VPC 中分別為 IPv4 和 IPv6 設定路由和安全性。如需如何將 VPC 設定為雙堆疊模式的詳細資訊，請參閱《Amazon VPC 使用者指南》中的遷移至 IPv6。

如果您為 VPC 設定網際網路閘道或傳出限定網際網路閘道，則可以在雙堆疊模式下使用 VPC。藉此，指派 IPv6 位址的工作就能透過網際網路閘道或僅限出口的網際網路閘道存取網際網路。NAT 閘道是選用。如需詳細資訊，請參閱 《Amazon VPC 使用者指南》 中的網際網路閘道和輸出限定網際網路閘道。

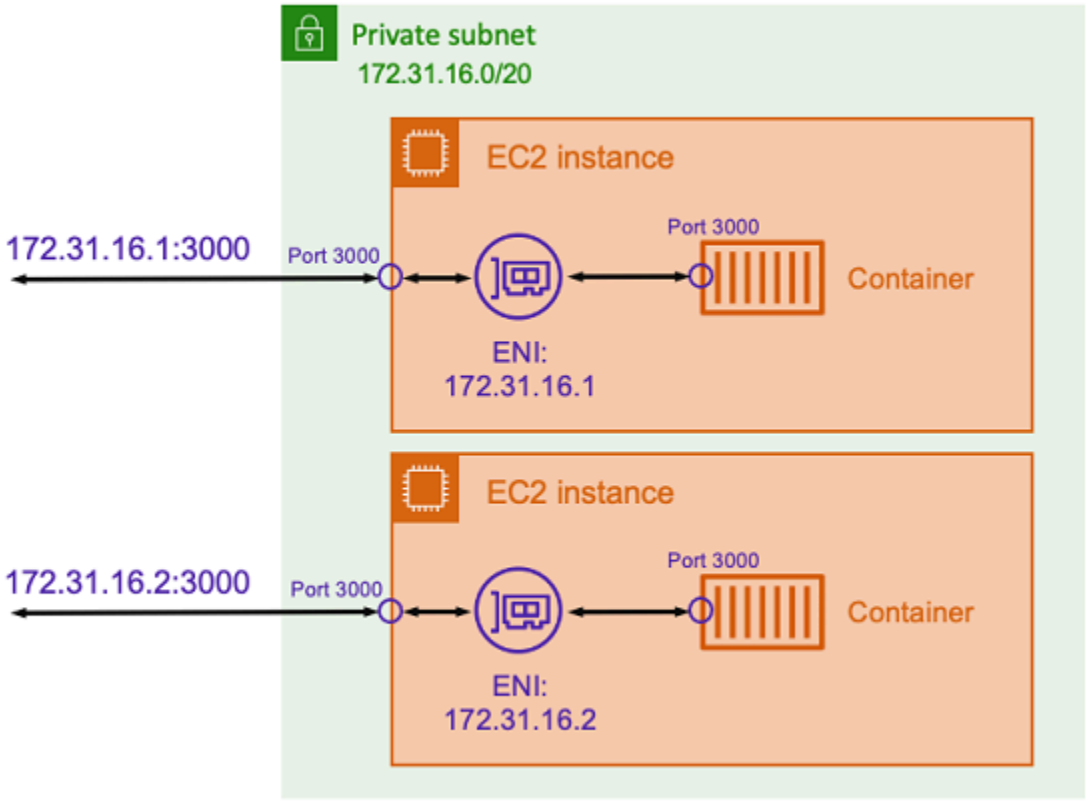
如果符合下列條件，將會為 Amazon ECS 任務指派 IPv6 地址：

- Amazon EC2 Linux 執行個體託管的任務正在使用版本 `1.45.0` 或更新版本的容器代理程式。如需如何檢查執行個體使用之代理程式版本以及視需要進行更新的相關資訊，請參閱 更新 Amazon ECS 容器代理程式。
- 此 `dualStackIPv6` 帳戶設定已啟用。如需詳細資訊，請參閱 透過帳戶設定存取 Amazon ECS 功能。
- 您的任務是使用 `awsvpc` 網路模式。
- 您的 VPC 和子網已針對 IPv6 進行設定。組態包含在指定子網路中建立的網路介面。如需如何將 VPC 設定為雙堆疊模式的詳細資訊，請參閱《Amazon VPC 使用者指南》中的遷移至 IPv6 和修改您子網路的公有 IPv6 定址屬性。

將 Amazon ECS 容器連接埠對應至 EC2 執行個體網路界面

僅 Amazon EC2 執行個體上託管的 Amazon ECS 任務支援 `host` 網路模式。使用 Fargate 上的 Amazon ECS 時不支援。

`host` 網路模式是 Amazon ECS 支援的最基本網路模式。使用託管模式，容器的聯網功能會直接繫結至執行容器的基礎主機。



假設您正在使用 Express 應用程式執行 Node.js 容器，該應用程式會接聽連接埠 3000 (類似於上圖中所示的連接埠)。使用 host 網路模式時，容器會使用基礎主機 Amazon EC2 執行個體的 IP 地址在連接埠 3000 上接收流量。我們不建議使用此模式。

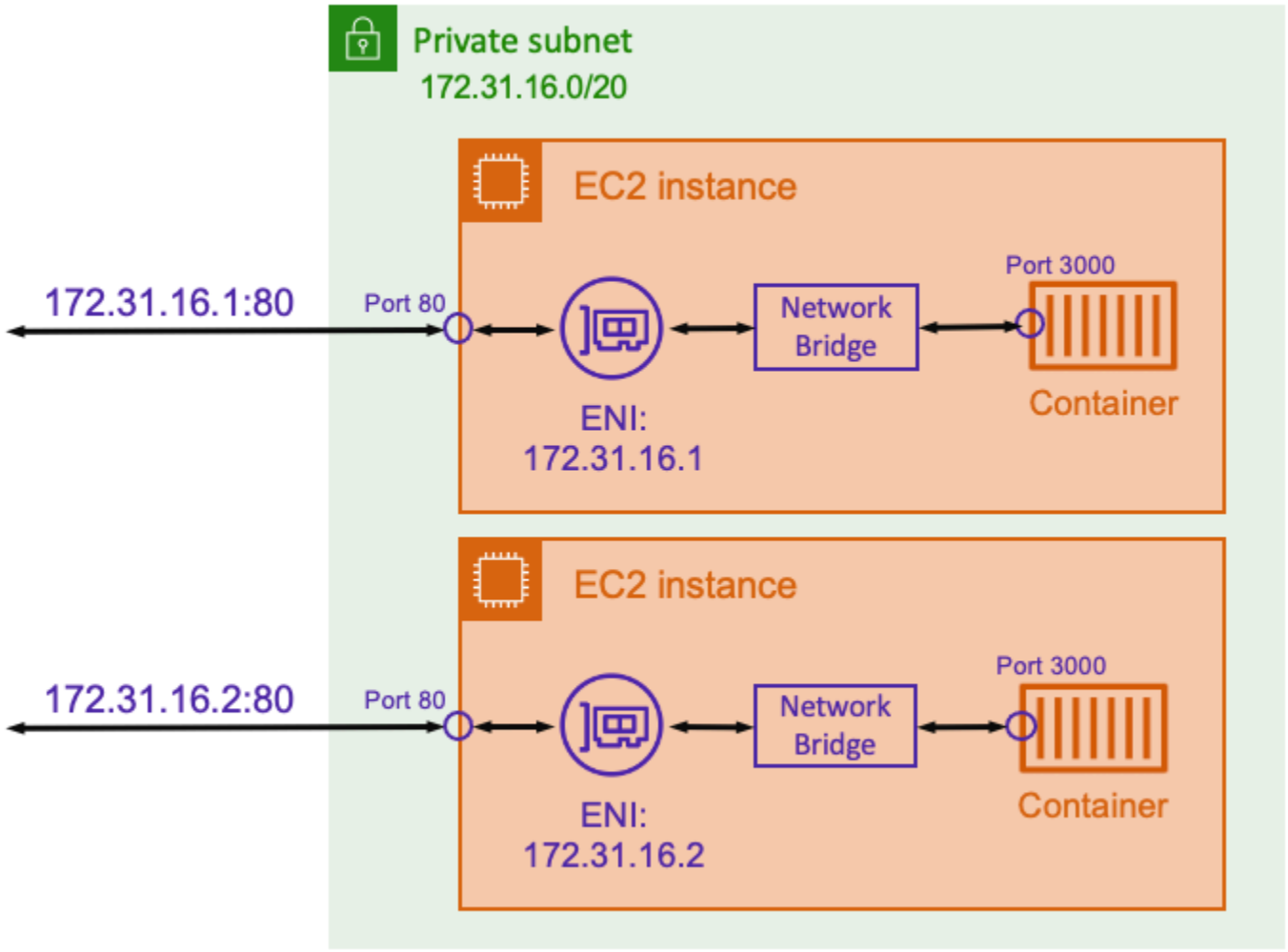
使用此網路模式存在重大的缺點。您不能在每台主機上執行多個任務的執行個體化。這是因為只有第一個任務可以繫結至 Amazon EC2 執行個體上所需的連接埠。使用 host 網路模式時，也無法重新對應容器連接埠。例如，如果應用程式需要接聽特定的連接埠號碼，則無法直接重新對應連接埠號碼。相反地，您必須透過變更應用程式組態來管理任何連接埠衝突。

使用 host 網路模式時也存在安全隱患。此模式允許容器模擬主機，並允許容器連線至主機上的私有迴路網路服務。

使用泊塢視窗的虛擬網路進行 Amazon ECS Linux 任務

僅 Amazon EC2 執行個體上託管的 Amazon ECS 任務支援 bridge 網路模式。

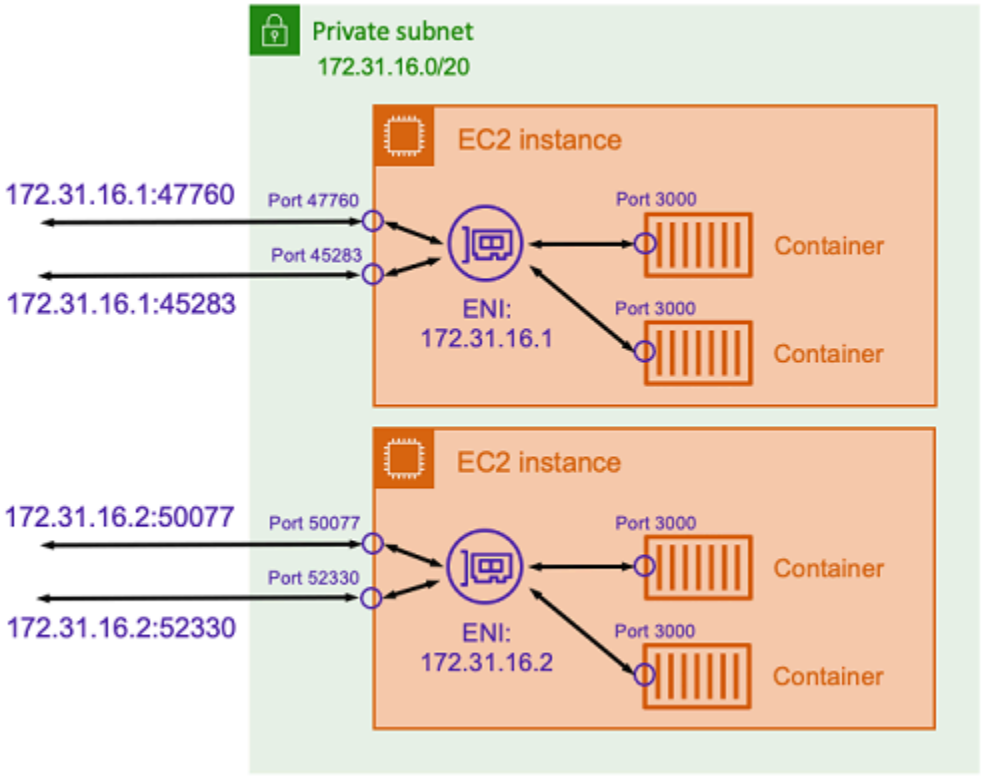
在 bridge 模式下，您正在使用虛擬網路橋接器，在主機和容器的網路之間建立一個層。如此一來，您就可以建立將主機連接埠重新對應至容器連接埠的連接埠映射。映射可以是靜態或動態模式。



透過靜態連接埠映射，您可以明確定義要對應至容器連接埠的主機連接埠。使用上述範例，主機上的 80 連接埠會對應至容器上的連接埠 3000。若要與容器化應用程式通訊，請將流量傳送至 Amazon EC2 執行個體 IP 地址的連接埠 80。從容器化應用程式的角度來看，其可以看到連接埠 3000 上的入站流量。

如果您只想變更流量連接埠，則適用於靜態連接埠映射。但是，這仍然與使用 host 網路模式具有相同的缺點。您不能在每台主機上執行多個任務的執行個體化。這是因為靜態連接埠映射僅允許單一容器對應至連接埠 80。

為了解決這個問題，可以考慮使用具有動態連接埠映射的 bridge 網路模式，如下圖所示。



透過不在連接埠映射中指定主機連接埠，您可以讓 Docker 從暫時連接埠範圍中選擇一個隨機、未使用的連接埠，並將其指派為容器的公有主機連接埠。例如，在容器上接聽連接埠 3000 的 Node.js 應用程式可能會被指派一個隨機的高編號連接埠，例如在 Amazon EC2 主機上的 47760。這樣做意味著您可以在主機上執行該容器的多個副本。此外，每個容器都可以在主機上指派自己的連接埠。容器的每個副本都會在連接埠 3000 上接收流量。不過，傳送流量至這些容器的用戶端會使用隨機指派的主機連接埠。

Amazon ECS 可協助您追蹤每個任務的隨機指派連接埠。它會自動更新負載平衡器目標群組和 AWS Cloud Map 服務探索，以取得工作 IP 位址和連接埠清單來達成此目標。這樣可以更輕鬆地使用透過動態連接埠的 `bridge` 模式運作的服務。

但是，使用 `bridge` 網路模式的缺點之一是難以鎖定服務對服務通訊。由於服務可能會指派給任何隨機、未使用的連接埠，因此必須在主機之間開放廣泛的連接埠範圍。然而，建立特定規則以使特定服務只能與另一個特定服務通訊並不容易。這些服務沒有可用於安全群組聯網規則的特定連接埠。

適用於 Fargate 啟動類型的 Amazon ECS 任務聯網選項

依預設，Fargate 上的每個 Amazon ECS 任務都會提供彈性網路介面 (ENI)，具有主要私有 IP 地址。使用公有子網路時，您可以選擇性地將公有 IP 位址指派給任務的 ENI。如果您的 VPC 設定為雙堆疊模式，而且您使用具有 IPv6 CIDR 區塊的子網路，則您的工作 ENI 也會收到 IPv6 位址。任務在指定

時間內只能有一個相關聯的 ENI。屬於同一個任務的容器也可透過 `localhost` 介面進行通訊。如需有關預設 VPC 的詳細資訊，請參閱《Amazon VPC 使用者指南》中的預設 VPC 和預設子網路。

若要讓 Fargate 上的任務提取容器映像，任務必須有通往網際網路的路由。以下說明如何確認您的任務具有通往網際網路的路由。

- 使用公有子網路時，您可將公有 IP 地址指派給任務 ENI。
- 使用私人子網路時，子網路可以連結 NAT 閘道。
- 使用 Amazon ECR 中託管的容器映像時，您可以將 Amazon ECR 設定為使用介面 VPC 端點，並且透過任務的私有 IPv4 位址進行映像提取 。如需詳細資訊，請參閱 《Amazon Elastic Container Registry 使用者指南》中的 Amazon ECR 介面 VPC 端點 (AWS PrivateLink)。

因為每項任務都會取得自己的 ENI，所以您可以利用聯網功能，例如 VPC 流程日誌，以便您監控任務的進出流量。如需詳細資訊，請參閱 Amazon VPC 使用者指南中的 VPC 流量日誌。

您也可以利用 AWS PrivateLink. 您可以設定 VPC 私人雲端介面端點，以便透過私有 IP 地址存取 Amazon ECS API。 AWS PrivateLink 將虛擬私人 VPC 和 Amazon ECS 之間的所有網路流量限制在 Amazon 網路上。您不需要網際網路閘道、NAT 裝置或虛擬私有閘道。如需詳細資訊，請參閱《Amazon ECS 最佳實務指南》中的 AWS PrivateLink。

如需如何搭配使用`NetworkConfiguration`資源的範例 AWS CloudFormation，請參閱the section called “使用獨立堆疊建立 Amazon ECS 資源”。

建立的 ENI 由 AWS Fargate全受管。此外，還有用來將許可授予 Fargate 的相關 IAM 政策。對於使用 Fargate 平台版本 `1.4.0` 或更新版本的任務，該任務會收到單一 ENI (也稱為任務 ENI)，所有網路流量會流經 VPC 中的 ENI。此流量記錄在您的 VPC 流程日誌中。對於使用 Fargate 平台版本 `1.3.0` 和舊版的任務，除了任務 ENI 以外，該任務也會收到個別 Fargate 擁有的 ENI，其用於某些網路流量，這些流量不會顯示在 VPC 流程日誌中。以下資料表描述網路流量行為，以及每個平台版本所需的 IAM 政策。

動作	使用 Linux 平台版本 **`1.3.0`** 和更早版本時的流量流程	使用 Linux 平台版本 **`1.4.0`** 時的流量流程	使用 Windows 平台版本 **`1.0.0`** 時的流量流程	IAM 許可
擷取 Amazon ECR 登入憑證	Fargate 擁有的 ENI	任務 ENI	任務 ENI	任務執行 IAM 角色

動作	使用 Linux 平台版本 **1.3.0** 和更早版本時的流量流程	使用 Linux 平台版本 **1.4.0** 時的流量流程	使用 Windows 平台版本 **1.0.0** 時的流量流程	IAM 許可
映像提取	任務 ENI	任務 ENI	任務 ENI	任務執行 IAM 角色
透過日誌驅動程式傳送日誌	任務 ENI	任務 ENI	任務 ENI	任務執行 IAM 角色
透過 FireLens Amazon ECS 傳送日誌	任務 ENI	任務 ENI	任務 ENI	任務 IAM 角色
從 Secrets Manager 或 Systems Manager 中擷取秘密	Fargate 擁有的 ENI	任務 ENI	任務 ENI	任務執行 IAM 角色
Amazon EFS 檔案系統流量	無	任務 ENI	任務 ENI	任務 IAM 角色
應用程式流量	任務 ENI	任務 ENI	任務 ENI	任務 IAM 角色

考量事項

使用任務聯網時，請考量下列事項。

- 需要 Amazon ECS 服務連結角色，才能為 Amazon ECS 提供代表您撥打其他 AWS 服務電話的許可。這個角色會在您建立叢集，或在 AWS Management Console中建立或更新服務時建立。如需詳細資訊，請參閱 使用 Amazon ECS 的服務連結角色。您也可以使用下列 AWS CLI 命令建立服務連結角色。

```
aws iam create-service-linked-role --aws-service-name ecs.amazonaws.com
```

- 在 VPC 上同時啟用 `enableDnsHostnames` 和 `enableDnsSupport` 選項時，Amazon ECS 會使用 Amazon 提供的 DNS 主機名稱填入任務的主機名稱。如果未啟用這些選項，任務的 DNS 主機名稱會設定為隨機的主機名稱。如需 VPC DNS 設定的詳細資訊，請參閱《Amazon VPC 使用者指南》中的搭配使用 DNS 與 VPC。
- 您為 `awsVpcConfiguration` 最多只能指定 16 個子網和 5 個安全群組。如需詳細資訊，請參閱 Amazon 彈性容器服務 API 參考中的AwsVpc組態。
- 您的帳戶無法手動分離或修改由 Fargate 建立並連接的 ENI。這是為了防止意外刪除與正在執行之任務相關聯的 ENI。若要釋出任務的 ENI，請停止該任務。
- 如果 VPC 子網已更新以變更其使用的 DHCP 選項集，您也無法將這些變更套用到使用 VPC 的現有任務。啟動新任務，其將在測試新變更接著停止舊任務的同時，接收新的設定以順利遷移 (如果不需要轉返)。
- 在具有 IPv6 CIDR 區塊的子網中啟動的任務只會在使用 Fargate 平台 Linux `1.4.0` 版或更新版或 Windows `1.0.0` 版時接收 IPv6 位址。
- 對於使用平台 Linux `1.4.0` 版或更新版或者 Windows `1.0.0` 版的任務，任務 ENI 支援巨型訊框。網路界面皆以最大傳輸單位 (MTU) 來設定，這是適合單一框架的最大酬載大小。MTU 越大，單一框架能容納的應用程式酬載越多，可降低每個框架的額外負荷並提高效率。當任務和目的地之間的網路路徑支援巨型訊框時，支援巨型訊框會降低額外負荷。
- 使用 Fargate 啟動類型的服務與任務僅支援 Application Load Balancer 和 Network Load Balancer。不支援 Classic Load Balancer。當您建立任何目標群組時，必須選擇 `ip` 做為目標類型，而不是選擇 `instance`。如需詳細資訊，請參閱 使用負載平衡來分配 Amazon ECS 服務流量。

在雙堆疊模式下使用 VPC

在雙堆疊模式中使用 VPC 時，您的任務可透過 IPv4 或 IPv6 或兩者進行通訊。IPv4 和 IPv6 地址彼此互相獨立。您必須在您的 VPC 中分別為 IPv4 和 IPv6 設定路由和安全。如需將 VPC 設定為雙堆疊模式的詳細資訊，請參閱 《Amazon VPC 使用者指南》 中的遷移至 IPv6。

如果符合下列條件，則會將 IPv6 位址指派給 Fargate 上的 Amazon ECS 任務：

- 您的 Amazon ECS `dualStackIPv6` 帳戶設定已開啟 (`enabled`)，以便在您要啟動任務的區域中啟動任務的 IAM 主體。此設定只能使用 API 或修改 AWS CLI。您可以透過設定帳戶預設設定，選擇為帳戶上的特定 IAM 主體或整個帳戶開啟此設定。如需詳細資訊，請參閱 透過帳戶設定存取 Amazon ECS 功能。
- 已對 IPv6 啟用 VPC 和子網路。如需如何將 VPC 設定為雙堆疊模式的詳細資訊，請參閱《Amazon VPC 使用者指南》中的遷移至 IPv6。

- 您的子網路已啟用以自動指派 IPv6 位址。如需詳細資訊，請參閱《Amazon VPC 使用者指南》中的修改子網路的 IPv6 地址屬性。
- 任務或服務使用適用於 Linux 的 Fargate 平台版本 `1.4.0` 或更高版本。

如果您將 VPC 設定網際網路閘道或傳出限定網際網路閘道，在 Fargate 上指派 IPv6 位址的 Amazon ECS 任務就可存取網際網路。不需要 NAT 閘道。如需詳細資訊，請參閱 《Amazon VPC 使用者指南》 中的網際網路閘道和輸出限定網際網路閘道。

Amazon ECS 任務的儲存選項

Amazon ECS 會根據您的需求，為您提供彈性、符合成本效益的 easy-to-use 資料儲存選項。Amazon ECS 支援下列適用於容器的資料磁碟區選項：

資料量	支援的啟動類型	支援的作業系統	儲存持續性	使用案例
Amazon Elastic Block Store (Amazon EBS)	Fargate，Amazon EC2	Linux	當附加到獨立任務時，可以保留。連接到服務維護的任務時是短暫的。	Amazon EBS 磁碟區為資料密集型容器化工作負載提供符合成本效益、耐用且高效能的區塊儲存。常見使用案例包括資料庫、虛擬桌面和根磁碟區等交易工作負載，以及輸送量密集型工作負載 (例如記錄處理和 ETL 工作負載)。如需詳細資訊，請參閱 使用 Amazon EBS 卷與 Amazon ECS。

資料量	支援的啟動類型	支援的作業系統	儲存持續性	使用案例
Amazon Elastic File System (Amazon EFS)	Fargate，Amazon EC2	Linux	持續	Amazon EFS 磁碟區提供簡單、可擴展且持久的共用檔案儲存，可與 Amazon ECS 任務搭配使用，這些任務會在您新增和移除檔案時自動成長和縮小。Amazon EFS 磁碟區支援並行，對於水平擴展並需要低延遲、高輸送量和一致性等儲存功能的容器化應用程式非常有用。read-after-write 常見使用案例包括工作負載，例如資料分析、媒體處理、內容管理和 Web 服務。如需詳細資訊，請參閱 搭配 Amazon ECS 使用 Amazon EFS 磁碟區。

資料量	支援的啟動類型	支援的作業系統	儲存持續性	使用案例
Amazon FSx for Windows File Server	Amazon EC2	Windows	持續	FSx for Windows File Server 磁碟區提供完全受管理的 Windows 檔案伺服器，您可以使用這些伺服器來佈建需要持續性、分散式、共用和靜態檔案儲存的 Windows 工作。常見的使用案例包括 .NET 應用程式，這些應用程式可能需要本機資料夾做為持續性儲存，Amazon FSx for Windows File Server 器的容器中提供本機資料夾，可讓多個容器在由 SMB 共用支援的相同檔案系統上進行讀寫。如需詳細資訊，請參閱 將 FSx 用於 FSx for Windows File Server 磁碟區與 Amazon ECS。

資料量	支援的啟動類型	支援的作業系統	儲存持續性	使用案例
Docker 磁碟區	Amazon EC2	Windows，Linux	持續	Docker 磁碟區是 Docker 容器執行階段的一項功能，可讓容器透過從主機的檔案系統掛載目錄來保留資料。Docker 磁碟區驅動程式 (也稱為外掛程式) 可用來整合容器磁碟區與外部儲存系統。Docker 卷可以由第三方驅動程序或內置`local`驅動程序管理。Docker 磁碟區的常見使用案例包括提供持續性資料磁碟區或在相同容器執行個體上不同容器上的不同位置共用磁碟區。如需詳細資訊，請參閱 搭配 Amazon ECS 使用碼頭磁碟區。

資料量	支援的啟動類型	支援的作業系統	儲存持續性	使用案例
綁定掛載	Fargate，Amazon EC2	Windows，Linux	短暫的	繫結掛載包含主機上的檔案或目錄，例如 Amazon EC2 執行個體或 AWS Fargate掛載到容器上的目錄。繫結掛載的常見使用案例包括與相同工作中的其他容器共用來源容器中的磁碟區，或在一或多個容器中掛載主機磁碟區或空磁碟區。如需詳細資訊，請參閱 搭配 Amazon ECS 使用綁定掛載。

使用 Amazon EBS 卷與 Amazon ECS

Amazon Elastic Block Store (Amazon EBS) 磁碟區可為資料密集型工作負載提供高可用性、符合成本效益、耐用的高效能區塊儲存。Amazon EBS 磁碟區可與 Amazon ECS 任務搭配使用，用於高輸送量和交易密集型應用程式。

在獨立工作啟動期間，您可以提供將用來將一個 EBS 磁碟區連接至工作的組態。在服務建立或更新期間，您可以提供組態，將每個工作附加一個 EBS 磁碟區至 ECS 服務所管理的每個工作。

透過在啟動時提供磁碟區設定，而不是在工作定義中提供磁碟區組態，您可以建立不受特定資料磁碟區類型或特定 EBS 磁碟區設定限制的工作定義。然後，您可以在不同的執行階段環境中重複使用工作定義 例如，您可以在部署生產工作負載期間提供更多輸送量，而不是預先產生的環境。

附加至 Amazon ECS 任務的 Amazon EBS 磁碟區是由 Amazon ECS 代表您管理。磁碟區可以使用 AWS Key Management Service (AWS KMS) 金鑰加密，以保護您的資料。您可以為附件配置新的空白磁碟區，也可以使用快照從現有磁碟區載入資料。

若要監控磁碟區的效能，您也可以使用 Amazon CloudWatch 指標。如需 Amazon EBS 磁碟區的 Amazon ECS 指標的詳細資訊，請參閱Amazon ECS 指標 CloudWatch 和 Amazon ECS 容器洞察指標。

如需有關 Amazon EBS 磁碟區的詳細資訊，請參閱 Amazon EBS 使用者指南中的 Amazon EBS 磁碟區。

AWS 區域 和 Amazon EBS 磁碟區的可用區域

Amazon EBS 磁碟區可以連接到 Amazon ECS 任務，如下所示： AWS 區域

區域名稱	區域代碼
美國東部 (維吉尼亞北部)	us-east-1
美國東部 (俄亥俄)	us-east-2
美國西部 (加州北部)	us-west-1
美國西部 (奧勒岡)	us-west-2
非洲 (開普敦)	af-south-1
亞太區域 (香港)	ap-east-1
亞太區域 (海德拉巴)	ap-south-2
亞太區域 (雅加達)	ap-southeast-3
亞太區域 (墨爾本)	ap-southeast-4
亞太區域 (孟買)	ap-south-1
亞太區域 (大阪)	ap-northeast-3
亞太區域 (首爾)	ap-northeast-2
亞太區域 (新加坡)	ap-southeast-1

區域名稱	區域代碼
亞太區域 (雪梨)	ap-southeast-2
亞太區域 (東京)	ap-northeast-1
加拿大 (中部)	ca-central-1
歐洲 (法蘭克福)	eu-central-1
歐洲 (愛爾蘭)	eu-west-1
歐洲 (倫敦)	eu-west-2
歐洲 (米蘭)	eu-south-1
歐洲 (巴黎)	eu-west-3
歐洲 (西班牙)	eu-south-2
歐洲 (斯德哥爾摩)	eu-north-1
歐洲 (蘇黎世)	eu-central-2
以色列 (特拉維夫)	il-central-1
中東 (巴林)	me-south-1
中東 (阿拉伯聯合大公國)	me-central-1
南美洲 (聖保羅)	sa-east-1

Important

您無法設定 Amazon EBS 磁碟區以附加到可用區域`euc1-az2`和`use1-az3`可用區域中的 Fargate Amazon ECS 任務。

考量事項

使用 Amazon EBS 磁碟區時，請考慮下列事項：

- Amazon EBS 磁碟區僅支援在 Fargate 上託管的 Linux 任務，以及在以 Amazon ECS 最佳化的 Amazon 機器映像 (AMI) 為Nitro基礎的 Linux 執行個體上託管的 EC2 啟動類型任務。如需執行個體類型的詳細資訊，請參閱 Amazon EC2 使用者指南中的執行個體類型。如需 Amazon ECS 啟動類型的詳細資訊，請參閱Amazon ECS 啟動類型。
- 對於在 Fargate 上託管的任務，平台版本`1.4.0`或更新版本 (Linux) 支援 Amazon EBS 磁碟區。如需詳細資訊，請參閱 適用於 Amazon ECS 的 Fargate Linux 平台版本。
- 對於在 Amazon EC2 Linux 執行個體上託管的任務，在 ECS 優化 AMI `20231219` 或更新版本上支援 Amazon EBS 磁碟區。如需詳細資訊，請參閱擷取 Amazon ECS 最佳化 AMI 中繼資料。
- 磁性 (`standard`) Amazon EBS 磁碟區類型不支援在 Fargate 上託管的任務。如需有關 Amazon EBS 磁碟區類型的詳細資訊，請參閱 Amazon EC2 使用者指南中的亞馬遜 EBS 磁碟區。
- 建立服務或在部署時設定磁碟區的獨立任務時，需要 Amazon ECS 基礎設施 IAM 角色。您可以將 AWS 受管 `AmazonECSInfrastructureRolePolicyForVolumes` IAM 政策附加到角色，也可以使用受管政策做為指南，建立和附加具有符合您特定需求的許可的自己政策。如需詳細資訊，請參閱 Amazon ECS 基礎設施 IAM 角色。
- 您最多可以將一個 Amazon EBS 磁碟區連接到每個 Amazon ECS 任務，而且該磁碟區必須是新磁碟區。您無法將現有的 Amazon EBS 磁碟區附加至任務。不過，您可以使用現有磁碟區的快照，在部署時設定新的 Amazon EBS 磁碟區。
- 您只能在部署時為使用滾動更新部署類型和複本排程策略的服務設定 Amazon EBS 磁碟區。
- Amazon ECS 會自動將保留的標籤`AmazonECSCreated`新增`AmazonECSManaged`至連接的磁碟區。如果您從磁碟區移除這些標籤，Amazon ECS 將無法代表您管理磁碟區。如需標記 Amazon EBS 磁碟區的詳細資訊，請參閱標記 Amazon EBS 磁碟區。如需標記 Amazon ECS 資源的詳細資訊，請參閱標記您的 Amazon ECS 資源。
- 不支援從包含分割區的 Amazon EBS 磁碟區快照佈建磁碟區。
- 連接至服務所管理之工作的磁碟區不會保留，而且一律會在工作終止時刪除。
- 您無法針對正在執行的 Amazon ECS 任務的附件設定 Amazon EBS 磁碟區。 AWS Outposts

將磁碟區組態延遲到 Amazon ECS 任務定義中的啟動時間

若要設定 Amazon EBS 磁碟區以連接到您的任務，您必須在任務定義中指定掛接點組態並命名磁碟區。您也必須設定`configuredAtLaunch`為，`true`因為無法在任務定義中為附件設定 Amazon EBS 磁碟區。相反地，Amazon EBS 磁碟區會在部署期間設定為連接。

下列任務定義顯示任務定義中`mountPoints`和`volumes`物件的語法。如需工作定義參數的詳細資訊，請參閱Amazon ECS 任務定義參數。若要使用此範例，請以您自己的資訊取代 *`user input placeholders`*。

若要使用 AWS Command Line Interface (AWS CLI) 註冊工作定義，請將範本儲存為 JSON 檔案，然後將檔案作為register-task-definition命令的輸入傳遞。

若要使用建立和註冊任務定義 AWS Management Console，請參閱使用主控台建立 Amazon ECS 任務定義。

```
{
    "family": "mytaskdef",
    "containerDefinitions": [
        {
            "name": "nginx",
            "image": "public.ecr.aws/nginx/nginx:latest",
            "networkMode": "awsvpc",
           "portMappings": [
                {
                    "name": "nginx-80-tcp",
                    "containerPort": 80,
                    "hostPort": 80,
                    "protocol": "tcp",
                    "appProtocol": "http"
                }
            ],
            "mountPoints": [
                {
                    "sourceVolume": "myEBSVolume",
                    "containerPath": "/mount/ebs",
                    "readOnly": true
                }
            ]
        }
    ],
    "volumes": [
        {
            "name": "myEBSVolume",
            "configuredAtLaunch": true
        }
    ],
    "requiresCompatibilities": [
        "FARGATE", "EC2"
    ],
    "cpu": "1024",
    "memory": "3072",
    "networkMode": "awsvpc"
```

```
}
```

`mountPoints`

類型：物件陣列

必要：否

容器中資料磁碟區的掛接點。此參數會映射到 Docker Remote API 的建立容器區段中的 `Volumes` 以及 docker run 的 `--volume` 選項。

Windows 容器可在 `$env:ProgramData` 所在的相同磁碟上掛載整個目錄。Windows 容器無法在不同的磁碟機上掛載目錄，而且無法跨磁碟機使用掛載點。您必須指定掛接點，才能將 Amazon EBS 磁碟區直接連接至 Amazon ECS 任務。

`sourceVolume`

類型：字串

必要：是 (當使用 `mountPoints` 時)

要掛載的磁碟區名稱。

`containerPath`

類型：字串

必要：是 (當使用 `mountPoints` 時)

要掛接磁碟區之容器中的路徑。

`readOnly`

類型：布林值

必要：否

如果此數值為 `true`，容器擁有磁碟區的唯讀存取權。如果此值為 `false`，則容器可寫入磁碟區。預設值為 `false`。

`name`

類型：字串

必要：否

磁碟區名稱。最多可使用 255 個字母 (大寫和小寫)、數字、連字號 (-) 和底線 (_)。此名稱會在容器定義mountPoints物件的sourceVolume參數中參照。

configuredAtLaunch

類型：布林值

必要：是，當您要使用將 EBS 磁碟區直接連接至工作時。

指定磁碟區是否可在啟動時進行設定。設定為時true，您可以在執行獨立工作時或建立或更新服務時設定磁碟區。設定為時true，您將無法在工作定義中提供其他磁碟區組態。必須提供此參數並將其設定true為，以設定 Amazon EBS 磁碟區以附加到任務。

為 Amazon ECS 加密存儲在 Amazon EBS 卷中的數據

您可以使用 AWS Key Management Service (AWS KMS) 來製作和管理保護資料的加密金鑰。Amazon EBS 磁碟區會在靜態時使用 AWS KMS keys加密。以下類型的數據被加密：

- 儲存在磁碟區上靜態的資料
- 磁碟 I/O
- 從磁碟區建立的快照
- 從快照建立的新磁碟區

您可以預設設定 Amazon EBS 加密，以便使用為帳戶設定的 KMS 金鑰加密建立並連接到任務的所有新磁碟區。如需有關預設情況下 Amazon EBS 加密和加密的詳細資訊，請參閱 Amazon EC2 使用者指南中的 Amazon EBS 加密。

附加到任務的 Amazon EBS 磁碟區可以使用別名alias/aws/ebs的預設值或對稱 AWS 受管金鑰 的客戶受管金鑰來加密。每個預設值對 AWS 帳戶 每個都 AWS 受管金鑰 是唯一的， AWS 區域 並且會自動建立。若要建立對稱的客戶受管金鑰，請遵循AWS KMS 開發人員指南中建立對稱加密 KMS 金鑰中的步驟。

客戶受管 KMS 金鑰政策

若要使用客戶受管金鑰加密連接至工作的 EBS 磁碟區，您必須設定 KMS 金鑰政策，以確保用於磁碟區組態的 IAM 角色具有使用金鑰的必要權限。

金鑰原則必須包含kms:CreateGrant和kms:GenerateDataKey*權限。kms:ReEncryptTo和kms:ReEncryptFrom權限對於加密使用快照建立的磁碟區是必要的。如果您只想為附件設定和加密新的空白磁碟區，您可以排除kms:ReEncryptTo和kms:ReEncryptFrom權限。

下列 JSON 程式碼片段顯示您可以附加至 KMS 金鑰原則的金鑰原則陳述式。使用這些陳述式將提供 ECS 使用金鑰來加密 EBS 磁碟區的存取權。若要使用範例原則陳述式，請以您自己*user input placeholders*的資訊取代。與往常一樣，只設定您需要的權限。

```
{
      "Effect": "Allow",
      "Principal": { "AWS": "arn:aws:iam::111122223333:role/ecsInfrastructureRole" },
      "Action": "kms:DescribeKey",
      "Resource":"*"
    },
    {
      "Effect": "Allow",
      "Principal": { "AWS": "arn:aws:iam::111122223333:role/ecsInfrastructureRole" },
      "Action": [
      "kms:GenerateDataKey*",
      "kms:ReEncryptTo",
      "kms:ReEncryptFrom"
      ],
      "Resource":"*",
      "Condition": {
        "StringEquals": {
          "kms:CallerAccount": "aws_account_id",
          "kms:ViaService": "ec2.region.amazonaws.com"
        },
        "ForAnyValue:StringEquals": {
          "kms:EncryptionContextKeys": "aws:ebs:id"
        }
      }
    },
    {
      "Effect": "Allow",
      "Principal": { "AWS": "arn:aws:iam::111122223333:role/ecsInfrastructureRole" },
      "Action": "kms:CreateGrant",
      "Resource":"*",
      "Condition": {
        "StringEquals": {
          "kms:CallerAccount": "aws_account_id",
          "kms:ViaService": "ec2.region.amazonaws.com"
```

```
        },
        "ForAnyValue:StringEquals": {
          "kms:EncryptionContextKeys": "aws:ebs:id"
        },
        "Bool": {
          "kms:GrantIsForAWSResource": true
        }
      }
    }
```

如需關鍵原則和權限的詳細資訊，請參閱《AWS KMS 開發人員指南》中的主要原則 AWS KMS和AWS KMS 權限。如需疑難排解與金鑰權限相關的 EBS 磁碟區附件問題，請參閱對 Amazon ECS 任務的 Amazon EBS 磁碟區附件進行疑難排解 。

在 Amazon ECS 部署時指定 Amazon EBS 磁碟區組態

在`configuredAtLaunch`參數設定為的情況下註冊任務定義後`true`，您可以在執行獨立任務或建立或更新服務時，在部署時設定 Amazon EBS 磁碟區。

若要混淆磁碟區，您可以使用 Amazon ECS API，或傳遞 JSON 檔案做為下列命令的輸入： AWS CLI

- `run-task`以執行獨立的 ECS 工作。
- `start-task`以在特定容器實例中運行獨立 ECS 任務。此命令不適用於 Fargate 啟動類型任務。
- `create-service`以建立新的 ECS 服務。
- `update-service`以更新現有服務。

> **Note**
>
> 若要讓任務中的容器寫入已掛接的 Amazon EBS 磁碟區，您必須以根使用者身分執行該容器。

您也可以使用設定 Amazon EBS 磁碟區。 AWS Management Console如需詳細資訊，請參閱以 Amazon ECS 任務的形式執行應用程式、使用主控台建立 Amazon ECS 服務及使用主控台更新 Amazon ECS 服務。

下列 JSON 程式碼片段顯示可在部署時設定的 Amazon EBS 磁碟區的所有參數。若要使用這些參數進行磁碟區組態，請以您自己*user input placeholders*的資訊取代。如需這些參數的詳細資訊，請參閱磁碟區組態。

```
"volumeConfigurations": [
        {
            "name": "ebs-volume",
            "managedEBSVolume": {
                "encrypted": true,
                "kmsKeyId": "arn:aws:kms:us-
east-1:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
                "volumeType": "gp3",
                "sizeInGiB": 10,
                "snapshotId": "snap-12345",
                "iops": 3000,
                "throughput": 125,
                "tagSpecifications": [
                    {
                        "resourceType": "volume",
                        "tags": [
                            {
                                "key": "key1",
                                "value": "value1"
                            }
                        ],
                        "propagateTags": "NONE"
                    }
                ],
                "roleArn": "arn:aws::iam:1111222333:role/ecsInfrastructureRole",
                 "terminationPolicy": {
                    "deleteOnTermination": true//can't be configured for service-
managed tasks, always true
                },
                "filesystemType": "ext4"
            }
        }
    ]
```

Important

請確定volumeName您在組態中指定的與您在作業定volumeName義中指定的相同。

如需有關檢查磁碟區附件狀態的資訊，請參閱對 Amazon ECS 任務的 Amazon EBS 磁碟區附件進行疑難排解。如需 EBS 磁碟區附件所需之 Amazon ECS 基礎設施 AWS Identity and Access Management (IAM) 角色的相關資訊，請參閱。Amazon ECS 基礎設施 IAM 角色

以下是顯示 Amazon EBS 磁碟區組態設定的 JSON 程式碼片段範例。這些例子可以通過將代碼段保存在 JSON 文件中，並將文件作為參數（使用參數）傳遞給 AWS CLI 命令來使用。`--cli-input-json file://`*filename*以您自己的資訊取代 *user input placeholders*。

設定獨立工作的磁碟區

下列程式碼片段顯示設定 Amazon EBS 磁碟區以附加到獨立任務的語法。下列 JSON 程式碼片段顯示設定`volumeType`、`sizeInGiB`、`encrypted`、和`kmsKeyId`設定的語法。JSON 檔案中指定的組態可用來建立 EBS 磁碟區，並將其附加至獨立工作。

```
{
   "cluster": "mycluster",
   "taskDefinition": "mytaskdef",
   "volumeConfigurations": [
        {
            "name": "datadir",
            "managedEBSVolume": {
                "volumeType": "gp3",
                "sizeInGiB": 100,
                "roleArn":"arn:aws:iam:1111222333:role/ecsInfrastructureRole",
                "encrypted": true,
                "kmsKeyId":
 "arn:aws:kms:region:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
            }
        }
   ]
}
```

建立服務時設定磁碟區

下列程式碼片段顯示針對由服務管理的任務的附件設定 Amazon EBS 磁碟區的語法。磁碟區是使用來自快照`snapshotId`。JSON 檔案中指定的組態可用來建立 EBS 磁碟區，並將其附加至服務所管理的每個工作。

```
{
   "cluster": "mycluster",
   "taskDefinition": "mytaskdef",
   "serviceName": "mysvc",
   "desiredCount": 2,
   "volumeConfigurations": [
        {
            "name": "myEbsVolume",
```

```
            "managedEBSVolume": {
              "roleArn":"arn:aws:iam:1111222333:role/ecsInfrastructureRole",
              "snapshotId": "snap-12345"
            }
        }
    ]
}
```

在服務更新時設定磁碟區

下列 JSON 程式碼片段顯示更新服務的語法，而該服務先前沒有為任務的附件設定 Amazon EBS 磁碟區。您必須提供任務定義修訂版本的 ARN，並將`true`其`configuredAtLaunch`設定為。下列 JSON 程式碼片段顯示設定`volumeType`、`sizeInGiB`、`throughput`、`iops`和和`filesystemType`設定的語法。此組態可用來建立 EBS 磁碟區，並將其連接至服務所管理的每項工作。

```
{
   "cluster": "mycluster",
   "taskDefinition": "mytaskdef",
   "serviceName": "mysvc",
   "desiredCount": 2,
  "volumeConfigurations": [
        {
            "name": "myEbsVolume",
            "managedEBSVolume": {
              "roleArn":"arn:aws:iam:1111222333:role/ecsInfrastructureRole",
               "volumeType": "gp3",
                "sizeInGiB": 100,
                 "iops": 3000,
                "throughput": 125,
                "filesystemType": "ext4"
            }
        }
    ]
}
```

將服務設定為不再使用 Amazon EBS 磁碟區

下列 JSON 程式碼片段顯示將服務更新為不再使用 Amazon EBS 磁碟區的語法。您必須提供`configuredAtLaunch`設定為的任務定義的 ARN`false`，或提供不含`configuredAtLaunch`參數的任務定義。您還必須提供一個空`volumeConfigurations`對象。

```
{
```

```
    "cluster": "mycluster",
    "taskDefinition": "mytaskdef",
    "serviceName": "mysvc",
    "desiredCount": 2,
    "volumeConfigurations": []
}
```

Amazon EBS 卷的終止政策

當 Amazon ECS 任務終止時，Amazon ECS 會使用該`deleteOnTermination`值來判斷是否應刪除與已終止任務相關聯的 Amazon EBS 磁碟區。依預設，工作終止時，會刪除附加至工作的 EBS 磁碟區。對於獨立工作，您可以將此設定變更為在工作終止時保留磁碟區。

> **Note**
>
> 連接至由服務管理之工作的磁碟區不會保留，而且一律會在工作終止時刪除。

標記 Amazon EBS 卷

您可以使用`tagSpecifications`物件標記 Amazon EBS 磁碟區。根據磁碟區是附加至獨立工作還是服務中的工作，您可以使用物件提供自己的標籤，並從作業定義或服務設定標籤的傳播。磁碟區可附加的標籤數目上限為 50。

> **Important**
>
> Amazon ECS 會自動將`AmazonECSCreated`和`AmazonECSManaged`保留的標籤附加到 Amazon EBS 磁碟區。這表示您可以控制磁碟區最多 48 個附加標籤的附加。這些額外的標籤可以是使用者定義、ECS 管理或傳播的標籤。

如果您想要將 Amazon ECS 管理的標籤新增至磁碟區，您必須在`UpdateService`、`CreateService`、`RunTask`或`StartTask`呼叫`true`中設定`enableECSManagedTags`為。如果您開啟 Amazon ECS 管理的標籤，Amazon ECS 會使用叢集和服務資訊 (`aws:ecs:`*`clusterName`*和) 自動標記磁碟區。`aws:ecs:`*`serviceName`*如需標記 Amazon ECS 資源的詳細資訊，請參閱標記您的 Amazon ECS 資源。

下列 JSON 程式碼片段顯示使用使用者定義標記連接至服務中每個任務之每個 Amazon EBS 磁碟區的語法。若要使用此範例建立服務，請以您自己*`user input placeholders`*的資訊取代。

```
{
   "cluster": "mycluster",
   "taskDefinition": "mytaskdef",
   "serviceName": "mysvc",
   "desiredCount": 2,
   "enableECSManagedTags": true,
   "volumeConfigurations": [
        {
            "name": "datadir",
            "managedEBSVolume": {
                "volumeType": "gp3",
                "sizeInGiB": 100,
                 "tagSpecifications": [
                    {
                        "resourceType": "volume",
                        "tags": [
                            {
                                "key": "key1",
                                "value": "value1"
                            }
                        ],
                        "propagateTags": "NONE"
                    }
                ]
                "roleArn":"arn:aws:iam:1111222333:role/ecsInfrastructureRole",
                "encrypted": true,
                "kmsKeyId":
 "arn:aws:kms:region:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
            }
        }
   ]
}
```

Important

您必須指定volume資源類型來標記 Amazon EBS 磁碟區。

適用於 Fargate 隨需任務的 Amazon EBS 磁碟區效能

Fargate 隨需任務可用的基準 Amazon EBS 磁碟區 IOPS 和輸送量取決於您為任務請求的 CPU 單位總數。如果您為 Fargate 任務請求 0.25，0.5 或 1 個虛擬 CPU 單元（vCPU），我們建議您配置一般用

途 SSD 卷（gp2或gp3）或硬盤驅動器（HDD）卷（st1或）。sc1如果您為 Fargate 任務請求超過 1 個 vCPU，則以下基準效能限制適用於連接到任務的 Amazon EBS 磁碟區。您可能會暫時獲得比以下限制更高的 EBS 效能。不過，我們建議您根據這些限制規劃工作負載。

要求的 CPU 單元 (以 vCPUs 為單位)	基準 Amazon EBS IOPS (16 KiB I/O)	基準 Amazon EBS 輸送量 (輸入 MiBps，128 KiB I/O)	基準頻寬 (以 Mbps 為單位)
2	3,000	75	360
4	5,000	120	1,150
8	10,000	250	2,300
16	15,000	500	4,500

> **Note**
>
> 當您設定 Amazon EBS 磁碟區以附加到 Fargate 任務時，Fargate 任務的 Amazon EBS 效能限制會在任務的暫時儲存和連接的磁碟區之間共用。

對 Amazon ECS 任務的 Amazon EBS 磁碟區附件進行疑難排解

您可能需要對 Amazon EBS 磁碟區的附件進行故障排除，或驗證 Amazon ECS 任務的附件。

檢查磁碟區附件狀態

您可以使用 AWS Management Console 來檢視 Amazon EBS 磁碟區附件到 Amazon ECS 任務的狀態。如果工作開始且附件失敗，您也會看到可用來進行疑難排解的狀態原因。建立的磁碟區將會被刪除，且工作將會停止。如需狀態原因的詳細資訊，請參閱Amazon EBS 磁碟區附加到 Amazon ECS 任務的狀態原因。

使用主控台檢視磁碟區的附件狀態和狀態原因

1. 開啟主控台，網址為 https://console.aws.amazon.com/ecs/v2。
2. 在 [叢集] 頁面上，選擇執行工作的叢集。叢集的詳細資訊頁面隨即顯示。
3. 在叢集的詳細資訊頁面上，選擇 [工作] 索引標籤。

4. 選擇您要檢視其磁碟區附件狀態的工作。如果您要檢查的工作已停止，您可能需要使用 [篩選所需狀態]，並選擇 [已停止]。
5. 在工作的詳細資訊頁面上，選擇磁碟區索引標籤。您將能夠在附件狀態下看到 Amazon EBS 磁碟區的附件狀態。如果磁碟區無法連接至作業，您可以選擇 [附件狀態] 下的狀態，以顯示失敗的原因。

您也可以使用 DescribeTasksAPI 來檢視工作的磁碟區附件狀態和相關的狀態原因。

服務和工作失敗

您可能會遇到非 Amazon EBS 磁碟區特有的服務或任務失敗，這些故障可能會影響磁碟區附件。如需詳細資訊，請參閱

- 服務事件訊息
- 已停止的工作錯誤碼
- API 失敗的原因

Amazon EBS 磁碟區附加到 Amazon ECS 任務的狀態原因

使用下列參考來修正當您設定 Amazon EBS 磁碟區以附加至 Amazon ECS 任務 AWS Management Console 時，可能會以狀態原因形式遇到的問題。如需在主控台中尋找這些狀態原因的詳細資訊，請參閱檢查磁碟區附件狀態。

*ECS ######## ECS #######arn: aw: IAM:: 1111222333: ##/ ECS##InfrastructureRole*請確認傳遞的角色與 Amazon ECS 具有適當的信任關係

此狀態原因會出現在下列案例中。

- 您提供 IAM 角色，但不附加必要的信任政策。如果角色沒有必要的信任政策，Amazon ECS 將無法存取您提供的 Amazon ECS 基礎設施 IAM 角色。該任務可能會卡在DEPROVISIONING狀態。如需必要信任原則的詳細資訊，請參閱Amazon ECS 基礎設施 IAM 角色。
- 您的 IAM 使用者沒有將 Amazon ECS 基礎設施角色傳遞給 Amazon ECS 的權限。該任務可能會卡在DEPROVISIONING狀態。若要避免此問題，您可以將PassRole權限附加至您的使用者。如需詳細資訊，請參閱 Amazon ECS 基礎設施 IAM 角色。
- 您的 IAM 角色沒有 Amazon EBS 磁碟區附件的必要許可。該任務可能會卡在DEPROVISIONING狀態。如需將 Amazon EBS 磁碟區附加到任務所需的特定許可的詳細資訊，請參閱Amazon ECS 基礎設施 IAM 角色。

> **Note**
>
> 由於角色傳播延遲，您也可能會看到這個錯誤訊息。如果等待幾分鐘後重試使用該角色並未修正問題，表示您可能錯誤設定角色的信任原則。

ECS 無法設定 EBS 磁碟區。遇到 IdempotentParameterMismatch「;「您提供的客戶端令牌與已刪除的資源相關聯。請使用不同的客戶端令牌。」

下列 AWS KMS 重要案例可能會導致`IdempotentParameterMismatch`訊息出現：

- 您可以指定無效的 KMS 金鑰 ARN、識別碼或別名。在這個案例中，工作可能會看似成功啟動，但工作最終會失敗，因為以非同步方式 AWS 驗證 KMS 金鑰。如需詳細資訊，請參閱 Amazon EC2 使用者指南中的亞馬遜 EBS 加密。
- 您提供的客戶受管金鑰缺少允許 Amazon ECS 基礎設施 IAM 角色使用金鑰進行加密的許可。若要避免金鑰原則權限問題，請參閱 Amazon EBS 磁碟區的資料加密中的範例 AWS KMS 金鑰政策。

您可以設定 Amazon EventBridge 將 Amazon EBS 磁碟區事件和 Amazon ECS 任務狀態變更事件傳送至目標，例如 Amazon CloudWatch 群組。然後，您可以使用這些事件來識別受影響磁碟區附件的特定客戶管理金鑰相關問題。如需詳細資訊，請參閱

- 如何建立用作 EventBridge 規則目標的 CloudWatch 記錄群組？ 關於 AWS Re：帖子。
- 工作狀態變更事件。
- EventBridgeAmazon EBS 用戶指南中的 Amazon EBS。

設定 EBS 磁碟區附件至您的工作時，ECS 逾時。

下列檔案系統格式案例會產生此訊息。

- 您在設定期間指定的檔案系統格式與工作的作業系統不相容。
- 您將 Amazon EBS 磁碟區設定為從快照建立，且快照的檔案系統格式與任務的作業系統不相容。對於從快照建立的磁碟區，您必須指定與建立快照時磁碟區所使用的相同檔案系統類型。

您可以利用 Amazon ECS 容器代理程式日誌，針對 Amazon EC2 啟動類型任務對此訊息進行疑難排解。如需詳細資訊，請參閱 Amazon ECS 日誌檔案位置和 Amazon ECS 日誌收集器。

搭配 Amazon ECS 使用 Amazon EFS 磁碟區

Amazon Elastic File System (Amazon EFS) 提供簡單且可擴展的檔案儲存體，可與 Amazon ECS 任務搭配使用。利用 Amazon EFS，儲存容量即可有彈性。儲存容量會隨著您新增和移除檔案時自動擴展和縮減。您的應用程式可在需要時具備所需的儲存容量。

您可以搭配使用 Amazon EFS 檔案系統與 Amazon ECS，來匯出整個容器執行個體之機群的檔案系統資料。如此一來，無論您的任務位於哪個執行個體，都可存取相同的持久性儲存體。您的任務定義也必須參考容器執行個體上所掛載的磁碟區，才可使用檔案系統。

如需教學，請參閱使用主控台為 Amazon ECS 設定 Amazon EFS 檔案系統。

考量事項

使用 Amazon EFS 磁碟區時，請考量下列事項：

- 對於使用 EC2 啟動類型的任務，透過 Amazon ECS 最佳化 AMI 版本 20191212 與容器代理程式 1.35.0 版，Amazon EFS 檔案系統支援被新增為公開預覽。不過，Amazon EFS 檔案系統支援廣泛使用 Amazon ECS 最佳化 AMI 20200319 版與容器代理程式 1.38.0 版，這包含 Amazon EFS 存取點和 IAM 授權功能。建議您使用 Amazon ECS 最佳化 AMI 版本 20200319 或更新版本以使用這些功能。如需詳細資訊，請參閱 Amazon ECS 優化的 Linux AMI。

  > **Note**
  >
  > 如果您建立自己的 AMI，則必須使用容器代理程式 1.38.0 或更新版本、`ecs-init` 1.38.0-1 版或更新版本，並在 Amazon EC2 執行個體上執行下列命令以啟用 Amazon ECS 磁碟區外掛程式。這些命令取決於您是否使用 Amazon Linux 2 或 Amazon Linux 作為基礎映像。
  >
  > Amazon Linux 2
  >
  > ```
  > yum install amazon-efs-utils
  > systemctl enable --now amazon-ecs-volume-plugin
  > ```
  >
  > Amazon Linux
  >
  > ```
  > yum install amazon-efs-utils
  > sudo shutdown -r now
  > ```

- 對於託管於 Fargate 的任務，平台版本 1.4.0 或更新版本 (Linux) 上支援 Amazon EFS 檔案系統。如需詳細資訊，請參閱 適用於 Amazon ECS 的 Fargate Linux 平台版本。

- 在 Fargate 上託管的任務使用 Amazon EFS 磁碟區時，Fargate 會建立負責管理 Amazon EFS 磁碟區的監督容器。監督容器使用少量的任務記憶體。查詢任務中繼資料第 4 版端點時，可看見監督容器。此外，它在 CloudWatch 容器見解中也會顯示為容器名稱`aws-fargate-supervisor`。如需使用 Amazon EC2 啟動類型時的詳細資訊，請參閱Amazon ECS 任務中繼資料端點版本 4。如需使用 Fargate 啟動類型時的詳細資訊，請參閱適用於 Fargate 上任務的 Amazon ECS 任務中繼資料端點第 4 版。
- 外部執行個體不支援使用 Amazon EFS 磁碟區或指定 `EFSVolumeConfiguration`。
- 建議您將代理程式組態檔案中的 `ECS_ENGINE_TASK_CLEANUP_WAIT_DURATION` 參數，設定為小於預設值 (約 1 小時) 的值。此變更有助於防止 EFS 掛載憑證過期，並允許清除未在使用中的掛載。如需詳細資訊，請參閱Amazon ECS 容器代理程式組態。

使用 Amazon EFS 存取點

Amazon EFS 存取點是應用程式特定的 EFS 檔案系統進入點，以管理應用程式存取共用資料集。如需有關 Amazon EFS 存取點及如何控制對它們的存取的詳細資訊，請參閱《Amazon Elastic File System 使用者指南》中的使用 Amazon EFS 存取點。

存取點可以針對透過存取點提出的所有檔案系統要求，強制執行使用者身分 (包括使用者的 POSIX 群組)。存取點也可以針對檔案系統強制使用不同的根目錄。如此一來，用戶端只能存取指定目錄或其子目錄中的資料。

> **Note**
>
> 建立 EFS 存取點時，您可以在檔案系統上指定要做為根目錄的路徑。在 Amazon ECS 任務定義中使用存取點 ID 參考 EFS 檔案系統時，必須忽略根目錄或將其設定為 `/`，這會強制執行 EFS 存取點上設定的路徑。

您可以使用 Amazon ECS 任務 IAM 角色，以強制執行特定應用程式使用特定存取點。透過結合 IAM 政策與存取點，您可以為應用程式提供特定資料集的安全存取。如需如何使用任務 IAM 角色的詳細資訊，請參閱Amazon ECS 任務 IAM 角色。

搭配 Amazon ECS 使用 Amazon EFS 磁碟區的最佳實務

將 Amazon EFS 與 Amazon ECS 搭配使用時，請注意下列最佳實務建議。

Amazon EFS 磁碟區的安全性和存取控制

Amazon EFS 提供存取控制功能，您可以使用這些功能來確保存放在 Amazon EFS 檔案系統中的資料安全無虞，而且只能從需要的應用程式存取。您可以透過啟用靜態和傳輸中的加密來保護資料。如需詳細資訊，請參閱 Amazon Elastic File System 使用者指南中的 Amazon EFS 的資料加密。

除了資料加密之外，您還可以使用 Amazon EFS 限制對檔案系統的存取。有三種方式可以在 EFS 中實作存取控制。

- 安全群組 — 使用 Amazon EFS 掛載目標，您可以設定用於允許和拒絕網路流量的安全群組。您可以設定連接到 Amazon EFS 的安全群組，以允許來自連接至 Amazon ECS 執行個體的安全群組的 NFS 流量 (連接埠 2049)，或在使用`awsvpc`網路模式時執行 Amazon ECS 任務。
- IAM — 您可以使用 IAM 限制對 Amazon EFS 檔案系統的存取。設定完成後，Amazon ECS 任務需要 IAM 角色才能存取檔案系統，才能掛接 EFS 檔案系統。如需詳細資訊，請參閱 Amazon 彈性檔案系統使用者指南中的使用 IAM 控制檔案系統資料存取。

  IAM 政策也可以強制執行預先定義的條件，例如要求用戶端在連線至 Amazon EFS 檔案系統時使用 TLS。如需詳細資訊，請參閱 Amazon 彈性檔案系統使用者指南中的用戶端適用的 Amazon EFS 條件金鑰。
- Amazon EFS 存取點 — Amazon EFS 存取點是 Amazon EFS 檔案系統中應用程式特定的進入點。您可以針對透過存取點發出的所有檔案系統要求，使用存取點強制執行使用者身分，包括使用者的 POSIX 群組。存取點也可以針對檔案系統強制使用不同的根目錄。這樣客戶端只能訪問指定目錄或其子目錄中的數據。

考慮在 Amazon EFS 檔案系統上實作所有三種存取控制，以獲得最大的安全性。例如，您可以將連接到 Amazon EFS 掛載點的安全群組設定為僅允許來自與您的容器執行個體或 Amazon ECS 任務相關聯的安全群組輸入 NFS 流量。此外，您可以將 Amazon EFS 設定為要求 IAM 角色存取檔案系統，即使連線來自允許的安全群組也一樣。最後，您可以使用 Amazon EFS 存取點強制執行 POSIX 使用者許可，並為應用程式指定根目錄。

下列任務定義程式碼片段顯示如何使用存取點掛接 Amazon EFS 檔案系統。

```
"volumes": [
    {
      "efsVolumeConfiguration": {
        "fileSystemId": "fs-1234",
        "authorizationConfig": {
          "acessPointId": "fsap-1234",
          "iam": "ENABLED"
```

```
        },
        "transitEncryption": "ENABLED",
        "rootDirectory": ""
      },
      "name": "my-filesystem"
    }
]
```

Amazon EFS 磁碟區的效能

Amazon EFS 提供兩種效能模式：一般用途和最大 I/O。一般用途適用於對延遲敏感的應用程式，例如內容管理系統和 CI/CD 工具。相較之下，Max I/O 檔案系統適用於資料分析、媒體處理和機器學習等工作負載。這些工作負載需要從數百個甚至數千個容器執行 parallel 作業，並且需要盡可能高的彙總輸送量和 IOPS。如需詳細資訊，請參閱 Amazon 彈性檔案系統使用者指南中的 Amazon EFS 效能模式。

某些延遲敏感的工作負載需要最大 I/O 效能模式所提供的較高 I/O 層級，以及一般用途效能模式所提供的較低延遲。對於這類工作負載，我們建議建立多個一般用途效能模式的檔案系統。如此一來，只要工作負載和應用程式能夠支援，您就可以將應用程式工作負載分散到所有這些檔案系統。

Amazon EFS 磁碟區輸送量

所有 Amazon EFS 檔案系統都有相關聯的計量輸送量，由使用佈建輸送量的檔案系統佈建輸送量量或使用大量輸送量的檔案系統存放在 EFS 標準或一個區域儲存類別中的資料量來決定。如需詳細資訊，請參閱 Amazon Elastic File System 使用者指南中的了解計量輸送量。

Amazon EFS 檔案系統的預設輸送量模式為爆量模式。使用成組分解模式時，檔案系統可用的輸送量會隨著檔案系統的成長而擴展或擴展。由於以檔案為基礎的工作負載通常會激增，需要一段時間的高輸送量，其餘時間需要較低的輸送量層級，因此 Amazon EFS 的設計可讓您在一段時間內達到高輸送量層級。此外，由於許多工作負載需要大量讀取，因此讀取作業的計量比例與其他 NFS 作業 (例如寫入) 的比例為 1:3。

所有 Amazon EFS 檔案系統可為每 TB 的 Amazon EFS 標準或 Amazon EFS 單區儲存提供每秒 50 MB 的一致基準效能。所有檔案系統 (無論大小為何) 都可以突發到 100 MB/s。具有超過 1TB EFS 標準或 EFS 單區域儲存空間的檔案系統，每 TB 可大幅提升至每秒 100 MB。由於讀取操作是以 1:3 的比率計量，因此每個 TiB 的讀取輸送量最多可驅動 300 MiBs /s。將資料新增到檔案系統時，檔案系統可用的最大輸送量會隨著 Amazon EFS 標準儲存類別中的儲存線性和自動擴展。如果您需要的輸送量超過儲存資料量所能達到的輸送量，則可以將佈建輸送量設定為工作負載所需的特定量。

檔案系統輸送量會在連接到檔案系統的所有 Amazon EC2 執行個體之間共用。例如，輸送量高達每秒 100 MB 的 1TB 檔案系統可以從單一 Amazon EC2 執行個體驅動每秒 100 MB 的磁碟機，每秒可以驅動每秒 10 MB。如需詳細資訊，請參閱 Amazon 彈性檔案系統使用者指南中的 Amazon EFS 效能。

優化 Amazon EFS 磁碟區的成本

Amazon EFS 可為您簡化擴展儲存的規模。Amazon EFS 檔案系統會隨著您新增更多資料而自動成長。特別是使用 Amazon EFS 爆量輸送量模式時，Amazon EFS 上的輸送量會隨著標準儲存類別中檔案系統的大小增加而擴展。若要改善輸送量，而不需要為 EFS 檔案系統上的佈建輸送量支付額外費用，您可以與多個應用程式共用 Amazon EFS 檔案系統。使用 Amazon EFS 存取點，您可以在共用的 Amazon EFS 檔案系統中實作儲存隔離。通過這樣做，即使應用程序仍然共享相同的文件系統，他們無法訪問數據，除非您授權它。

隨著資料成長，Amazon EFS 可協助您將不常存取的檔案自動移至較低的儲存類別。Amazon EFS 標準不常存取 (IA) 儲存類別可降低未每天存取檔案的儲存成本。它可以在不犧牲 Amazon EFS 提供的高可用性、高耐用性、彈性和 POSIX 檔案系統存取權的情況下完成這項工作。如需詳細資訊，請參閱 Amazon 彈性檔案系統使用者指南中的 Amazon EFS 儲存類別。

請考慮使用 Amazon EFS 生命週期政策，將不常存取的檔案移至 Amazon EFS IA 儲存體，以自動節省成本。如需詳細資訊，請參閱《Amazon Elastic File System 使用者指南》中的 Amazon EFS 生命週期管理。

建立 Amazon EFS 檔案系統時，您可以選擇 Amazon EFS 是跨多個可用區域 (標準) 複寫資料，還是以冗餘方式將資料存放在單一可用區域內。相較於 Amazon EFS 標準儲存類別，Amazon EFS 單區儲存類別可大幅降低儲存成本。對於不需要異地同步備份恢復的工作負載，請考慮使用 Amazon EFS 單區儲存類別。您可以將不常存取的檔案移至 Amazon EFS 單區域不常存取，進一步降低 Amazon EFS 單區域儲存的成本。如需詳細資訊，請參閱 Amazon EFS 不頻繁存取。

Amazon EFS 大量資料保護

Amazon EFS 會以冗餘方式將您的資料存放在使用標準儲存類別的檔案系統的多個可用區域。如果您選取 Amazon EFS 單一區域儲存類別，您的資料會以冗餘方式存放在單一可用區域中。此外，Amazon EFS 的設計可在一年內提供 99.999999999% (11 9 年代) 的耐用性。

與任何環境一樣，最佳做法是進行備份，並建立防止意外刪除的保護措施。對於 Amazon EFS 資料，最佳實務包括使用 AWS Backup. 使用 Amazon EFS One Zone 區域儲存類別的檔案系統會設定為在建立檔案系統時預設自動備份檔案，除非您選擇停用此功能。如需詳細資訊，請參閱 Amazon 彈性檔案系統使用者指南中的 Amazon EFS 的資料保護。

在 Amazon ECS 任務定義中指定 Amazon EFS 檔案系統

若要為您的容器使用 Amazon EFS 檔案系統磁碟區，您必須在任務定義中指定磁碟區並掛載點組態。以下任務定義 JSON 片段說明容器 `volumes` 和 `mountPoints` 物件的語法。

```
{
    "containerDefinitions": [
        {
            "name": "container-using-efs",
            "image": "amazonlinux:2",
            "entryPoint": [
                "sh",
                "-c"
            ],
            "command": [
                "ls -la /mount/efs"
            ],
            "mountPoints": [
                {
                    "sourceVolume": "myEfsVolume",
                    "containerPath": "/mount/efs",
                    "readOnly": true
                }
            ]
        }
    ],
    "volumes": [
        {
            "name": "myEfsVolume",
            "efsVolumeConfiguration": {
                "fileSystemId": "fs-1234",
                "rootDirectory": "/path/to/my/data",
                "transitEncryption": "ENABLED",
                "transitEncryptionPort": integer,
                "authorizationConfig": {
                    "accessPointId": "fsap-1234",
                    "iam": "ENABLED"
                }
            }
        }
    ]
}
```

`efsVolumeConfiguration`

類型：物件

必要：否

只有使用 Amazon EFS 磁碟區時才會指定此參數。

`fileSystemId`

類型：字串

必要：是

要使用的 Amazon EFS 檔案系統識別碼。

`rootDirectory`

類型：字串

必要：否

在 Amazon EFS 檔案系統中的目錄，其將掛載作為主機內的根目錄。如果省略此參數，使用 Amazon EFS 磁碟區的根目錄。指定 `/` 的效果與忽略此參數的效果相同。

> **Important**
>
> 如果在 `authorizationConfig` 中指定 EFS 存取點，則必須忽略根目錄參數或設定為 `/`，這將強制執行 EFS 存取點上設定的路徑。

`transitEncryption`

類型：字串

有效值：`ENABLED | DISABLED`

必要：否

指定是否要對在 Amazon ECS 主機和 Amazon EFS 伺服器之間傳輸中的 Amazon EFS 資料啟用加密功能。如果使用 Amazon EFS IAM 授權，則必須啟用傳輸加密。如果省略此參數，系統會使用 `DISABLED` 的預設值。如需詳細資訊，請參閱《Amazon Elastic File System 使用者指南》中的加密傳輸中的資料。

`transitEncryptionPort`

類型：整數

必要：否

在 Amazon ECS 主機和 Amazon EFS 伺服器之間傳送加密資料時所使用的連接埠。如果您未指定傳輸加密連接埠，它會使用 Amazon EFS 掛載協助程式使用的連接埠選擇策略。如需詳細資訊，請參閱《Amazon Elastic File System 使用者指南》中的 EFS 掛載協助程式。

`authorizationConfig`

類型：物件

必要：否

Amazon EFS 檔案系統的授權組態詳細資訊。

`accessPointId`

類型：字串

必要：否

要使用的存取點 ID。如果指定了存取點，則必須省略 `efsVolumeConfiguration` 中的根目錄值，或設定為 `/`，這將強制執行在 EFS 存取點上設定的路徑。如果使用存取點，則必須在 `EFSVolumeConfiguration` 中啟用傳輸加密。如需詳細資訊，請參閱《Amazon Elastic File System 使用者指南》中的使用 Amazon EFS 存取點。

`iam`

類型：字串

有效值：`ENABLED` | `DISABLED`

必要：否

掛載 Amazon ECS 檔案系統時，指定是否使用任務定義中定義的 Amazon EFS 任務 IAM 角色。如果已啟用，必須在 `EFSVolumeConfiguration` 中啟用傳輸加密。如果省略此參數，系統會使用 `DISABLED` 的預設值。如需詳細資訊，請參閱任務的 IAM 角色。

使用主控台為 Amazon ECS 設定 Amazon EFS 檔案系統

瞭解如何將 Amazon Elastic File System (Amazon EFS) 檔案系統與 Amazon ECS 搭配使用。

步驟 1：建立 Amazon ECS 叢集。

使用下列步驟來建立 Amazon ECS 叢集。

建立新叢集 (Amazon ECS 主控台)

在開始之前，請指派適當的 IAM 許可。如需詳細資訊，請參閱 the section called “Amazon ECS 集群示例”。

1. 開啟主控台，網址為 https://console.aws.amazon.com/ecs/v2。
2. 從導覽列中選取要使用的「區域」。
3. 在導覽窗格中，選擇叢集。
4. 在 Clusters (叢集) 頁面上，選擇 Create cluster (建立叢集)。
5. 在叢集組態下的叢集名稱中，輸入叢集名稱的 `EFS-tutorial`。
6. (選用) 若要變更任務和服務啟動所在的 VPC 和子網路，請在 Networking (聯網) 下，執行下列任一操作：
   - 若要移除子網路，請在 Subnets (子網路) 下，對您要移除之每一個子網路選擇 X。
   - 若要變更為非 default (預設) VPC，請在 VPC 下，選擇現有的 VPC，然後在 Subnets (子網路) 下選擇各個子網路。
7. 若要將 Amazon EC2 執行個體新增至叢集，展開 Infrastructure (基礎設施)，然後選取 Amazon EC2 執行個體。接下來，設定作為容量提供者的 Auto Scaling 群組：
   - 若要建立 Auto Scaling 群組，請從 Auto Scaling group (ASG) (Auto Scaling 群組 (ASG)) 中選取 Create new group (建立新群組)，然後提供有關該群組的下列詳細資訊：
     - 在作業系統/架構中，選擇 Amazon Linux 2。
     - 對於 EC2 instance type (EC2 執行個體類型)，選擇 `t2.micro`。

       對於 SSH key pair (SSH 金鑰對)，選擇在連線到執行個體時證明您身分的金鑰對。
     - 在容量中，輸入 1。
8. 選擇建立。

步驟 2：為 Amazon EC2 執行個體和 Amazon EFS 檔案系統建立安全群組

在此步驟中，您將為 Amazon EC2 執行個體建立安全群組，以允許連接埠 80 上的入站網路流量，並為 Amazon EFS 檔案系統建立一個允許從容器執行個體進行入站存取的安全群組。

使用下列選項為 Amazon EC2 執行個體建立安全群組：

- 安全群組名稱 - 為您的安全群組輸入唯一的名稱。
- VPC - 選擇您先前為叢集所確認的 VPC。
- 傳入規則
  - 類型 - HTTP
  - 資源來源 - 0.0.0.0/0。

使用下列選項為 Amazon EFS 檔案系統建立安全群組：

- 安全群組名稱 - 為您的安全群組輸入唯一的名稱。例如 `EFS-access-for-sg-dc025fa2`。
- VPC - 選擇您先前為叢集所確認的 VPC。
- 傳入規則
  - 類型 - NFS
  - 來源 - 使用您為執行個體建立的安全群組 ID 進行自訂。

如需如何建立安全群組的詳細資訊，請參閱 Amazon EC2 使用者指南中的建立安全群組。

步驟 3：建立 Amazon EFS 檔案系統

在此步驟中，您將建立一個 Amazon EFS 檔案系統。

建立 Amazon ECS 任務的 Amazon EFS 檔案系統。

1. 開啟 Amazon Elastic File System 主控台，網址為 https://console.aws.amazon.com/efs/。
2. 選擇 Create file system (建立檔案系統)。
3. 輸入檔案系統的名稱，然後選擇容器執行個體所託管的 VPC。根據預設，指定之 VPC 中的每個子網路，都會收到一個使用該 VPC 之預設安全群組的掛載目標。然後，選擇自訂。

   > **Note**
   >
   > 本教學假設您的 Amazon EFS 檔案系統、Amazon ECS 叢集、容器執行個體和任務都位於同一個 VPC 中。如需有關從不同 VPC 掛接檔案系統的詳細資訊，請參閱 Amazon EFS 使用者指南中的逐步解說：從不同 VPC 掛接檔案系統。

4. 在檔案系統設定頁面上，設定選用設定，然後在效能設定下，為您的檔案系統選擇高載輸送量模式。設定完設定之後，請選取下一步。

a. (選用) 為您的檔案系統新增標籤。例如，您可以在 Name (名稱) 鍵旁的 Value (值) 欄中輸入名稱，藉此為檔案系統指定不會重複的一名稱。

b. (選用) 啟用生命週期管理，以避免在不常存取的儲存體上浪費成本。如需詳細資訊，請參閱《Amazon Elastic File System 使用者指南》中的 EFS 生命週期管理。

c. (選用) 啟用加密。選取此核取方塊以啟用靜態 Amazon EFS 檔案系統的加密。

5. 在網路存取頁面的掛載目標下，將每個可用區域的現有安全群組組態替換為您為 步驟 2：為 Amazon EC2 執行個體和 Amazon EFS 檔案系統建立安全群組 中的檔案系統建立的安全群組，然後選擇下一步。
6. 您不需要為本教學課程設定檔案系統政策，因此您可以選擇下一步略過此區段。
7. 檢閱您的檔案系統選項，然後選擇建立以完成程序。
8. 從檔案系統畫面中，記錄檔案系統 ID。在下一個步驟中，您將會在 Amazon ECS 任務定義中參考這個值。

步驟 4：將內容新增至 Amazon EFS 檔案系統

在此步驟中，您會將 Amazon EFS 檔案系統掛載到 Amazon EC2 執行個體，並將內容新增到其中。這是為了在本教學課程中進行測試，以說明資料的持久性。使用此功能時，通常會將應用程式或其他方法的資料寫入 Amazon EFS 檔案系統。

建立 Amazon EC2 執行個體並掛載 Amazon EFS 檔案系統

1. 在 https://console.aws.amazon.com/ec2/ 開啟 Amazon EC2 主控台。
2. 選擇 Launch Instance (啟動執行個體)。
3. 在應用程式和作業系統映像 (Amazon Machine Image) 下，選取 Amazon Linux 2 AMI (HVM)。
4. 在執行個體類型中，保留預設執行個體類型 `t2.micro`。
5. 在金鑰對 (登入) 下，選取用於 SSH 存取執行個體的金鑰對。
6. 在網路設定下，選取為 Amazon EFS 檔案系統和 Amazon ECS 叢集指定的 VPC。選取子網路和在 步驟 2：為 Amazon EC2 執行個體和 Amazon EFS 檔案系統建立安全群組 中建立的執行個體安全群組。設定執行個體的安全群組。確認已啟用自動指派公有 IP。
7. 在設定儲存下，選擇檔案系統的編輯按鈕，然後選擇 EFS。選取您在 步驟 3：建立 Amazon EFS 檔案系統 中建立的檔案系統。您可以選擇性地變更掛載點，或保留預設值。

> **Important**
>
> 您必須先選擇子網路，然後才能將檔案系統新增至執行個體。

8. 清除自動建立並連接安全群組。保持選取另一個核取方塊。選擇 Add shared file system (新增共用檔案系統)。
9. 在 Advanced Details (進階詳細資訊) 下，確保透過 Amazon EFS 檔案系統掛載步驟自動填入使用者資料指令碼。
10. 在摘要下，確定執行個體數量為 1。選擇啟動執行個體。
11. 在啟動執行個體頁面上，選擇檢視所有執行個體以查看執行個體的狀態。初始時，您的執行個體狀態為 PENDING。當狀態變更為 RUNNING 且執行個體通過所有狀態檢查之後，執行個體即可供使用。

現在，您可以連線到 Amazon EC2 執行個體，並將內容新增至 Amazon EFS 檔案系統。

連線到 Amazon EC2 執行個體，並將內容新增至 Amazon EFS 檔案系統。

1. 您建立之 Amazon EC2 執行個體的 SSH。如需詳細資訊，請參閱 Amazon EC2 使用者指南中的 Connect 到 Linux 執行個體。
2. 從終端機視窗，執行 df -T 命令，確認 Amazon EFS 檔案系統已掛載。在下列輸出中，我們已反白顯示 Amazon EFS 檔案系統掛載。

```
$ df -T
Filesystem     Type            1K-blocks    Used        Available Use% Mounted on
devtmpfs       devtmpfs           485468       0           485468   0% /dev
tmpfs          tmpfs              503480       0           503480   0% /dev/shm
tmpfs          tmpfs              503480     424           503056   1% /run
tmpfs          tmpfs              503480       0           503480   0% /sys/fs/
cgroup
/dev/xvda1     xfs               8376300 1310952          7065348  16% /
127.0.0.1:/    nfs4     9007199254739968       0 9007199254739968   0% /mnt/efs/fs1
tmpfs          tmpfs              100700       0           100700   0% /run/
user/1000
```

3. 導覽至掛載 Amazon EFS 檔案系統所在的目錄。在上述範例中，即為 /mnt/efs/fs1。
4. 建立名為 index.html 且具有下列內容的檔案：

```
<html>
```

```
    <body>
        <h1>It Works!</h1>
        <p>You are using an Amazon EFS file system for persistent container
 storage.</p>
    </body>
</html>
```

步驟 5：建立任務定義

以下任務定義會建立名稱為 `efs-html` 的資料磁碟區。`nginx` 容器會將主機資料磁碟區掛載於 NGINX 根目錄 `/usr/share/nginx/html`。

使用 Amazon ECS 主控台建立新的任務定義

1. 開啟主控台，網址為 https://console.aws.amazon.com/ecs/v2。
2. 在導覽窗格中，選擇 Task Definitions (任務定義)。
3. 選擇 Create new task definitio (建立新任務定義)、Create new task definition with JSON (使用 JSON 建立新的任務定義)。
4. 在 JSON 編輯器方塊中，複製並貼上以下 JSON 文字，以 Amazon EFS 檔案系統的 ID 取代 `fileSystemId`。

```
{
    "containerDefinitions": [
        {
            "memory": 128,
            "portMappings": [
                {
                    "hostPort": 80,
                    "containerPort": 80,
                    "protocol": "tcp"
                }
            ],
            "essential": true,
            "mountPoints": [
                {
                    "containerPath": "/usr/share/nginx/html",
                    "sourceVolume": "efs-html"
                }
            ],
            "name": "nginx",
            "image": "nginx"
```

```
        }
    ],
    "volumes": [
        {
            "name": "efs-html",
            "efsVolumeConfiguration": {
                "fileSystemId": "fs-1324abcd",
                "transitEncryption": "ENABLED"
            }
        }
    ],
    "family": "efs-tutorial",
    "executionRoleArn":"arn:aws::iam::111122223333:role/ecsTaskExecutionRole"
}
```

> **Note**
>
> 您可以將下列許可新增至 Amazon ECS 任務執行 IAM 角色，以允許 Amazon ECS 代理程式在啟動時找到 Amazon EFS 檔案系統並將其掛接到任務。
>
> - `elasticfilesystem:ClientMount`
> - `elasticfilesystem:ClientWrite`
> - `elasticfilesystem:DescribeMountTargets`
> - `elasticfilesystem:DescribeFileSystems`

5. 選擇建立。

步驟 6：執行任務並檢視結果

現在，已建立您的 Amazon EFS 檔案系統，並且提供 NGINX 容器的 Web 內容，您可以使用建立的任務定義來執行任務。NGINX Web 伺服器會隨即提供您簡單的 HTML 頁面。如果您在 Amazon EFS 檔案系統中更新了內容，這些變更也會傳播至所有也掛載該檔案系統的容器。

任務會在您為叢集定義的子網路中執行。

使用主控台執行任務並檢視結果

1. 開啟主控台，網址為 https://console.aws.amazon.com/ecs/v2。
2. 在 Clusters (叢集) 頁面上，選取要在哪個叢集中執行獨立任務。

決定您要從中啟動服務的資源。

啟動服務的資源	步驟
叢集	a. 在叢集頁面，選取您要在哪個叢集中建立服務。 b. 在 Tasks (任務) 標籤中，選擇 Run new task (執行新任務)。
啟動類型	a. 在 Task (任務) 頁面中，選擇任務定義。 b. 如果有多個修訂版，則請選取修訂版。 c. 依序選擇 Create (建立)、Run task (執行任務)。

3. (選用) 選擇已排程的任務在叢集基礎設施中的分佈方式。展開 Compute configuration (運算組態)，然後執行下列操作：

分發方法	步驟
啟動類型	a. 在運算選項區段中，選取啟動類型。 b. 在啟動類型中，選擇 EC2。

4. 針對 Application type (應用程式類型)，選擇 Task (任務)。
5. 在任務定義中，選擇您先前建立的 `efs-tutorial` 任務定義。
6. 在所需任務中，輸入 1。
7. 選擇建立。
8. 在叢集頁面上，選擇基礎設施。
9. 在容器執行個體下，選擇要連線的容器執行個體。
10. 在容器執行個體頁面的聯網下，記錄執行個體的公有 IP 或公有 DNS。

11. 開啟瀏覽器並輸入公有 IP 地址。您應該看到下列訊息：

```
It works!
You are using an Amazon EFS file system for persistent container storage.
```

> **Note**
>
> 如果您未看到該訊息，請確認您容器執行個體的安全群組，允許連接埠 80 上的入站網路流量，且檔案系統的安全群組允許來自容器執行個體的入站存取。

將 FSx 用於 FSx for Windows File Server 磁碟區與 Amazon ECS

FSx for Windows File Server 提供完全受管理的 Windows 檔案伺服器，這些伺服器由 Windows 檔案系統支援。搭配使用 FSx for Windows File Server 與 ECS 時，您可以使用持續、分散、共用、靜態的檔案儲存來佈建 Windows 任務。如需詳細資訊，請參閱什麼是 FSx for Windows File Server？。

> **Note**
>
> 使用 Amazon ECS 最佳化 Windows Server 2016 Full AMI 的 EC2 執行個體不支援 FSx for Windows File Server ECS 任務磁碟區。
>
> 您無法在 Fargate 組態的 Windows 容器中使用適用於 Windows 檔案伺服器磁碟區的 FSx。相反，您可以修改容器以在啟動時裝載它們。

您可以使用 FSx for Windows File Server 來部署需要存取共用外部儲存體、高可用性區域儲存體或高輸送量儲存體的 Windows 工作負載。您可以將一或多個 FSx for Windows File Server 檔案系統磁碟區的 FSx 掛接到在 Amazon ECS 視窗執行個體上執行的 Amazon ECS 容器。您可以在單一 Amazon ECS 任務中，在多個 Amazon ECS 容器之間共用適用於 Windows 檔案伺服器檔案系統磁碟區的 FSx。

若要讓 FSx for Windows File Server 與 ECS 搭配使用，在任務定義中包含 FSx for Windows File Server 檔案系統 ID 和相關資訊。如下列任務定義 JSON 片段範例。在建立並執行任務定義之前，您必須準備好以下事項。

- 加入有效的網域的 ECS Windows EC2 執行個體。它可以由現場部署活動目錄或在 Amazon EC2 上自託管的活動目錄託管。AWS Directory Service for Microsoft Active Directory

- AWS Secrets Manager 秘密或 Systems Manager 參數，其中包含用來加入作用中目錄網域並附加 Windows 檔案伺服器檔案系統 FSx 的認證。憑證值是您在建立 Active Directory 時輸入的名稱和密碼憑證。

如需相關教學，請參閱了解如何設定適用於 FSx for Windows File Server 檔案系 Amazon 的 FSx。

考量事項

在使用 FSx for Windows File Server 磁碟區時，請考量下列事項：

- FSx for Windows File Server 與 Amazon ECS 僅支援 Windows Amazon EC2 執行個體。不支援 Linux Amazon EC2 執行個體。
- FSx for Windows File Server 與 Amazon ECS 不支援 AWS Fargate。
- 具有 `awsvpc` 網路模式的 FSx for Windows File Server 與 Amazon ECS 需要容器代理程式的版本 `1.54.0` 或更新版本。
- Amazon ECS 任務使用的磁碟機代號的上限為 23。具有 FSx for Windows File Server 磁碟區的每個任務都會取得指派給它的磁碟機代號。
- 根據預設，任務資源清除時間為任務結束後的 3 小時。即使沒有任務正在使用它，任務建立的檔案映射會持續 3 小時。使用 Amazon ECS 環境變數 `ECS_ENGINE_TASK_CLEANUP_WAIT_DURATION` 可設定預設清理時間。如需詳細資訊，請參閱 Amazon ECS 容器代理程式組態。
- 任務通常只會在與 FSx for Windows File Server 檔案系統相同的 VPC 中執行。但是，如果 Amazon ECS 叢集 VPC 與 FSx for Windows File Server 檔案系統之間透過 VPC 對等互連建立了網路連線，則可以跨 VPC 支援。
- 透過設定 VPC 安全群組，可在網路層級控制對 FSx for Windows File Server 檔案系統的存取。只有在已加入 Active Directory 網域且已正確設定作用中目錄安全群組的 EC2 執行個體上託管的工作，才能存取 Windows 檔案伺服器檔案共用的 FSx。如果安全群組設定錯誤，Amazon ECS 無法啟動任務，並顯示下列錯誤訊息：`unable to mount file system fs-id`」
- FSx for Windows File Server 與 AWS Identity and Access Management (IAM) 整合，以控制 IAM 使用者和群組可對 Windows 檔案伺服器資源的特定 FSx 執行的動作。透過用戶端授權，客戶可以定義允許或拒絕存取特定 FSx for Windows File Server 檔案系統的 IAM 角色，選擇性地要求唯讀存取，以及選擇性地允許或禁止從用戶端對檔案系統進行根存取。如需詳細資訊，請參閱《Amazon FSx Windows 使用者指南》中的安全性。

將 FSx FSx for Windows File Server 與 Amazon ECS 搭配使用的最佳實務

當您將 FSx FSx for Windows File Server 與 Amazon ECS 搭配使用時，請注意下列最佳實務建議。

適用於 Windows 檔案伺服器 FSx 的安全性和存取控制

FSx for Windows File Server 提供下列存取控制功能，您可以使用這些功能來確保儲存在 FSx 中的 Windows 檔案伺服器檔案系統中的資料是安全的，而且只能從需要它的應用程式存取。

適用於 Windows 檔案伺服器磁碟區的 FSx 資料加密

FSx for Windows File Server 的檔案系統支援兩種加密形式。它們是傳輸中的數據和靜態加密的加密。對應於支援 SMB 通訊協定 3.0 或更新版本的容器執行個體上的檔案共用，支援傳輸中的資料加密。建立 Amazon FSx 檔案系統時，系統會自動啟用靜態資料加密。當您存取檔案系統時，Amazon FSx 會使用 SMB 加密自動加密傳輸中的資料，而無需修改應用程式。如需詳細資訊，請參閱 Amazon FSx 適用於 Windows 檔案伺服器使用者指南的 Amazon FSx 中的資料加密。

使用視窗 ACL 進行資料夾層級的存取控制

視窗 Amazon EC2 執行個體使用活動目錄登入資料存取亞馬遜 FSx 檔案共用。它使用標準的 Windows 存取控制清單 (ACL) 進行精細的檔案層級和資料夾層級存取控制。您可以建立多個認證，每個認證都用於共用中的特定資料夾，以對應至特定工作。

在下列範例中，工作可以App01使用 Secrets Manager 中儲存的認證存取資料夾。它的 Amazon 資源名稱（ARN）是1234.

```
"rootDirectory": "\\path\\to\\my\\data\App01",
"credentialsParameter": "arn-1234",
"domain": "corp.fullyqualified.com",
```

在另一個範例中，工作可以App02使用 Secrets Manager 中儲存的認證存取資料夾。其 ARN 是 6789。

```
"rootDirectory": "\\path\\to\\my\\data\App02",
"credentialsParameter": "arn-6789",
"domain": "corp.fullyqualified.com",
```

在 Amazon ECS 任務定義中指定 FSx for Windows File Server 檔案系統的 FSx

若要將 FSx for Windows File Server 檔案系統磁碟區用於您的容器，請在任務定義中指定磁碟區並掛載點組態。以下任務定義 JSON 片段說明容器 `volumes` 和 `mountPoints` 物件的語法。

```
{
    "containerDefinitions": [
        {
            "entryPoint": [
                "powershell",
                "-Command"
            ],
            "portMappings": [],
            "command": ["New-Item -Path C:\\fsx-windows-dir\\index.html -ItemType file
 -Value '<html> <head> <title>Amazon ECS Sample App</title> <style>body {margin-top:
 40px; background-color: #333;} </style> </head><body> <div style=color:white;text-
align:center> <h1>Amazon ECS Sample App</h1> <h2>It Works!</h2> <p>You are using Amazon
 FSx for Windows File Server file system for persistent container storage.</p>' -
Force"],
            "cpu": 512,
            "memory": 256,
            "image": "mcr.microsoft.com/windows/servercore/iis:windowsservercore-
ltsc2019",
            "essential": false,
            "name": "container1",
            "mountPoints": [
                {
                    "sourceVolume": "fsx-windows-dir",
                    "containerPath": "C:\\fsx-windows-dir",
                    "readOnly": false
                }
            ]
        },
        {
            "entryPoint": [
                "powershell",
                "-Command"
            ],
            "portMappings": [
                {
                    "hostPort": 443,
                    "protocol": "tcp",
                    "containerPort": 80
                }
            ],
            "command": ["Remove-Item -Recurse C:\\inetpub\\wwwroot\\* -Force; Start-
Sleep -Seconds 120; Move-Item -Path C:\\fsx-windows-dir\\index.html -Destination C:\
\inetpub\\wwwroot\\index.html -Force; C:\\ServiceMonitor.exe w3svc"],
```

```
            "mountPoints": [
                {
                    "sourceVolume": "fsx-windows-dir",
                    "containerPath": "C:\\fsx-windows-dir",
                    "readOnly": false
                }
            ],
            "cpu": 512,
            "memory": 256,
            "image": "mcr.microsoft.com/windows/servercore/iis:windowsservercore-
ltsc2019",
            "essential": true,
            "name": "container2"
        }
    ],
    "family": "fsx-windows",
    "executionRoleArn": "arn:aws:iam::111122223333:role/ecsTaskExecutionRole",
    "volumes": [
        {
            "name": "fsx-windows-dir",
            "fsxWindowsFileServerVolumeConfiguration": {
                "fileSystemId": "fs-0eeb5730b2EXAMPLE",
                "authorizationConfig": {
                    "domain": "example.com",
                    "credentialsParameter": "arn:arn-1234"
                },
                "rootDirectory": "share"
            }
        }
    ]
}
```

`FSxWindowsFileServerVolumeConfiguration`

類型：物件

必要：否

當您為任務儲存使用 FSx for Windows File Server 檔案系統時，指定此參數。

`fileSystemId`

類型：字串

必要：是

要使用的 FSx for Windows File Server 檔案系統 ID。

`rootDirectory`

類型：字串

必要：是

FSx for Windows File Server 檔案系統中的目錄，其將掛載做為主機內的根目錄。

`authorizationConfig`

`credentialsParameter`

類型：字串

必要：是

授權憑證選項：

- Secrets Manager 秘密的 Amazon Resource Name (ARN)。
- Systems Manager 參數的 Amazon Resource Name (ARN)。

`domain`

類型：字串

必要：是

由 AWS Directory Service for Microsoft Active Directory(AWS Managed Microsoft AD) 目錄或自我託管的 EC2 活動目錄託管的完整網域名稱。

儲存 Windows 檔案伺服器磁碟區認證 FSx 的方法

有兩種不同的方法可存放憑證，以便與憑證參數搭配使用。

- AWS Secrets Manager 秘密

  您可以使用「其他類型的密碼」類別，在 AWS Secrets Manager 主控台中建立此認證。您可以為每個鍵/值對、使用者名稱/管理員以及密碼/##新增一列。

- Systems Manager 參數

  您可以在 Systems Manager 參數主控台中建立此憑證，方法是在下列範例程式碼片段的表單中輸入文字。

```
{
  "username": "admin",
  "password": "password"
}
```

任務定義 FSxWindowsFileServerVolumeConfiguration 參數中的 credentialsParameter 保留秘密 ARN 或 Systems Manager 參數 ARN。如需詳細資訊，請參閱 《Secrets Manager 使用者指南》中的什麼是 AWS Secrets Manager 和 Systems Manager 使用者指南中的Systems Manager 參數存放區。

了解如何設定適用於 FSx for Windows File Server 檔案系 Amazon 的 FSx

了解如何啟動 Amazon ECS 最佳化 Windows 執行個體，該執行個體託管 FSx for Windows File Server 檔案系統和可存取檔案系統的容器。要做到這一點，你首先創建一個 AWS Directory Service AWS 管理的 Microsoft 活動目錄。然後，您可以使用 Amazon EC2 執行個體和任務定義建立適用於 Windows 檔案伺服器檔案伺服器的 FSx 檔案伺服器檔案系統和叢集。您可以設定容器的任務定義，以使用 FSx for Windows File Server 檔案系統。最後，您將測試檔案系統。

每次啟動或刪除 Active Directory 或 FSx for Windows File Server 檔案系統時，需要 20 至 45 分鐘。請準備保留至少 90 分鐘以完成教學課程，或在幾個工作階段中完成教學課程。

教學課程的先決條件

- 管理使用者。請參閱設定以使用 Amazon ECS。 。
- (選用) 用於透過 RDP 存取連線至 EC2 Windows 執行個體的 PEM 金鑰對。如需如何建立金鑰對的資訊，請參閱 Windows 執行個體使用者指南中的 Amazon EC2 金鑰對和 Windows 執行個體。
- 具有至少一個公有、一個私有子網路和一個安全群組的 VPC。您可以使用預設 VPC。您不需要 NAT 閘道或裝置。 AWS Directory Service 不支援與 Active Directory 搭配的網路位址轉譯 (NAT)。為了可以運作，Active Directory、FSx for Windows File Server 檔案系統、ECS 叢集和 EC2 執行個體必須位於 VPC 內。如需 VPC 和 Active Directory 的相關資訊，請參閱 AWS 受管 Microsoft AD 先決條件中的 Amazon VPC 主控台精靈組態。
- IAM ecsInstanceRole 和 ecsTaskExecution角色許可與您的帳戶相關聯。這些服務連結的角色可讓服務代表您進行 API 呼叫，並存取容器、機密、目錄和檔案伺服器。

步驟 1：建立 IAM 存取角色

使用 AWS Management Console建立叢集。

1. 請參Amazon ECS 容器執行個體 IAM 角色閱檢查您是否擁有 ecsInstanceRole 並查看如果沒有的話如何創建一個。
2. 建議您針對實際生產環境中的最低許可自訂角色政策。為了完成本教學課程的工作，請確認以下 AWS 託管策略已附加到您的 ECS InstanceRole。連接政策 (如果尚未連接)。

   - 亚马逊 ContainerServicefor
   - 亞馬遜核心 ManagedInstance
   - 亞馬遜訪DirectoryService問

   附加 AWS 受管理的策略。

   a. 開啟 IAM 主控台。
   b. 在導覽窗格中，選擇 Roles (角色)。
   c. 選擇 AWS 受管角色。
   d. 選擇 Permissions, Attach policies. (許可、連接政策。)。
   e. 若要縮小可連接的政策，請使用 Filter (篩選)。
   f. 選取適用的政策，然後選擇 Attach Policy (連接政策)。
3. 請參Amazon ECS 任務執行 IAM 角色閱檢查您是否有 ecs，TaskExecutionRole 並了解如何創建一個如果沒有 ECS。

   建議您針對實際生產環境中的最低許可自訂角色政策。為了完成本教學課程的工作，請確認下列 AWS 受管理的策略已附加到您的 ecs TaskExecution Role。連接政策 (如果尚未連接)。使用上一節中提供的程序來附加 AWS 受管理的策略。

   - SecretsManagerReadWrite
   - 亞馬遜 SxRead OnlyAccess
   - 亞馬遜訪ReadOnly問
   - AmazonECS TaskExecution RolePolicy

步驟 2：建立 Windows Active Directory (AD)

1. 請遵循《Directory Service 管理指南》中建立 AWS 受管理的 AD AWS 目錄中所述的步驟。使用您在本教學課程中指定的 VPC。在步驟 3 建立您的 AWS 受管 AD 目錄中，請儲存使用者名稱和密碼，以便在後續步驟中使用。另外，請記下完整網域名稱，以供後續步驟使用。您可以繼續在建立 Active Directory 時完成以下步驟。
2. 建立要在下列步驟中使用的 AWS 密碼管理員密碼。如需詳細資訊，請參閱 AWS Secrets Manager 使用指南中的 AWS Secrets Manager 入門。

   a. 開啟 Secrets Manager 主控台。
   b. 按一下 Store a new secret (存放新機密)。
   c. 選取 Other type of secrets (其他機密類型)。
   d. 對於 Secret key/value (機密鍵/值)，在第一個資料列中建立具有值 **admin** 的索引鍵 **username**。按一下 + Add row (+ 新增列)。
   e. 在新資料列中，建立索引鍵 **password**。針對值，請輸入您在建立 AWS 受管理 AD 目錄的步驟 3 中輸入的密碼。
   f. 按一下 Next (下一步) 按鈕。
   g. 提供機密名稱和描述。按一下 Next (下一步)。
   h. 按一下 Next (下一步)。按一下 Store (存放)。
   i. 從 Secrets (機密) 頁面清單上，按一下您剛才建立的機密。
   j. 儲存新機密的 ARN，以便在後續步驟中使用。
   k. 您可以繼續進行下一個步驟，同時建立 Active Directory。

步驟 3：驗證並更新安全群組

在此步驟中，您會驗證並更新您正在使用的安全群組的規則。為此，您可以使用為 VPC 建立的預設安全群組。

驗證並更新安全群組。

您需要建立或編輯安全群組，以便連接埠可以接收和傳送資料，請參閱 FSx for Windows File Server 使用者指南中的 Amazon VPC 安全群組。您可以藉由建立安全群組傳入規則完成此操作，規則顯示在以下傳入規則資料表的第一列。此規則允許來自網路界面 (及其關聯執行個體) 的傳入流量，而這些網路界面會指派給安全群組。您建立的所有雲端資源都位於同一 VPC 內，並連接至相同的安全群組。因此，此規則允許所需的流量往來於 FSx for Windows File Server 檔案系統、Active Directory 和 ECS 執行個體。其他傳入規則允許流量提供網站和 RDP 存取服務，以便連線至您的 ECS 執行個體。

下表顯示本教學課程所需的安全群組傳入規則。

Type	通訊協定	連接埠範圍	來源
所有流量	全部	全部	*sg-securitygroup*
HTTPS	TCP	443	0.0.0.0/0
RDP	TCP	3389	您的筆記型電腦 IP 地址

下表顯示本教學課程所需的安全群組傳出規則。

Type	通訊協定	連接埠範圍	目的地
所有流量	全部	全部	0.0.0.0/0

1. 開啟 EC2 主控台並從左側選單選取 Security Groups (安全群組)。
2. 從現在顯示的安全群組清單中，選取您要用於本教學課程的安全群組左側的核取方塊。

   隨即顯示您的安全群組詳細資訊。
3. 編輯傳入和傳出規則，方法是選取 Inbound rules (傳入規則) 或 Outbound rules (傳出規則) 索引標籤，然後選擇 Edit inbound rules (編輯傳入規則) 或 Edit outbound rules (編輯傳出規則) 按鈕。編輯規則以與上表中顯示的規則相符。稍後在本教學課程中建立 EC2 執行個體後，請使用 EC2 執行個體的公有 IP 地址編輯傳入規則 RDP 來源，詳情請參與《Amazon EC2 Windows 執行個體使用者指南》中的連線至 Windows 執行個體。

步驟 4：建立 FSx for Windows File Server 檔案系統

在您的安全群組經過驗證和更新，且您的 Active Directory 已建立並處於作用中狀態之後，請在與您的 Active Directory 相同的 VPC 中建立 FSx for Windows File Server 檔案系統。請依照下列步驟，建立 FSx for Windows File Server 檔案系統，以供您的 Windows 任務使用。

建立第一個檔案系統。

1. 開啟 Amazon FSx 主控台。
2. 在儀表板上，選擇 Create file system (建立檔案系統) 以啟動檔案系統建立精靈。
3. 在 Select file system type (選取檔案系統類型) 頁面中，選擇 FSx for Windows File Server，然後選擇 Next (下一步)。Create file system (建立檔案系統) 頁面隨即顯示。
4. 在 File system details (檔案系統詳細資訊) 區段中，輸入檔案系統的名稱。命名檔案系統可讓您更輕鬆地尋找和管理檔案系統。您最多可使用 256 個 Unicode 字元。允許的字元如下：英文字母、數字、空格和特殊字元加號 (+)、減號 (-)、等號 (=)、句點 (.)、下劃線 (_)、冒號 (:) 和斜線 (/)。
5. 對於 Deployment type (部署類型)，請選擇 Single-AZ (單一可用區)，部署在單一可用區域中部署的檔案系統。單一可用區 2 是最新一代的單一可用區域檔案系統，支援 SSD 和 HDD 儲存。
6. 對於 Storage type (儲存類型)，請選擇 EBS。
7. 對於 Storage capacity (儲存容量)，請輸入最小儲存容量。
8. 保留 Throughput capacity (輸送容量) 的預設設定。
9. 在「網路和安全性」區段中，選擇您為 AWS Directory Service 目錄選擇的相同 Amazon VPC。
10. 對於 VPC Security Groups (VPC 安全群組) 下，選擇您在步驟 3：驗證並更新您的安全群組中驗證的安全群組。
11. 對於 Windows authentication (Windows 身分驗證)，選擇 AWS 受管 Microsoft Active Directory，然後從清單中選擇您的 AWS Directory Service 目錄。
12. 對於 Encryption (加密)，請保留 aws/fsx (default) (aws/fsx (預設)) 的預設 Encryption key (加密金鑰) 設定。
13. 請保留 Maintenance preferences (維護偏好設定) 的預設設定。
14. 按一下 Next (下一步) 按鈕。
15. 檢閱顯示在 Create file system (建立檔案系統) 頁面上的檔案系統組態。請注意建立檔案系統後，您可以修改哪些檔案系統設定，以供參考。選擇 Create file system (建立檔案系統)。
16. 請注意檔案系統 ID。您會在稍後的步驟中需要使用。

    您可以在建立 FSx for Windows File Server 檔案系統時，繼續建立叢集和 EC2 執行個體的後續步驟。

步驟 5：建立 Amazon ECS 叢集。

使用 Amazon ECS 主控台建立叢集

1. 開啟主控台，網址為 https://console.aws.amazon.com/ecs/v2。
2. 從導覽列中選取要使用的「區域」。
3. 在導覽窗格中，選擇叢集。
4. 在 Clusters (叢集) 頁面上，選擇 Create cluster (建立叢集)。
5. 在叢集組態下的叢集名稱中，輸入 windows-fsx-cluster。
6. 展開基礎設施，清除 AWS Fargate (無伺服器)，然後選取 Amazon EC2 執行個體。
    - 若要建立 Auto Scaling 群組，請從 Auto Scaling group (ASG) (Auto Scaling 群組 (ASG)) 中選取 Create new group (建立新群組)，然後提供有關該群組的下列詳細資訊：
        - 在作業系統/架構中，選擇 Windows Server 2019 Core。
        - 對於 EC2 執行個體類型，選擇 t2.medium 或 t2.micro。
7. 選擇建立。

步驟 6：建立 Amazon ECS 最佳化的 Amazon EC2 執行個體

建立 Amazon ECS Windows 容器執行個體。

建立 Amazon ECS 執行個體

1. 使用 `aws ssm get-parameters` 命令擷取託管 VPC 的區域 AMI 名稱。如需詳細資訊，請參閱擷取 Amazon ECS 最佳化 AMI 中繼資料。
2. 使用 Amazon EC2 主控台來啟動執行個體。
    a. 前往 https://console.aws.amazon.com/ec2/ 開啟 Amazon EC2 主控台。
    b. 從導覽列中選取要使用的「區域」。
    c. 在 EC2 儀表板中，選擇 Launch instance (啟動執行個體)。
    d. 在 Name (名稱) 輸入唯一的名稱。
    e. 針對應用程式和作業系統映像 (Amazon Machine Image)，在搜尋欄位中，輸入您擷取到的 AMI 名稱。
    f. 對於執行個體類型，選擇 t2.medium 或 t2.micro。
    g. 在 Key pair (login) (金鑰對 (登入)) 欄位中，選擇一個金鑰對。如果您未指定金鑰對，則

h. 在網路設定下，針對 VPC 和子網路，選擇您的 VPC 和公有子網路。

i. 在 Network settings (網路設定) 下的 Security group (安全群組) 欄位中，選擇一個現有的安全群組，或建立一個新的安全群組。確定您選擇的安全群組具有在 教學課程的先決條件 中定義的輸入和輸出規則

j. 在 Network settings (網路設定) 下的 Auto-assign Public IP (自動指派公有 IP) 欄位中，選取 Enable (啟用)。

k. 展開進階詳細資訊，然後針對網域加入目錄，選取您建立的 Active Directory 的 ID。啟動 EC2 執行個體時，此選項網域會加入您的 AD。

l. 在 [進階詳細資料] 下方，針對 IAM 執行個體設定檔選擇 ecs InstanceRole。

m. 利用下列使用者資料設定您的 Amazon ECS 容器執行個體。在 Advanced details (進階詳細資料) 下的 User data (使用者資料) 欄位中，貼入下列指令碼，以您的叢集名稱取代 *cluster_name*。

```
<powershell>
Initialize-ECSAgent -Cluster windows-fsx-cluster -EnableTaskIAMRole
</powershell>
```

n. 準備就緒後，請選取 acknowledgment (確認) 欄位，再選擇 Launch Instances (啟動執行個體)。

o. 會有確認頁面讓您知道您的執行個體正在啟動。選擇 View Instances (檢視執行個體) 關閉確認頁面並返回主控台。

3. 開啟主控台，網址為 https://console.aws.amazon.com/ecs/v2。

4. 在導覽窗格中選擇叢集，然後選擇 windows-fsx-cluster。

5. 選擇基礎設施索引標籤，並確認您的執行個體已在 windows-fsx-cluster 叢集註冊。

步驟 7：註冊 Windows 任務定義

您必須先註冊任務定義，才能在您的 Amazon ECS 叢集中執行 Windows 容器。以下任務定義範例會顯示簡單的網頁。任務會啟動兩個可存取 FSx 檔案系統的容器。第一個容器會將 HTML 檔案寫入檔案系統。第二個容器從檔案系統下載 HTML 檔案並為網頁提供服務。

1. 開啟主控台，網址為 https://console.aws.amazon.com/ecs/v2。

2. 在導覽窗格中，選擇 Task Definitions (任務定義)。

3. 選擇 Create new task definitio (建立新任務定義)、Create new task definition with JSON (使用 JSON 建立新的任務定義)。

4. 在 JSON 編輯器方塊中，取代任務執行角色的值和有關 FSx 檔案系統的詳細資訊，然後選擇儲存。

```
{
    "containerDefinitions": [
        {
            "entryPoint": [
                "powershell",
                "-Command"
            ],
            "portMappings": [],
            "command": ["New-Item -Path C:\\fsx-windows-dir\\index.html -ItemType
 file -Value '<html> <head> <title>Amazon ECS Sample App</title> <style>body
 {margin-top: 40px; background-color: #333;} </style> </head><body> <div
 style=color:white;text-align:center> <h1>Amazon ECS Sample App</h1> <h2>It
 Works!</h2> <p>You are using Amazon FSx for Windows File Server file system for
 persistent container storage.</p>' -Force"],
            "cpu": 512,
            "memory": 256,
            "image": "mcr.microsoft.com/windows/servercore/iis:windowsservercore-
ltsc2019",
            "essential": false,
            "name": "container1",
            "mountPoints": [
                {
                    "sourceVolume": "fsx-windows-dir",
                    "containerPath": "C:\\fsx-windows-dir",
                    "readOnly": false
                }
            ]
        },
        {
            "entryPoint": [
                "powershell",
                "-Command"
            ],
            "portMappings": [
                {
                    "hostPort": 443,
                    "protocol": "tcp",
                    "containerPort": 80
                }
            ],
```

```
            "command": ["Remove-Item -Recurse C:\\inetpub\\wwwroot\\* -Force;
 Start-Sleep -Seconds 120; Move-Item -Path C:\\fsx-windows-dir\\index.html -
Destination C:\\inetpub\\wwwroot\\index.html -Force; C:\\ServiceMonitor.exe
 w3svc"],
            "mountPoints": [
                {
                    "sourceVolume": "fsx-windows-dir",
                    "containerPath": "C:\\fsx-windows-dir",
                    "readOnly": false
                }
            ],
            "cpu": 512,
            "memory": 256,
            "image": "mcr.microsoft.com/windows/servercore/iis:windowsservercore-
ltsc2019",
            "essential": true,
            "name": "container2"
        }
    ],
    "family": "fsx-windows",
    "executionRoleArn": "arn:aws:iam::111122223333:role/ecsTaskExecutionRole",
    "volumes": [
        {
            "name": "fsx-windows-dir",
            "fsxWindowsFileServerVolumeConfiguration": {
                "fileSystemId": "fs-0eeb5730b2EXAMPLE",
                "authorizationConfig": {
                    "domain": "example.com",
                    "credentialsParameter": "arn:arn-1234"
                },
                "rootDirectory": "share"
            }
        }
    ]
}
```

步驟 8：執行任務並檢視結果

在執行任務之前，請確認 FSx for Windows File Server 檔案系統的狀態為 Available (可用)。可用之後，您可以使用您建立的任務定義來執行任務。透過使用檔案系統建立相互之間隨機顯示 HTML 檔案的容器，以啟動任務。隨機顯示後，Web 伺服器提供簡單的 HTML 頁面。

> **Note**
>
> 您可能無法從 VPN 內連線至網站。

使用 Amazon ECS 主控台執行任務並檢視結果。

1. 開啟主控台，網址為 https://console.aws.amazon.com/ecs/v2。
2. 在導覽窗格中選擇叢集，然後選擇 windows-fsx-cluster。
3. 選擇任務索引標籤，然後選擇執行新任務。
4. 針對 Launch Type (啟動類型)，選擇 EC2。
5. 在部署組態下，針對任務定義選擇 fsx-windows，然後選擇建立。
6. 當您的任務狀態為執行中時，請選擇任務 ID。
7. 在容器下，當容器 1 狀態為已停止時，選取容器 2 以檢視容器的詳細資訊。
8. 在 container2 的容器詳細資訊下，選取網路繫結，然後按一下與容器關聯的外部 IP 地址。您的瀏覽器將會開啟並顯示以下訊息。

```
Amazon ECS Sample App
It Works!
You are using Amazon FSx for Windows File Server file system for persistent
 container storage.
```

> **Note**
>
> 可能需要幾分鐘的時間才會顯示訊息。如果幾分鐘後您未看到此訊息，請檢查您是否未在 VPN 中執行，並確認容器執行個體的安全群組，允許連接埠 443 上的傳入網路 HTTP 流量。

步驟 9：清除

> **Note**
>
> 刪除 FSx for Windows File Server 檔案系統或 AD 需要 20 到 45 分鐘的時間。您必須等到 FSx for Windows File Server 檔案系統刪除操作完成後，才能開始 AD 刪除操作。

刪除 FSx for Windows File Server 檔案系統。

1. 開啟 Amazon FSx 主控台
2. 選取您剛建立之 FSx for Windows File Server 檔案系統左側的選項按鈕。
3. 選擇動作。
4. 選取 Delete file system (刪除檔案系統)。

刪除 AD。

1. 開啟 AWS Directory Service 主控台。
2. 選擇您剛建立之 AD 左側的選項按鈕。
3. 選擇動作。
4. 選擇 Delete directory (刪除目錄)。

刪除叢集。

1. 開啟主控台，網址為 https://console.aws.amazon.com/ecs/v2。
2. 在導覽窗格中選擇叢集，然後選擇 fsx-windows-cluster。
3. 選擇 Delete cluster (刪除叢集)。
4. 輸入短語，然後選擇刪除。

終止 EC2 執行個體。

1. 開啟 Amazon EC2 主控台。
2. 從左側選單中，選取 Instances (執行個體)。
3. 勾選您建立之 EC2 執行個體左側的核取方塊。
4. 按一下執行個體狀態和終止執行個體。

刪除密碼。

1. 開啟 Secrets Manager 主控台。
2. 選取您為此逐步解說所建立的機密。
3. 按一下 Actions (動作)。
4. 選擇 Delete secret (刪除機密)。

搭配 Amazon ECS 使用碼頭磁碟區

使用 Docker 磁碟區時，可以使用內建的 `local` 驅動程式或第三方磁碟區驅動程式。Docker 磁碟區是由 Docker 管理，並且會在容器執行個體上的 `/var/lib/docker/volumes` 中建立一個目錄，其中包含磁碟區資料。

若要使用 Docker 磁碟區，請在您的任務定義中指定 `dockerVolumeConfiguration`。如需詳細資訊，請參閱使用磁碟區。

Docker 磁碟區的一些常用案例如下：

- 提供為與容器搭配使用的持久性資料磁碟區
- 在相同的容器執行個體上不同容器的不同位置共用定義的資料磁碟區
- 定義空的非持久性資料磁碟區，並將它掛載在相同任務內的多個容器上
- 提供資料磁碟區給由第三方驅動程式所管理的任務

使用 Docker 磁碟區的注意事項

使用 Docker 磁碟區時，請考量下列事項：

- 只有在使用 EC2 啟動類型或外部執行個體時才支援 Docker 磁碟區。
- Windows 容器只支援使用 `local` 驅動程式。
- 如果使用第三方驅動程式，確保先在容器執行個體上安裝及使用它，容器代理程式才會啟動。如果在啟動代理程式之前沒有啟動第三方驅動程式，您可以使用下列命令之一重新啟動容器代理程式：
  - 對於 Amazon ECS 最佳化 Amazon Linux 2 AMI：

    ```
    sudo systemctl restart ecs
    ```

  - 對於 Amazon ECS 最佳化 Amazon Linux AMI：

    ```
    sudo stop ecs && sudo start ecs
    ```

在 Amazon ECS 任務定義中指定泊塢視窗磁碟區

您必須先在任務定義中指定磁碟區和掛載點組態，您的容器才可以使用資料磁碟區。本節說明容器的磁碟區組態。對於使用 Docker 磁碟區的任務，指定 `dockerVolumeConfiguration`。對於使用綁定掛載主機磁碟區的任務，指定 `host` 和選用的 `sourcePath`。

以下任務定義 JSON 說明容器 volumes 和 mountPoints 物件的語法。

```
{
    "containerDefinitions": [
        {
            "mountPoints": [
                {
                    "sourceVolume": "string",
                    "containerPath": "/path/to/mount_volume",
                    "readOnly": boolean
                }
            ]
        }
    ],
    "volumes": [
        {
            "name": "string",
            "dockerVolumeConfiguration": {
                "scope": "string",
                "autoprovision": boolean,
                "driver": "string",
                "driverOpts": {
                    "key": "value"
                },
                "labels": {
                    "key": "value"
                }
            }
        }
    ]
}
```

name

類型：字串

必要：否

磁碟區名稱。最多可使用 255 個字母 (大寫和小寫)、數字、連字號 (-) 和底線 (_)。此名稱會在容器定義mountPoints物件的sourceVolume參數中參照。

dockerVolumeConfiguration

類型：DockerVolume配置對象

必要：否

此參數只有使用 Docker 磁碟區時才會指定。只有在 EC2 執行個體上執行任務時，才支援 Docker 磁碟區。Windows 容器僅支援`local`驅動程式的使用。若要使用綁定掛載，請指定 `host`。

`scope`

類型：字串

有效值: `task` | `shared`

必要：否

決定生命週期的 Docker 磁碟區範圍。範圍受限於 `task` 的 Docker 磁碟區，會在任務啟動時自動佈建，以及在任務停止時銷毀。範圍為 `shared` 的 Docker 磁碟區會在任務停止之後保留。

`autoprovision`

類型：布林值

預設值：`false`

必要：否

若此數值為 `true`，Docker 磁碟區便得以建立 (若它尚不存在)。只有在是時才會使用此欄`scope`位`shared`。如果`scope`是`task`，則必須省略此參數或設定為`false`。

`driver`

類型：字串

必要：否

要使用的 Docker 磁碟區驅動程式。驅動程式值必須與 Docker 提供的驅動程式名稱相符，因為此名稱用於工作放置。如果驅動程式是使用 Docker 外掛程式 CLI 安裝的，請使用`docker plugin ls`從容器執行個體擷取驅動程式名稱。如果使用其他方法安裝驅動程式，請使用 Docker 外掛程式探索來擷取驅動程式名稱。如需詳細資訊，請參閱 Docker 外掛程式探索。此參數會映射到 Docker Remote API 的建立磁碟區區段中的 `Driver` 以及 `docker volume create` 的 `--driver` 選項。

`driverOpts`

類型：字串

必要：否

要通過的 Docker 驅動程序特定選項的地圖。此參數會映射到 Docker Remote API 的建立磁碟區區段中的 DriverOpts 以及 docker volume create 的 --opt 選項。

labels

類型：字串

必要：否

自訂中繼資料，新增到您的 Docker 磁碟區。此參數會映射到 Docker Remote API 的建立磁碟區區段中的 Labels 以及 docker volume create 的 --label 選項。

mountPoints

類型：物件陣列

必要：否

容器中資料磁碟區的掛接點。此參數會映射到 Docker Remote API 的建立容器區段中的 Volumes 以及 docker run 的 --volume 選項。

Windows 容器可在 $env:ProgramData 所在的相同磁碟上掛載整個目錄。Windows 容器無法在不同的磁碟機上掛載目錄，而且無法跨磁碟機使用掛載點。您必須指定掛接點，才能將 Amazon EBS 磁碟區直接連接至 Amazon ECS 任務。

sourceVolume

類型：字串

必要：是 (當使用 mountPoints 時)

要掛載的磁碟區名稱。

containerPath

類型：字串

必要：是 (當使用 mountPoints 時)

要掛接磁碟區之容器中的路徑。

readOnly

類型：布林值

必要：否

如果此數值為 `true`，容器擁有磁碟區的唯讀存取權。如果此值為 `false`，則容器可寫入磁碟區。預設值為 `false`。

泊塢視窗磁碟區範例

為使用 Docker 磁碟區的容器提供暫時性儲存

在此範例中，容器使用任務完成後清除的空白資料磁碟區。舉一個使用案例範例，您的容器在任務期間需要存取一些暫存檔案儲存位置。使用 Docker 磁碟區可達成此任務。

1. 在任務定義 `volumes` 區段，以 `name` 和 `DockerVolumeConfiguration` 值定義資料磁碟區。在本範例中，我們指定範圍為 `task`，所以會在任務停止後刪除磁碟區，並使用內建的 `local` 驅動程式。

```
"volumes": [
    {
        "name": "scratch",
        "dockerVolumeConfiguration" : {
            "scope": "task",
            "driver": "local",
            "labels": {
                "scratch": "space"
            }
        }
    }
]
```

2. 在 `containerDefinitions` 區段中定義容器，並使用 `mountPoints` 值來參考已定義的磁碟區名稱，使用 `containerPath` 值將磁碟區掛載於容器。

```
"containerDefinitions": [
    {
        "name": "container-1",
        "mountPoints": [
            {
              "sourceVolume": "scratch",
              "containerPath": "/var/scratch"
            }
        ]
    }
]
```

使用 Docker 磁碟區提供容器的持久性儲存

在此範例中，您要多個容器使用一個共用磁碟區，並且要在使用它的任何單一任務停止之後加以保留。正在使用內建的 `local` 驅動程式。如此一來，磁碟區仍受限於容器執行個體的生命週期。

1. 在任務定義 `volumes` 區段，以 `name` 和 `DockerVolumeConfiguration` 值定義資料磁碟區。在此範例中，指定 `shared` 範圍，以便持續保留磁碟區，請將 `autoprovision` 設定為 `true`。如此一來，已建立該磁碟區以供使用。然後，還可以使用內建 `local` 驅動程式。

```
"volumes": [
    {
        "name": "database",
        "dockerVolumeConfiguration" : {
            "scope": "shared",
            "autoprovision": true,
            "driver": "local",
            "labels": {
                "database": "database_name"
            }
        }
    }
]
```

2. 在 `containerDefinitions` 區段中定義容器，並使用 `mountPoints` 值來參考已定義的磁碟區名稱，使用 `containerPath` 值將磁碟區掛載於容器。

```
"containerDefinitions": [
    {
        "name": "container-1",
        "mountPoints": [
        {
          "sourceVolume": "database",
          "containerPath": "/var/database"
        }
      ]
    },
    {
      "name": "container-2",
      "mountPoints": [
        {
          "sourceVolume": "database",
          "containerPath": "/var/database"
        }
```

```
      ]
    }
  ]
```

使用 Docker 磁碟區提供容器的 NFS 持久性儲存

在此範例中，容器使用 NFS 資料磁碟區，該資料磁碟區在任務啟動時自動掛載並在任務停止時自動卸載。這可使用 Docker 內建 `local` 驅動程式加以實現。一個範例使用案例是，您可能有本機 NFS 儲存體，並且需要從 ECS Anywhere 任務進行存取。這可以使用具有 NFS 驅動程式選項的 Docker 磁碟區來達成。

1. 在任務定義 `volumes` 區段，以 `name` 和 `DockerVolumeConfiguration` 值定義資料磁碟區。在此範例中，指定 `task` 範圍，以便在任務停止後卸載磁碟區。使用 `local` 驅動程式並相應地使用 `type`、`device` 和 `o` 選項設定 `driverOpts`。使用 NFS 伺服器端點取代 `NFS_SERVER`。

```
"volumes": [
       {
           "name": "NFS",
           "dockerVolumeConfiguration" : {
               "scope": "task",
               "driver": "local",
               "driverOpts": {
                   "type": "nfs",
                   "device": "$NFS_SERVER:/mnt/nfs",
                   "o": "addr=$NFS_SERVER"
               }
           }
       }
   ]
```

2. 在 `containerDefinitions` 區段中定義容器，並使用 `mountPoints` 值來參考已定義的磁碟區名稱，使用 `containerPath` 值將磁碟區掛載在容器上。

```
"containerDefinitions": [
       {
           "name": "container-1",
           "mountPoints": [
               {
                 "sourceVolume": "NFS",
                 "containerPath": "/var/nfsmount"
               }
```

```
            ]
        }
    ]
```

搭配 Amazon ECS 使用綁定掛載

透過繫結掛載，主機上的檔案或目錄 (例如 Amazon EC2 執行個體) 會掛載到容器中。在 Fargate 和 Amazon EC2 執行個體上託管的任務支援綁定掛載。繫結裝載會繫結至使用它們之容器的生命週期。使用綁定掛載的所有容器停止之後，例如當任務停止時，資料就會被移除。對於在 Amazon EC2 執行個體上託管的任務，可以在任務定義中指定和可選`sourcePath`值，將資料`host`與主機 Amazon EC2 執行個體的生命週期相關聯。如需詳細資訊，請參閱 Docker 文件中的使用綁定掛載。

以下是綁定掛載的常用案例。

- 若要提供空白的資料磁碟區以便在一個或多個容器中掛載。
- 若要在一個或多個容器中掛載主機資料磁碟區。
- 若要與相同任務中的其他容器共用來源容器的資料磁碟區。
- 若要將 Dockerfile 中的路徑及其內容公開給一個或多個容器。

使用綁定掛載時的考量

使用綁定掛載時，請考量下列事項。

- 根據預設， AWS Fargate 使用平台版本`1.4.0`或更新版本 (Linux) `1.0.0` 或更新版本 (Windows) 託管的工作會收到至少 20 GiB 的暫時儲存，以供繫結裝載使用。您可以透過在工作定義中指定`ephemeralStorage`參數，將暫時儲存的總量增加到最多 200 GiB。
- 若要在執行任務時將 Dockerfile 中的檔案公開至資料磁碟區，Amazon ECS 資料平面會尋找 `VOLUME` 指令。如果在 `VOLUME` 指令中指定的絕對路徑與在任務定義中指定的 `containerPath` 相同，則 `VOLUME` 指令路徑中的資料會複製到資料磁碟區。在下列 Dockerfile 範例中，`/var/log/exported` 目錄中名為 `examplefile` 的檔案會寫入主機，然後掛載在容器內。

```
FROM public.ecr.aws/amazonlinux/amazonlinux:latest
RUN mkdir -p /var/log/exported
RUN touch /var/log/exported/examplefile
VOLUME ["/var/log/exported"]
```

根據預設，磁碟區許可設定為 0755 和擁有者設定為 root。您可以在 Dockerfile 中自訂這些許可。在下列範例中，將目錄的擁有者設定為 node。

```
FROM public.ecr.aws/amazonlinux/amazonlinux:latest
RUN yum install -y shadow-utils && yum clean all
RUN useradd node
RUN mkdir -p /var/log/exported && chown node:node /var/log/exported
RUN touch /var/log/exported/examplefile
USER node
VOLUME ["/var/log/exported"]
```

- 對於在 Amazon EC2 執行個體上託管的任務，當 host 和 sourcePath 值未指定時，Docker 常駐程式會為您管理綁定掛載。當沒有任何容器參考此綁定掛載時，Amazon ECS 容器代理程式任務清除服務最終會予以刪除。根據預設，這會在容器退出的 3 小時後發生。不過，您可以使用 ECS_ENGINE_TASK_CLEANUP_WAIT_DURATION 代理程式變數設定此持續時間。如需詳細資訊，請參閱 Amazon ECS 容器代理程式組態。如果您需要此資料的保留時間超過容器的生命週期，請為綁定掛載指定 sourcePath 值。

在 Amazon ECS 任務定義中指定繫結裝載

對於託管在 Fargate 或 Amazon EC2 執行個體上的 Amazon ECS 任務，下列任務定義 JSON 程式碼片段會顯示任務定義volumesmountPoints、和ephemeralStorage物件的語法。

```
{
   "family": "",
   ...
   "containerDefinitions" : [
      {
         "mountPoints" : [
            {
               "containerPath" : "/path/to/mount_volume",
               "sourceVolume" : "string"
            }
          ],
          "name" : "string"
       }
    ],
    ...
    "volumes" : [
       {
```

```
            "name" : "string"
        }
    ],
    "ephemeralStorage": {
    "sizeInGiB": integer
    }
}
```

對於託管於 Amazon EC2 執行個體上的 Amazon ECS 任務，您可以在指定任務磁碟區詳細資訊時使用選用的 host 參數和 sourcePath。指定時，它會將綁定掛載任務的生命週期，而不是容器。

```
"volumes" : [
    {
        "host" : {
            "sourcePath" : "string"
        },
        "name" : "string"
    }
]
```

下面會更詳細地描述每個任務定義參數。

name

類型：字串

必要：否

磁碟區名稱。最多可使用 255 個字母 (大寫和小寫)、數字、連字號 (-) 和底線 (_)。此名稱會在容器定義mountPoints物件的sourceVolume參數中參照。

host

必要：否

host 參數用於將綁定掛載的生命週期綁定到主機 Amazon EC2 執行個體，而非任務，以及它的儲存位置。如果 host 參數是空的，則 Docker 常駐程式會為您的資料磁碟區指派主機路徑，但其相關聯的容器停止執行後，不保證會保留資料。

Windows 容器可在 $env:ProgramData 所在的相同磁碟上掛載整個目錄。

> **Note**
>
> 只有在使用 Amazon EC2 執行個體上託管的任務時，才支援此sourcePath參數。

`sourcePath`

類型：字串

必要：否

使用 `host` 參數時，指定 `sourcePath` 以宣告在主機 Amazon EC2 執行個體上提供給容器的路徑。如果此參數是空的，則 Docker 常駐程式會為您指派主機路徑。如果 `host` 參數包含 `sourcePath` 檔案位置，資料磁碟區將保留在主機 Amazon EC2 執行個體上的指定位置，直到您手動將其刪除為止。如果 `sourcePath` 值不存在於主機 Amazon EC2 執行個體，Docker 常駐程式將建立該值。如果位置存在，將匯出來源路徑資料夾的內容。

`mountPoints`

類型：物件陣列

必要：否

容器中資料磁碟區的掛接點。此參數會映射到 Docker Remote API 的建立容器區段中的 `Volumes` 以及 docker run 的 `--volume` 選項。

Windows 容器可在 `$env:ProgramData` 所在的相同磁碟上掛載整個目錄。Windows 容器無法在不同的磁碟機上掛載目錄，而且無法跨磁碟機使用掛載點。您必須指定掛接點，才能將 Amazon EBS 磁碟區直接連接至 Amazon ECS 任務。

`sourceVolume`

類型：字串

必要：是 (當使用 `mountPoints` 時)

要掛載的磁碟區名稱。

`containerPath`

類型：字串

必要：是 (當使用 `mountPoints` 時)

要掛接磁碟區之容器中的路徑。

`readOnly`

類型：布林值

必要：否

如果此數值為 `true`，容器擁有磁碟區的唯讀存取權。如果此值為 `false`，則容器可寫入磁碟區。預設值為 `false`。

`ephemeralStorage`

類型：物件

必要：否

為任務配置的暫時性儲存量。對於 AWS Fargate 使用平台版本`1.4.0`或更新版本 (Linux) 或更新版本 (Windows) 主控的工作，此參數可用來擴充暫時儲存空間總量，超出預設數量。`1.0.0`

您可以使用副駕駛 CLI CloudFormation、 AWS SDK 或 CLI 來指定繫結掛載的暫時儲存。

綁定掛載範例

若要為 Fargate 任務分配更多的暫時性儲存量

對於託管於使用平台版本 `1.4.0` 或更新版本 (Linux) 或者 `1.0.0` (Windows) 的 Fargate 上的 Amazon ECS 任務，您可以為任務中的容器分配比預設暫時性儲存量更多的空間以供使用。此範例可以整合到其他範例中，為您的 Fargate 任務分配更多的暫時性儲存。

- 在任務定義中，定義 `ephemeralStorage` 物件。`sizeInGiB` 必須是 21 和 200 之間的整數並以 GiB 為單位來表示。

```
"ephemeralStorage": {
    "sizeInGiB": integer
}
```

若要為一個或多個容器提供空白的資料磁碟區

在某些情況下，您希望在任務中為容器提供一些暫存空間。例如，您可能有兩個資料庫容器，在任務期間需要存取相同的暫存檔案儲存位置。這可以使用綁定掛載來達成。

1. 在任務定義 `volumes` 區段中，以名稱 `database_scratch` 定義綁定掛載。

```
  "volumes": [
    {
      "name": "database_scratch"
    }
  ]
```

2. 在 containerDefinitions 區段中，建立資料庫容器定義。如此一來，它們就可以掛載磁碟區。

```
"containerDefinitions": [
    {
      "name": "database1",
      "image": "my-repo/database",
      "cpu": 100,
      "memory": 100,
      "essential": true,
      "mountPoints": [
        {
          "sourceVolume": "database_scratch",
          "containerPath": "/var/scratch"
        }
      ]
    },
    {
      "name": "database2",
      "image": "my-repo/database",
      "cpu": 100,
      "memory": 100,
      "essential": true,
      "mountPoints": [
        {
          "sourceVolume": "database_scratch",
          "containerPath": "/var/scratch"
        }
      ]
    }
  ]
```

若要將 Docker 文件中的路徑及其內容公開給容器

在此範例中，您有一個 Dockerfile，它會寫入您想要掛載在容器內的資料。此範例適用於 Fargate 或 Amazon EC2 執行個體上託管的任務。

1. 建立 Dockerfile。下列範例使用公有 Amazon Linux 2 容器映像，並在我們想要在容器內部掛載的 `/var/log/exported` 目錄中建立名為 `examplefile` 的檔案。`VOLUME` 指令應該指定一個絕對路徑。

```
FROM public.ecr.aws/amazonlinux/amazonlinux:latest
RUN mkdir -p /var/log/exported
RUN touch /var/log/exported/examplefile
VOLUME ["/var/log/exported"]
```

   根據預設，磁碟區許可設定為 `0755` 和擁有者設定為 `root`。可以在 Docker 檔案中變更這些許可。在下列範例中，`/var/log/exported` 目錄的擁有者設定為 `node`。

```
FROM public.ecr.aws/amazonlinux/amazonlinux:latest
RUN yum install -y shadow-utils && yum clean all
RUN useradd node
RUN mkdir -p /var/log/exported && chown node:node /var/log/exported
USER node
RUN touch /var/log/exported/examplefile
VOLUME ["/var/log/exported"]
```

2. 在任務定義 `volumes` 區段中，以名稱 `application_logs` 定義一個磁碟區。

```
"volumes": [
  {
    "name": "application_logs"
  }
]
```

3. 在 `containerDefinitions` 區段中，建立應用程式容器定義。如此一來，它們就可以掛載儲存。`containerPath` 值必須符合 Dockerfile `VOLUME` 指令中指定的絕對路徑。

```
"containerDefinitions": [
  {
    "name": "application1",
    "image": "my-repo/application",
    "cpu": 100,
```

```
      "memory": 100,
      "essential": true,
      "mountPoints": [
        {
          "sourceVolume": "application_logs",
          "containerPath": "/var/log/exported"
        }
      ]
    },
    {
      "name": "application2",
      "image": "my-repo/application",
      "cpu": 100,
      "memory": 100,
      "essential": true,
      "mountPoints": [
        {
          "sourceVolume": "application_logs",
          "containerPath": "/var/log/exported"
        }
      ]
    }
  ]
```

若要為與主機 Amazon EC2 執行個體生命週期相關聯的容器提供空資料磁碟區

對於在 Amazon EC2 執行個體上託管的任務，您可以使用綁定掛載，並將資料與主機 Amazon EC2 執行個體的生命週期相關聯。您可以使用 `host` 參數並指定 `sourcePath` 值來達成此操作。位在 `sourcePath` 的任何檔案都會出現在值為 `containerPath` 的容器中。寫入 `containerPath` 值的任何檔案都會寫入主機 Amazon EC2 執行個體上的 `sourcePath` 值。

> **Important**
>
> Amazon ECS 不會在 Amazon EC2 執行個體之間同步您的儲存。使用持久性儲存的任務可以放置在您有可用容量之叢集中的任何 Amazon EC2 執行個體。如果您的任務在停止和重新啟動後需要持續性儲存，請務必在任務啟動時使用 AWS CLI start-task 命令指定相同的 Amazon EC2 執行個體。也可使用 Amazon EFS 磁碟區以供持久性儲存。如需詳細資訊，請參閱 搭配 Amazon ECS 使用 Amazon EFS 磁碟區。

1. 在任務定義 volumes 區段，以 name 和 sourcePath 值定義綁定掛載。在下列範例中，主機 Amazon EC2 執行個體包含要掛載在容器內的 /ecs/webdata 的資料。

```
"volumes": [
  {
    "name": "webdata",
    "host": {
      "sourcePath": "/ecs/webdata"
    }
  }
]
```

2. 在 containerDefinitions 區段中定義容器，並使用 mountPoints 值來參考綁定掛載名稱，使用 containerPath 值在容器上掛載綁定掛載。

```
"containerDefinitions": [
  {
    "name": "web",
    "image": "nginx",
    "cpu": 99,
    "memory": 100,
    "portMappings": [
      {
        "containerPort": 80,
        "hostPort": 80
      }
    ],
    "essential": true,
    "mountPoints": [
      {
        "sourceVolume": "webdata",
        "containerPath": "/usr/share/nginx/html"
      }
    ]
  }
]
```

在不同位置的多個容器中掛載已定義的磁碟區

您可以在任務定義中定義資料磁碟區，並將該磁碟區掛載到不同容器的不同位置。例如，您的主機容器有一個位於 /data/webroot 的網站資料夾。建議您以唯讀方式將該資料磁碟區掛載到有不同文件根目錄的兩個不同 Web 伺服器上。

1. 在任務定義 volumes 區段中，以名稱 webroot 和來源路徑 /data/webroot 定義資料磁碟區。

```
"volumes": [
  {
    "name": "webroot",
    "host": {
      "sourcePath": "/data/webroot"
    }
  }
]
```

2. 在 containerDefinitions 區段中，使用 mountPoints 值為每個 Web 伺服器定義容器，這些值會建立 webroot 磁碟區與指向該容器文件根目錄之 containerPath 值的關聯性。

```
"containerDefinitions": [
  {
    "name": "web-server-1",
    "image": "my-repo/ubuntu-apache",
    "cpu": 100,
    "memory": 100,
    "portMappings": [
      {
        "containerPort": 80,
        "hostPort": 80
      }
    ],
    "essential": true,
    "mountPoints": [
      {
        "sourceVolume": "webroot",
        "containerPath": "/var/www/html",
        "readOnly": true
      }
    ]
  },
  {
    "name": "web-server-2",
```

```
      "image": "my-repo/sles11-apache",
      "cpu": 100,
      "memory": 100,
      "portMappings": [
        {
          "containerPort": 8080,
          "hostPort": 8080
        }
      ],
      "essential": true,
      "mountPoints": [
        {
          "sourceVolume": "webroot",
          "containerPath": "/srv/www/htdocs",
          "readOnly": true
        }
      ]
    }
  ]
```

使用 **volumesFrom** 掛載來自其他容器的磁碟區

針對在 Amazon EC2 執行個體上託管的任務，您可以在容器上定義一或多個磁碟區，然後在相同的任務中的不同的容器定義中使用 volumesFrom 參數，將所有來自 sourceContainer 的磁碟區掛載在其原始定義的掛載點。volumesFrom 參數適用於在任務定義中定義的磁碟區，以及使用 Dockerfile 內建在映像中的磁碟區。

1. (選用) 若要共用內建在映像的磁碟區，請使用 Dockerfile 中的 VOLUME 指令。以下範例 Dockerfile 使用 httpd 映像，然後新增磁碟區，再將之掛載到 Apache 文件根中的 dockerfile_volume。該資料夾由 httpd Web 伺服器使用。

   ```
   FROM httpd
   VOLUME ["/usr/local/apache2/htdocs/dockerfile_volume"]
   ```

   您可以使用此 Dockerfile 建立映像，並將之推送到儲存庫，例如 Docker Hub，然後在您的任務定義中使用。在以下步驟中使用的示例my-repo/httpd_dockerfile_volume圖像是使用前面的 Dockerfile 構建的。
2. 建立任務定義，定義您容器的其他磁碟區及掛載點。在這個範例 volumes 區段中，您要建立一個名為 empty 的空磁碟區，它是由 Docker 常駐程式管理。您還要定義一個稱為 host_etc 的主機磁碟區。它會匯出主機容器執行個體的 /etc 資料夾。

```
{
  "family": "test-volumes-from",
  "volumes": [
    {
      "name": "empty",
      "host": {}
    },
    {
      "name": "host_etc",
      "host": {
        "sourcePath": "/etc"
      }
    }
  ],
```

在容器定義區段中，建立一個容器，掛載之前定義的磁碟區。在此範例中，web 容器掛載 empty 和 host_etc 磁碟區。這是使用由 Dockerfile 中的磁碟區建立的映像的容器。

```
"containerDefinitions": [
    {
      "name": "web",
      "image": "my-repo/httpd_dockerfile_volume",
      "cpu": 100,
      "memory": 500,
      "portMappings": [
        {
          "containerPort": 80,
          "hostPort": 80
        }
      ],
      "mountPoints": [
        {
          "sourceVolume": "empty",
          "containerPath": "/usr/local/apache2/htdocs/empty_volume"
        },
        {
          "sourceVolume": "host_etc",
          "containerPath": "/usr/local/apache2/htdocs/host_etc"
        }
      ],
      "essential": true
```

```
    },
```

建立另一個容器，使用 volumesFrom 掛載與 web 容器相關聯的所有磁碟區。web 容器上的所有磁碟區同樣掛載在 busybox 容器上。這包括在 Dockerfile 中指定的磁碟區，該磁碟區用於建置 my-repo/httpd_dockerfile_volume 映像。

```
    {
      "name": "busybox",
      "image": "busybox",
      "volumesFrom": [
        {
          "sourceContainer": "web"
        }
      ],
      "cpu": 100,
      "memory": 500,
      "entryPoint": [
        "sh",
        "-c"
      ],
      "command": [
        "echo $(date) > /usr/local/apache2/htdocs/empty_volume/date && echo $(date)
 > /usr/local/apache2/htdocs/host_etc/date && echo $(date) > /usr/local/apache2/
htdocs/dockerfile_volume/date"
      ],
      "essential": false
    }
  ]
}
```

執行此任務時，兩個容器會掛載磁碟區，而 busybox 容器中的 command 會將日期和時間寫入檔案。這個檔案在每個磁碟區資料夾中稱為 date。然後，在 web 容器所顯示的網站上，就能看到這些資料夾。

> **Note**
>
> 由於 busybox 容器執行快速命令，然後結束，所以在容器定義中必須設為 "essential": false。否則，它結束時會停止整個任務。

管理 Amazon ECS 上的容器交換記憶體空間

透過 Amazon ECS，您可在容器層級控制 Linux Amazon EC2 執行個體上的交換記憶體空間用量。使用每個容器交換組態，任務定義內的每個容器都可以啟用或停用交換。對於已啟用交換的容器，可以限制所使用的最大交換空間量。例如，對延遲要求嚴格的容器可以停用交換。相較之下，具有高暫態記憶體需求的容器可以開啟交換功能，以減少容器負載下時發 out-of-memory 生錯誤的機會。

容器的交換組態由下列容器定義參數管理：

`maxSwap`

容器可以使用的交換記憶體總量 (以 MiB 為單位)。此參數將會轉換為 docker run 的 `--memory-swap` 選項，其中值是容器記憶體與 `maxSwap` 值的總和。

如果將 `maxSwap` 值指定為 `0`，容器不會使用交換。接受的值為 `0` 或任何正整數。如果省略 `maxSwap` 參數，容器使用其執行所在的容器執行個體的交換組態。必須設定 `maxSwap` 值，才能使用 `swappiness` 參數。

`swappiness`

您可藉此調整容器的記憶體交換行為。`swappiness` 的值若為 `0` 將導致交換不會發生 (除非有需要)。為 `100` 的 `swappiness` 值導致積極地交換頁面。接受的值為介於 `0` 與 `100` 之間的整數。如果未指定 `swappiness` 參數，則會使用預設值 `60`。如果未對 `maxSwap` 指定值，則會忽略此參數。此參數會映射到 docker run 的 `--memory-swappiness` 選項。

在以下範例中，提供了 JSON 語法。

```
"containerDefinitions": [{
        ...
        "linuxParameters": {
            "maxSwap": integer,
            "swappiness": integer
        },
        ...
}]
```

考量事項

當您使用每個容器交換組態時，請考量下列事項。

- 必須在託管任務的 Amazon EC2 執行個體上啟用和配置交換空間，容器才能使用這些空間。根據預設，Amazon ECS 最佳化 AMI 沒有啟用交換功能。您必須在執行個體上啟用交換，才能使用此功能。如需詳細資訊，請參閱 Amazon EC2 使用者指南中的執行個體存放區交換磁碟區或如何使用交換檔分配記憶體做為 Amazon EC2 執行個體中的交換空間？ 。
- 只有指定 EC2 啟動類型的任務定義才支援交換空間容器定義參數。僅供 Fargate 上的 Amazon ECS 使用的任務定義不支援這些參數。
- 只有 Linux 容器才支援此功能。目前不支援 Windows 容器。
- 如果從任務定義中省略 `maxSwap` 和 `swappiness` 容器定義參數，每個容器都有一個為 `60` 的預設 `swappiness` 值。此外，總交換使用量限制為容器記憶體的兩倍。
- 如果您在 Amazon Linux 2023 上使用任務，則不支援 `swappiness` 參數。

Amazon ECS 任務定義 Fargate 啟動類型的差異

為了使用 Fargate，您必須將任務定義配置為使用 Fargate 啟動類型。使用 Fargate 時，還有其他考慮因素。

任務定義參數

使用 Fargate 啟動類型的任務不支援所有可用的 Amazon ECS 任務定義參數。有些參數完全不予以支援，而其他參數對 Fargate 任務會有不同的行為。

下列任務定義參數在 Fargate 任務中無效：

- `disableNetworking`
- `dnsSearchDomains`
- `dnsServers`
- `dockerSecurityOptions`
- `extraHosts`
- `gpu`
- `ipcMode`
- `links`
- `placementConstraints`
- `privileged`

- `maxSwap`
- `swappiness`

下列任務定義參數在 Fargate 任務中有效，但應注意其限制：

- `linuxParameters` - 當指定套用於容器的特定於 Linux 的選項時，對於 `capabilities`，您可以新增的唯一功能是 `CAP_SYS_PTRACE`。不支援 `devices`、`sharedMemorySize` 和 `tmpfs` 參數。如需詳細資訊，請參閱 Linux 參數。
- `volumes` - Fargate 任務僅支援繫結掛載主機磁碟區，所以不支援 `dockerVolumeConfiguration` 參數。如需詳細資訊，請參閱 磁碟區。
- `cpu` - 對於 AWS Fargate 上的 Windows 容器，值不可少於 1 個 vCPU。

為了確保您的任務定義通過驗證可與 Fargate 搭配使用，您可以在登錄任務定義時指定下列項目：

- 在中 AWS Management Console，對於「需要相容性」欄位，指定。`FARGATE`
- 在中 AWS CLI，指定選`--requires-compatibilities`項。
- 在 Amazon ECS API 中，指定 `requiresCompatibilities` 標記。

作業系統與架構

當您為 AWS Fargate 設定任務和容器定義，則必須指定容器執行的作業系統。AWS Fargate 支援以下作業系統：

- Amazon Linux 2

> **Note**
>
> Linux 容器僅使用來自主機作業系統的核心和核心組態。例如，核心組態包括 `sysctl` 系統控制項。Linux 容器映像可以從包含任何 Linux 發行版本的檔案和程式的基礎映像製作。如果 CPU 架構相符，您可以從任何作業系統上的任何 Linux 容器映像執行容器。

- Windows Server 2019 Full
- Windows Server 2019 Core
- Windows Server 2022 Full
- Windows Server 2022 Core

當您在 AWS Fargate 上執行 Windows 容器時，必須具有 X86_64 CPU 架構。

當您在 AWS Fargate 上執行 Linux 容器時，對於基於 ARM 的應用程式，可以使用 X86_64 CPU 架構或 ARM64 架構。如需詳細資訊，請參閱 the section called “64 位元 ARM 工作負載的工作定義”。

任務 CPU 和記憶體

AWS Fargate 的 Amazon ECS 任務定義需要您指定任務層級的 CPU 和記憶體。雖然您也可以指定 Fargate 任務之容器層級的 CPU 和記憶體，但這是選用的。只要在任務層級指定這些資源，就可以滿足大多數使用案例。下表顯示有效的任務層級 CPU 和記憶體組合。您可以將工作定義中的記憶體值指定為 MiB 或 GB 的字串。例如，您可以將記憶體值指定為 `3072` MiB 或 `3 GB` GB。您可以將 JSON 檔案中的 CPU 值指定為以 CPU 單位或虛擬 CPU (vCPUs) 為單位的字串。例如，您可以使用 CPU 單位或 `1 vCPU` vCPUs 來指定 CPU 值。`1024`

CPU 數值	記憶體數值	AWS Fargate 支援的作業系統
256 (.25 vCPU)	512 MiB、1 GB、2 GB	Linux
512 (.5 vCPU)	1 GB、2 GB、3 GB、4 GB	Linux
1024 (1 vCPU)	2 GB、3 GB、4 GB、5 GB、6 GB、7 GB、8 GB	Linux、Windows
2048 (2 vCPU)	介於 4 GB 與 16 GB 之間，以 1 GB 為單位遞增	Linux、Windows
4096 (4 vCPU)	介於 8 GB 與 30 GB 之間，以 1 GB 為單位遞增	Linux、Windows
8192 (8 vCPU) **Note** 此選項需要 Linux 平台 `1.4.0` 或更新版本。	介於 16 GB 與 60 GB 之間，以 4 GB 為單位遞增	Linux
16384 (16vCPU)	介於 32 GB 與 120 GB 之間，以 8 GB 為單位遞增	Linux

CPU 數值	記憶體數值	AWS Fargate 支援的作業系統

> **Note**
>
> 此選項需要 Linux 平台 `1.4.0` 或更新版本。

任務聯網

AWS Fargate 的 Amazon ECS 任務需要 `awsvpc` 網路模式，該網路模式會為每個任務提供彈性網絡介面。當您使用此網路模式執行任務或建立服務時，必須指定一或多個子網路來連接網路界面，以及指定一或多個安全群組來套用至網路界面。

如果使用公有子網路，請決定是否提供網路界面的公有 IP 地址。若要讓公有子網路中的 Fargate 任務提取容器映像，需要透過路由至網際網路，或可將請求路由至網際網路的 NAT 閘道，將公有 IP 地址指派給任務的彈性網路界面。若要讓私有子網路中的 Fargate 任務提取容器映像，您需要子網路中的 NAT 閘道以將請求路由到網際網路。在 Amazon ECR 中託管容器映像時，可以將 Amazon ECR 設定為使用介面 VPC 端點。在這種情況下，任務的私有 IPv4 地址會用於映像提取。如需 Amazon ECR 介面端點的詳細資訊，請參閱 Amazon Elastic Container Registry 使用者指南中的 Amazon ECR 介面 VPC 端點 (AWS PrivateLink)。

以下是 Fargate 服務`networkConfiguration`區段的範例：

```
"networkConfiguration": {
   "awsvpcConfiguration": {
      "assignPublicIp": "ENABLED",
      "securityGroups": [ "sg-12345678" ],
      "subnets": [ "subnet-12345678" ]
   }
}
```

任務資源限制

AWS Fargate 上適用於 Linux 容器的 Amazon ECS 任務定義支援 `ulimits` 參數定義要為容器設定的資源限制。

AWS Fargate 上適用於 Windows 的 Amazon ECS 任務定義不支援 `ulimits` 參數定義要為容器設定的資源限制。

在 Fargate 上託管的 Amazon ECS 任務會使用作業系統設定的預設資源限制值，但是 nofile 資源限制參數除外。nofile 資源限制會對容器可使用的開放檔案數量設限。在 Fargate 上，預設的 nofile 軟限制為 1024，硬限制為 65535。您可以將兩個限制的值設定為 1048576。

以下是範例任務定義程式碼片段，示範如何定義已加倍的自訂 nofile 限制：

```
"ulimits": [
    {
       "name": "nofile",
       "softLimit": 2048,
       "hardLimit": 8192
    }
]
```

如需可調整之其他資源限制的詳細資訊，請參閱 資源限制。

日誌

事件記錄

Amazon ECS 記錄它採取的操作。 EventBridge您可以使用 Amazon ECS 事件 EventBridge 來接收有關 Amazon ECS 叢集、服務和任務目前狀態的近乎即時的通知。此外，您可以自動執行動作來回應這些事件。如需詳細資訊，請參閱 使用以下方式自動回應 Amazon ECS 錯誤 EventBridge。

任務生命週期日誌記錄

在 Fargate 上執行的任務會發佈時間戳記，以透過任務生命週期的狀態追蹤任務。您可以在中的工作詳細資料中查看時間戳記， AWS Management Console 並在 AWS CLI 和 SDK 中描述工作。例如，您可以使用時間戳記來評估任務下載容器映像所花費的時間，並決定是否應最佳化容器映像大小，或使用 Seekable OCI 索引。如需有關容器映像實務的詳細資訊，請參閱 Amazon ECS 容器映像的最佳實務。

應用程式日誌記錄

AWS Fargate 的 Amazon ECS 任務定義支援日誌組態的 awslogs、splunk 和 awsfirelens 日誌驅動程式。

awslogs日誌驅動程式會設定您的 Fargate 任務，以便將日誌資訊傳送到 Amazon 日誌。CloudWatch 以下顯示任務定義片段，其中設定了 awslogs 日誌驅動程式：

```
"logConfiguration": {
   "logDriver": "awslogs",
```

```
    "options": {
       "awslogs-group" : "/ecs/fargate-task-definition",
       "awslogs-region": "us-east-1",
       "awslogs-stream-prefix": "ecs"
    }
}
```

如需有關在工作定義中使用記awslogs錄驅動程式將容器記錄檔傳送至 CloudWatch 記錄的詳細資訊，請參閱將 Amazon ECS 日誌傳送到 CloudWatch 。

如需任務定義中 awsfirelens 日誌驅動程式的詳細資訊，請參閱 將 Amazon ECS 日誌傳送到 AWS 服務或 AWS Partner。

如需在任務定義中使用 splunk 日誌驅動程式的詳細資訊，請參閱 splunk日誌驅動。

任務儲存體

對於 Fargate 上託管的 Amazon ECS 任務，支援下列儲存類型：

- Amazon EBS 磁碟區為資料密集型容器化工作負載提供符合成本效益、耐用且高效能的區塊儲存。如需詳細資訊，請參閱 使用 Amazon EBS 卷與 Amazon ECS。
- 適用於持久性儲存的 Amazon EFS 磁碟區。如需詳細資訊，請參閱 搭配 Amazon ECS 使用 Amazon EFS 磁碟區。
- 用於暫時性儲存的繫結掛載。如需詳細資訊，請參閱 搭配 Amazon ECS 使用綁定掛載。

使用 Seekable OCI (SOCI) 延遲載入容器映像

在使用 Linux 平台版本 1.4.0 的 Fargate 上，Amazon ECS 任務可以使用 Seekable OCI (SOCI) 來協助更快地啟動任務。使用 SOCI 時，容器只需耗費幾秒鐘就可以開始提取映像，因此在背景下載影像時，可為環境設定和應用程式執行個體化提供時間。這稱為延遲載入。當 Fargate 啟動 Amazon ECS 任務時，Fargate 會自動偵測任務中映像是否存在 SOCI 索引，並啟動容器，而無需等待整個映像下載完成。

對於在沒有 SOCI 索引的情況下執行的容器，容器映像會在容器啟動之前完全下載。此行為在 Fargate 的所有其他平台版本以及 Amazon EC2 執行個體上的 Amazon ECS 最佳化 AMI 上均相同。

Seekable OCI 索引

可搜索的 OCI（SOCI）是一種開源技術，可以通過懶惰地加載容 AWS 器映像來更快地啟動容器。SOCI 的運作原理是在現有容器映像中建立檔案索引 (SOCI 索引)。此索引有助於更快地啟動容

器，提供在下載整個映像之前從容器映像中擷取個別檔案的功能。SOCI 索引必須以成品的形式儲存在與容器登錄中的映像相同的儲存庫中。您應該只使用來自信任來源的 SOCI 索引，因為索引是映像內容的權威來源。如需詳細資訊，請參引入 Seekable OCI 用於延遲載入容器映像。

考量事項

如果您希望 Fargate 使用 SOCI 索引在任務中延遲載入容器映像，請考慮以下幾點：

- 只有在 Linux 平台版本 `1.4.0` 上執行的任務才能使用 SOCI 索引。不支援在 Fargate 上執行 Windows 容器的任務。
- 支援在 X86_64 或 ARM64 CPU 架構上執行的任務。採用 ARM64 架構的 Linux 任務不支援 Fargate Spot 容量提供者。
- 任務定義中的容器映像均須在與映像相同的容器登錄中具有 SOCI 索引。
- 任務定義中的容器映像均須儲存在相容的映像登錄中。以下列出相容的登錄：
  - Amazon ECR 私有登錄檔。
- 僅支援使用 gzip 壓縮或未壓縮的容器映像。不支援使用 zstd 壓縮的容器映像。
- 我們建議您嘗試延遲載入大於 250 MiB 壓縮大小的容器映像。您不太可能看到載入較小映像的時間減少。
- 由於延遲載入可能會變更任務開始所需的時間，因此您可能需要變更各種逾時，例如 Elastic Load Balancing 的運作狀態檢查寬限期。
- 如果您想要防止容器映像延遲載入，請從容器登錄中刪除 SOCI 索引。如果任務中的容器映像不符合其中一項考量事項，則會透過預設方法下載該容器映像。

建立 Seekable OCI 索引

對於延遲加載容器映像，它需要沿著容器映像一側創建並存儲在容器映像存儲庫中的 SOCI 索引（元數據文件）。要創建和推送 SOCI 索引，您可以使用開源社交快照 CLI 工具。 GitHub或者，您可以部署 CloudFormation AWS SOCI 索引產生器。這是一種無伺服器解決方案，可在將容器映像推送至 Amazon ECR 時自動建立和推送 SOCI 索引。如需有關解決方案和安裝步驟的詳細資訊，請參閱上 GitHub的 CloudFormation AWS SOCI 索引產生器。 CloudFormation AWS SOCI 索引產生器是自動化 SOCI 入門的一種方式，而開放原始碼 soci 工具在索引產生方面具有更大的彈性，以及在持續整合和持續交付 (CI/CD) 管線中整合索引產生的能力。

> **Note**
>
> 若要為映像建立 SOCI 索引，映像必須存在於執行 `soci-snapshotter` 的電腦上的 containerd 映像存放區中。如果映像位於 Docker 映像存放區中，則無法找到該映像。

驗證任務是否使用延遲載入

若要驗證任務是否使用 SOCI 延遲載入，請從任務內部檢查任務中繼資料端點。當您查詢任務中繼資料端點版本 4 時，您從中查詢的容器的預設路徑中有一個 `Snapshotter` 欄位。此外，`/task` 路徑中的每個容器都有 `Snapshotter` 欄位。此欄位的預設值為`overlayfs`，如果使用 SOCI，`soci`則此欄位會設定為。

執行 Windows 的 EC2 執行個體的 Amazon ECS 任務定義差異

在 EC2 Windows 執行個體上執行的任務不支援所有可用的 Amazon ECS 任務定義參數。某些參數完全不受支援，而其他參數的行為則不同。

下列任務定義參數不支援 Amazon EC2 視窗任務定義：

- `containerDefinitions`
  - `disableNetworking`
  - `dnsServers`
  - `dnsSearchDomains`
  - `extraHosts`
  - `links`
  - `linuxParameters`
  - `privileged`
  - `readonlyRootFilesystem`
  - `user`
  - `ulimits`
- `volumes`
  - `dockerVolumeConfiguration`
- `cpu`

  我們建議為 Windows 容器指定容器層級的 CPU。

- `memory`

  我們建議為 Windows 容器指定容器層級的記憶體。

- `proxyConfiguration`
- `ipcMode`
- `pidMode`
- `taskRoleArn`

  EC2 Windows 執行個體功能上任務的 IAM 角色需要額外的組態，但這類組態大部分類似於為 Linux 容器執行個體上的任務設定 IAM 角色。如需更多資訊，請參閱the section called “ Amazon EC2 窗口實例附加配置”。

使用主控台建立 Amazon ECS 任務定義

您可以使用主控台或編輯 JSON 檔案來建立工作定義。

JSON 驗證

Amazon ECS 主控台 JON 編輯器會對 JSON 檔案的以下方面進行驗證：

- 該文件是一個有效的 JSON 文件。
- 該文件不包含任何無關鍵字。
- 檔案包含`familyName`參數。
- 下至少有一個項目`containerDefinitions`。

AWS CloudFormation 堆疊

下列行為適用於 2023 年 1 月 12 日之前在新的 Amazon ECS 主控台中建立的任務定義。

建立任務定義時，Amazon ECS 主控台會自動建立名稱開頭為的 CloudFormation堆疊。`ECS-Console-V2-TaskDefinition-`如果您使用 AWS CLI 或 AWS SDK 取消註冊任務定義，則必須手動刪除任務定義堆疊。若要取得更多資訊，請參閱《使用指南》中的AWS CloudFormation 〈刪除堆疊〉。

在 2023 年 1 月 12 日之後建立的任務定義不會自動為其建立 CloudFormation 堆疊。

程序

Amazon ECS console

1. 開啟主控台，網址為 https://console.aws.amazon.com/ecs/v2。
2. 在導覽窗格中，選擇 Task Definitions (任務定義)。
3. 在 [建立新任務定義] 功能表上，選擇 [建立新作業定義]。
4. 在任務定義系列中，請為任務定義指定唯一名稱。
5. 在啟動類型中，選擇應用程式環境。主控台預設值為 AWS Fargate(無伺服器)。Amazon ECS 使用此值執行驗證，以確保任務定義參數對基礎設施類型有效。
6. 對於 Operating system/Architecture (作業系統/架構)，選擇適用於任務的作業系統和 CPU 架構。

   若要在 64 位元 ARM 架構上執行您的工作，請選擇 Linu x/ARM64。如需詳細資訊，請參閱 the section called “執行時間平台”。

   若要在 Windows 容器上執行 AWS Fargate 任務，請選擇支援的 Windows 作業系統。如需詳細資訊，請參閱 the section called “作業系統與架構”。
7. 在 Task size (任務大小) 中，選擇要為任務預留的 CPU 和記憶體大小。CPU 值指定為 vCPU，記憶體則指定為 GB。

   對於在 Fargate 上託管的任務，下表顯示了有效的 CPU 和記憶體組合。

CPU 數值	記憶體數值	AWS Fargate 支援的作業系統
256 (.25 vCPU)	512 MiB、1 GB、2 GB	Linux
512 (.5 vCPU)	1 GB、2 GB、3 GB、4 GB	Linux
1024 (1 vCPU)	2 GB、3 GB、4 GB、5 GB、6 GB、7 GB、8 GB	Linux、Windows
2048 (2 vCPU)	介於 4 GB 與 16 GB 之間，以 1 GB 為單位遞增	Linux、Windows

CPU 數值	記憶體數值	AWS Fargate 支援的作業系統
4096 (4 vCPU)	介於 8 GB 與 30 GB 之間，以 1 GB 為單位遞增	Linux、Windows
8192 (8 vCPU) **Note** 此選項需要 Linux 平台 `1.4.0` 或更新版本。	介於 16 GB 與 60 GB 之間，以 4 GB 為單位遞增	Linux
16384 (16vCPU) **Note** 此選項需要 Linux 平台 `1.4.0` 或更新版本。	介於 32 GB 與 120 GB 之間，以 8 GB 為單位遞增	Linux

對於在 Amazon EC2 上託管的任務，支援的任務 CPU 值介於 128 個 CPU 單位 (0.125 vCPU) 到 10240 個 CPU 單位 (10 vCPU) 之間。若要以 GB 為單位指定記憶體值，請在值之後輸入 GB。例如，若要將「記憶體」值設定為 3GB，請輸入 3 GB。

> **Note**
>
> Windows 容器會忽略任務層級的 CPU 和記憶體參數。

8. 在 Network mode (網路模式) 中選擇要使用的網路模式。預設網路模式為 awsvpc 模式。如需詳細資訊，請參閱 Amazon ECS 任務聯網。

   如果您選擇橋接器，請在連接埠對應下，針對主機連接埠，輸入容器執行個體上要保留給容器的連接埠號碼。

9. (選擇性) 展開「工作角色」區段以設定工作的 AWS Identity and Access Management (IAM) 角色：

   a. 在 Task role (任務角色) 中，選擇要指派給任務的 IAM 角色。任務 IAM 角色為任務中的容器提供許可，以呼叫 AWS API 操作。

   b. 在任務執行角色中，選擇角色。

      如需有關何時使用任務執行角色的資訊，請參閱 the section called “任務執行 IAM 角色”。如果您不需要角色，請選擇「無」。

10. 請為您任務定義中要定義的每個容器完成以下步驟。

    a. 在 Name (名稱) 中，輸入容器的名稱。

    b. 在 Image URI (映像 URI) 中，輸入要用來啟動容器的映像。Amazon ECR 公共圖庫登錄中的映像只能使用 Amazon ECR 公共註冊表名稱來指定。例如，如果`public.ecr.aws/ecs/amazon-ecs-agent:latest`已指定，則會使用託管在 Amazon ECR 公共圖庫上的 Amazon Linux 容器。對於所有其他儲存庫，請使用`repository-url/image:tag`或`repository-url/image@digest`格式來指定存放庫。

    c. 如果您的映像位於 Amazon ECR 以外的私有登錄中，請在私有登錄下，開啟私有登錄身分驗證。然後，在 Secrets Manager ARN 或名稱中，輸入密碼的 Amazon Resource Name (ARN)。

    d. 對於基本容器，如果您的任務定義有兩個或多個容器定義，您可以指定是否應將容器視為必要容器。當容器標記為「基本」時，如果該容器停止，則會停止工作。每個任務定義必須至少包含一個基本容器。

    e. 連接埠映射允許容器存取主機上的連接埠，以傳送或接收流量。在 Port mappings (連接埠映射) 下執行以下其中一項動作：

       - 若您使用 awsvpc 網路模式，請在 Container port (容器連接埠) 和 Protocol (通訊協定) 中，選擇要用於容器的連接埠映射。
       - 若您使用 bridge 網路模式，請在 Container port (容器連接埠) 和 Protocol (通訊協定) 中，選擇要用於容器的連接埠映射。

       選擇 Add more port mappings (新增更多連接埠映射)，以指定其他容器連接埠映射。

    f. 若要授予容器對其根檔案系統的唯讀存取權限，請在唯讀根檔案系統中選取唯讀。

    g. (選擇性) 若要定義不同於工作層級值的容器層級 CPU、GPU 和記憶體限制，請在資源配置限制下執行下列動作：

- 對於 CPU，請輸入 Amazon ECS 容器代理程式為該容器保留的 CPU 單元數目。
- 在 GPU 中，輸入容器執行個體的 GPU 單元數量。

  支援 GPU 的 Amazon EC2 執行個體，每個 GPU 都有 1 個 GPU 單元。如需詳細資訊，請參閱 the section called “GPU 工作負載的工作定義”。
- 在記憶體硬體限制中，輸入要呈現給容器的記憶體容量 (以 GB 為單位)。如果容器嘗試超過硬限制，容器將會停止。
- Docker20.10.0 或更新版本的精靈會為容器保留至少 6 MB (MiB) 的記憶體，因此請勿為容器指定少於 6 MiB 的記憶體。

  Docker19.03.13-ce 或更早版本的精靈會為容器保留至少 4 MiB 的記憶體，因此請勿為容器指定少於 4 MiB 的記憶體。
- 在記憶體軟限制中，輸入為容器保留的記憶體軟限制 (GB)。

  當系統記憶體爭用時，Docker 會嘗試將容器記憶體保持在此軟性限制。如果您未指定任務層級的記憶體，您必須為記憶體硬限制和記憶體軟限制之一 (或兩者) 指定非零整數。如果同時指定兩者，記憶體硬限制必須大於記憶體軟限制。

  視窗容器不支援此功能。

h. (選用) 展開環境變數區段，以指定要注入到容器中的環境變數。您可以使用鍵值對個別指定環境變數，也可以指定託管在 Amazon S3 儲存貯體中的環境變數檔案來大量指定環境變數。若要取得有關如何格式化環境變數檔案的資訊，請參閱〈〉將個別環境變數傳遞至 Amazon ECS 容器。

   當您指定密碼儲存的環境變數時，對於 Key，請輸入密碼名稱。然後 ValueFrom，輸入系統管理器參數存儲秘密或 Secrets Manager 秘密的完整 ARN

i. (選用) 選擇使用日誌收集選項來指定日誌組態。每個可用的日誌驅動程序式都有要指定的日誌驅動程序選項。預設選項會將容器日誌傳送到 Amazon CloudWatch 日誌。其他記錄驅動程式選項是使用來設定 AWS FireLens。如需詳細資訊，請參閱 將 Amazon ECS 日誌傳送到 AWS 服務或 AWS Partner。

   下方更詳細地描述了每個容器日誌目的地。

   - Amazon CloudWatch — 設定任務以將容器日誌傳送到CloudWatch 日誌。系統會提供預設的記錄驅動程式選項，以代表您建立 CloudWatch 記錄群組。若要指定不同的日誌群組名稱，請變更驅動程式選項值。

- 將防護記錄匯出至 Splunk — 設定工作，以將容器記錄傳送至遠端服務的Splunk驅動程式。您必須輸入Splunk網路服務的 URL。T Splunk oken 被指定為秘密選項，因為它可以被視為敏感資料。
- 將日誌匯出至 Amazon 資料防火軟管 — 設定任務以將容器日誌傳送到 Firehose。提供預設的記錄驅動程式選項，可將記錄檔傳送至 Firehose 傳送串流。若要指定不同的交付串流名稱，請變更驅動程式選項值。
- 將日誌匯出到 Amazon Kinesis Data Streams — 設定任務以將容器日誌傳送到 Kinesis Data Streams。提供預設的記錄驅動程式選項，可將記錄檔傳送至 Kinesis Data Streams。若要指定不同的串流名稱，請變更驅動程式選項值。
- 將日誌匯出到 Amazon OpenSearch 服務 — 設定任務以將容器日誌傳送到 OpenSearch 服務網域。務必提供日誌驅動程式選項。
- 將日誌匯出到 Amazon S3 — 設定任務以將容器日誌傳送到 Amazon S3 儲存貯體。提供預設日誌驅動程式選項，但您必須指定有效的 Amazon S3 儲存貯體名稱。

j. (選用) 設定其他容器參數。

若要設定此選項	執行此作業	
Healthcheck 這些是確定容器是否健康的命令。如需詳細資訊，請參閱 使用容器運作狀態檢查判斷 Amazon ECS 任務運作狀態。	展開 HealthCheck，然後設定下列項目： • 在 Command (命令) 欄位中，輸入以逗號分隔的命令清單。您可以用 CMD 作為此命令的開頭，如此能直接執行命令引數；或以 CMD-SHELL 為開頭，藉以使用容器預設的 Shell 來執行命令。如果均尚未指定，則使用 CMD。 • 在 Interval (間隔) 欄位中，輸入每次運作狀態檢查之間的秒數。有效值介於 5 到 30。 • 在 Timeout (逾時) 欄位中，輸入在判定為失敗之前，等待運作狀態檢查成功執行的時間 (以秒為單位)。有效值介於 2 到 60。 • 在 Start period (開始期間) 欄位中，輸入在運作狀態檢查命令執行之前，等待容器引導的時間 (以秒為單位)。有效值介於 0 到 300。 •	

若要設定此選項	執行此作業	
	在 Retries (重試次數) 欄位中，輸入失敗時重試運作狀態檢查命令的次數。有效值介於 1 和 10 之間。	
容器逾時 這些選項決定何時啟動和停止容器。	展開容器逾時，然後設定以下項目： • 若要設定放棄解析容器相依性之前的等待時間，請在 [啟動逾時] 中輸入秒數。 • 若要設定容器本身未正常結束時在容器停止之前等待的時間，請在停止逾時中輸入秒數。	

若要設定此選項	執行此作業	
容器網路設定 這些選項決定是否要在容器內使用聯網功能。	展開容器網路設定，然後設定以下項目： • 若要停用容器聯網功能，請選取關閉聯網。 • 若要設定顯示給容器的 DNS 伺服器 IP 地址，請在 DNS 伺服器中，以個別的行輸入每個伺服器的 IP 地址。 • 若要設定 DNS 網域以搜尋顯示給容器的non-fully-qualified 主機名稱，請在 DNS 搜尋網域中，在不同的行中輸入每個網域。 模式是 **^[a-zA-Z0-9-.]{0,253}[a-zA-Z0-9]$** 。 • 若要設定容器主機名稱，請在主機名稱中輸入容器主機名稱。 • 若要新增附加至容器上 `/etc/hosts` 檔案的主機名稱和 IP 地址映射，請選擇新增額外主機，然後在主機名稱和 IP 地址中輸入主機名稱和 IP 地址。	

若要設定此選項	執行此作業	
Docker配置 這些會覆寫中的值Dockerfile。	展開Docker組態，然後設定下列項目： • 在命令中，輸入容器的可執行命令。 此參數會對應至Docker遠端 API 的 [建立容器] 區段Cmd中，並將選COMMAND項對應至docker run 。此參數會覆寫中的CMD指令Dockerfile。 • 在「進入點」中，輸Docker入傳遞至容器的「入口點」。 此參數會對應至Docker遠端 API 的 [建立容器] 區段Entrypoint 中，並將選--entrypoint 項對應至docker run 。此參數會覆寫中的ENTRYPOINT 指令Dockerfile。 • 在工作目錄中，輸入容器將執行所提供的任何進入點和命令指示的目錄。 此參數會對應至Docker遠端 API 的 [建立容器] 區段WorkingDir 中，	

若要設定此選項	執行此作業	
	並將選--workdir 項對應至docker run。此參數會覆寫中的 WORKDIR指令Dockerfile。	
Ulimits 這些值會覆寫作業系統的預設資源配額設定。 此參數會映射到 Docker Remote API 的建立容器區段中的 Ulimits 以及 docker run 的 --ulimit 選項。	展開 [資源限制] (ulimits)，然後選擇 [新增] ulimit。在「限制名稱」中，選擇限制。然後，在軟限制和硬限制中輸入值。 若要新增其他ulimits，請選擇 [新增] ulimit。	
Docker標籤 此選項會將中繼資料新增至您的容器。 此參數會映射到 Docker Remote API 的建立容器區段中的 Labels 以及 docker run 的 --label 選項。	展開Docker標籤，選擇 [新增金鑰值配對]，然後輸入 [金鑰] 和 [值]。 若要新增其他Docker標籤，請選擇 [新增金鑰值組]。	

若要設定此選項	執行此作業	
容器啟動順序 此選項定義容器啟動和關閉的相依性。容器可包含多個相依性。	展開啟動相依性順序，然後設定下列項目： a. 選擇新增容器相依性。 b. 在容器中，選擇容器。 c. 在條件中，選擇啟動相依性條件。 若要新增其他相依性，請選擇新增容器相依性。	

k. (選用) 選擇 Add more containers (新增更多容器)，以新增其他容器至任務定義。

11. (選擇性)「儲存體」區段可用來擴充 Fargate 上託管之工作的暫時儲存容量。您也可以使用此區段來新增工作的資料磁碟區組態。

    - 若要將 Fargate 任務的可用暫時性儲存擴展到大於 20 gibibytes (GiB) 的預設值，請在 Amount (數量) 中輸入值，可輸入的最大數值為 200 GiB。

12. (選擇性) 若要新增工作定義的資料磁碟區組態，請選擇 [新增磁碟區]，然後依照下列步驟執行。

    a. 在 Volume name (磁碟區名稱) 中，輸入資料磁碟區的名稱。在建立容器掛載點時會使用資料磁碟區名稱。

    b. 對於磁碟區組態，請選取要在建立工作定義時或部署期間設定磁碟區。

    > **Note**
    >
    > 建立任務定義時可設定的磁碟區包括繫結掛載Docker、Amazon EFS 和適用於 Windows 檔案伺服器的 Amazon FSx。執行任務時或建立或更新服務時，可在部署時設定的磁碟區包括 Amazon EBS。

    c. 對於磁碟區類型，請選取與所選組態類型相容的磁碟區類型，然後設定磁碟區類型。

磁碟區類型	步驟	
綁定安裝	a. 選擇 Add mount point (新增掛載點)，然後設定下列項目： • 對於 Container (容器)，選擇掛載點的容器。 • 在來源磁碟區中，選擇要掛載到容器的資料磁碟區。 • 在 Container path (容器路徑) 中，輸入容器的路徑，以掛載磁碟區。 • 在唯讀中，選取容器是否具有對磁碟區的唯讀存取權。 b. 若要新增其他掛載點，請 Add mount point (新增掛載點)。	

磁碟區類型	步驟	
EFS	a. 在 File system ID (檔案系統 ID) 欄位中，選擇 Amazon EFS 檔案系統 ID。 b. (選用) 在 Root directory (根目錄) 欄位中，輸入 Amazon EFS 檔案系統中的目錄，其將掛載作為主機內的根目錄。如果省略此參數，使用 Amazon EFS 磁碟區的根目錄。 如果您計劃使用 EFS 存取點，請將此欄位留空。 c. (選用) 在 Access point (存取點) 欄位中，選擇要使用的存取點 ID。 d. (選用) 若要加密 Amazon EFS 檔案系統與 Amazon ECS 主機之間的資料，或在掛載磁碟區時使用任務執行角色，請選擇 Advanced configurations (進階組態)，然後設定下列項目： • 若要加密 Amazon EFS 檔案系統與 Amazon ECS 主機之間的資料，請選取 Transit encryption (傳輸加	

磁碟區類型	步驟	
	密)，然後在 Port (連接埠) 欄位中輸入要在 Amazon ECS 主機與 Amazon EFS 伺服器之間傳送加密資料時使用的連接埠。如果您未指定傳輸加密連接埠，它會使用 Amazon EFS 掛載協助程式使用的連接埠選擇策略。如需詳細資訊，請參閱《Amazon Elastic File System 使用者指南》中的 EFS 掛載協助程式。 • 若要在掛載 Amazon ECS 檔案系統時，使用任務定義中定義的 Amazon EFS 任務 IAM 角色，請選擇 IAM authorization (IAM 授權)。 e. 選擇 Add mount point (新增掛載點)，然後設定下列項目： • 對於 Container (容器)，選擇掛載點的容器。 • 在來源磁碟區中，選擇要掛載到容器的資料磁碟區。	

磁碟區類型	步驟	
	• 在 Container path (容器路徑) 中，輸入容器的路徑，以掛載磁碟區。 • 在唯讀中，選取容器是否具有對磁碟區的唯讀存取權。 f. 若要新增其他掛載點，請 Add mount point (新增掛載點)。	

磁碟區類型	步驟	
Docker	a. 在驅動程式中，輸入Docker磁碟區組態。Windows 容器僅支援使用本機驅動程式。若要使用綁定掛載，請指定主機。 b. 在 Scope (範圍) 欄位中，選擇磁碟區生命週期。 • 若要在任務開始與停止時讓生命週期持續，請選擇 Task (任務)。 • 若要讓磁碟區在任務停止後可持續執行，請選擇 Shared (共享)。 c. 選擇 Add mount point (新增掛載點)，然後設定下列項目： • 對於 Container (容器)，選擇掛載點的容器。 • 在來源磁碟區中，選擇要掛載到容器的資料磁碟區。 • 在 Container path (容器路徑) 中，輸入容器的路徑，以掛載磁碟區。	

磁碟區類型	步驟	
	• 在唯讀中，選取容器是否具有對磁碟區的唯讀存取權。 d. 若要新增其他掛載點，請 Add mount point (新增掛載點)。	

磁碟區類型	步驟	
FSx for Windows File Server 專用 FSx	a. 在檔案系統 ID 中，選擇 FSx for Windows File Server 檔案系統 ID。 b. 針對根目錄，請輸入目錄，輸入 FSx for Windows File Server 系統內的目錄，以掛載為主機內的根目錄。 c. 在憑證參數中，請選擇憑證的儲存方式。 • 若要使用 AWS Secrets Manager，請輸入秘 Secrets Manager 碼的 Amazon 資源名稱 (ARN)。 • 若要使用 AWS Systems Manager，請輸入 Systems Manager 參數的 Amazon 資源名稱 (ARN)。 d. 對於網域，請輸入由 AWS Directory Service for Microsoft Active Directory (AWS Managed Microsoft AD) 目錄或自我託管的 EC2 Active Directory 託管的完整網域名稱。 e.	

磁碟區類型	步驟	
	選擇 Add mount point (新增掛載點)，然後設定下列項目： • 對於 Container (容器)，選擇掛載點的容器。 • 在來源磁碟區中，選擇要掛載到容器的資料磁碟區。 • 在 Container path (容器路徑) 中，輸入容器的路徑，以掛載磁碟區。 • 在唯讀中，選取容器是否具有對磁碟區的唯讀存取權。 f. 若要新增其他掛載點，請 Add mount point (新增掛載點)。	

磁碟區類型	步驟	
Amazon EBS	a. 選擇 Add mount point (新增掛載點)，然後設定下列項目： • 對於 Container (容器)，選擇掛載點的容器。 • 在來源磁碟區中，選擇要掛載到容器的資料磁碟區。 • 在 Container path (容器路徑) 中，輸入容器的路徑，以掛載磁碟區。 • 在唯讀中，選取容器是否具有對磁碟區的唯讀存取權。 b. 若要新增其他掛載點，請 Add mount point (新增掛載點)。	

13. 若要從另一個容器新增磁碟區，請選擇新增磁碟區來源，然後設定下列項目：
    - 在容器中，選擇容器。
    - 在來源中，選擇包含您要掛載的磁碟區的容器。
    - 在唯讀中，選取容器是否具有對磁碟區的唯讀存取權。
14. (選擇性) 若要使用 AWS Distro for OpenTelemetry整合來設定應用程式追蹤和指標收集設定，請展開監控，然後選取使用指標收集來收集任務的指標，並將其傳送至 Prometheus 的 Amazon CloudWatch 或 Amazon 受管服務。選取此選項後，Amazon ECS 會建立預先設定為

傳送應用程式指標的 AWS Distro for OpenTelemetry容器附屬工具。如需詳細資訊，請參閱 使用應用程式指標關聯 Amazon ECS 應用程式效能。

a. 選取 Amazon CloudWatch 後，您的自訂應用程式指標會以自訂指標的 CloudWatch 形式路由到。如需詳細資訊，請參閱 將應用程式指標匯出到 Amazon CloudWatch。

> **Important**
>
> 將應用程式指標匯出到 Amazon 時 CloudWatch，您的任務定義需要具有所需許可的任務 IAM 角色。如需詳細資訊，請參閱 與 Amazon OpenTelemetry 集成的 AWS 發行版所需的 IAM 許可 CloudWatch。

b. 若您選取 Amazon Managed Service and Prometheus (Prometheus libraries instrumentation) (Amazon Managed Service for Prometheus (Prometheus 程式庫檢測))，則您的任務層級 CPU、記憶體、網路，以及儲存指標和自定應用程式指標，都將路由至 Amazon Managed Service for Prometheus。對於 Workspace 遠端寫入端點，請輸入Prometheus工作區的遠端寫入端點 URL。對於抓取目標，請輸入 AWS Distro for OpenTelemetry收集器可用於抓取指標數據的主機和端口。如需詳細資訊，請參閱 將應用程式指標匯出至 Amazon Managed Service for Prometheus。

> **Important**
>
> 若將應用程式指標匯出至 Amazon Managed Service for Prometheus，您的任務定義需要具有所需許可的任務 IAM 角色。如需詳細資訊，請參閱 AWS 發行版所需的 IAM 許可，以便與 Prometheus 的 Amazon 託管服務 OpenTelemetry 整合。

c. 當您選取 Prometheus 的 Amazon 受管服務 (OpenTelemetry 儀器) 時，您的任務層級 CPU、記憶體、網路和儲存指標以及您的自訂應用程式指標都會路由至 Prometheus 的 Amazon 受管服務。對於 Workspace 遠端寫入端點，請輸入Prometheus工作區的遠端寫入端點 URL。如需詳細資訊，請參閱 將應用程式指標匯出至 Amazon Managed Service for Prometheus。

> **Important**
>
> 若將應用程式指標匯出至 Amazon Managed Service for Prometheus，您的任務定義需要具有所需許可的任務 IAM 角色。如需詳細資訊，請參閱 AWS 發行版所需的 IAM 許可，以便與 Prometheus 的 Amazon 託管服務 OpenTelemetry 整合。

15. (選用) 展開 Tags (標籤) 區段，將標籤作為鍵值對新增至任務定義中。
    - [新增標籤] 選擇新增標籤，然後執行下列操作：
      - 在索引鍵中，輸入索引鍵名稱。
      - 在值中，進入索引鍵值。
    - [移除標籤] 在標籤旁邊，選擇 移除標籤。
16. 選擇建立來註冊任務定義。

Amazon ECS console JSON editor

1. 開啟主控台，網址為 https://console.aws.amazon.com/ecs/v2。
2. 在導覽窗格中，選擇 Task Definitions (任務定義)。
3. 在 [建立新工作定義] 功能表上，選擇 [使用 JSON 建立新工作定義]。
4. 在 JSON 編輯工具方塊中，編輯您的 JSON 檔案，

   JSON 必須通過 the section called “JSON 驗證” 中指定的驗證檢查。
5. 選擇 建立。

使用主控台更新 Amazon ECS 任務定義

任務定義修訂版是目前任務定義的複本，並以新參數值取代現有參數值。您未修改的所有參數都位於新修訂版中。

若要更新任務定義，請建立任務定義修訂版。如果任務定義是用於服務，則您必須更新該服務才能使用更新的任務定義。

在建立修訂版時，您可以修改以下容器屬性和環境屬性。

- 容器映像 URI
- 連接埠映射
- 環境變數
- 任務大小
- 容器大小
- 任務角色

- 任務執行角色
- 磁碟區和容器掛載點
- 私有登錄檔

JSON 驗證

Amazon ECS 主控台 JON 編輯器會對 JSON 檔案的以下方面進行驗證：

- 檔案為有效的 JSON 檔案
- 檔案不包含任何無關的金鑰
- 該檔案包含 `familyName` 參數
- `containerDefinitions` 下至少有一個項目

程序

Amazon ECS console

1. 開啟主控台，網址為 https://console.aws.amazon.com/ecs/v2。
2. 從導覽列中選擇包含您任務定義的區域。
3. 在導覽窗格中，選擇 Task Definitions (任務定義)。
4. 選擇任務定義。
5. 選擇任務定義修訂版，然後選擇建立新修訂版、建立新修訂版。
6. 在 Create new task definition revision (建立新任務定義修訂版) 頁面上進行變更。例如，若要變更現有的容器定義 (例如容器映像、記憶體限制或連接埠映射)，請選取容器，然後選擇進行變更。
7. 驗證資訊，然後選擇更新。
8. 如果您的任務定義是用於服務，請使用更新的任務定義來更新您的服務。如需詳細資訊，請參閱 使用主控台更新 Amazon ECS 服務。

Amazon ECS console JSON editor

1. 開啟主控台，網址為 https://console.aws.amazon.com/ecs/v2。
2. 在導覽窗格中，選擇 Task Definitions (任務定義)。

3. 選擇 Create new revision (建立新修訂版)，以及 Create new revision with JSON (使用 JSON 建立新修訂版)。
4. 在 JSON 編輯工具方塊中，編輯您的 JSON 檔案，

   JSON 必須通過 the section called “JSON 驗證” 中指定的驗證檢查。
5. 選擇 建立。

使用主控台取消註冊 Amazon ECS 任務定義修訂

當您不再需要 Amazon ECS 中的特定任務定義修訂時，可以取消註冊任務定義修訂，以便當您要執行任務或更新服務時，該修訂不再顯示在 `ListTaskDefinition` API 呼叫或主控台中。

當您取消註冊任務定義修訂時，會立即標示為 `INACTIVE`。參考 `INACTIVE` 任務定義修訂的現有任務和服務會繼續執行，而不會中斷。參考 `INACTIVE` 任務定義修訂的現有服務仍然可以透過修改服務所需的計數進行擴展或縮減。

您不能使用 `INACTIVE` 任務定義修訂來執行新任務或建立新服務。您也無法更新現有服務以參考 `INACTIVE` 任務定義修訂 (雖然取消註冊後最多有 10 分鐘的空檔，但這些限制在此期間尚未生效)。

> **Note**
>
> 取消註冊任務系列中的所有修訂時，任務定義系列會移至 `INACTIVE` 清單。新增 `INACTIVE` 任務定義修訂，會將任務定義系列移至 `ACTIVE` 清單。
> 此時，`INACTIVE` 任務定義修訂會在您的帳戶中無限期地保持可發現狀態。不過，此行為未來可能變更。因此，你不應該依賴持續超出任何相關任務和服務生命週期的 `INACTIVE` 任務定義修訂。

AWS CloudFormation 堆疊

下列行為適用於 2023 年 1 月 12 日之前在新的 Amazon ECS 主控台中建立的任務定義。

建立任務定義時，Amazon ECS 主控台會自動建立名稱開頭為的 CloudFormation堆疊。`ECS-Console-V2-TaskDefinition-`如果您使用 AWS CLI 或 AWS SDK 取消註冊任務定義，則必須手動刪除任務定義堆疊。若要取得更多資訊，請參閱《使用指南》中的AWS CloudFormation 〈刪除堆疊〉。

在 2023 年 1 月 12 日之後建立的任務定義不會自動為其建立 CloudFormation 堆疊。

程序

取消註冊新任務定義 (Amazon ECS 主控台)

1. 開啟主控台，網址為 https://console.aws.amazon.com/ecs/v2。
2. 從導覽列中選擇包含您任務定義的區域。
3. 在導覽窗格中，選擇 Task Definitions (任務定義)。
4. 在 Task Definitions (任務定義) 頁面，選擇包含您想要取消註冊之一個或多個修訂版的任務定義系列。
5. 在任務定義名稱頁面上，選取要刪除的修訂版，然後選擇動作、取消註冊。
6. 驗證 Deregister (取消註冊) 視窗中的資訊，然後選擇 Deregister (取消註冊) 以完成操作。

使用主控台刪除 Amazon ECS 任務定義修訂

當 Amazon ECS 中不再需要特定任務定義修訂時，您可以刪除任務定義修訂。

任務定義修訂版刪除後，任務定義修訂版會立即從 INACTIVE 狀態轉變為 DELETE_IN_PROGRESS 狀態。參考 DELETE_IN_PROGRESS 任務定義修訂版的現有任務和服務會繼續執行，不會中斷。

您不能使用狀態為 DELETE_IN_PROGRESS 的任務定義修訂版來執行新任務或建立新服務。您也無法更新現有服務以參考狀態為 DELETE_IN_PROGRESS 的任務定義修訂版。

當您刪除所有 INACTIVE 任務定義修訂時，任務定義名稱不會顯示在主控台中，也不會在 API 中傳回。如果任務定義修訂版本DELETE_IN_PROGRESS處於狀態，則任務定義名稱會顯示在主控台中，並在 API 中傳回。Amazon ECS 會保留任務定義名稱，下次您使用該名稱建立任務定義時，修訂版本將會遞增。

可以封鎖刪除的 Amazon ECS 資源

如果有任何 Amazon ECS 資源依賴於任務定義修訂版本，則任務定義刪除請求將不會完成。以下資源可能會阻止任務定義遭刪除：

- Amazon ECS 任務 - 需要任務定義才能讓任務保持良好的運作狀態。
- Amazon ECS 部署和任務集 - 當為 Amazon ECS 部署或任務集啟動擴展事件時，需要任務定義。

如果您的任務定義保持在DELETE_IN_PROGRESS狀態，您可以使用控制台或 AWS CLI 來識別，然後停止阻止任務定義刪除的資源。

移除封鎖的資源後刪除任務定義

移除封鎖任務定義刪除的資源後，以下規則適用：

- Amazon ECS 任務 - 任務停止後，任務定義刪除最多可能需要 1 小時才能完成。
- Amazon ECS 部署和任務集 - 刪除部署或任務集後，刪除任務定義最多可能需要 24 小時才能完成。

程序

刪除任務定義 (Amazon ECS 主控台)

您必須先取消註冊任務定義修訂版，然後才能刪除它。如需詳細資訊，請參閱 the section called “使用主控台取消註冊任務定義修訂版”。

1. 開啟主控台，網址為 https://console.aws.amazon.com/ecs/v2。
2. 從導覽列中選擇包含您任務定義的區域。
3. 在導覽窗格中，選擇 Task Definitions (任務定義)。
4. 在任務定義頁面，選擇包含您想要刪除之一或多個修訂版的任務定義系列。
5. 在 [工作定義名稱] 頁面上，選取要刪除的修訂，然後選擇 [動作] > [刪除]。

   如果「刪除」無法使用，您必須取消註冊任務定義。
6. 確認 [刪除] 確認方塊中的資訊，然後選擇 [刪除] 以完成。

Amazon ECS 任務定義使用案例

進一步瞭解如何撰寫各種 AWS 服務和功能的工作定義。

視您的工作負載而定，需要設定某些作業定義參數。此外，對於 EC2 啟動類型，您必須選擇為工作負載引入的特定執行個體。

主題

- GPU 工作負載的 Amazon ECS 任務定義
- 適用於視訊轉碼工作負載的 Amazon ECS 任務定義
- 適用於 AWS 神經元機器學習工作負載的 Amazon ECS 任務定義
- 適用於深度學習執行個體的 Amazon ECS 任務定義
- 適用於 64 位元 ARM 工作負載的 Amazon ECS 任務定義

- 將 Amazon ECS 日誌傳送到 CloudWatch
- 將 Amazon ECS 日誌傳送到 AWS 服務或 AWS Partner
- 在 Amazon ECS 中使用非AWS 容器映像
- 將個別環境變數傳遞至 Amazon ECS 容器
- 將環境變量傳遞到 Amazon ECS 容器
- 將敏感資料傳遞至 Amazon ECS 容器

GPU 工作負載的 Amazon ECS 任務定義

您建立具有支援 GPU 容器執行個體的叢集時，Amazon ECS 支援使用 GPU 的工作負載。使用 p2、p3、p5、g3、g4 和 g5 執行個體類型的 Amazon EC2 GPU 型容器執行個體可讓您存取 NVIDIA GPU。如需詳細資訊，請參閱 Amazon EC2 使用者指南中的 Linux 加速運算執行個體。

Amazon ECS 提供的 GPU 最佳化 AMI 可與預先設定的 NVIDIA 核心驅動程式和 Docker GPU 執行時間搭配使用。如需詳細資訊，請參閱 Amazon ECS 優化的 Linux AMI。

您可以在容器定義中指定 GPU 數量，以便在容器層級考量任務放置。Amazon ECS 會對支援 GPU 的可用容器執行個體排程並將實體 GPU 固定至適當容器以獲得最佳效能。

支援下列 Amazon EC2 GPU 型執行個體類型。如需詳細資訊，請參閱 Amazon EC2 P2 執行個體、亞馬遜 EC2 P3 執行個體、亞馬 Amazon EC2 EC2 P4d 執行個體、亞 Amazon EC2 G3 執行個體、Amazon EC2 G4 執行個體、Amazon EC2 G5 執行個體和 Amazon EC2 G6 執行個體。

執行個體類型	GPU	GPU 記憶體 (GiB)	vCPU	記憶體 (GiB)
p3.2xlarge	1	16	8	61
p3.8xlarge	4	64	32	244
p3.16xlarge	8	128	64	488
p3dn.24xlarge	8	256	96	768
p4d.24xlarge	8	320	96	1152
p5.48xlarge	8	640	192	2048

執行個體類型	GPU	GPU 記憶體 (GiB)	vCPU	記憶體 (GiB)
g3s.xlarge	1	8	4	30.5
g3.4xlarge	1	8	16	122
g3.8xlarge	2	16	32	244
g3.16xlarge	4	32	64	488
g4dn.xlarge	1	16	4	16
g4dn.2xlarge	1	16	8	32
g4dn.4xlarge	1	16	16	64
g4dn.8xlarge	1	16	32	128
g4dn.12xlarge	4	64	48	192
g4dn.16xlarge	1	16	64	256
g5.xlarge	1	24	4	16
g5.2xlarge	1	24	8	32
g5.4xlarge	1	24	16	64
g5.8xlarge	1	24	32	128
g5.16xlarge	1	24	64	256
g5.12xlarge	4	96	48	192
g5.24xlarge	4	96	96	384
g5.48xlarge	8	192	192	768
大	1	24	4	16
大	1	24	8	32

執行個體類型	GPU	GPU 記憶體 (GiB)	vCPU	記憶體 (GiB)
大	1	24	16	64
大	1	24	32	128
大	1	24	64	256
大	4	96	48	192
大	4	96	48	192
大	8	192	192	768
g6. 金屬	8	192	192	768
GR6.4 x 大	1	24	16	128
克 6.8 倍大	1	24	32	256

您可以透過查詢 AWS Systems Manager 參數存放區 API，擷取適用於 Amazon ECS 最佳化 AMI 的 Amazon 機器映像 (AMI) 識別碼。若使用此參數，您無需手動查詢 Amazon ECS 最佳化 AMI ID。如需有關 Systems Manager 參數存放區 API 的詳細資訊，請參閱GetParameter。您使用的使用者必須擁有 `ssm:GetParameter` IAM 許可，才能擷取 Amazon ECS 最佳化 AMI 中繼資料。

```
aws ssm get-parameters --names /aws/service/ecs/optimized-ami/amazon-linux-2/gpu/
recommended --region us-east-1
```

考量事項

> **Note**
>
> 對 g2 執行個體系列類型的支援已被棄用。
> p2 執行個體系列類型僅在早於 Amazon ECS GPU 最佳化 AMI `20230912` 之前的版本上支援。如果您需要繼續使用 p2 執行個體，請參閱 您需要 P2 執行個體時要採取的動作。
> 在這兩種執行個體系列類型上就地更新 NVIDIA/CUDA 驅動程式，將導致潛在的 GPU 工作負載失敗。

建議您開始在 Amazon ECS 上使用 GPU 前考量下列事項。

- 您的叢集可以混合 GPU 和非 GPU 容器執行個體。
- 您可在外部執行個體上執行 GPU 工作負載。在您叢集中註冊外部執行個體時，請確保安裝指令碼中包含 `--enable-gpu` 標記。如需詳細資訊，請參閱 將外部執行個體註冊到 Amazon ECS 叢集。
- 您必須在代理程式組態檔案中將 `ECS_ENABLE_GPU_SUPPORT` 設定為 `true`。如需詳細資訊，請參閱 the section called “容器代理程式組態”。
- 執行任務或建立服務時，您可以在設定任務放置限制條件時使用執行個體類型屬性，來決定任務在哪個容器執行個體啟動。這樣便可更有效地使用您的資源。如需詳細資訊，請參閱 Amazon ECS 如何在容器執行個體上放置任務。

  以下範例會在預設叢集的 `g4dn.xlarge` 容器執行個體啟動任務。

  ```
  aws ecs run-task --cluster default --task-definition ecs-gpu-task-def \
       --placement-constraints type=memberOf,expression="attribute:ecs.instance-type ==
    g4dn.xlarge" --region us-east-2
  ```

- 對於在容器定義中指定 GPU 資源需求的每個容器，Amazon ECS 將容器執行時間設定為 NVIDIA 容器執行時間。
- NVIDIA 容器執行時間需要在容器中設定一些環境變數才能正常運作。如需這些環境變數的清單，請參閱使用 Docker 進行特殊組態。Amazon ECS 將 `NVIDIA_VISIBLE_DEVICES` 環境變數值設定為 Amazon ECS 指派給容器的 GPU 裝置 ID 清單。對於其他必要的環境變數，Amazon ECS 不會對其進行設定。因此，請確保容器映像會設定它們，或在容器定義中設定它們。
- Amazon ECS GPU 最佳化 AMI 版本 `20230929` 和更新版本支援 p5 執行個體類型系列。
- Amazon ECS GPU 最佳化 AMI 版本 `20230913` 和更新版本支援 g4 執行個體類型系列。如需詳細資訊，請參閱 Amazon ECS 優化的 Linux AMI。Amazon ECS 主控台的 Create Cluster (建立叢集) 工作流程中不支援此系列。若要使用這些執行個體類型，您必須使用 Amazon EC2 主控台或 API AWS CLI，並將執行個體手動註冊到叢集。
- p4d.24xlarge 執行個體類型僅適用於 CUDA 11 或更新版本。
- Amazon ECS GPU 最佳化 AMI 已啟用 IPv6，這會在使用 yum 時導致問題。透過設定 yum 以搭配使用 IPv4 與下列命令，可解決此問題。

  ```
  echo "ip_resolve=4" >> /etc/yum.conf
  ```

- 在建置一個不使用 NVIDIA/CUDA 基礎映像的容器映像時，必須將 `NVIDIA_DRIVER_CAPABILITIES` 容器執行時間變數設定為下列其中一個值：
  - `utility,compute`

- `all`

  如需如何設定變數的詳細資訊，請參閱 NVIDIA 網站上的控制 NVIDIA 容器執行時間。

- Windows 容器不支援 GPU。

啟動適用於 Amazon ECS 的 GPU 容器執行個體

若要在 Amazon ECS 上使用 GPU 執行個體，您需要建立啟動範本、使用者資料檔案，然後啟動執行個體。

然後，您可以執行使用針對 GPU 設定的工作定義的工作。

使用啟動範本

您可以建立啟動範本。

- 建立使用 Amazon ECS 最佳化 GPU AMI 識別碼適用於 AMI 的啟動範本。如需如何建立啟動範本的詳細資訊，請參閱使用 Amazon EC2 使用者指南中定義的參數建立新的啟動範本。

  將上一步的 AMI ID 用於 Amazon 機器映像。如需如何使用 Systems Manager 參數指定 AMI ID 的詳細資訊，請參閱 Amazon EC2 使用者指南中的在啟動範本中指定 Systems Manager 參數。

  將以下內容新增至啟動範本中的使用者資料。以您的叢集名稱取代 *cluster-name*。

  ```
  #!/bin/bash
  echo ECS_CLUSTER=cluster-name >> /etc/ecs/ecs.config;
  echo ECS_ENABLE_GPU_SUPPORT=true >> /etc/ecs/ecs.config
  ```

使用 AWS CLI

您可以使用 AWS CLI 來啟動容器執行個體。

1. 建立稱為 `userdata.toml` 的檔案。此檔案會用於執行個體使用者資料。以您的叢集名稱取代 *cluster-name*。

   ```
   #!/bin/bash
   echo ECS_CLUSTER=cluster-name >> /etc/ecs/ecs.config;
   echo ECS_ENABLE_GPU_SUPPORT=true >> /etc/ecs/ecs.config
   ```

2. 執行下列命令以取得 GPU AMI 識別碼。您會在以下步驟中使用此 ID。

```
aws ssm get-parameters --names /aws/service/ecs/optimized-ami/amazon-linux-2/gpu/
recommended --region us-east-1
```

3. 執行下列命令以啟動 GPU 執行個體。請記得替換以下參數：

   - 將*###*替換為執行個體將在其中啟動的私有或公有子網路的 ID。
   - 使用上一個步驟的 *AMI ##### gpu_* ami。
   - 將 *t3.large* 替換為您要使用的執行個體類型。
   - 將 *region* 替換為區域代碼。

```
aws ec2 run-instances --key-name ecs-gpu-example \
   --subnet-id subnet \
   --image-id gpu_ami \
   --instance-type t3.large \
   --region region \
   --tag-specifications 'ResourceType=instance,Tags=[{Key=GPU,Value=example}]' \
   --user-data file://userdata.toml \
   --iam-instance-profile Name=ecsInstanceRole
```

4. 執行下列命令來驗證容器執行個體是否已註冊至叢集。當您執行此命令時，請記得替代下列參數：

   - 將 *cluster* 替代為叢集名稱。
   - 將 *region* 替換為您的區域代碼。

```
aws ecs list-container-instances --cluster cluster-name --region region
```

在 Amazon ECS 任務定義中指定 GPU

若要使用容器執行個體的 GPU 和 Docker GPU 執行時間，確保您在任務定義中指定容器所需的 GPU 數量。放置支援 GPU 的容器後，Amazon ECS 容器代理程式會將所需數量的實體 GPU 固定至適當的容器。為任務中所有容器保留的 GPU 數量不可超過任務啟動所在之容器執行個體上可用的 GPU 數量。如需詳細資訊，請參閱 使用主控台建立 Amazon ECS 任務定義。

> **Important**
>
> 如果未在任務定義中指定 GPU 要求，該任務會使用預設 Docker 執行時間。

下列顯示任務定義中 GPU 要求的 JSON 格式：

```
{
  "containerDefinitions": [
     {
        ...
        "resourceRequirements" : [
            {
               "type" : "GPU",
               "value" : "2"
            }
        ],
     },
...
}
```

以下範例會示範指定 GPU 要求的 Docker 容器語法。此容器會使用 2 個 GPU，執行 `nvidia-smi` 公用程式，然後結束。

```
{
  "containerDefinitions": [
    {
      "memory": 80,
      "essential": true,
      "name": "gpu",
      "image": "nvidia/cuda:11.0.3-base",
      "resourceRequirements": [
         {
           "type":"GPU",
           "value": "2"
         }
      ],
      "command": [
        "sh",
        "-c",
        "nvidia-smi"
      ],
      "cpu": 100
    }
  ],
  "family": "example-ecs-gpu"
}
```

您需要 P2 執行個體時要採取的動作

如果您需要使用 P2 執行個體，您可以使用下列其中一個選項來繼續使用執行個體。

您必須修改這兩個選項的執行個體使用者資料。如需詳細資訊，請參閱 Amazon EC2 使用者指南中的使用執行個體使用者資料。

使用最新支援的 GPU 最佳化 AMI

您可以使用 GPU 最佳化 AMI 的 20230906 版本，並將下列項目新增至執行個體使用者資料。

以您的叢集名稱取代 cluster-name。

```
#!/bin/bash
echo "exclude=*nvidia* *cuda*" >> /etc/yum.conf
echo "ECS_CLUSTER=cluster-name" >> /etc/ecs/ecs.config
```

使用最新的 GPU 最佳化 AMI，並更新使用者資料

您可以將下列內容新增至執行個體使用者資料。這將解除安裝 Nvidia 535/Cuda12.2 驅動程式，然後安裝 Nvidia 470/Cuda11.4 驅動程式並修復該版本。

```
#!/bin/bash
yum remove -y cuda-toolkit* nvidia-driver-latest-dkms*
tmpfile=$(mktemp)
cat >$tmpfile <<EOF
[amzn2-nvidia]
name=Amazon Linux 2 Nvidia repository
mirrorlist=\$awsproto://\$amazonlinux.\$awsregion.\$awsdomain/\$releasever/amzn2-
nvidia/latest/\$basearch/mirror.list
priority=20
gpgcheck=1
gpgkey=https://developer.download.nvidia.com/compute/cuda/repos/rhel7/
x86_64/7fa2af80.pub
enabled=1
exclude=libglvnd-*
EOF

mv $tmpfile /etc/yum.repos.d/amzn2-nvidia-tmp.repo
yum install -y system-release-nvidia cuda-toolkit-11-4 nvidia-driver-latest-
dkms-470.182.03
yum install -y libnvidia-container-1.4.0 libnvidia-container-tools-1.4.0 nvidia-
container-runtime-hook-1.4.0 docker-runtime-nvidia-1
```

```
echo "exclude=*nvidia* *cuda*" >> /etc/yum.conf
nvidia-smi
```

建立您自己的 P2 相容 GPU 最佳化 AMI

您可以建立與 P2 執行個體相容的自訂 Amazon ECS GPU 最佳化 AMI，然後使用 AMI 啟動 P2 執行個體。

1. 執行下列命令以複製 `amazon-ecs-ami repo`。

   ```
   git clone https://github.com/aws/amazon-ecs-ami
   ```

2. 設定所需的 Amazon ECS 代理程式，並在 `release.auto.pkrvars.hcl` 或 `overrides.auto.pkrvars.hcl` 中取得 Amazon Linux AMI 版本。
3. 執行以下命令建立私有 P2 相容的 EC2 AMI。

   將區域替換為具有執行個體區域的區域。

   ```
   REGION=region make al2keplergpu
   ```

4. 將 AMI 與下列執行個體使用者資料搭配使用，以連線至 Amazon ECS 叢集。

   以您的叢集名稱取代 cluster-name。

   ```
   #!/bin/bash
   echo "ECS_CLUSTER=cluster-name" >> /etc/ecs/ecs.config
   ```

適用於視訊轉碼工作負載的 Amazon ECS 任務定義

若要在 Amazon ECS 上使用影片轉碼工作負載，請註冊 Amazon EC2 VT1 執行個體。註冊這些執行個體後，您可以在 Amazon ECS 將即時和預先渲染的影片轉碼工作負載作為任務執行。Amazon EC2 VT1 執行個體使用 Xilinx U30 媒體轉碼卡來加速即時和預先渲染的影片轉碼工作負載。

> **Note**
>
> 如需如何在 Amazon ECS 以外的容器中執行影片轉碼工作負載的說明，請參閱 Xilinx 文件。

考量事項

當您開始在 Amazon ECS 上部署 VT1 之前，請考量下列事項：

- 您的叢集可包含 VT1 和非 VT1 執行個體組合。
- 您需要使用具有加速 AVC (H.264) 和 HEVC (H.265) 解碼器的 Xilinx U30 媒體轉碼卡的 Linux 應用程式。

> **Important**
>
> 使用其他解碼器的應用程式在 VT1 執行個體上可能沒有提升效能。

- U30 卡上只能執行一個轉碼任務。每張卡都有兩個與其關聯的裝置。只要您的每個 VT1 執行個體都有轉碼卡，您就可以執行所需數量的轉碼任務。
- 建立服務或執行獨立任務時，您可以在設定任務置放限制條件時使用執行個體類型屬性。這可確保在您指定的容器執行個體上啟動任務。此舉有助於確保您有效地運用資源，並確保您的影片轉碼工作負載任務位於 VT1 執行個體上。如需詳細資訊，請參閱 Amazon ECS 如何在容器執行個體上放置任務。

  在以下範例中，在您的 `default` 叢集的 `vt1.3xlarge` 執行個體上執行任務。

```
aws ecs run-task \
     --cluster default \
     --task-definition vt1-3xlarge-xffmpeg-processor \
     --placement-constraints type=memberOf,expression="attribute:ecs.instance-type ==
 vt1.3xlarge"
```

- 您可以設定容器，以便在主機容器執行個體上使用可用的特定 U30 卡。您可使用 `linuxParameters` 參數並指定裝置詳細資訊來實現這一操作。如需詳細資訊，請參閱 任務定義需求。

使用 VT1 AMI

您有兩種可選的選項，在 Amazon EC2 上執行 Amazon ECS 容器執行個體的 AMI。第一個選項是使用 AWS Marketplace上的 Xilinx 正式 AMI。第二個選項是從範本儲存庫建置自己的 AMI。

- 賽靈思提供 AMI 上的. AWS Marketplace

- Amazon ECS 提供了一個範本儲存庫，您可以使用該範本儲存庫為影片轉碼工作負載建置 AMI。此 AMI 隨附 Xilinx U30 驅動程式。您可以在GitHub上找到包含封裝程式指令碼的儲存庫。如需 Packer 的詳細資訊，請參閱 Packer documentation (《Packer 文件》)。

任務定義需求

若要在 Amazon ECS 上執行視影片轉碼容器，您的任務定義必須包含使用加速 H.264/AVC 和 H.265/HEVC 解碼器的影片轉碼應用程式。您可以按照 Xil GitHub inx 上的步驟構建容器映像。

任務定義必須根據執行個體類型專門設定。執行個體類型為 3xlarge、6xlarge 和 24xlarge。您必須設定容器，以便在主機容器執行個體上使用可用的特定 Xilinx U30 裝置。您可以使用 `linuxParameters` 參數進行該動作。下表詳細 SoCs 說明每個執行個體類型專屬的卡片和裝置。

執行個體類型	vCPU	RAM (GiB)	U30 加速器卡	可定址 XCU30 SoC 裝置	裝置路徑
vt1.3xlarge	12	24	1	2	`/dev/dri/renderD128 ,/dev/dri/renderD129`
vt1.6xlarge	24	48	2	4	`/dev/dri/renderD128 ,/dev/dri/renderD129 ,/dev/dri/renderD130 ,/dev/dri/renderD131`

執行個體類型	vCPU	RAM (GiB)	U30 加速器卡	可定址 XCU30 SoC 裝置	裝置路徑
vt1.24xlarge	96	182	8	16	/dev/dri/renderD128 ,/dev/dri/renderD129 ,/dev/dri/renderD130 ,/dev/dri/renderD131 ,/dev/dri/renderD132 ,/dev/dri/renderD133 ,/dev/dri/renderD134 ,/dev/dri/renderD135 ,/dev/dri/renderD136 ,/dev/dri/renderD137 ,/dev/dri/renderD13

執行個體類型	vCPU	RAM (GiB)	U30 加速器卡	可定址 XCU30 SoC 裝置	裝置路徑
					8 ,/dev/dri/renderD139 ,/dev/dri/renderD140 ,/dev/dri/renderD141 ,/dev/dri/renderD142 ,/dev/dri/renderD143

Important

如果任務定義列出 EC2 執行個體沒有的裝置，則任務無法執行。當任務失敗時，下列錯誤訊息會出現在 `stoppedReason`：`CannotStartContainerError: Error response from daemon: error gathering device information while adding custom device "/dev/dri/renderD130": no such file or directory`

在 Amazon ECS 任務定義中指定視訊轉碼

在以下範例中，提供了用於 Amazon EC2 上 Linux 容器的任務定義的語法。此任務定義適用於遵循 Xilinx 文件提供的程序建置的容器映像。如果您使用此範例，請用自己的映像替換 `image`，接著將影片檔案複製到 `/home/ec2-user` 目錄的執行個體。

vt1.3xlarge

1. 使用下列內容建立名為 vt1-3xlarge-ffmpeg-linux.json 的文字檔案。

```
{
    "family": "vt1-3xlarge-xffmpeg-processor",
    "requiresCompatibilities": ["EC2"],
    "placementConstraints": [
        {
            "type": "memberOf",
            "expression": "attribute:ecs.os-type == linux"
        },
        {
            "type": "memberOf",
            "expression": "attribute:ecs.instance-type == vt1.3xlarge"
        }
    ],
    "containerDefinitions": [
        {
            "entryPoint": [
                "/bin/bash",
                "-c"
            ],
            "command": ["/video/ecs_ffmpeg_wrapper.sh"],
            "linuxParameters": {
                "devices": [
                    {
                        "containerPath": "/dev/dri/renderD128",
                        "hostPath": "/dev/dri/renderD128",
                        "permissions": [
                            "read",
                            "write"
                        ]
                    },
                    {
                        "containerPath": "/dev/dri/renderD129",
                        "hostPath": "/dev/dri/renderD129",
                        "permissions": [
                            "read",
                            "write"
                        ]
                    }
                ]
```

```
            },
            "mountPoints": [
                {
                    "containerPath": "/video",
                    "sourceVolume": "video_file"
                }
            ],
            "cpu": 0,
            "memory": 12000,
            "image": "0123456789012.dkr.ecr.us-west-2.amazonaws.com/aws/xilinx-
xffmpeg",
            "essential": true,
            "name": "xilinix-xffmpeg"
        }
    ],
    "volumes": [
        {
            "name": "video_file",
            "host": {"sourcePath": "/home/ec2-user"}
        }
    ]
}
```

2. 註冊任務定義。

```
aws ecs register-task-definition --family vt1-3xlarge-xffmpeg-processor --cli-
input-json file://vt1-3xlarge-xffmpeg-linux.json --region us-east-1
```

vt1.6xlarge

1. 使用下列內容建立名為 vt1-6xlarge-ffmpeg-linux.json 的文字檔案。

```
{
    "family": "vt1-6xlarge-xffmpeg-processor",
    "requiresCompatibilities": ["EC2"],
    "placementConstraints": [
        {
            "type": "memberOf",
            "expression": "attribute:ecs.os-type == linux"
        },
        {
            "type": "memberOf",
```

```
            "expression": "attribute:ecs.instance-type == vt1.6xlarge"
        }
    ],
    "containerDefinitions": [
        {
            "entryPoint": [
                "/bin/bash",
                "-c"
            ],
            "command": ["/video/ecs_ffmpeg_wrapper.sh"],
            "linuxParameters": {
                "devices": [
                    {
                        "containerPath": "/dev/dri/renderD128",
                        "hostPath": "/dev/dri/renderD128",
                        "permissions": [
                            "read",
                            "write"
                        ]
                    },
                    {
                        "containerPath": "/dev/dri/renderD129",
                        "hostPath": "/dev/dri/renderD129",
                        "permissions": [
                            "read",
                            "write"
                        ]
                    },
                    {
                        "containerPath": "/dev/dri/renderD130",
                        "hostPath": "/dev/dri/renderD130",
                        "permissions": [
                            "read",
                            "write"
                        ]
                    },
                    {
                        "containerPath": "/dev/dri/renderD131",
                        "hostPath": "/dev/dri/renderD131",
                        "permissions": [
                            "read",
                            "write"
                        ]
                    }
```

```
                ]
            },
            "mountPoints": [
                {
                    "containerPath": "/video",
                    "sourceVolume": "video_file"
                }
            ],
            "cpu": 0,
            "memory": 12000,
            "image": "0123456789012.dkr.ecr.us-west-2.amazonaws.com/aws/xilinx-
xffmpeg",
            "essential": true,
            "name": "xilinix-xffmpeg"
        }
    ],
    "volumes": [
        {
            "name": "video_file",
            "host": {"sourcePath": "/home/ec2-user"}
        }
    ]
}
```

2. 註冊任務定義。

```
aws ecs register-task-definition --family vt1-6xlarge-xffmpeg-processor --cli-
input-json file://vt1-6xlarge-xffmpeg-linux.json --region us-east-1
```

vt1.24xlarge

1. 使用下列內容建立名為 vt1-24xlarge-ffmpeg-linux.json 的文字檔案。

```
{
    "family": "vt1-24xlarge-xffmpeg-processor",
    "requiresCompatibilities": ["EC2"],
    "placementConstraints": [
        {
            "type": "memberOf",
            "expression": "attribute:ecs.os-type == linux"
        },
        {
```

```
            "type": "memberOf",
            "expression": "attribute:ecs.instance-type == vt1.24xlarge"
        }
    ],
    "containerDefinitions": [
        {
            "entryPoint": [
                "/bin/bash",
                "-c"
            ],
            "command": ["/video/ecs_ffmpeg_wrapper.sh"],
            "linuxParameters": {
                "devices": [
                    {
                        "containerPath": "/dev/dri/renderD128",
                        "hostPath": "/dev/dri/renderD128",
                        "permissions": [
                            "read",
                            "write"
                        ]
                    },
                    {
                        "containerPath": "/dev/dri/renderD129",
                        "hostPath": "/dev/dri/renderD129",
                        "permissions": [
                            "read",
                            "write"
                        ]
                    },
                    {
                        "containerPath": "/dev/dri/renderD130",
                        "hostPath": "/dev/dri/renderD130",
                        "permissions": [
                            "read",
                            "write"
                        ]
                    },
                    {
                        "containerPath": "/dev/dri/renderD131",
                        "hostPath": "/dev/dri/renderD131",
                        "permissions": [
                            "read",
                            "write"
                        ]
```

```
                    },
                    {
                        "containerPath": "/dev/dri/renderD132",
                        "hostPath": "/dev/dri/renderD132",
                        "permissions": [
                            "read",
                            "write"
                        ]
                    },
                    {
                        "containerPath": "/dev/dri/renderD133",
                        "hostPath": "/dev/dri/renderD133",
                        "permissions": [
                            "read",
                            "write"
                        ]
                    },
                    {
                        "containerPath": "/dev/dri/renderD134",
                        "hostPath": "/dev/dri/renderD134",
                        "permissions": [
                            "read",
                            "write"
                        ]
                    },
                    {
                        "containerPath": "/dev/dri/renderD135",
                        "hostPath": "/dev/dri/renderD135",
                        "permissions": [
                            "read",
                            "write"
                        ]
                    },
                    {
                        "containerPath": "/dev/dri/renderD136",
                        "hostPath": "/dev/dri/renderD136",
                        "permissions": [
                            "read",
                            "write"
                        ]
                    },
                    {
                        "containerPath": "/dev/dri/renderD137",
                        "hostPath": "/dev/dri/renderD137",
```

```
                        "permissions": [
                            "read",
                            "write"
                        ]
                    },
                    {
                        "containerPath": "/dev/dri/renderD138",
                        "hostPath": "/dev/dri/renderD138",
                        "permissions": [
                            "read",
                            "write"
                        ]
                    },
                    {
                        "containerPath": "/dev/dri/renderD139",
                        "hostPath": "/dev/dri/renderD139",
                        "permissions": [
                            "read",
                            "write"
                        ]
                    },
                    {
                        "containerPath": "/dev/dri/renderD140",
                        "hostPath": "/dev/dri/renderD140",
                        "permissions": [
                            "read",
                            "write"
                        ]
                    },
                    {
                        "containerPath": "/dev/dri/renderD141",
                        "hostPath": "/dev/dri/renderD141",
                        "permissions": [
                            "read",
                            "write"
                        ]
                    },
                    {
                        "containerPath": "/dev/dri/renderD142",
                        "hostPath": "/dev/dri/renderD142",
                        "permissions": [
                            "read",
                            "write"
                        ]
```

```
                    },
                    {
                        "containerPath": "/dev/dri/renderD143",
                        "hostPath": "/dev/dri/renderD143",
                        "permissions": [
                            "read",
                            "write"
                        ]
                    }
                ]
            },
            "mountPoints": [
                {
                    "containerPath": "/video",
                    "sourceVolume": "video_file"
                }
            ],
            "cpu": 0,
            "memory": 12000,
            "image": "0123456789012.dkr.ecr.us-west-2.amazonaws.com/aws/xilinx-
xffmpeg",
            "essential": true,
            "name": "xilinix-xffmpeg"
        }
    ],
    "volumes": [
        {
            "name": "video_file",
            "host": {"sourcePath": "/home/ec2-user"}
        }
    ]
}
```

2. 註冊任務定義。

```
aws ecs register-task-definition --family vt1-24xlarge-xffmpeg-processor --cli-
input-json file://vt1-24xlarge-xffmpeg-linux.json --region us-east-1
```

適用於 AWS 神經元機器學習工作負載的 Amazon ECS 任務定義

您可將 Amazon EC2 Trn1、Amazon EC2 Inf1 和 Amazon EC2 Inf2 執行個體註冊至叢集，用於機器學習工作負載。

Amazon EC2 Trn1 執行個體採用 AWS Trainium 晶片。這些執行個體為雲端中的機器學習提供高效能和低成本的訓練。您可以在 Trn1 執行個體上使用 AWS Neuron 打造機器學習架構，來訓練機器學習推論模型。然後，您可以在 Inf1 例證或 Inf2 例證上執行模型，以使用推論晶片的加速度。 AWS

Amazon EC2 Inf1 執行個體和 Inf2 執行個體由 AWS Inferentia 晶片提供支援。這些晶片在雲端提供高效能和最低成本的推論。

機器學習模型使用 AWS Neuron 部署至容器，這是一款特殊軟體開發套件 (SDK)。SDK 由編譯器、執行階段和剖析工具組成，可最佳化機器學習晶片的 AWS 機器學習效能。 AWS 神經元支援常用的機器學習架構，例如 TensorFlow PyTorch、和 Apache MXNet。

考量事項

當您開始在 Amazon ECS 上部署 Neuron 之前，請考量下列事項：

- 您的叢集可包含 Trn1、Inf1、Inf2 和其他執行個體的組合。
- 您需要在使用支援 AWS 神經元的機器學習架構的容器中使用 Linux 應用程式。

> **Important**
>
> 使用其他架構的應用程式在 Trn1、Inf1 和 Inf2 執行個體上的效能可能不會有很大提升。

- 每個 AWS Trainium 或 AWS Inferentia 晶片一次僅可執行一個推論或推論訓練任務。對於 Inf1，每個晶片都有 4 NeuronCores 個。對於 Trn1 和 In2，每個芯片都有 2。 NeuronCores每個 Trn1、Inf1 和 Inf2 執行個體上有多少個晶片就可以執行多少個任務。
- 建立服務或執行獨立任務時，您可以在設定任務置放限制條件時使用執行個體類型屬性。這可確保在您指定的容器執行個體上啟動任務。藉此可協助您最佳化整體資源使用率，並確保推論工作負載的任務都位於 Trn1、Inf1 或 Inf2 執行個體上。如需詳細資訊，請參閱 Amazon ECS 如何在容器執行個體上放置任務。

  在以下範例中，在您的 `default` 叢集的 `Inf1.xlarge` 執行個體上執行任務。

```
aws ecs run-task \
     --cluster default \
     --task-definition ecs-inference-task-def \
     --placement-constraints type=memberOf,expression="attribute:ecs.instance-type ==
 Inf1.xlarge"
```

- 無法在任務定義中定義 Neuron 資源需求。而是將容器設定為使用主機容器執行個體上可用的特定 AWS Trainium 或 AWS 推論晶片。您可使用 `linuxParameters` 參數並指定裝置詳細資訊來執行此動作。如需詳細資訊，請參閱 任務定義需求。

使用 Amazon ECS 優化 Amazon Linux 2023（神經元）AMI

Amazon ECS 提供了一個 Amazon ECS 優化 AMI，該 AMI 基於 Amazon Linux 2023，用於工作負載和推論工作負載 AWS 。 AWS 它配備了 AWS 神經元驅動程序和運行時的 Docker。此 AMI 使得在 Amazon ECS 上執行機器學習推論工作負載更輕鬆。

我們建議您在啟動 Amazon EC2 Trn1、InF1 和 In2 執行個體時使用 Amazon ECS 優化的亞馬遜 Linux 2023 (神經元) AMI。

您可以使用以下命令檢索當前 Amazon ECS 優化的 Amazon Linux 2023（神經元）AMI。 AWS CLI

```
aws ssm get-parameters --names /aws/service/ecs/optimized-ami/amazon-linux-2023/neuron/
recommended
```

以下區域支援 Amazon ECS 最佳化 Amazon Linux 2023 (神經元) AMI：

- 美國東部 (維吉尼亞北部)
- 美國東部 (俄亥俄)
- 美國西部 (加利佛尼亞北部)
- 美國西部 (奧勒岡)
- 亞太區域 (孟買)
- 亞太區域 (大阪)
- 亞太區域 (首爾)
- 亞太區域 (東京)
- 亞太區域 (新加坡)
- 亞太區域 (悉尼)
- 加拿大 (中部)
- 歐洲 (法蘭克福)
- 歐洲 (愛爾蘭)
- 歐洲 (倫敦)
- 歐洲 (巴黎)

- 歐洲 (斯德哥爾摩)
- 南美洲 (聖保羅)

使用 Amazon ECS 優化 Amazon Linux 2（神經元）AMI

Amazon ECS 提供了一個 Amazon ECS 優化 AMI，該 AMI 基於 Amazon Linux 2，用於工作負載和推 AWS 論工作負載。 AWS 它配備了 AWS 神經元驅動程序和運行時的 Docker。此 AMI 使得在 Amazon ECS 上執行機器學習推論工作負載更輕鬆。

建議在啟動 Amazon EC2 Trn1、Inf1 和 Inf2 執行個體時，使用 Amazon ECS 最佳化 Amazon Linux 2 (Neuron) AMI。

您可以使用以下命令檢索當前 Amazon ECS 優化 Amazon Linux 2（神經元）AMI。 AWS CLI

```
aws ssm get-parameters --names /aws/service/ecs/optimized-ami/amazon-linux-2/inf/
recommended
```

Amazon ECS 最佳化 Amazon Linux 2 (神經元) AMI 在以下區域支援:

- 美國東部 (維吉尼亞北部)
- 美國東部 (俄亥俄)
- 美國西部 (加利佛尼亞北部)
- 美國西部 (奧勒岡)
- 亞太區域 (孟買)
- 亞太區域 (大阪)
- 亞太區域 (首爾)
- 亞太區域 (東京)
- 亞太區域 (新加坡)
- 亞太區域 (悉尼)
- 加拿大 (中部)
- 歐洲 (法蘭克福)
- 歐洲 (愛爾蘭)
- 歐洲 (倫敦)
- 歐洲 (巴黎)

- 歐洲 (斯德哥爾摩)
- 南美洲 (聖保羅)

任務定義需求

若要在 Amazon ECS 上部署 Neuron，您的任務定義必須包含提供推論模型之預先建置容器的容器定義。 TensorFlow它由 AWS Deep Learning Containers 提供。此容器包含 AWS 神經元運行時和 TensorFlow 服務應用程序。啟動時，此容器會從 Amazon S3 擷取您的模型、使用儲存的模型啟動神經元 TensorFlow 服務，然後等待預測請求。在下面的例子中，容器映像具有 TensorFlow 1.15 和 Ubuntu 18.04。維護針對神經元優化的預先構建 Deep Learning Containers 的完整列表。 GitHub如需詳細資訊，請參閱使用 AWS 神經元服 TensorFlow 務。

```
763104351884.dkr.ecr.us-east-1.amazonaws.com/tensorflow-inference-neuron:1.15.4-neuron-py37-ubuntu18.04
```

或者，您可以建置自己的 Neuron 附屬容器映像。如需詳細資訊，請參閱AWS Deep Learning AMI 開發人員指南中的教學課程：神經元 TensorFlow 服務。

任務定義必須根據單一執行個體類型專門設定。您必須將容器設定為使用主機容器執行個體上可用的特定 AWS Trainium 或 AWS 推論裝置。您可以使用 `linuxParameters` 參數進行該動作。下表詳細說明特定於每種執行個體類型的晶片。

執行個體類型	vCPU	RAM (GiB)	AWS ML 加速器芯片	裝置路徑
trn1.2xlarge	8	32	1	`/dev/neuron0`
trn1.32xlarge	128	512	16	`/dev/neuron0 ,/dev/neuron1 ,/dev/neuron2 ,/dev/neuron3 ,/dev/neuron4 ,/dev/neuron5 ,/dev/neur`

執行個體類型	vCPU	RAM (GiB)	AWS ML 加速器芯片	裝置路徑
				on6 ,/dev/neuron7 ,/dev/neuron8 ,/dev/neuron9 ,/dev/neuron10 ,/dev/neuron11 ,/dev/neuron12 ,/dev/neuron13 ,/dev/neuron14 ,/dev/neuron15
inf1.xlarge	4	8	1	/dev/neuron0
inf1.2xlarge	8	16	1	/dev/neuron0
inf1.6xlarge	24	48	4	/dev/neuron0 ,/dev/neuron1 ,/dev/neuron2 ,/dev/neuron3

執行個體類型	vCPU	RAM (GiB)	AWS ML 加速器芯片	裝置路徑
inf1.24xlarge	96	192	16	/dev/neuron0 ,/dev/neuron1 ,/dev/neuron2 ,/dev/neuron3 ,/dev/neuron4 ,/dev/neuron5 ,/dev/neuron6 ,/dev/neuron7 ,/dev/neuron8 ,/dev/neuron9 ,/dev/neuron10 ,/dev/neuron11 ,/dev/neuron12 ,/dev/neuron13 ,/dev/neuron14 ,/dev/neuron15
inf2.xlarge	8	16	1	/dev/neuron0
inf2.8xlarge	32	64	1	/dev/neuron0

執行個體類型	vCPU	RAM (GiB)	AWS ML 加速器芯片	裝置路徑
inf2.24xlarge	96	384	6	/dev/neuron0 ,/dev/neuron1 ,/dev/neuron2 ,/dev/neuron3 ,/dev/neuron4 ,/dev/neuron5 ,
inf2.48xlarge	192	768	12	/dev/neuron0 ,/dev/neuron1 ,/dev/neuron2 ,/dev/neuron3 ,/dev/neuron4 ,/dev/neuron5 ,/dev/neuron6 ,/dev/neuron7 ,/dev/neuron8 ,/dev/neuron9 ,/dev/neuron10 ,/dev/neuron11

在 Amazon ECS 任務定義中指定 AWS 神經元機器學習

以下是 `inf1.xlarge` 的 Linux 任務定義範例，顯示要使用的語法。

```
{
    "family": "ecs-neuron",
    "requiresCompatibilities": ["EC2"],
```

```
    "placementConstraints": [
        {
            "type": "memberOf",
            "expression": "attribute:ecs.os-type == linux"
        },
        {
            "type": "memberOf",
            "expression": "attribute:ecs.instance-type == inf1.xlarge"
        }
    ],
    "executionRoleArn": "${YOUR_EXECUTION_ROLE}",
    "containerDefinitions": [
        {
            "entryPoint": [
                "/usr/local/bin/entrypoint.sh",
                "--port=8500",
                "--rest_api_port=9000",
                "--model_name=resnet50_neuron",
                "--model_base_path=s3://your-bucket-of-models/resnet50_neuron/"
            ],
            "portMappings": [
                {
                    "hostPort": 8500,
                    "protocol": "tcp",
                    "containerPort": 8500
                },
                {
                    "hostPort": 8501,
                    "protocol": "tcp",
                    "containerPort": 8501
                },
                {
                    "hostPort": 0,
                    "protocol": "tcp",
                    "containerPort": 80
                }
            ],
            "linuxParameters": {
                "devices": [
                    {
                        "containerPath": "/dev/neuron0",
                        "hostPath": "/dev/neuron0",
                        "permissions": [
                            "read",
```

```
                            "write"
                        ]
                    }
                ],
                "capabilities": {
                    "add": [
                        "IPC_LOCK"
                    ]
                }
            },
            "cpu": 0,
            "memoryReservation": 1000,
            "image": "763104351884.dkr.ecr.us-east-1.amazonaws.com/tensorflow-
inference-neuron:1.15.4-neuron-py37-ubuntu18.04",
            "essential": true,
            "name": "resnet50"
        }
    ]
}
```

適用於深度學習執行個體的 Amazon ECS 任務定義

若要在 Amazon ECS 上使用深度學習工作負載，請將 Amazon EC2 DL1 執行個體註冊到您的叢集。Amazon EC2 DL1 執行個體由 Habana 實驗室 (Intel 公司) 的 Gaudi 加速器提供。使用 Habana SynapseAI SDK 連線到 Habana Gaudi 加速器。SDK 支援常用的機器學習架構，以 TensorFlow 及 PyTorch.

考量事項

當您開始在 Amazon ECS 上部署 DL1 之前，請考量下列事項：

- 您的叢集可包含 DL1 和非 DL1 執行個體組合。
- 建立服務或執行獨立任務時，您可以在設定任務置放限制條件時使用執行個體類型屬性，以確定任務於指定的容器執行個體啟動。藉此可確保您的資源得到有效利用，並確保深度學習工作負載的任務位於 DL1 執行個體上。如需詳細資訊，請參閱 Amazon ECS 如何在容器執行個體上放置任務。

  以下範例在 `default` 叢集的 `dl1.24xlarge` 執行個體上執行任務。

```
aws ecs run-task \
     --cluster default \
     --task-definition ecs-dl1-task-def \
```

```
    --placement-constraints type=memberOf,expression="attribute:ecs.instance-type ==
dl1.24xlarge"
```

使用 DL1 AMI

對於在 Amazon EC2 DL1 執行個體上執行 Amazon ECS 的 AMI，您有三個選項：

- AWS Marketplace 由哈瓦那在這裡提供的 AMI。
- 由 Amazon Web Services 提供的 Habana 深度學習 AMI。由於其不包含在內，您需要單獨安裝 Amazon ECS 容器代理程式。
- 使用封裝程式建置GitHub軟體庫所提供的自訂 AMI。如需詳細資訊，請參閱 Packer documentation (《Packer 文件》)。

在 Amazon ECS 任務定義中指定深度學習

若要在 Amazon ECS 上執行 Habana Gaudi 加速 Deep Learning Containers，您的任務定義必須包含預先建置容器的容器定義，該容器為深度學習模型提供 TensorFlow 或 PyTorch 使用深度學習容器提供的 Habana SynapseAI 提供的深度學習模型。 AWS

下面的容器映像有 TensorFlow 2.7.0 和 Ubuntu 20.04。針對 Habana Gaudi 加速器進行最佳化的預先建置 Deep Learning Containers 的完整清單保留在上。 GitHub如需詳細資訊，請參閱 Habana Training Containers (Habana 訓練容器)。

```
763104351884.dkr.ecr.us-east-1.amazonaws.com/tensorflow-training-habana:2.7.0-hpu-py38-
synapseai1.2.0-ubuntu20.04
```

以下是 Amazon EC2 上的 Linux 容器任務定義範例，顯示了要使用的語法。此範例使用包含 Habana 實驗室系統管理介面工具 (HL-SMI) 的映像，請參閱：`vault.habana.ai/gaudi-docker/1.1.0/ubuntu20.04/habanalabs/tensorflow-installer-tf-cpu-2.6.0:1.1.0-614`

```
{
    "family": "dl-test",
    "requiresCompatibilities": ["EC2"],
    "placementConstraints": [
        {
            "type": "memberOf",
            "expression": "attribute:ecs.os-type == linux"
        },
        {
```

```
            "type": "memberOf",
            "expression": "attribute:ecs.instance-type == dl1.24xlarge"
        }
    ],
    "networkMode": "host",
    "cpu": "10240",
    "memory": "1024",
    "containerDefinitions": [
        {
            "entryPoint": [
                "sh",
                "-c"
            ],
            "command": ["hl-smi"],
            "cpu": 8192,
            "environment": [
                {
                    "name": "HABANA_VISIBLE_DEVICES",
                    "value": "all"
                }
            ],
            "image": "vault.habana.ai/gaudi-docker/1.1.0/ubuntu20.04/habanalabs/
tensorflow-installer-tf-cpu-2.6.0:1.1.0-614",
            "essential": true,
            "name": "tensorflow-installer-tf-hpu"
        }
    ]
}
```

適用於 64 位元 ARM 工作負載的 Amazon ECS 任務定義

Amazon ECS 支援使用 64 位元 ARM 應用程式。您可以在採用 AWS Graviton 2 處理器的平台上執行您的應用程式。該平台適用多種工作負載。其包括各種工作負載，例如應用程式伺服器、微型服務、高效能運算、CPU 型機器學習推論、影片編碼、電子設計自動化、遊戲、開放原始碼資料庫和記憶體內快取。

考量事項

在您開始部署使用 64 位元 ARM 架構的任務定義之前，請考量下列事項：

- 應用程式可使用 Fargate 或 EC2 啟動類型。
- 採用 ARM64 架構的 Linux 任務不支援 Fargate Spot 容量提供者。

- 應用程式僅能使用 Linux 作業系統。
- 對於 Fargate 類型，應用程式必須使用 Fargate 平台版本 `1.4.0` 或更新版本。
- 這些應用程序可以使用Fluent Bit或用 CloudWatch 於監視。
- 對於 Fargate 啟動類型，下列項目 AWS 區域 不支援 64 位元 ARM 工作負載：
  - 美國東部 (維吉尼亞北部)、`use1-az3` 可用區域
- 對於 Amazon EC2 啟動類型，請參閱以下內容，以驗證您所在區域是否支援您要使用的執行個體類型：
  - Amazon EC2 M6g 執行個體
  - Amazon EC2 T4g 執行個體
  - Amazon EC2 C6g 執行個體
  - Amazon EC2 R6gd 執行個體
  - Amazon EC2 X2gd 執行個體

  您也可以使用 Amazon EC2 `describe-instance-type-offerings` 命令搭配篩選條件，查看您所在區域的執行個體優惠。

  ```
  aws ec2 describe-instance-type-offerings --filters Name=instance-
  type,Values=instance-type --region region
  ```

  以下範例檢查美國東部 (維吉尼亞北部) (us-east-1) 區域中的 M6 執行個體類型可用性。

  ```
  aws ec2 describe-instance-type-offerings --filters "Name=instance-type,Values=m6*" --
  region us-east-1
  ```

  如需詳細資訊，請describe-instance-type-offerings 參閱 Amazon EC2 命令列參考中的。

在 Amazon ECS 任務定義中指定 ARM 架構

若要使用 ARM 架構，請為 `cpuArchitecture` 任務定義參數指定 `ARM64`。

在以下範例中，ARM 架構是在任務定義中指定的。其為 JSON 格式。

```
{
    "runtimePlatform": {
        "operatingSystemFamily": "LINUX",
        "cpuArchitecture": "ARM64"
```

```
    },
...
}
```

以下為顯示"hello world" 的 ARM 架構的任務定義範例。

```
{
 "family": "arm64-testapp",
 "networkMode": "awsvpc",
 "containerDefinitions": [
    {
        "name": "arm-container",
        "image": "arm64v8/busybox",
        "cpu": 100,
        "memory": 100,
        "essential": true,
        "command": [ "echo hello world" ],
        "entryPoint": [ "sh", "-c" ]
    }
 ],
 "requiresCompatibilities": [ "FARGATE" ],
 "cpu": "256",
 "memory": "512",
 "runtimePlatform": {
        "operatingSystemFamily": "LINUX",
        "cpuArchitecture": "ARM64"
  },
 "executionRoleArn": "arn:aws:iam::123456789012:role/ecsTaskExecutionRole"
}
```

將 Amazon ECS 日誌傳送到 CloudWatch

您可以在工作中設定容器，將記錄資訊傳送至 CloudWatch 記錄檔。如果您為任務使用 Fargate 啟動類型，您可檢視容器的日誌。如果您使用 EC2 啟動類型，您可在單一便利位置檢視容器的不同日誌，並防止容器日誌佔用容器執行個體的磁碟空間。

> **Note**
>
> 任務中容器所記錄的資訊類型，絕大部分取決於其 ENTRYPOINT 命令。在預設情況下，擷取的日誌會顯示您在本機執行容器時，通常會在互動式終端機中看見的命令輸出，其為 STDOUT 和 STDERR I/O 串流。日awslogs誌驅動程序只是將這些日誌從 Docker 傳遞到CloudWatch 日

誌。如需 Docker 日誌處理方式 (包括擷取不同檔案資料或串流的替代方法) 的詳細資訊，請參閱 Docker 文件中的檢視容器或服務的日誌。

若要將系統日誌從 Amazon ECS 容器執行個體傳送到 CloudWatch 日誌，請參閱 Amazon 日 CloudWatch 誌使用者指南中的監控日誌檔和 CloudWatch 日誌配額。

Fargate 啟動類型

如果您對任務使用 Fargate 啟動類型，您需將必要的 `logConfiguration` 參數新增至任務定義，以開啟 `awslogs` 日誌驅動程式。如需詳細資訊，請參閱 Amazon ECS 任務定義範例：將日誌路由到 CloudWatch。

針對 Fargate 上的 Windows 容器，當任何作業定義參數具有特殊字元時，請執行下列其中一個選項，`& \ < > ^ |` 例如：

- 在整個參數字符串周圍添加帶有雙引號的 escape (\)

  範例

  ```
  "awslogs-multiline-pattern": "\"^[|DEBUG|INFO|WARNING|ERROR\"",
  ```

- 在每個特殊字符周圍添加一個轉義（^）字符

  範例

  ```
  "awslogs-multiline-pattern": "^^[^|DEBUG^|INFO^|WARNING^|ERROR",
  ```

EC2 啟動類型

如果您對任務使用 EC2 啟動類型，並且想要開啟 `awslogs` 日誌驅動程式，則 Amazon ECS 容器執行個體需要至少 1.9.0 版的容器代理程式。如需如何檢查代理程式版本及更新至最新版本的資訊，請參閱「更新 Amazon ECS 容器代理程式」。

> **Note**
>
> 您必須使用 Amazon ECS 最佳化 AMI 或具有至少套件版本`1.9.0-1`的自訂 AMI。`ecs-init`使用自訂 AMI 時，您必須在`docker run`陳述式或環境變數檔案中使用下列環境變數來指定啟動代理程式時，Amazon EC2 執行個體上可使用`awslogs`記錄驅動程式。

```
ECS_AVAILABLE_LOGGING_DRIVERS=["json-file","awslogs"]
```

您的 Amazon ECS 容器執行個體也需要可用來啟動容器執行個體之 IAM 角色的 `logs:CreateLogStream` 和 `logs:PutLogEvents` 許可。在 Amazon ECS 中啟用 `awslogs` 日誌驅動程式支援前，如果您已建立 Amazon ECS 容器執行個體角色，您可能需要新增此許可。`ecsTaskExecutionRole` 在其被指派給任務時使用，且應該包含正確的許可。如需有關工作執行角色的資訊，請參閱Amazon ECS 任務執行 IAM 角色。如果您的容器執行個體使用容器執行個體的受管 IAM 政策，則您的容器執行個體應該具有正確的許可。如需容器執行個體的受管 IAM 政策的相關資訊，請參閱Amazon ECS 容器執行個體 IAM 角色。

Amazon ECS 任務定義範例：將日誌路由到 CloudWatch

您必須在工作定義中指定容器的記`awslogs`錄驅動程式 CloudWatch，才能將記錄檔傳送至容器。如需記錄參數的詳細資訊，請參閱 儲存與記錄

隨後的任務定義 JSON 具有為每個容器指定的 `logConfiguration` 物件。一個用於將日誌發送到名為的日誌組的 WordPress 容器`awslogs-wordpress`。另一個用於 MySQL 容器，該容器將日誌發送到名為 `awslogs-mysql` 的日誌群組。兩個容器使用的日誌串流前綴皆為 `awslogs-example`。

```
{
    "containerDefinitions": [
        {
            "name": "wordpress",
            "links": [
                "mysql"
            ],
            "image": "wordpress",
            "essential": true,
            "portMappings": [
                {
                    "containerPort": 80,
                    "hostPort": 80
                }
            ],
            "logConfiguration": {
                "logDriver": "awslogs",
                "options": {
                    "awslogs-create-group": "true",
                    "awslogs-group": "awslogs-wordpress",
```

```
                    "awslogs-region": "us-west-2",
                    "awslogs-stream-prefix": "awslogs-example"
                }
            },
            "memory": 500,
            "cpu": 10
        },
        {
            "environment": [
                {
                    "name": "MYSQL_ROOT_PASSWORD",
                    "value": "password"
                }
            ],
            "name": "mysql",
            "image": "mysql",
            "cpu": 10,
            "memory": 500,
            "essential": true,
            "logConfiguration": {
                "logDriver": "awslogs",
                "options": {
                    "awslogs-create-group": "true",
                    "awslogs-group": "awslogs-mysql",
                    "awslogs-region": "us-west-2",
                    "awslogs-stream-prefix": "awslogs-example",
                    "mode": "non-blocking",
                    "max-buffer-size": "25m"
                }
            }
        }
    ],
    "family": "awslogs-example"
}
```

在容器定義記錄檔組態中使用記awslogs錄驅動程式註冊工作定義之後，您可以執行工作或使用該工作定義建立服務，以開始將記錄檔傳送至 CloudWatch 記錄檔。如需詳細資訊，請參閱 以 Amazon ECS 任務的形式執行應用程式 及 使用主控台建立 Amazon ECS 服務。

將 Amazon ECS 日誌傳送到 AWS 服務或 AWS Partner

您可以使 FireLens 用 Amazon ECS 使用任務定義參數將日誌路由到 AWS 服務或 AWS Partner Network (APN) 目的地以進行日誌儲存和分析。這 AWS Partner Network 是一個由合作夥伴組成的

全球社群，利用計畫、專業知識和資源來建置、行銷和銷售客戶產品。如需詳細資訊，請參閱 AWS Partner。FireLens 使用 Fluentd 和 Fluent Bit。我們提供 AWS for Fluent Bit 映像，或者您也可以使用自己的 Fluentd 或 Fluent Bit 映像。

使 FireLens 用 Amazon ECS 時，請考慮下列事項：

- 建議您將記錄容器名稱新增`my_service_`至記錄檔容器名稱，以便在主控台中輕鬆區分容器名稱。
- Amazon ECS 預設會在應用程式容器和容器之間新增起始容器順序相依性。FireLens當您在應用程式容器和容器之間指定容器順序時，會覆寫預設的起始容器順序。FireLens
- FireLensAmazon ECS 適用於在 Linux 上託管的任務和 Linux 上的 Amazon EC2 AWS Fargate 上託管的任務都受到支持。Windows 容器不支援 FireLens。

  如需如何為 Windows 容器設定集中式記錄的相關資訊，請參閱 Centralized logging for Windows containers on Amazon ECS using Fluent Bit (《使用 Fluent Bit 為 Amazon ECS 上的 Windows 容器設定集中式記錄》)。
- 您可以使用 AWS CloudFormation 模板。為 Amazon ECS FireLens 進行配置。若要取得更多資訊，請參閱AWS CloudFormation 使用指南AWS::ECS::TaskDefinition FirelensConfiguration中的〈
- FireLens偵聽連接埠24224，因此為了確保在工作之外無法連線到 FireLens 記錄路由器，您不得允許任務使用的安全性群組24224中的連接埠上的輸入流量。對於使用 `awsvpc` 網路模式的任務，這是與任務相關聯的安全群組。對於使用 `host` 網路模式的任務，這是與託管任務的 Amazon EC2 執行個體相關聯的安全群組。對於使用 `bridge` 網路模式的任務，請不要建立使用連接埠 24224 的任何連接埠映射。
- 對於使用`bridge`網路模式的工作，具有 FireLens 組態的容器必須在任何依賴該容器的應用程式容器啟動之前啟動。若要控制容器的起始順序，請在工作定義中使用相依性條件。如需詳細資訊，請參閱 容器相依性。

  > **Note**
  >
  > 如果您在容器定義中搭 FireLens 配組態使用相依性條件參數，請確定每個容器都有`START`或`HEALTHY`條件需求。

- 根據預設，FireLens 將叢集和任務定義名稱以及叢集的 Amazon Resource Name (ARN) 作為中繼資料索引鍵新增到 stout/stderr 容器紀錄。以下是中繼資料格式的範例。

```
"ecs_cluster": "cluster-name",
"ecs_task_arn": "arn:aws:ecs:region:111122223333:task/cluster-
name/f2ad7dba413f45ddb4EXAMPLE",
```

```
"ecs_task_definition": "task-def-name:revision",
```

如果您不想在紀錄中使用中繼資料，請在任務定義 `firelensConfiguration` 部分中將 `false` 設定為 `enable-ecs-log-metadata`。

```
"firelensConfiguration":{
   "type":"fluentbit",
   "options":{
      "enable-ecs-log-metadata":"false",
      "config-file-type":"file",
      "config-file-value":"/extra.conf"
}
```

若要使用此功能，您必須為任務建立 IAM 角色，以提供使用任 AWS 務所需的任何服務所需的許可。例如，如果容器將日誌路由到 Firehose，則該任務需要調用 `firehose:PutRecordBatch` API 的權限。如需詳細資訊，請參閱 《IAM 使用者指南》中的新增和移除 IAM 身分許可。

在下列情況下，您的任務可能還需要 Amazon ECS 任務執行角色。如需詳細資訊，請參閱 Amazon ECS 任務執行 IAM 角色。

- 如果您的任務託管在 Fargate 上，而且您要從 Amazon ECR 提取容器映像，或從日誌組態 AWS Secrets Manager 中參考敏感資料，則必須包含任務執行 IAM 角色。
- 當您使用 Amazon S3 中託管的自訂組態檔案時，您的任務執行 IAM 角色必須包含`s3:GetObject`許可。

如需如何搭配 Amazon ECS 使用多個組態檔案 (包括您託管的檔案或 Amazon S3 中的檔案) 的詳細資訊，請參閱 ECS 上流利位元的初始化程序、多組態支援。

設定 Amazon ECS 日誌以獲得高輸送量

建立作業定義時，可以指定記憶體中緩衝的記錄行數，方法是指定中的`log-driver-buffer-limit`值。如需詳細資訊，請參閱 Docker 文件中的 Fluentd 登入驅動程式。

當輸送量很高時，請使用此選項，因為 Docker 可能會耗盡緩衝區記憶體並捨棄緩衝區訊息，因此可以新增訊息。

使用具有緩衝區限制選項 FireLens 的 Amazon ECS 時，請考慮下列事項：

- Amazon EC2 啟動類型和具有平台版本的 `1.4.0` 或更新版本的 Fargate 啟動類型支援此選項。

- 只有在 logDriver 設定為 awsfirelens 時，此選項才有效。
- 預設緩衝區限制為1048576記錄行。
- 有效值0和536870912記錄行。
- 此緩衝區使用的最大記憶體容量是每個記錄行的大小和緩衝區大小的乘積。例如，如果應用程式的記錄行平均為 2 KiB，則最多8會使用 4096 的緩衝區限制。在工作層級配置的記憶體總量應該大於除了記錄驅動程式記憶體緩衝區之外，配置給所有容器的記憶體數量。

在任務定義中指定 awsfirelens 日誌驅動程式時，Amazon ECS 代理程式會將下列環境變數插入容器中：

FLUENT_HOST

指派給 FireLens 容器的 IP 位址。

FLUENT_PORT

流利轉送通訊協定正在接聽的連接埠。

您可以使用 FLUENT_HOST 和 FLUENT_PORT 環境變數從程式碼直接登入日誌路由器，而不是透過 stdout。有關更多信息，請參閱流體日誌記錄器。 GitHub

以下顯示指定的語法log-driver-buffer-limit。替換my_service_為您的服務名稱：

```
{
    "containerDefinitions": [
        {
            "essential": true,
            "image": "906394416424.dkr.ecr.us-west-2.amazonaws.com/aws-for-fluent-
bit:stable",
            "name": "my_service_log_router",
            "firelensConfiguration": {
                "type": "fluentbit"
            },
            "logConfiguration": {
                "logDriver": "awslogs",
                "options": {
                    "awslogs-group": "firelens-container",
                    "awslogs-region": "us-west-2",
                    "awslogs-create-group": "true",
                    "awslogs-stream-prefix": "firelens"
```

```
                }
            },
            "memoryReservation": 50
        },
        {
            "essential": true,
            "image": "httpd",
            "name": "app",
            "logConfiguration": {
                "logDriver": "awsfirelens",
                "options": {
                    "Name": "firehose",
                    "region": "us-west-2",
                    "delivery_stream": "my-stream",
                    "log-driver-buffer-limit": "51200"
                }
            },
            "dependsOn": [
                {
                    "containerName": "log_router",
                    "condition": "START"
                }
            ],
            "memoryReservation": 100
        }
    ]
}
```

AWS 適用於 Amazon ECS 的Fluent Bit映像儲存庫

AWS 為 CloudWatch 日誌和 Firehose 提供帶有插件的Fluent Bit圖像。我們建議您將 Fluent Bit 用作日誌路由器，因為它的資源使用率低於 Fluentd。如需詳細資訊，請參閱流利位元的CloudWatch 日誌和Amazon Kinesis Firehose 的流利位元。

流利位元影像可在 Amazon ECR 公共圖庫和 Amazon ECR 儲存庫中的 Amazon ECR 上 AWS 區域 取得，以實現高可用性。AWS

Amazon ECR Public Gallery

Fluent Bit圖像可在 Amazon ECR 公共畫廊上 AWS 找到。這是下載 for Fluent Bit image 的建議位置，因為它是一個公共存儲庫，並且可以從所有人使用 AWS 區域。 AWS 如需詳細資訊，請參閱 Amazon ECR Public Gallery 上的 aws-for-fluent-bit。

Linux

Amazon ECR 公共Fluent Bit圖庫中的圖像支持具有或x86-64架構的 Amazon Linux 操作系統。 AWS ARM 64

您可以透過指定具有所需Fluent Bit影像標籤的存放庫 URL，從 Amazon ECR 公用圖庫提取圖像。 AWS 在 Amazon ECR Public Gallery 的映像標籤索引標籤中可找到可用的映像標籤。

以下顯示 Docker CLI 要使用的語法。

```
docker pull public.ecr.aws/aws-observability/aws-for-fluent-bit:tag
```

例如，您可以使用此 Docker CLI 命令 AWS 為Fluent Bit映像提取最新的穩定檔。

```
docker pull public.ecr.aws/aws-observability/aws-for-fluent-bit:stable
```

> **Note**
>
> 允許未經驗證的提取，但速率限制低於已驗證的提取。要在提取之前使用您的 AWS 帳戶進行身份驗證，請使用以下命令。
>
> ```
> aws ecr-public get-login-password --region us-east-1 | docker login --username
>  AWS --password-stdin public.ecr.aws
> ```

Windows

Amazon ECR 公共Fluent Bit圖庫中的 For 映像支援具有下列作業系統的AMD64架構： AWS

- Windows Server 2022 Full
- Windows Server 2022 Core
- Windows Server 2019 Full
- Windows Server 2019 Core

在 AWS Fargate 上的 Windows 容器不支持 FireLens。

您可以透過指定具有所需Fluent Bit影像標籤的存放庫 URL，從 Amazon ECR 公用圖庫提取圖像。 AWS 在 Amazon ECR Public Gallery 的映像標籤索引標籤中可找到可用的映像標籤。

以下顯示 Docker CLI 要使用的語法。

```
docker pull public.ecr.aws/aws-observability/aws-for-fluent-bit:tag
```

例如，您可以使用此 Docker CLI 命令 AWS 為Fluent Bit映像提取最新的穩定檔。

```
docker pull public.ecr.aws/aws-observability/aws-for-fluent-bit:windowsservercore-
stable
```

> **Note**
>
> 允許未經驗證的提取，但速率限制低於已驗證的提取。要在提取之前使用您的 AWS 帳戶進行身份驗證，請使用以下命令。
>
> ```
> aws ecr-public get-login-password --region us-east-1 | docker login --username
>  AWS --password-stdin public.ecr.aws
> ```

Amazon ECR

流利位元影像可在 Amazon ECR 上取得，以提供高可用性。 AWS 這些圖像在大多數可用 AWS 區域，包括 AWS GovCloud (US).

Linux

可以使 AWS 用以下命令檢索 Fluent 位圖像 URI 的最新穩定版。

```
aws ssm get-parameters \
      --names /aws/service/aws-for-fluent-bit/stable \
      --region us-east-1
```

您可以使用下列指令來列 AWS 出 Fluent 位元影像的所有版本，以查詢「Systems Manager 參數存放區」參數。

```
aws ssm get-parameters-by-path \
      --path /aws/service/aws-for-fluent-bit \
      --region us-east-1
```

您可以參考 Systems Manager 參數存放區名稱，在 AWS CloudFormation 範本中參考 Fluent 位元影像的最新穩定 AWS 檔。以下是範例：

```
Parameters:
  FireLensImage:
    Description: Fluent Bit image for the FireLens Container
    Type: AWS::SSM::Parameter::Value<String>
    Default: /aws/service/aws-for-fluent-bit/stable
```

Windows

可以使 AWS 用以下命令檢索 Fluent 位圖像 URI 的最新穩定版。

```
aws ssm get-parameters \
      --names /aws/service/aws-for-fluent-bit/windowsservercore-stable \
      --region us-east-1
```

您可以使用下列指令來列 AWS 出 Fluent 位元影像的所有版本，以查詢「Systems Manager 參數存放區」參數。

```
aws ssm get-parameters-by-path \
      --path /aws/service/aws-for-fluent-bit/windowsservercore \
      --region us-east-1
```

您可以參考「Systems Manager」參數存放區名稱，在 AWS CloudFormation 範本中參考 Fluent 位元影像的最新穩定 AWS 檔。以下是範例：

```
Parameters:
  FireLensImage:
    Description: Fluent Bit image for the FireLens Container
    Type: AWS::SSM::Parameter::Value<String>
    Default: /aws/service/aws-for-fluent-bit/windowsservercore-stable
```

Amazon ECS 任務定義範例：將日誌路由到 FireLens

若要搭配 FireLens 使用自訂日誌路由，您必須在工作定義中指定下列項目：

- 包含 FireLens 組態的日誌路由器容器。我們建議將容器標示為 `essential`。
- 包含指定日誌驅動程式的日誌配置的一或多個應用 `awsfirelens` 程式容器。
- 任務IAM 角色 Amazon Resource Name (ARN)，其中包含路由日誌所需的任務許可。

使用建立新工作定義時 AWS Management Console，有一個 FireLens整合區段可讓您輕鬆新增記錄路由器容器。如需詳細資訊，請參閱 使用主控台建立 Amazon ECS 任務定義。

Amazon ECS 轉換日誌組態並產生 Fluentd 或 Fluent Bit 輸出組態。輸出配置掛載在流利位和 `/fluentd/etc/fluent.conf` 流利位 `/fluent-bit/etc/fluent-bit.conf` 的日誌路由容器中。

> **Important**
>
> FireLens 會監聽連接埠 24224。因此，若要確保無法在工作之外連線到 FireLens 記錄路由器，您不得允許任務使用的安全性群組24224中連接埠上的輸入流量。對於使用 `awsvpc` 網路模式的任務，這是與任務相關聯的安全群組。對於使用 `host` 網路模式的任務，這是與託管任務的 Amazon EC2 執行個體相關聯的安全群組。對於使用 `bridge` 網路模式的任務，請不要建立使用連接埠 24224 的任何連接埠映射。

根據預設，Amazon ECS 會在日誌項目中新增其他欄位，以協助識別日誌的來源。

- `ecs_cluster` - 任務所屬叢集的名稱。
- `ecs_task_arn` – 容器所屬任務的完整 Amazon Resource Name (ARN)。
- `ecs_task_definition` - 任務正在使用的任務定義名稱和修訂版本。
- `ec2_instance_id`：託管容器的 Amazon EC2 執行個體 ID。此欄位僅用於使用 EC2 啟動類型的任務。

`false`如果您不想要中繼資料，可以`enable-ecs-log-metadata`將設定為。

下列工作定義範例會定義使用 Fluent Bit 將其記錄路由傳送至記錄的記錄路由器容器。 CloudWatch 它還定義了一個應用程式容器，該容器使用日誌組態將日誌路由到 Amazon Data Firehose，並將用於緩衝事件的記憶體設定為 2 MiB。

> **Note**
>
> 如需更多範例任務定義，請參閱上 GitHub的 Amazon ECS FireLens 範例。

```
{
  "family": "firelens-example-firehose",
  "taskRoleArn": "arn:aws:iam::123456789012:role/ecs_task_iam_role",
```

```
  "containerDefinitions": [
    {
      "essential": true,
      "image": "906394416424.dkr.ecr.us-west-2.amazonaws.com/aws-for-fluent-
bit:stable",
      "name": "log_router",
      "firelensConfiguration": {
        "type": "fluentbit",
        "options": {
          "enable-ecs-log-metadata": "true"
        }
      },
      "logConfiguration": {
        "logDriver": "awslogs",
        "options": {
          "awslogs-group": "firelens-container",
          "awslogs-region": "us-west-2",
          "awslogs-create-group": "true",
          "awslogs-stream-prefix": "firelens"
        }
      },
      "memoryReservation": 50
    },
    {
      "essential": true,
      "image": "httpd",
      "name": "app",
      "logConfiguration": {
        "logDriver": "awsfirelens",
        "options": {
          "Name": "firehose",
          "region": "us-west-2",
          "delivery_stream": "my-stream",
          "log-driver-buffer-limit": "2097152"
        }
      },
      "memoryReservation": 100
    }
  ]
}
```

在 `logConfiguration` 物件中指定為選項的索引鍵值組，用來產生 Fluentd 或 Fluent Bit 輸出組態。以下是來自 Fluent Bit 輸出定義的程式碼範例。

```
[OUTPUT]
    Name   firehose
    Match  app-firelens*
    region us-west-2
    delivery_stream my-stream
```

> **Note**
>
> FireLens 管理match配置。您不會在任務match定義中指定組態。

使用自訂規劃檔

您可以指定自訂規劃檔。組態檔格式是您所使用日誌路由器的原生格式。如需詳細資訊，請參閱 流利配置檔案語法 和 流利位元組態檔案。

在您的自訂組態檔案中，對於使用 bridge 或 awsvpc 網路模式的任務，不要透過 TCP 設定 Fluentd 或 Fluent Bit 向前輸入，因為 FireLens 將其新增到輸入組態中。

您的 FireLens 組態必須包含下列選項，才能指定自訂組態檔案：

`config-file-type`

自訂組態檔案的來源位置。可用選項為 s3 或 file。

> **Note**
>
> 託管於的工作 AWS Fargate 僅支援file組態檔案類型。

`config-file-value`

自訂組態檔案的來源。如果使用 s3 組態檔案類型，則組態檔案值是 Amazon S3 儲存貯體和檔案的完整 ARN。如果使用 file 組態檔案類型，組態檔案值就是容器映像中或掛載在容器中的磁碟區上的組態檔案的完整路徑。

> **Important**
>
> 使用自訂組態檔案時，您必須指定與 FireLens 使用的路徑不同的路徑。Amazon ECS 會為 Fluent Bit 保留 `/fluent-bit/etc/fluent-bit.conf` filepath，並為 Fluentd 保留 `/fluentd/etc/fluent.conf`。

下列範例顯示指定自訂組態時所需的語法。

> **Important**
>
> 若要指定託管於 Amazon S3 的自訂組態檔案，請確定您已建立具有適當許可的任務執行 IAM 角色。

以下顯示指定自訂組態時所需的語法。

```
{
  "containerDefinitions": [
    {
      "essential": true,
      "image": "906394416424.dkr.ecr.us-west-2.amazonaws.com/aws-for-fluent-
bit:stable",
      "name": "log_router",
      "firelensConfiguration": {
        "type": "fluentbit",
        "options": {
          "config-file-type": "s3 | file",
          "config-file-value": "arn:aws:s3:::mybucket/fluent.conf | filepath"
        }
      }
    }
  ]
}
```

> **Note**
>
> 託管於的工作 AWS Fargate 僅支援`file`組態檔案類型。

在 Amazon ECS 中使用非AWS 容器映像

使用私人登錄來儲存您的認證 AWS Secrets Manager，然後在您的工作定義中參考它們。這提供了一種方法來參照存在於私人登錄中的容器映像檔，而這些映像檔之外需 AWS 要在您的工作定義中進行驗證。託管於 Fargate、Amazon EC2 執行個體以及使用 Amazon ECS Anywhere 的外部執行個體上的任務支援此功能。

> **Important**
>
> 如果您的任務定義參考存放在 Amazon ECR 中的映像，則此主題不適用。如需詳細資訊，請參閱 《Amazon Elastic Container Registry 使用者指南》中的搭配使用 Amazon ECR 映像與 Amazon ECS。

對於託管於 Amazon EC2 執行個體上的任務，此功能要求具備版本 `1.19.0` 或更新的容器代理程式。不過，我們建議您使用最新版的容器代理程式。如需如何檢查代理程式版本及更新至最新版本的資訊，請參閱「更新 Amazon ECS 容器代理程式」。

對於託管於 Fargate 上的任務，此功能需要平台版本 `1.2.0` 或更新版本。如需相關資訊，請參閱適用於 Amazon ECS 的 Fargate Linux 平台版本。

在您的容器定義內，使用您所建立的秘密的詳細資訊來指定 `repositoryCredentials` 物件。參照的密碼可以來自與使用密碼的任務不同 AWS 區域 或不同的帳戶。

> **Note**
>
> 使用 Amazon ECS API 或 AWS SDK 時 AWS CLI，如果密碼存在於與您正在啟動的任務 AWS 區域 相同，則您可以使用完整的 ARN 或密碼名稱。如果此秘密已存在於不同帳戶中，則必須指定秘密的完整 ARN。使用時 AWS Management Console，必須始終指定密碼的完整 ARN。

以下是任務定義的程式碼片段，其會顯示所需的參數：

用**##########################購。**

```
"containerDefinitions": [
    {
        "image": "private-repo/private-image",
```

```
        "repositoryCredentials": {
            "credentialsParameter":
 "arn:aws:secretsmanager:region:aws_account_id:secret:secret_name"
        }
    }
]
```

> **Note**
>
> 啟用私有登錄檔身分驗證的另一個方法，是使用 Amazon ECS 容器代理程式環境變數來向私有登錄檔進行身分驗證。只有託管於 Amazon EC2 執行個體上的任務才支援此方法。如需詳細資訊，請參閱 為私有碼頭映像設定 Amazon ECS 容器執行個體 。

若要使用私人登錄

1. 任務定義必須具有任務執行角色。這可讓容器代理程式提取容器映像。如需詳細資訊，請參閱 Amazon ECS 任務執行 IAM 角色。

   若要將存取提供給您建立的秘密，請將以下許可字作為內嵌政策，新增到任務執行角色。如需詳細資訊，請參閱新增和移除 IAM 政策。

   - `secretsmanager:GetSecretValue`
   - `kms:Decrypt` - 只有在您的金鑰使用自訂 KMS 金鑰而非預設金鑰時，才需要此項目。您的自訂金鑰的 Amazon Resource Name (ARN) 必須新增為資源。

   下列為新增許可的內嵌政策範例。

   ```
   {
       "Version": "2012-10-17",
       "Statement": [
           {
               "Effect": "Allow",
               "Action": [
                   "kms:Decrypt",
                   "secretsmanager:GetSecretValue"
               ],
               "Resource": [

    "arn:aws:secretsmanager:<region>:<aws_account_id>:secret:secret_name",
   ```

```
                "arn:aws:kms:<region>:<aws_account_id>:key/key_id"
            ]
        }
    ]
}
```

2. 用 AWS Secrets Manager 於為您的私人登錄認證建立密碼。若要取得有關如何建立密碼的資訊，請參閱《AWS Secrets Manager 使用指南》中的〈建立 AWS Secrets Manager 密碼〉。

   使用下列格式輸入您的私人登錄認證：

```
{
  "username" : "privateRegistryUsername",
  "password" : "privateRegistryPassword"
}
```

3. 註冊任務定義。如需詳細資訊，請參閱 the section called “使用主控台建立任務定義”。

將個別環境變數傳遞至 Amazon ECS 容器

> **Important**
>
> 建議您將機密資料儲存在機 AWS Secrets Manager 密或參 AWS Systems Manager 數存放區參數中。如需詳細資訊，請參閱 將敏感資料傳遞至 Amazon ECS 容器。
> 任務定義中指定的環境變數可供所有使用者和角色讀取，系統允許這些使用者和角色對任務定義執行 `DescribeTaskDefinition` 動作。

您可以將環境變數透過下列方式傳遞至容器：

- 個別方式：使用 `environment` 容器定義參數。這會映射到 docker run 的 `--env` 選項。
- 大批方式：使用 `environmentFiles` 容器定義參數列出內含環境變數的一個或多個檔案。檔案必須託管在 Amazon S3 中。這會映射到 docker run 的 `--env-file` 選項。

以下任務定義片段示範如何指定個別環境變數。

```
{
    "family": "",
    "containerDefinitions": [
```

```
        {
            "name": "",
            "image": "",
            ...
            "environment": [
                {
                    "name": "variable",
                    "value": "value"
                }
            ],
            ...
        }
    ],
    ...
}
```

將環境變量傳遞到 Amazon ECS 容器

> **Important**
>
> 建議您將機密資料儲存在機 AWS Secrets Manager 密或參 AWS Systems Manager 數存放區參數中。如需詳細資訊，請參閱 將敏感資料傳遞至 Amazon ECS 容器。
> 環境變數檔案是 Simple Storage Service (Amazon S3) 中的物件，所有 Amazon S3 安全考量事項均適用。
> 您無法在 Fargate 上的 Windows 容器和 Windows 容器上使用該`environmentFiles`參數。

您可以建立環境變數檔案並將其存放在 Amazon S3 中，以將環境變數傳遞至您的容器。

透過在檔案中指定環境變數，您可以批次導入環境變數。在容器定義中指定 `environmentFiles` 物件，其中具有內含環境變數檔案的 Amazon S3 儲存貯體清單。

Amazon ECS 不會對環境變數強制執行大小限制，但是大型環境變數檔案可能會填滿磁碟空間。使用環境變數檔案的每個任務都會導致檔案複本下載到磁碟。Amazon ECS 會移除檔案做為任務清理的一部分。

如需有關支援之環境變數的資訊，請參閱進階容器定義參數 - 環境。

在容器定義中指定環境變數檔案時，請考量下列事項。

- 對於 Amazon EC2 上的 Amazon ECS 任務，您的容器執行個體需要版本 1.39.0 或更新的容器代理程式，才能使用此功能。如需如何檢查代理程式版本及更新至最新版本的資訊，請參閱「更新 Amazon ECS 容器代理程式」。
- 對於 AWS Fargate 特上的 Amazon ECS 任務，您的任務必須使用平台版本1.4.0或更新版本 (Linux) 才能使用此功能。如需詳細資訊，請參閱 適用於 Amazon ECS 的 Fargate Linux 平台版本。

  驗證作業系統平台是否支援該變數。如需詳細資訊，請參閱the section called “容器定義”及the section called “其他任務定義參數”。
- 檔案必須使用 .env 副檔名和 UTF-8 編碼。
- 每個工作定義最多有 10 個檔案。
- 環境檔案中的每一行都必須包含 VARIABLE=VALUE 格式的環境變數。空格或引號會包含為 Amazon ECS 檔案值的一部分。以 # 開頭的行會被視為註解，而忽略。如需環境變數檔案語法的詳細資訊，請參閱《在檔案中宣告預設環境變數》。

  下列為適當的語法。

  ```
  #This is a comment and will be ignored
  VARIABLE=VALUE
  ENVIRONMENT=PRODUCTION
  ```
- 如果有在容器定義中使用 environment 參數指定的環境變數，它們的優先順序高於環境檔案中包含的變數。
- 如果指定內含相同變數的多個環境檔案，則處理順序為先進入者為優先。這表示會使用變數的第一個值，並忽略重複變數的後續值。建議您使用唯一的變數名稱。
- 如果將環境檔案指定為容器覆寫，則會使用該檔案。此外，在容器定義中指定的任何其他環境檔案都將被忽略。
- 下列規則適用於 Fargate 啟動類型：
  - 檔案的處理方式類似於本機 Docker 環境檔案。
  - 不支援 Shell 逸出處理。
  - 容器進入點會解譯 VARIABLE 值。

所需的 IAM 許可

需要 Amazon ECS 任務執行角色，才能使用此功能。這可讓容器代理程式從 Amazon S3 中提取環境變數檔案。如需詳細資訊，請參閱 Amazon ECS 任務執行 IAM 角色。

若要將存取提供給您建立的 Amazon S3 物件，請將以下許可做為內嵌政策，手動新增到任務執行角色。使用 Resource 參數可將許可範圍限定到包含環境變數檔案的 Amazon S3 儲存貯體。如需詳細資訊，請參閱新增和移除 IAM 政策。

- s3:GetObject
- s3:GetBucketLocation

在下列範例中，許可會新增到內嵌政策。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject"
      ],
      "Resource": [
        "arn:aws:s3:::examplebucket/folder_name/env_file_name"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetBucketLocation"
      ],
      "Resource": [
        "arn:aws:s3:::examplebucket"
      ]
    }
  ]
}
```

範例

以下任務定義片段示範如何指定環境變數檔案。

```
{
    "family": "",
    "containerDefinitions": [
        {
```

```
            "name": "",
            "image": "",
            ...
            "environmentFiles": [
                {
                    "value": "arn:aws:s3:::s3_bucket_name/envfile_object_name.env",
                    "type": "s3"
                }
            ],
            ...
        }
    ],
    ...
}
```

將敏感資料傳遞至 Amazon ECS 容器

您可以安全地將敏感資料 (例如資料庫憑證) 傳遞至容器。

您可以使用 Secrets Manager 或做為 Systems Manager 參數存放區中的參數來儲存密碼。

您可以透過程式設計方式從應用程式或使用環境變數擷取密碼。

若要開始，請先將機密資料儲存為 Secret Manager 中的密碼，或在 Systems Manager 員參數存放區中儲存為參數。然後，使用以下其中一種方式向容器公開秘密。

主題

- Amazon ECS 中密碼管理的最佳實務
- 在 Amazon ECS 中以程式設計方式擷取秘密管理員
- 在 Amazon ECS 中以程式設計方式擷取 Systems Manager 參數存放
- 透過 Amazon ECS 環境變數擷取秘密管理員機密
- 透過 Amazon ECS 環境變數擷取 Systems Manager 參數
- 擷取 Amazon ECS 日誌記錄組態的密碼
- 使用 Amazon ECS 中的 Secrets Manager 密碼指定敏感資料

Amazon ECS 中密碼管理的最佳實務

秘密 (例如 API 金鑰和資料庫憑證) 經常被應用程式用來存取其他系統。其通常由使用者名稱和密碼、憑證或 API 金鑰組成。對這些秘密的存取應限制在使用 IAM 的特定 IAM 主體，並在執行期插入容器。

秘密可以從 AWS Secrets Manager 和 Amazon EC2 Systems Manager 參數存放區無縫插入到容器中。這些秘密可以在您的任務中引用為以下任何內容。

1. 被引用為使用 `secrets` 容器定義參數的環境變數。
2. 如果您的日誌平台需要驗證，則被引用為 `secretOptions`。如需詳細資訊，請參閱紀錄組態選項。
3. 如果從中提取容器的登錄檔需要驗證，則使用 `repositoryCredentials` 容器定義參數的映像會將其參照為提取的秘密。從 Amazon ECR Public Gallery 提取映像時使用此方法。如需詳細資訊，請參閱任務的私有登錄檔身分驗證。

秘密推薦

建議您在設定秘密管理時執行下列動作。

使用 AWS Secrets Manager 或 Amazon EC2 Systems Manager 參數存放區來存放秘密材料

您應該將 API 金鑰、資料庫登入資料和其他秘密材料安全地存放在 Amazon EC2 Systems Manager 參數存放區中， AWS Secrets Manager 或作為加密參數存放區中存放。這些服務之所以類似，因為它們都是用 AWS KMS 來加密敏感資料的受管鍵值存放區。 AWS Secrets Manager不過，也包含自動輪換密碼、產生隨機密碼，以及跨 AWS 帳戶共用密碼的功能。如果您認為這些功能很重要，請使用 AWS Secrets Manager ，否則請加密參數。

> **Note**
>
> 參考來自 AWS Secrets Manager 或 Amazon EC2 Systems Manager 參數存放區的任務需要任務執行角色，其政策可授予 Amazon ECS 存取所需密碼的權限，以及用於加密和解密該機密的 AWS KMS 金鑰 (如果適用)。

> **Important**
>
> 不會自動輪換任務中參考的秘密。如果您的秘密發生變更，您必須強制執行新部署或啟動新任務，才能擷取最新的秘密值。如需詳細資訊，請參閱下列主題：
>
> - AWS Secrets Manager：注入資料作為環境變數
> - Amazon EC2 Systems Manager 參數存放區：注入資料作為環境變數

從加密的 Amazon S3 儲存貯體擷取資料

由於環境變數的值可能會在日誌中意外洩漏並在執行 `docker inspect` 時顯示，因此您應該將秘密儲存在加密的 Amazon S3 儲存貯體中，並使用任務角色限制對這些秘密的存取。執行此動作時，必須寫入應用程式，才能讀取 Amazon S3 儲存貯體中的秘密。如需說明，請參閱對 Amazon S3 儲存貯體設定預設伺服器端加密行為。

使用附屬容器將秘密掛載至磁碟區

因為環境變數會有較高的資料外洩風險，因此您應該執行附屬容器來讀取您的密碼， AWS Secrets Manager 然後將它們寫入共用磁碟區。透過使用 Amazon ECS 容器排序，此容器可以在應用程式容器之前執行和結束。執行此操作時，應用程式容器隨後會掛載寫入秘密的磁碟區。就像 Amazon S3 儲存貯體方法一樣，必須寫入您的應用程式才能從共用磁碟區讀取秘密。由於磁碟區的範圍限定在任務，因此會在任務停止後自動刪除磁碟區。有關附屬容器的範例，請參閱 aws-secret-sidecar-injector 專案。

> **Note**
>
> 在 Amazon EC2 上，寫入秘密的磁碟區可以使用 AWS KMS 客戶受管金鑰加密。開啟時 AWS Fargate，磁碟區儲存會使用服務受管金鑰自動加密。

其他資源

- 將秘密傳遞至 Amazon ECS 任務中的容器
- Chamber (密室) 是用於在 Amazon EC2 Systems Manager 參數存放區儲存秘密的包裝程式

在 Amazon ECS 中以程式設計方式擷取秘密管理員

使用 Secrets Manager 保護敏感資料並在資料庫憑證、API 金鑰和其他秘密的整個生命週期進行輪換、管理和擷取。

您可以使用 Secrets Manager 來儲存敏感資料，而不是在應用程式中以純文字對敏感資訊進行硬式編碼。

我們建議使用這種方法擷取敏感資料，因為使用這種方法後，隨後如果 Secrets Manager 秘密有更新，應用程式會自動擷取最新版本的秘密。

在 Secrets Manager 中建立秘密。建立 Secret Manager 秘密後，更新應用程式程式碼即可擷取秘密。

保護 Secrets Manager 中的敏感資料之前，請先檢閱下列考量事項。

- 僅支援儲存文字資料的密碼，這些密碼是使用 CreateSecretAPI `SecretString` 參數建立的密碼。不支援儲存二進位資料的密碼，這些密碼是使用 CreateSecretAPI `SecretBinary` 參數建立的機密。
- 使用介面 VPC 端點來增強安全控制。您必須建立 Secrets Manager 的介面 VPC 端點。如需有關 VPC 端點的資訊，請參閱《AWS Secrets Manager 使用者指南》中的建立VPC 端點。
- 任務所使用的 VPC 必須使用 DNS 解析。

所需的 IAM 許可

若要使用此功能，您必須擁有 Amazon ECS 任務角色，並在您的任務定義中參考該角色。如需詳細資訊，請參閱 Amazon ECS 任務 IAM 角色。

若要允許存取您建立的 Secrets Manager 秘密，請將以下許可新增到任務執行角色。如需有關如何管理許可的相關資訊，請參閱《IAM 使用者指南》中的新增和移除 IAM 身分許可。

- `secretsmanager:GetSecretValue` – 如果參考 Secrets Manager 秘密，則此項目為必要。新增從 Secrets Manager 擷取密碼的許可。

下列政策範例新增必要許可。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "secretsmanager:GetSecretValue"
      ],
      "Resource": [
        "arn:aws:secretsmanager:region:aws_account_id:secret:secret_name"
      ]
    }
  ]
}
```

建立 Secrets Manager 秘密

您可以使用 Secrets Manager 主控台為您的敏感資料建立秘密。如需有關如何建立秘密的詳細資訊，請參閱《AWS Secrets Manager 使用者指南》中的建立 AWS Secrets Manager 秘密。

更新應用程式以程式設計方式擷取 Secrets Manager 秘密

您可以直接從應用程式呼叫 Secrets Manager API 來擷取秘密。如需相關資訊，請參閱AWS Secrets Manager 使用指南 AWS Secrets Manager中的擷取密碼。

若要擷取儲存在中的敏感資料 AWS Secrets Manager，請參閱 SDK 程式碼範例程式庫中 AWS Secrets Manager 使用 AWSAWS SDK 的程式碼範例。

在 Amazon ECS 中以程式設計方式擷取 Systems Manager 參數存放

Systems Manager Parameter Store 提供安全的秘密儲存和管理。您可以將密碼、資料庫字串、EC2 執行個體 ID 和 AMI ID 等資料以及授權代碼儲存為參數值。您存放的值可以是純文字或加密資料。

您可以使用 Secrets Manager 來儲存敏感資料，而不是在應用程式中以純文字對敏感資訊進行硬式編碼。

我們建議使用這種擷取機密資料的方法，因為如果 Systems Manager 參數存放區參數隨後更新，應用程式會自動擷取最新版本。

在 Secrets Manager 中建立秘密。建立 Secret Manager 秘密後，更新應用程式程式碼即可擷取秘密。

在保護 Systems Manager Parameter Store 中的敏感資料之前，請先檢閱以下考量事項。

- 僅支援儲存文字資料的秘密。不支援儲存二進位資料的秘密。
- 使用介面 VPC 端點來增強安全控制。
- 任務所使用的 VPC 必須使用 DNS 解析。

所需的 IAM 許可

若要使用此功能，您必須擁有 Amazon ECS 任務角色，並在您的任務定義中參考該角色。這可讓容器代理程式提取必要的 Systems Manager 資源。如需詳細資訊，請參閱 Amazon ECS 任務 IAM 角色。

> **Important**
>
> 對於使用 EC2 啟動類型的任務，您必須使用 ECS 代理程式組態變數 `ECS_ENABLE_AWSLOGS_EXECUTIONROLE_OVERRIDE=true`，才能使用此功能。您可以在建立容器執行個體期間將其新增至 `./etc/ecs/ecs.config` 檔案，也可以將其新增至現有的執行個體，然後重新啟動 ECS 代理程式。如需詳細資訊，請參閱 Amazon ECS 容器代理程式組態。

若要允許存取您建立的 Systems Manager Parameter Store 參數，請將以下許可當作政策，手動新增到任務執行角色。如需有關如何管理許可的相關資訊，請參閱《IAM 使用者指南》中的新增和移除 IAM 身分許可。

- `ssm:GetParameters` – 如果參考任務定義中的 Systems Manager Parameter Store 參數，則此項目為必要。新增擷取 Systems Manager 參數的許可。
- `secretsmanager:GetSecretValue` – 如果直接參考 Secrets Manager 秘密，或者 Systems Manager Parameter Store 參數參考任務定義中的 Secrets Manager 秘密，則此項目為必要。新增從 Secrets Manager 擷取密碼的許可。
- `kms:Decrypt` – 只有在秘密使用客戶受管金鑰而非預設金鑰時，此項目為必要。您的自訂金鑰的 ARN 應該新增為資源。新增解密客戶受管金鑰的許可。

下列政策範例新增必要許可：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ssm:GetParameters",
        "secretsmanager:GetSecretValue",
        "kms:Decrypt"
      ],
      "Resource": [
        "arn:aws:ssm:region:aws_account_id:parameter/parameter_name",
        "arn:aws:secretsmanager:region:aws_account_id:secret:secret_name",
        "arn:aws:kms:region:aws_account_id:key/key_id"
      ]
    }
  ]
}
```

建立 參數

您可以使用 Systems Manager 主控台為您的敏感資料建立 Systems Manager Parameter Store 參數。如需詳細資訊，請參閱《AWS Systems Manager 使用者指南》中的建立 Systems Manager 參數 (主控台) 或建立 Systems Manager 參數 (AWS CLI)。

更新應用程式，以程式設計方式擷取 Systems Manager Parameter Store 秘密

若要擷取儲存在 Systems Manager 參數存放區參數中的機密資料，請參閱 SDK 程式碼範例程式碼程式庫中使用 AWSAWS SDK 的 Systems Manager 程式碼範例。

透過 Amazon ECS 環境變數擷取秘密管理員機密

當您將秘密作為環境變數插入時，可以指定秘密的完整內容、秘密內的特定 JSON 金鑰或要插入的特定秘密版本。這可協助您控制向容器公開的敏感資料。如需有關秘密版本控制的詳細資訊，請參閱《AWS Secrets Manager 使用者指南》中的 AWS Secrets Manager重要術語和概念。

使用環境變數將 Secret Manager 密碼插入容器時，應考慮下列事項。

- 敏感資料會在初次啟動容器時，嵌入您的容器。如果後續更新或輪換秘密，則容器不會自動收到更新的值。您必須啟動新的任務，或如果任務是服務的一部分，您可以更新服務，並使用 Force new deployment (強制新的部署) 選項強制服務來啟動新的任務。
- 對於上的 Amazon ECS 任務 AWS Fargate，應考慮下列事項：
  - 若要將秘密的完整內容作為環境變數或在日誌組態中插入，您必須使用平台版本 `1.3.0` 或更新版本。如需相關資訊，請參閱適用於 Amazon ECS 的 Fargate Linux 平台版本。
  - 若要將特定 JSON 金鑰或秘密版本作為環境變數或在日誌組態中插入，您必須使用平台版本 `1.4.0` 或更新版本 (Linux)，或者 `1.0.0` (Windows)。如需相關資訊，請參閱適用於 Amazon ECS 的 Fargate Linux 平台版本。
- 對於 EC2 上的 Amazon ECS 任務，應考慮下列事項：
  - 若要使用秘密的特定 JSON 索引鍵或版本插入秘密，您的容器執行個體必須有 `1.37.0` 版或更新版本的容器代理程式。不過，我們建議您使用最新版的容器代理程式。如需檢查代理程式版本及更新至最新版本的資訊，請參閱「更新 Amazon ECS 容器代理程式」。

    若要插入秘密的完整內容做為環境變數，或在日誌組態中插入秘密，則容器執行個體必須有 `1.22.0` 版或更新版本的容器代理程式。
- 使用介面 VPC 私人雲端端點來增強安全性控制，並透過私有子網路連線至 Secrets Manager。您必須建立 Secrets Manager 的介面 VPC 端點。如需有關 VPC 端點的資訊，請參閱《AWS Secrets Manager 使用者指南》中的建立VPC 端點。如需有關使用 Secrets Manager 和 Amazon VPC 的詳細資訊，請參閱如何在 Amazon VPC 中連接到 Secrets Manager 服務。
- 對於設定為使用 `awslogs` 日誌記錄驅動程式的 Windows 任務，您也必須在容器執行個體上設定 `ECS_ENABLE_AWSLOGS_EXECUTIONROLE_OVERRIDE` 環境變數。這可以透過使用者資料完成，語法如下：

```
<powershell>
```

```
[Environment]::SetEnvironmentVariable("ECS_ENABLE_AWSLOGS_EXECUTIONROLE_OVERRIDE",
 $TRUE, "Machine")
Initialize-ECSAgent -Cluster <cluster name> -EnableTaskIAMRole -LoggingDrivers
 '["json-file","awslogs"]'
</powershell>
```

IAM 許可

若要使用此功能，您必須擁有 Amazon ECS 任務執行角色，並在您的任務定義中參考它。如需詳細資訊，請參閱 Amazon ECS 任務執行 IAM 角色。

若要將存取提供給您建立的 Secrets Manager 秘密，請將以下許可做為內嵌政策，手動新增到任務執行角色。如需詳細資訊，請參閱新增和移除 IAM 政策。

- `secretsmanager:GetSecretValue` - 如果您要參考 Secrets Manager 秘密，則需要此項目。新增從 Secrets Manager 擷取密碼的許可。
- `kms:Decrypt` – 只有在秘密使用客戶受管金鑰而非預設金鑰時，此項目為必要。客戶受管金鑰的 ARN 應該新增為資源。新增解密客戶受管金鑰的許可。

下列政策範例新增必要許可：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "secretsmanager:GetSecretValue",
        "kms:Decrypt"
      ],
      "Resource": [
        "arn:aws:secretsmanager:region:aws_account_id:secret:secret_name",
        "arn:aws:kms:region:aws_account_id:key/key_id"
      ]
    }
  ]
}
```

建立 AWS Secrets Manager 秘密

您可以使用 Secrets Manager 主控台為您的敏感資料建立秘密。若要取得更多資訊，請參閱《AWS Secrets Manager 使用指南》中的〈建立 AWS Secrets Manager 密碼〉。

將環境變數新增至容器定義

您可以在容器定義中指定下列項目：

- 包含要在容器中設定之環境變數名稱的 `secrets` 物件。
- Secrets Manager 秘密的 Amazon Resource Name (ARN)
- 包含要提供給容器之敏感資料的其他參數

下列範例示範了必須為 Secrets Manager 秘密指定的完整語法。

```
arn:aws:secretsmanager:region:aws_account_id:secret:secret-name:json-key:version-stage:version-id
```

以下部分說明其他參數。這些參數雖然是選用，但如果您不使用，則必須包含冒號：來使用預設值。以下提供範例深入說明。

`json-key`

使用您要設為環境變數值的值，來指定金鑰/值對中的金鑰名稱。僅支援 JSON 格式的值。如果您沒有指定 JSON 金鑰，則會使用秘密的完整內容。

`version-stage`

指定您要使用之秘密版本的預備標籤。如果指定了版本預備標籤，就無法指定版本 ID。如果未指定版本階段，則預設會擷取具有 `AWSCURRENT` 階段標籤的秘密。

預備標籤會用來在不同版本的秘密更新或輪換時加以追蹤。每個版本的秘密都有一或多個預備標籤和 ID。如需詳細資訊，請參閱 《AWS Secrets Manager 使用者指南》中的 AWS Secrets Manager 的重要術語和概念。

`version-id`

針對您要使用的秘密版本，指定其唯一識別符。如果指定了版本 ID，就無法指定版本預備標籤。如果未指定版本 ID，則預設會擷取具有 `AWSCURRENT` 階段標籤的秘密。

版本 ID 會用來在不同版本的秘密更新或輪換時加以追蹤。每個版本的秘密都有 ID。如需詳細資訊，請參閱 《AWS Secrets Manager 使用者指南》中的 AWS Secrets Manager的重要術語和概念。

容器定義範例

下列範例示範您可以在容器定義中參考 Secrets Manager 秘密的方法。

Example 參考完整秘密

以下是任務定義的程式碼片段，顯示參考 Secrets Manager 秘密全文時的格式。

```
{
  "containerDefinitions": [{
    "secrets": [{
      "name": "environment_variable_name",
      "valueFrom": "arn:aws:secretsmanager:region:aws_account_id:secret:secret_name-
AbCdEf"
    }]
  }]
}
```

若要從容器內存取此秘密的值，您需要呼叫 $environment_variable_name。

Example 參考秘密中的特定金鑰

以下示範 get-secret-value 命令的範例輸出，會顯示秘密的內容，以及與其相關的版本預備標籤和版本 ID。

```
{
    "ARN": "arn:aws:secretsmanager:region:aws_account_id:secret:appauthexample-AbCdEf",
    "Name": "appauthexample",
    "VersionId": "871d9eca-18aa-46a9-8785-981ddEXAMPLE",
    "SecretString": "{\"username1\":\"password1\",\"username2\":\"password2\",
\"username3\":\"password3\"}",
    "VersionStages": [
        "AWSCURRENT"
    ],
    "CreatedDate": 1581968848.921
}
```

在 ARN 結尾指定金鑰名稱，來在容器定義中參考上一個輸出的特定金鑰。

```
{
  "containerDefinitions": [{
    "secrets": [{
      "name": "environment_variable_name",
      "valueFrom": "arn:aws:secretsmanager:region:aws_account_id:secret:appauthexample-AbCdEf:username1::"
    }]
  }]
}
```

Example 參考特定秘密版本

以下示範 describe-secret 命令的範例輸出，會顯示秘密的未加密內容，以及所有版本秘密的中繼資料。

```
{
    "ARN": "arn:aws:secretsmanager:region:aws_account_id:secret:appauthexample-AbCdEf",
    "Name": "appauthexample",
    "Description": "Example of a secret containing application authorization data.",
    "RotationEnabled": false,
    "LastChangedDate": 1581968848.926,
    "LastAccessedDate": 1581897600.0,
    "Tags": [],
    "VersionIdsToStages": {
        "871d9eca-18aa-46a9-8785-981ddEXAMPLE": [
            "AWSCURRENT"
        ],
        "9d4cb84b-ad69-40c0-a0ab-cead3EXAMPLE": [
            "AWSPREVIOUS"
        ]
    }
}
```

在 ARN 結尾指定金鑰名稱，來在容器定義中參考上一個輸出的特定版本預備標籤。

```
{
  "containerDefinitions": [{
    "secrets": [{
      "name": "environment_variable_name",
      "valueFrom": "arn:aws:secretsmanager:region:aws_account_id:secret:appauthexample-AbCdEf::AWSPREVIOUS:"
```

```
    }]
  }]
}
```

在 ARN 結尾指定金鑰名稱，來在容器定義中參考上一個輸出的特定版本 ID。

```
{
  "containerDefinitions": [{
    "secrets": [{
      "name": "environment_variable_name",
      "valueFrom": "arn:aws:secretsmanager:region:aws_account_id:secret:appauthexample-
AbCdEf:::9d4cb84b-ad69-40c0-a0ab-cead3EXAMPLE"
    }]
  }]
}
```

Example 參考秘密的特定金鑰和版本預備標籤

以下說明如何參考秘密中的特定金鑰和特定版本預備標籤。

```
{
  "containerDefinitions": [{
    "secrets": [{
      "name": "environment_variable_name",
      "valueFrom": "arn:aws:secretsmanager:region:aws_account_id:secret:appauthexample-
AbCdEf:username1:AWSPREVIOUS:"
    }]
  }]
}
```

若要指定特定的金鑰和版本 ID，請使用下列語法。

```
{
  "containerDefinitions": [{
    "secrets": [{
      "name": "environment_variable_name",
      "valueFrom": "arn:aws:secretsmanager:region:aws_account_id:secret:appauthexample-
AbCdEf:username1::9d4cb84b-ad69-40c0-a0ab-cead3EXAMPLE"
    }]
  }]
}
```

如需有關如何使用環境變數中指定的秘密建立任務定義的資訊，請參閱使用主控台建立 Amazon ECS 任務定義。

透過 Amazon ECS 環境變數擷取 Systems Manager 參數

Amazon ECS 可讓您將敏感資料插入容器，方法是將敏感資料儲存在參數存放區參數中 AWS Systems Manager ，然後在容器定義中參照這些資料。

使用環境變數將 Systems Manager 密碼插入容器時，請考慮下列事項。

- 敏感資料會在初次啟動容器時，嵌入您的容器。如果後續更新或輪換秘密，則容器不會自動收到更新的值。您必須啟動新的任務，或如果任務是服務的一部分，您可以更新服務，並使用 Force new deployment (強制新的部署) 選項強制服務來啟動新的任務。
- 對於上的 Amazon ECS 任務 AWS Fargate，應考慮下列事項：
  - 若要將秘密的完整內容作為環境變數或在日誌組態中插入，您必須使用平台版本 `1.3.0` 或更新版本。如需相關資訊，請參閱適用於 Amazon ECS 的 Fargate Linux 平台版本。
  - 若要將特定 JSON 金鑰或秘密版本作為環境變數或在日誌組態中插入，您必須使用平台版本 `1.4.0` 或更新版本 (Linux)，或者 `1.0.0` (Windows)。如需相關資訊，請參閱適用於 Amazon ECS 的 Fargate Linux 平台版本。
- 對於 EC2 上的 Amazon ECS 任務，應考慮下列事項：
  - 若要使用秘密的特定 JSON 索引鍵或版本插入秘密，您的容器執行個體必須有 `1.37.0` 版或更新版本的容器代理程式。不過，我們建議您使用最新版的容器代理程式。如需檢查代理程式版本及更新至最新版本的資訊，請參閱「更新 Amazon ECS 容器代理程式」。

    若要插入秘密的完整內容做為環境變數，或在日誌組態中插入秘密，則容器執行個體必須有 `1.22.0` 版或更新版本的容器代理程式。
- 使用介面 VPC 端點來增強安全控制。您必須為 Systems Manager 建立介面 VPC 端點。如需有關 VPC 端點的資訊，請參閱《AWS Systems Manager 使用者指南》中的建立VPC 端點。
- 對於設定為使用 `awslogs` 日誌記錄驅動程式的 Windows 任務，您也必須在容器執行個體上設定 `ECS_ENABLE_AWSLOGS_EXECUTIONROLE_OVERRIDE` 環境變數。這可以透過 User Data (使用者資料) 完成，語法如下：

```
<powershell>
[Environment]::SetEnvironmentVariable("ECS_ENABLE_AWSLOGS_EXECUTIONROLE_OVERRIDE",
 $TRUE, "Machine")
Initialize-ECSAgent -Cluster <cluster name> -EnableTaskIAMRole -LoggingDrivers
 '["json-file","awslogs"]'
```

```
</powershell>
```

IAM 許可

若要使用此功能，您必須擁有 Amazon ECS 任務執行角色，並在您的任務定義中參考它。這可讓容器代理程式提取必要的 Systems Manager 資源。如需詳細資訊，請參閱 Amazon ECS 任務執行 IAM 角色。

> **Important**
>
> 對於使用 EC2 啟動類型的任務，您必須使用 ECS 代理程式組態變數 `ECS_ENABLE_AWSLOGS_EXECUTIONROLE_OVERRIDE=true`，才能使用此功能。您可以在建立容器執行個體期間將其新增至 `./etc/ecs/ecs.config` 檔案，也可以將其新增至現有的執行個體，然後重新啟動 ECS 代理程式。如需詳細資訊，請參閱 Amazon ECS 容器代理程式組態。

若要提供您所建立之「Systems Manager 參數存放區」參數的存取權，請手動將下列權限新增至工作執行角色。如需有關如何管理許可的相關資訊，請參閱《IAM 使用者指南》中的新增和移除 IAM 身分許可。

- `ssm:GetParameters` – 如果參考任務定義中的 Systems Manager Parameter Store 參數，則此項目為必要。新增擷取 Systems Manager 參數的許可。
- `secretsmanager:GetSecretValue` – 如果直接參考 Secrets Manager 秘密，或者 Systems Manager Parameter Store 參數參考任務定義中的 Secrets Manager 秘密，則此項目為必要。新增從 Secrets Manager 擷取密碼的許可。
- `kms:Decrypt` – 只有在秘密使用客戶受管金鑰而非預設金鑰時，此項目為必要。您的自訂金鑰的 ARN 應該新增為資源。新增解密客戶受管金鑰的許可。

下列政策範例新增必要許可：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ssm:GetParameters",
```

```
        "secretsmanager:GetSecretValue",
        "kms:Decrypt"
      ],
      "Resource": [
        "arn:aws:ssm:region:aws_account_id:parameter/parameter_name",
        "arn:aws:secretsmanager:region:aws_account_id:secret:secret_name",
        "arn:aws:kms:region:aws_account_id:key/key_id"
      ]
    }
  ]
}
```

建立系 Systems Manager 參數

您可以使用 Systems Manager 主控台為您的敏感資料建立 Systems Manager Parameter Store 參數。如需詳細資訊，請參閱《AWS Systems Manager 使用者指南》中的建立 Systems Manager 參數 (主控台) 或建立 Systems Manager 參數 (AWS CLI)。

將環境變數新增至容器定義

在您的容器定義內，將 `secrets` 指定為要在容器中設定的環境變數名稱，以及 Systems Manager 參數存放區參數 (含有要呈現給容器的敏感資料) 的完整 ARN。如需詳細資訊，請參閱 `secrets`。

以下是任務定義的程式碼片段，顯示參考 Systems Manager 參數存放區參數的格式。如果 Systems Manager 參數存放區參數與您要啟動的任務位於相同區域中，則您可以使用參數的完整 ARN 或名稱。如果參數存在於不同區域，則請指定完整 ARN。

```
{
  "containerDefinitions": [{
    "secrets": [{
      "name": "environment_variable_name",
      "valueFrom": "arn:aws:ssm:region:aws_account_id:parameter/parameter_name"
    }]
  }]
}
```

如需有關如何使用環境變數中指定的秘密建立任務定義的資訊，請參閱使用主控台建立 Amazon ECS 任務定義。

擷取 Amazon ECS 日誌記錄組態的密碼

您可以使用中的`secretOptions`參數`logConfiguration`來傳遞用於記錄的敏感資料。

您可以將秘密存儲在 Secrets Manager 或 Systems Manager 中。

使用 Secrets Manager

在您的容器定義內，指定 `logConfiguration` 時，您可指定 `secretOptions`，在該參數中，需提供要在容器中設定的日誌驅動程式選項名稱，以及含有要呈現給容器之敏感資料的 Secrets Manager 秘密之完整 ARN。

以下是任務定義的程式碼片段，顯示參考 Secrets Manager 秘密時的格式。

```
{
  "containerDefinitions": [{
    "logConfiguration": [{
      "logDriver": "splunk",
      "options": {
        "splunk-url": "https://your_splunk_instance:8088"
      },
      "secretOptions": [{
        "name": "splunk-token",
        "valueFrom": "arn:aws:secretsmanager:region:aws_account_id:secret:secret_name-
AbCdEf"
      }]
    }]
  }]
}
```

使用 Systems Manager

您可以將敏感資料插入日誌組態。在您的容器定義內，指定 `logConfiguration` 時，您可用要在容器中設定的日誌驅動程式選項名稱指定 `secretOptions`，以及 Systems Manager 參數存放區參數 (含有要呈現給容器的敏感資料) 的完整 ARN。

> **Important**
>
> 如果 Systems Manager 參數存放區參數與您要啟動的任務位於相同區域中，則您可以使用參數的完整 ARN 或名稱。如果參數存在於不同區域，則請指定完整 ARN。

以下是任務定義的程式碼片段，顯示參考 Systems Manager 參數存放區參數的格式。

```
{
```

```
  "containerDefinitions": [{
    "logConfiguration": [{
      "logDriver": "fluentd",
      "options": {
        "tag": "fluentd demo"
      },
      "secretOptions": [{
        "name": "fluentd-address",
        "valueFrom": "arn:aws:ssm:region:aws_account_id:parameter:/parameter_name"
      }]
    }]
  }]
}
```

使用 Amazon ECS 中的 Secrets Manager 密碼指定敏感資料

Amazon ECS 可讓您將敏感資料插入容器，方法是將敏感資料存放在機 AWS Secrets Manager 密中，然後在容器定義中參考這些資料。如需詳細資訊，請參閱 將敏感資料傳遞至 Amazon ECS 容器。

了解如何建立 Secrets Manager 密碼、參考 Amazon ECS 任務定義中的密碼，然後透過查詢顯示機密內容的容器內的環境變數來驗證其運作正常。

必要條件

本教學課程假設已完成下列先決條件：

- 已完成「設定以使用 Amazon ECS。」中的步驟。
- 您的 AWS 使用者具有建立所述 Secrets Manager 和 Amazon ECS 資源所需的 IAM 許可。

步驟 1：建立 Secrets Manager 秘密

您可以使用 Secrets Manager 主控台為您的敏感資料建立秘密。在本教學課程中，我們將建立基本秘密，以供存放容器中稍後參考的使用者名稱和密碼。如需詳細資訊，請參閱 AWS Secrets Manager 使用者指南中的建立基本秘密。

要儲存在此秘密中的鍵/值對是教學課程結尾處容器中的環境變量值。

儲存秘密 ARN，以在後續步驟的任務執行 IAM 政策和任務定義中參考。

步驟 2：更新您的任務執行 IAM 角色

若要讓 Amazon ECS 從您的 Secrets Manager 機密擷取敏感資料，您必須擁有 Amazon ECS 任務執行角色，並在您的任務定義中參考它。這可讓容器代理程式提取必要的 Secrets Manager 資源。如果您尚未建立任務執行 IAM 角色，請參閱 Amazon ECS 任務執行 IAM 角色。

下列步驟假設您已建立並正確設定任務執行 IAM 角色。

更新您的任務執行 IAM 角色

使用 IAM 主控台，以所需的許可更新您的任務執行角色。

1. 在以下網址開啟 IAM 主控台：https://console.aws.amazon.com/iam/。
2. 在導覽窗格中，選擇角色。
3. 搜尋 `ecsTaskExecutionRole` 的角色清單並加以選取。
4. 選擇 Permissions (許可)、Add inline policy (新增內嵌政策)。
5. 選擇 JSON 索引標籤並指定以下 JSON 文字，確保您為步驟 1 建立的 Secrets Manager 機密指定完整的 ARN。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "secretsmanager:GetSecretValue"
            ],
            "Resource": [

 "arn:aws:secretsmanager:region:aws_account_id:secret:username_value"
            ]
        }
    ]
}
```

6. 選擇檢閱政策。針對 Name (名稱) 指定 `ECSSecretsTutorial`，然後選擇 Create policy (建立政策)。

步驟 3：建立 Amazon ECS 任務定義

您可以使用 Amazon ECS 主控台來建立一個參考 Secrets Manager 秘密的任務定義。

建立一個指定秘密的任務定義

使用 IAM 主控台，以所需的許可更新您的任務執行角色。

1. 開啟主控台，網址為 https://console.aws.amazon.com/ecs/v2。
2. 在導覽窗格中，選擇 Task Definitions (任務定義)。
3. 選擇 Create new task definitio (建立新任務定義)、Create new task definition with JSON (使用 JSON 建立新的任務定義)。
4. 在 JSON 編輯器方塊中輸入以下任務定義 JSON 文字，確保您為步驟 1 建立的 Secrets Manager 密碼和步驟 2 更新的任務執行 IAM 角色，指定完整的 ARN。選擇儲存。
5.

```
{
    "executionRoleArn": "arn:aws:iam::aws_account_id:role/ecsTaskExecutionRole",
    "containerDefinitions": [
        {
            "entryPoint": [
                "sh",
                "-c"
            ],
            "portMappings": [
                {
                    "hostPort": 80,
                    "protocol": "tcp",
                    "containerPort": 80
                }
            ],
            "command": [
                "/bin/sh -c \"echo '<html> <head> <title>Amazon ECS Sample
 App</title> <style>body {margin-top: 40px; background-color: #333;} </style> </
head><body> <div style=color:white;text-align:center> <h1>Amazon ECS Sample App</
h1> <h2>Congratulations!</h2> <p>Your application is now running on a container in
 Amazon ECS.</p> </div></body></html>' >  /usr/local/apache2/htdocs/index.html &&
 httpd-foreground\""
            ],
            "cpu": 10,
            "secrets": [
                {
                    "valueFrom":
 "arn:aws:secretsmanager:region:aws_account_id:secret:username_value",
                    "name": "username_value"
                }
            ],
```

```
            "memory": 300,
            "image": "httpd:2.4",
            "essential": true,
            "name": "ecs-secrets-container"
        }
    ],
    "family": "ecs-secrets-tutorial"
}
```

6. 選擇建立。

步驟 4：建立 Amazon ECS 叢集

您可以使用 Amazon ECS 主控台建立一個叢集，其中包含要執行任務的容器執行個體。如果您的現有叢集有至少一個向其註冊的容器執行個體，並有可用資源可執行為此教學課程建立的一個任務定義執行個體，您可以跳到下一個步驟。

在本教學課程中，我們將使用 Amazon ECS 最佳化 Amazon Linux 2 AMI 建立具有一個 `t2.micro` 容器執行個體的叢集。

如需有關如何建立 EC2 啟動類型叢集的相關資訊，請參閱 the section called “為 Amazon EC2 啟動類型建立叢集”。

步驟 5：執行 Amazon ECS 任務

您可以透過 Amazon ECS 主控台，使用您建立的任務定義來執行任務。在本教學課程中，我們將會執行使用 EC2 啟動類型的任務，並使用我們在前一個步驟中建立的叢集。

如需有關如何執行任務的資訊，請參閱 the section called “將應用程式作為工作執行”。

步驟 6：驗證

您可以使用下列步驟，驗證是否已成功完成所有步驟，以及是否已在您的容器中正確建立環境變數。

驗證是否已建立環境變數

1. 尋找您容器執行個體的公有 IP 或 DNS 地址。
    a. 開啟主控台，網址為 https://console.aws.amazon.com/ecs/v2。
    b. 在導覽窗格中，選擇叢集，然後選擇您建立的叢集。
    c. 選擇基礎設施，然後選擇容器執行個體。

d. 記錄您執行個體的公有 IP 或公有 DNS。

2. 如果您使用的是 macOS 或 Linux 電腦，請使用下列命令連線到您的執行個體，並替換為您私有金鑰的路徑及執行個體的公有地址：

```
$ ssh -i /path/to/my-key-pair.pem ec2-user@ec2-198-51-100-1.compute-1.amazonaws.com
```

如需有關使用 Windows 電腦的詳細資訊，請參閱 Amazon EC2 使用者指南中的使用 PuTTY 從視窗連接到 Linux 執行個體。

> **Important**
>
> 如需連線至執行個體時發生任何問題的詳細資訊，請參閱 Amazon EC2 使用者指南中的連線至執行個體疑難排解。

3. 列出在執行個體上執行的容器。請記下 `ecs-secrets-tutorial` 容器的容器 ID。

```
docker ps
```

4. 使用上一個步驟輸出中的容器 ID，連接到 `ecs-secrets-tutorial` 容器。

```
docker exec -it container_ID /bin/bash
```

5. 使用 `echo` 命令來列印環境變數的值。

```
echo $username_value
```

如果教學課程成功，您應該會看到以下輸出：

```
password_value
```

> **Note**
>
> 或者，您可以使用 `env` (或 `printenv`) 命令列出您容器中的所有環境變數。

步驟 7：清除

完成此教學課程時，建議您清除相關聯的資源，以免未使用的資源產生費用。

清除資源

1. 開啟主控台，網址為 https://console.aws.amazon.com/ecs/v2。
2. 在導覽窗格中，選擇叢集。
3. 在叢集頁面上，選擇叢集。
4. 選擇 Delete Cluster (刪除叢集)。
5. 在確認方塊中，輸入 delete **####**，然後選擇刪除。
6. 在以下網址開啟 IAM 主控台：https://console.aws.amazon.com/iam/。
7. 在導覽窗格中，選擇角色。
8. 搜尋 `ecsTaskExecutionRole` 的角色清單並加以選取。
9. 選擇「權限」，然後選擇 ECS SecretsTutorial 旁邊的 X。選擇移除。
10. 開啟位於的 Secrets Manager 主控台https://console.aws.amazon.com/secretsmanager/。
11. 選取您所建立的 username_value 秘密，然後選擇 Actions (動作)、Delete secret (刪除秘密)。

Amazon ECS 任務定義參數

任務定義分為單獨的部分：任務系列、 AWS Identity and Access Management (IAM) 任務角色、網路模式、容器定義、磁碟區、任務放置限制和啟動類型。任務定義中需要系列和容器定義。相對的，任務角色、網路模式、磁碟區、任務置放限制及啟動類型則為選用的。

您可以在 JSON 檔案中使用這些參數來設定任務定義。

以下是每個任務定義參數更詳細的描述。

系列

`family`

類型：字串

必要：是

當您註冊任務定義時，您會指定它的系列，這類似於任務定義以修訂版號碼指定的多個版本名稱。在特定系列註冊的第一個任務定義會得到修訂版 1，之後註冊的任何任務定義，則得到連續的修訂版號碼。

啟動類型

當您註冊任務定義時，您可以指定 Amazon ECS 驗證任務定義所依據的啟動類型。如果任務定義不會依據指定的相容性進行驗證，則會傳回用戶端例外狀況。如需詳細資訊，請參閱 Amazon ECS 啟動類型。

任務定義允許使用以下參數。

`requiresCompatibilities`

類型：字串陣列

必要：否

有效值: `EC2` | `FARGATE` | `EXTERNAL`

驗證任務定義所依據的啟動類型。這會啟動檢查，以確保任務定義中使用的所有參數都能滿足啟動類型的要求。

任務角色

`taskRoleArn`

類型：字串

必要：否

當您註冊任務定義時，您可以為 IAM 角色提供任務角色，該角色可讓任務許可中的容器，代您呼叫其關聯政策中指定的 AWS API。如需詳細資訊，請參閱 Amazon ECS 任務 IAM 角色。

在您啟動 Amazon ECS 最佳化 Windows Server AMI 時，Windows 上任務的 IAM 角色要求設定 `-EnableTaskIAMRole` 選項。您的容器還必須執行一些組態程式碼，以使用此功能。如需詳細資訊，請參閱 Amazon EC2 窗口實例附加配置。

任務執行角色

`executionRoleArn`

類型：字串

必要：有條件

任務執行角色的 Amazon 資源名稱 (ARN)，可授與 Amazon ECS 容器代理程式的權限，以代表您進行 AWS API 呼叫。

> **Note**
>
> 視任務需求而定，任務執行 IAM 角色是必要的項目。如需詳細資訊，請參閱 Amazon ECS 任務執行 IAM 角色。

網路模式

`networkMode`

類型：字串

必要：否

任務中的容器所使用的 Docker 聯網模式。對於在 Amazon EC2 Linux 執行個體上託管的 Amazon ECS 任務，有效值為 `none`、`bridge`、`awsvpc` 以及 `host`。如果未指定網路模式，預設網路模式為 `bridge`。對於在 Amazon EC2 Windows 執行個體上託管的 Amazon ECS 任務，有效值為 `default` 和 `awsvpc`。如果未指定網路模式，會使用 `default` 網路模式。對於在 Fargate 上託管的 Amazon ECS 任務，需要使用`awsvpc`網路模式。

如果網路模式設為 `none`，任務的容器即沒有外部連線，並且無法在容器定義中指定連接埠映射。

如果網路模式為 `bridge`，任務會使用 Linux 上的 Docker 內建虛擬網路，該虛擬網路會在託管任務的每個 Amazon EC2 執行個體內執行。Linux 上的內建虛擬網路會使用 `bridge` Docker 網路驅動程式。

如果網路模式為 `host`，任務會使用主機網路，並直接將容器連接埠映射至託管任務的 Amazon EC2 執行個體 ENI，以略過 Docker 的內建虛擬網路。動態連接埠映射無法在此網路模式下使用。任務定義中使用此模式的容器必須指定特定 `hostPort` 數字。主機上的連接埠號碼無法由多個任務使用。因此，您無法在單一 Amazon EC2 執行個體上執行同一個任務定義的多個任務。

> **Important**
>
> 執行使用 `host` 網路模式的任務時，為了獲得更好的安全性，不要使用根使用者 (UID 0) 執行容器。作為最佳安全實務，請一律使用非根使用者。

對於 Amazon EC2 啟動類型，如果網路模式為awsvpc，則會將任務分配一個 elastic network interface，您必須在建立服務或使用任務定義執行任務NetworkConfiguration時指定一個。如需詳細資訊，請參閱 適用於 EC2 啟動類型的 Amazon ECS 任務聯網選項。

如果網路模式為 default，任務會使用 Windows 上的 Docker 內建虛擬網路，該虛擬網路會在託管任務的每個 Amazon EC2 執行個體內執行。Windows 上的內建虛擬網路會使用 nat Docker 網路驅動程式。

對於 Fargate 啟動類型，當網路模式為時awsvpc，工作會配置 elastic network interface，而且您必須在建立服務或使用工作定義執行工作NetworkConfiguration時指定一個。如需詳細資訊，請參閱 Fargate 工作網路。awsvpc 網路模式為容器提供最高聯網效能，因為它們使用的是 Amazon EC2 網路堆疊。公開的容器連接埠會直接映射至連接的彈性網路介面連接埠。因此，您無法使用動態主機連接埠對應。

host 和 awsvpc 網路模式為容器提供最高聯網效能，因為它們使用的是 Amazon EC2 網路堆疊。使用 host 和 awsvpc 網路模式，公開的容器連接埠會直接映射到對應的主機連接埠 (適用於 host 網路模式) 或已連接的彈性網路介面連接埠 (適用於 awsvpc 網路模式)。因此，您無法使用動態主機連接埠對應。

如果使用 Fargate 啟動類型，則需要 awsvpc 網路模式。如果使用 EC2 啟動類型，允許的網路模式取決於基礎 EC2 執行個體的作業系統。如果是 Linux，則可以使用任何網路模式。如果是 Windows，則會使用 default 和 awsvpc 模式。

執行時間平台

operatingSystemFamily

類型：字串

必要：有條件

預設：LINUX

在 Fargate 上託管的 Amazon ECS 任務需要此參數。

在註冊任務定義時，您會指定作業系統系列。

在 Fargate 上託管的 Amazon ECS 任務的有效值為 LINUX、WINDOWS_SERVER_2019_FULL、WINDOWS_SERVER_2019_CORE、WINDOWS_SERVER_2022_ 和 WINDOWS_SERVER_2022_CORE 。

在 EC2 上託管的 Amazon ECS 任務的有效值為 LINUX、WINDOWS_SERVER_2022_CORE、WINDOWS_SERVER_2022_FULL、WINDOWS_SERVER_2019_ 以及 WINDOWS_SERVER_2019_CORE、WINDOWS_SERVER_2016_FULL、WINDOWS_SERVER_2004_CORE 和 WINDOWS_SERVER_20H2_CORE。

服務中使用的所有任務定義都必須與此參數具有相同的值。

如果任務定義為服務的一部分，此值必須與服務 platformFamily 值相符。

cpuArchitecture

類型：字串

必要：有條件

預設：X86_64

在 Fargate 上託管的 Amazon ECS 任務需要此參數。如果參數保留為 null，則會在 Fargate 上託管的任務啟動時自動指派預設值。

在註冊任務定義時，您會指定 CPU 架構。有效值為 X86_64 和 ARM64。

服務中使用的所有任務定義都必須與此參數具有相同的值。

若您具有 Fargate 啟動類型或 EC2 啟動類型的 Linux 任務，則可將值設定為 ARM64。如需詳細資訊，請參閱 the section called “64 位元 ARM 工作負載的工作定義”。

任務大小

當您註冊任務定義時，您可以指定任務使用的 CPU 和記憶體總計。這和容器定義層級的 cpu 和 memory 值是分開的。對於在 Amazon EC2 執行個體上託管的任務，這些欄位是選用欄位。對於在 Fargate (Linux 和 Windows 皆可) 上託管的任務，這些欄位為必填欄位，而且受支援的 cpu 和 memory 具有特定值。

> **Note**
>
> Windows 容器會忽略任務層級的 CPU 和記憶體參數。我們建議為 Windows 容器指定容器層級的資源。

任務定義允許使用以下參數：

`cpu`

類型：字串

必要：有條件

> **Note**
>
> Windows 容器不支援此參數。

針對任務呈現的 CPU 單位硬性限制。您可以將 JSON 檔案中的 CPU 值指定為以 CPU 單位或虛擬 CPU (vCPUs) 為單位的字串。例如，您可以使用 CPU 單位或 1 vCPU vCPUs 來指定 CPU 值。`1024`註冊任務定義時，vCPU 值會轉換為整數，指出 CPU 單位。

對於在 EC2 或外部執行個體上執行的任務，此欄位是選用欄位。如果您的叢集沒有任何一個已註冊的容器執行個體具有可用的請求 CPU 單位，則任務會失敗。在 EC2 或外部執行個體上執行的任務支援值介於 `0.125` vCPU 和 `10` vCPU 之間。

對於在 Fargate 上 (Linux 和 Windows 容器皆可) 執行的任務，此欄位為必填欄位，而且您必須使用下列其中一個值，以決定 `memory` 參數的受支援值範圍。下表顯示有效的任務層級 CPU 和記憶體組合。

CPU 數值	記憶體數值	AWS Fargate 支援的作業系統
256 (.25 vCPU)	512 MiB、1 GB、2 GB	Linux
512 (.5 vCPU)	1 GB、2 GB、3 GB、4 GB	Linux
1024 (1 vCPU)	2 GB、3 GB、4 GB、5 GB、6 GB、7 GB、8 GB	Linux、Windows
2048 (2 vCPU)	介於 4 GB 與 16 GB 之間，以 1 GB 為單位遞增	Linux、Windows
4096 (4 vCPU)	介於 8 GB 與 30 GB 之間，以 1 GB 為單位遞增	Linux、Windows
8192 (8 vCPU)	介於 16 GB 與 60 GB 之間，以 4 GB 為單位遞增	Linux

CPU 數值	記憶體數值	AWS Fargate 支援的作業系統
Note 此選項需要 Linux 平台 1.4.0 或更新版本。		
16384 (16vCPU) Note 此選項需要 Linux 平台 1.4.0 或更新版本。	介於 32 GB 與 120 GB 之間，以 8 GB 為單位遞增	Linux

`memory`

類型：字串

必要：有條件

> **Note**
>
> Windows 容器不支援此參數。

要呈現給工作的記憶體硬性限制。您可以將工作定義中的記憶體值指定為以 MB (MiB) 或 GB 為單位的字串。例如，您可以將記憶體值指定為 3072 MiB 或 3 GB GB。註冊任務定義時，GB 值會轉換為整數，指出 MiB。

對於在 Amazon EC2 執行個體上託管的任務，此欄位為選用欄位，可使用任意值。如果指定任務層級的記憶體值，則容器層級的記憶體值是選用的。如果您的叢集沒有任何一個已註冊的容器執行個體具有可用的請求記憶體，則任務會失敗。您可以盡可能為特定執行個體類型的任務提供最多的記憶體，以將資源使用率最大化。如需詳細資訊，請參閱 保留 Amazon ECS Linux 容器執行個體記憶體。

對於在 Fargate 上 (Linux 和 Windows 皆可) 託管的任務，此欄位為必填欄位，而且您必須使用下列其中一個值，以決定 cpu 參數的受支援值的範圍：

記憶體值 (單位為 MiB，以 GB 為單位的近似等值)	CPU 數值	Fargate 支援的作業系統
512 (0.5 GB)、1024 (1 GB)、2048 (2 GB)	256 (.25 vCPU)	Linux
1024 (1 GB)、2048 (2 GB)、3072 (3 GB)、4096 (4 GB)	512 (.5 vCPU)	Linux
2048 (2 GB)、3072 (3 GB)、4096 (4GB)、5120 (5 GB)、6144 (6 GB)、7168 (7 GB)、8192 (8 GB)	1024 (1 vCPU)	Linux、Windows
介於 4096 (4 GB) 與 16384 (16 GB) 之間，以 1024 (1 GB) 為單位遞增	2048 (2 vCPU)	Linux、Windows
介於 8192 (8 GB) 與 30720 (30 GB) 之間，以 1024 (1 GB) 為單位遞增	4096 (4 vCPU)	Linux、Windows
介於 16 GB 與 60 GB 之間，以 4 GB 為單位遞增 **Note** 此選項需要 Linux 平台 `1.4.0` 或更新版本。	8192 (8 vCPU)	Linux
介於 32 GB 與 120 GB 之間，以 8 GB 為單位遞增	16384 (16vCPU)	Linux

記憶體值 (單位為 MiB，以 GB 為單位的近似等值)	CPU 數值	Fargate 支援的作業系統
Note 此選項需要 Linux 平台 1.4.0 或更新版本。		

容器定義

註冊任務定義時，您必須指定容器定義清單，以傳遞到容器執行個體上的 Docker 常駐程式。容器定義允許使用以下參數。

主題

- 標準容器定義參數
- 進階容器定義參數
- 其他容器定義參數

標準容器定義參數

下列任務定義參數為必要參數，或是會在大部分的容器定義中使用。

主題

- 名稱
- 映像
- 記憶體
- 連接埠映射
- 私有儲存庫憑證

名稱

name

類型：字串

必要：是

容器的名稱。可以包含最多可達 255 個字元 (大小寫)、數字、連字號和底線。如果您在任務定義中連結多個容器，則一個容器的 name 可以輸入另一個容器的 links。這是為了連結容器。

映像

image

類型：字串

必要：是

用來啟動容器的映像。此字串會直接傳遞至 Docker 常駐程式。根據預設，Docker Hub 登錄檔中的映像為可用。您也可以使用 *repository-url*/*image*:*tag* 或 *repository-url*/*image*@*digest* 指定其他存放庫。允許最多 255 個字元 (大小寫)、數字、連字號、底線、等號、句號、正斜線、井號。此參數會映射至 Docker Remote API 的建立容器區段中的 Image 以及 docker run 的 IMAGE 參數。

- 當新的任務啟動時，Amazon ECS 容器代理會提取最新版本的指定映像和標籤，以供容器使用。但是，儲存庫映像後續的更新將不會散佈到已在執行中的任務。
- 支援私有登錄檔中的映像。如需詳細資訊，請參閱 在 Amazon ECS 中使用非AWS 容器映像。
- 可使用完整的 registry/repository:tag 或 registry/repository@digest 命名慣例來指定 Amazon ECR 儲存庫中的映像 (例如 *aws_account_id*.dkr.ecr.*region*.amazonaws.com/*my-web-app*:*latest* 或 *aws_account_id*.dkr.ecr.*region*.amazonaws.com/*my-web-app*@*sha256:94afd1f2e64d908bc90dbca0035a5b567EXAMPLE*)。
- Docker Hub 上官方儲存庫中的映像，使用的是單一名稱 (例如，ubuntu 或 mongo)。
- Docker Hub 上的其他儲存庫中的映像要求使用組織名稱 (例如，amazon/amazon-ecs-agent)。
- 其他線上儲存庫中的映像更進一步要求使用網域名稱 (例如，quay.io/assemblyline/ubuntu)。

記憶體

memory

類型：整數

必要：否

提供給容器使用的記憶體數量 (MiB)。如果您的容器嘗試使用超過此處指定的記憶體，容器便會終止。為任務中所有容器預留的記憶體總量必須低於任務 `memory` 值 (如果已指定一個)。此參數會映射到 Docker Remote API 的建立容器區段中的 `Memory` 以及 docker run 的 `--memory` 選項。

如果使用 Fargate 啟動類型，這是選用參數。

如果使用 EC2 啟動類型，您必須指定任務層級的記憶體值或容器層級的記憶體值。如果您同時指定容器層級的 `memory` 和 `memoryReservation` 值,則 `memory` 值必須大於 `memoryReservation` 值。如果指定 `memoryReservation`，則會從放置容器的容器執行個體可用記憶體資源中減去該值。否則，即使用 `memory` 的值。

Docker 20.10.0 或更新版本的常駐程式會為容器保留最低 6 MiB 的記憶體。因此，不要為容器指定少於 6 MiB 的記憶體。

Docker 19.03.13-ce 或更舊版本的常駐程式會為容器保留最低 4 MiB 的記憶體。因此，不要為容器指定少於 4 MiB 的記憶體。

> **Note**
>
> 若您嘗試盡可能為特定執行個體類型的任務提供最多的記憶體，以將資源使用率最大化，請參閱「保留 Amazon ECS Linux 容器執行個體記憶體」。

`memoryReservation`

類型：整數

必要：否

為容器保留的記憶體軟性限制 (MiB)。當系統記憶體爭用時，Docker 會嘗試將容器記憶體保持在此軟性限制。但是，您的容器可以在需要時使用更多記憶體。容器可以使用最多達到以 `memory` 參數指定的硬性限制 (如適用)，或容器執行個體上的所有可用記憶體，以先達到者為準。此參數會映射到 Docker Remote API 的建立容器區段中的 `MemoryReservation` 以及 docker run 的 `--memory-reservation` 選項。

如果未指定任務層級的記憶體值，您必須為容器定義中的 `memory` 或 `memoryReservation` 之一或兩者指定非零整數。若您同時指定兩者，`memory` 必須大於 `memoryReservation`。如果指定

`memoryReservation`，則會從放置容器的容器執行個體可用記憶體資源中減去該值。否則，即使用 `memory` 的值。

例如，若您的容器通常會使用 128 MiB 的記憶體，但會偶爾短期爆量到 256 MiB 的記憶體。您可以將 `memoryReservation` 設為 128 MiB，並將 `memory` 硬性限制設為 300 MiB。此組態讓容器只從容器執行個體的剩餘資源中保留 128 MiB 的記憶體。同時，其還允許容器在需要時使用更多的記憶體資源。

> **Note**
>
> Windows 容器不支援此參數。

Docker 20.10.0 或更新版本的常駐程式會為容器保留最低 6 MiB 的記憶體。因此，不要為容器指定少於 6 MiB 的記憶體。

Docker 19.03.13-ce 或更舊版本的常駐程式會為容器保留最低 4 MiB 的記憶體。因此，不要為容器指定少於 4 MiB 的記憶體。

> **Note**
>
> 若您嘗試盡可能為特定執行個體類型的任務提供最多的記憶體，以將資源使用率最大化，請參閱「保留 Amazon ECS Linux 容器執行個體記憶體」。

連接埠映射

`portMappings`

類型：物件陣列

必要：否

連接埠映射可讓容器存取主機容器執行個體上的連接埠，以傳送或接收流量。

針對使用 `awsvpc` 網路模式的任務定義，只指定 `containerPort`。`hostPort` 可維持空白，或其值必須與 `containerPort` 相同。

Windows 上的連接埠映射會使用 NetNAT 閘道地址，而非 `localhost`。Windows 沒有連接埠映射回送，所以您無法從主機本身存取容器的映射連接埠。

此參數的大部分欄位 (containerPort、hostPort、protocol) 會映射到 Docker Remote API 的建立容器區段中的 PortBindings 以及 docker run 的 --publish 選項。若任務定義的網路模式設定為 host，則主機連接埠必須為未定義，或符合連接埠映射中的容器連接埠。

> **Note**
>
> 任務達到 RUNNING 狀態後，手動和自動主機及容器連接埠指派會顯示在下列位置：
>
> - 主控台：選取任務的容器說明的 Network Bindings (網路繫結) 區段。
> - AWS CLI：describe-tasks 命令輸出的 networkBindings 區段。
> - API：DescribeTasks 的回應。
> - 中繼資料：任務中繼資料端點。

`appProtocol`

類型：字串

必要：否

用於連接埠映射的應用程式通訊協定。此參數僅適用於 Service Connect。建議您將此參數設定為與您應用程式使用的通訊協定一致。如果您設定此參數，Amazon ECS 會將通訊協定專屬連線處理新增至 Service Connect Proxy。如果您設定此參數，Amazon ECS 會在 Amazon ECS 主控台和中新增通訊協定特定遙測。 CloudWatch

如果您沒有為此參數設定值，則會使用 TCP。不過，Amazon ECS 不會針對 TCP 新增通訊協定特定遙測。

如需詳細資訊，請參閱 the section called “ Service Connect”。

有效的通訊協定值："HTTP" | "HTTP2" | "GRPC"

`containerPort`

類型：整數

必要：是 (當使用 portMappings 時)

容器上的連接埠號碼，該號碼繫結到使用者指定或自動指派的主機連接埠。

如果在使用 Fargate 啟動類型的任務中使用容器，則必須使用 containerPort 來指定公開連接埠。

對於 Fargate 上的 Windows 容器，您不得將連接埠 3150 用於 containerPort。這是因為其已保留。

假設您在 EC2 啟動類型的任務中使用容器，並且指定容器連接埠而非主機連接埠。然後，您的容器會自動在暫時性連接埠範圍中接收到主機連接埠。如需詳細資訊，請參閱 hostPort。使用此方式自動指派的連接埠映射不包含在容器執行個體 100 個保留連接埠的配限制中。

containerPortRange

類型：字串

必要：否

綁定到動態映射主機連接埠範圍之容器上的連接埠號碼範圍。

您只能使用 register-task-definition API 來設定此參數。此選項可在 portMappings 參數中使用。如需詳細資訊，請參閱在 參考資料AWS Command Line Interface 中的 register-task-definition。

指定 containerPortRange 時，以下規則適用：

- 您必須使用 bridge 網路模式或 awsvpc 網路模式。
- 此參數適用於 EC2 和 AWS Fargate 啟動類型。
- 此參數適用於 Linux 和 Windows 作業系統。
- 容器執行個體必須至少具有 1.67.0 版的容器代理程式以及至少 1.67.0-1 版的 ecs-init 套件。
- 每個容器最多可以指定 100 個連接埠範圍。
- 您不會指定 hostPortRange。hostPortRange 的值設定如下：
  - 對於具有 awsvpc 網路模式的任務中的容器，hostPort 會設定為與 containerPort 相同的值。這是靜態映射策略。
  - 對於具有 bridge 網路模式的任務中的容器，Amazon ECS 代理程式會從預設暫時性範圍尋找開放的主機連接埠並傳送到 Docker，以將主機連接埠綁定到容器連接埠。
- containerPortRange 有效值介於 1 到 65535。
- 一個連接埠只能包含在每個容器的一個連接埠映射中。
- 您無法指定重疊的連接埠範圍。
- 範圍中的第一個連接埠必須小於範圍中的最後一個連接埠。
- Docker 建議您在擁有大量連接埠時關閉 Docker 常駐程式組態檔案中的 Docker 代理。

如需詳細資訊，請參閱上的問題 #11185 GitHub。

如需有關如何在 Docker 常駐程式組態檔案中關閉 Docker 代理的詳細資訊，請參閱《Amazon ECS 開發人員指南》中的 Docker 常駐程式。

您可以呼叫 `DescribeTasks` 來檢視 `hostPortRange`，此即綁定到容器連接埠的主機連接埠。

傳送到 EventBridge的 Amazon ECS 任務事件中不包含連接埠範圍。如需詳細資訊，請參閱 the section called “使用以下方式自動回應 Amazon ECS 錯誤 EventBridge”。

`hostPortRange`

類型：字串

必要：否

與網路繫結搭配使用之主機上的連接埠號碼範圍。這由 Docker 指派，並由 Amazon ECS 代理程式交付。

`hostPort`

類型：整數

必要：否

要為您的容器保留之容器執行個體上的連接埠號碼。

如果在使用 Fargate 啟動類型的任務中使用容器，`hostPort` 可維持空白，或其值與 `containerPort` 相同。

假設您在 EC2 啟動類型的任務中使用容器。您可以為容器連接埠對應指定非保留的主機連接埠。這稱為靜態主機連接埠對應。或者，您可以在指定 `containerPort` 時省略 `hostPort` (或將其設定為 `0`)。您的容器會針對您的容器執行個體作業系統和 Docker 版本，自動接收暫時性連接埠範圍內的連接埠。這稱為動態主機連接埠對應。

Docker 1.6.0 版和更新版本的預設暫時性連接埠範圍列在 `/proc/sys/net/ipv4/ip_local_port_range` 下的執行個體上。如果此核心參數不可用，會使用預設的 `49153-65535` 暫時性連接埠範圍。請勿嘗試在暫時性連接埠範圍中指定主機連接埠。這是因為其已保留以便自動指派。一般而言，低於 `32768` 的連接埠便會位於暫時性連接埠範圍之外。

SSH 的預設保留連接埠為 22，Docker 連接埠為 2375 和 2376，Amazon ECS 容器代理程式連接埠則為 `51678-51680`。任何使用者先前為執行中任務指定的主機連接埠也會在執行任務

時保留。在任務停止後，便會釋放主機連接埠。目前預留的連接埠會顯示在 describe-container-instances 輸出的 `remainingResources` 中。容器執行個體一次最多可預留 100 個連接埠，包括預設的預留連接埠。自動指派的連接埠不計入 100 個預留連接埠的限制。

`name`

類型：字串

必要：否，需要在服務中設定 Service Connect

用於連接埠映射的名稱。此參數僅適用於 Service Connect。此參數是您在服務的 Service Connect 組態中使用的名稱。

如需詳細資訊，請參閱 使用服務 Connect 以簡短名稱連接 Amazon ECS 服務。

下列範例中使用了 Service Connect 的兩個必要欄位。

```
"portMappings": [
    {
        "name": string,
        "containerPort": integer
    }
]
```

`protocol`

類型：字串

必要：否

用於連接埠映射的協定。有效值為 `tcp` 和 `udp`。預設值為 `tcp`。

> **Important**
>
> Service Connect 僅支援 `tcp`。請記住，如果未設定此欄位，則會隱含 `tcp`。

> **Important**
>
> 只有在使用 1.2.0 版或更新版本的 Amazon ECS 容器代理程式 (例如 `amzn-ami-2015.03.c-amazon-ecs-optimized` AMI)，或是使用已更新到 1.3.0 版或更

新版本的容器代理程式而啟動的容器執行個體上，才能使用 UDP 支援。若要將您的容器代理更新到最新版本，請參閱「更新 Amazon ECS 容器代理程式」。

若您指定主機連接埠，請使用以下語法。

```
"portMappings": [
    {
        "containerPort": integer,
        "hostPort": integer
    }
    ...
]
```

若您希望取得自動指派的主機連接埠，請使用以下語法。

```
"portMappings": [
    {
        "containerPort": integer
    }
    ...
]
```

私有儲存庫憑證

`repositoryCredentials`

類型：RepositoryCredentials 物件

必要：否

私有登錄檔身分驗證的儲存庫登入資料。

如需詳細資訊，請參閱 在 Amazon ECS 中使用非AWS 容器映像。

`credentialsParameter`

類型：字串

必要：是 (當使用 `repositoryCredentials` 時)

包含私有儲存庫憑證密碼的 Amazon Resource Name (ARN)。

如需詳細資訊，請參閱 在 Amazon ECS 中使用非AWS 容器映像。

> **Note**
>
> 當您使用 Amazon ECS API 或 AWS 軟體開發套件時，如果密碼與您正在啟動的任務位於相同的區域中，則您可以使用完整 ARN 或機密名稱。 AWS CLI使用時 AWS Management Console，您必須指定密碼的完整 ARN。

以下是任務定義的程式碼片段，其會顯示所需的參數：

```
"containerDefinitions": [
    {
        "image": "private-repo/private-image",
        "repositoryCredentials": {
            "credentialsParameter":
 "arn:aws:secretsmanager:region:aws_account_id:secret:secret_name"
        }
    }
]
```

進階容器定義參數

以下進階容器定義參數會為用於在 Amazon ECS 容器執行個體上啟動容器的 docker run 命令提供延伸功能。

主題

- 運作狀態檢查
- 環境
- Network settings (網路設定)
- 儲存與記錄
- 安全
- 資源限制

- Docker 標籤

運作狀態檢查

`healthCheck`

用於容器的容器運作狀態檢查命令和相關的設定參數。如需詳細資訊，請參閱 使用容器運作狀態檢查判斷 Amazon ECS 任務運作狀態。

`command`

表示容器執行的命令的字串陣列，用於確定運作狀態是否良好。此字串陣列的開頭可以是 CMD，如此能直接執行命令引數；或是 CMD-SHELL，藉以使用容器預設的 shell 來執行命令。如果均尚未指定，則使用 CMD。

在中註冊工作定義時 AWS Management Console，請使用逗號分隔的指令清單。這些命令會在建立任務定義後轉換為字串。以下為運作狀態檢查的範例輸入。

```
CMD-SHELL, curl -f http://localhost/ || exit 1
```

使用 AWS Management Console JSON 面板、或 API 註冊任務定義時 AWS CLI，請將命令清單括在括弧中。以下為運作狀態檢查的範例輸入。

```
[ "CMD-SHELL", "curl -f http://localhost/ || exit 1" ]
```

結束代碼 0 (沒有 `stderr` 輸出) 表示成功，而任何非零的結束代碼則表示失敗。如需詳細資訊，請參閱 Docker Remote API 建立容器區段中的 `HealthCheck`。

`interval`

每個運作狀態檢查的時間間隔 (以秒為單位)。您可以指定 5 至 300 秒之間的值。預設 值為 30 秒。

`timeout`

判定為失敗之前，等待運作狀態檢查成功執行的時間 (以秒為單位)。您可以指定 2 至 60 秒之間的值。預設值為 5 秒。

`retries`

容器運作狀態判定為不良之前，重試失敗的運作狀態檢查的次數上限。您可以指定 1 至 10 次嘗試。預設值為重試三次。

`startPeriod`

選用的寬限期，讓容器有時間引導，不將失敗的運作狀態檢查計入重試次數上限。您可以指定 0 至 300 秒之間的值。預設情況下，`startPeriod` 是停用的。

環境

`cpu`

類型：整數

必要：否

Amazon ECS 容器代理程式保留給容器的 `cpu` 數量。在 Linux 上，此參數會映射到 Docker Remote API 的建立容器區段中的 `CpuShares` 以及 docker run 的 `--cpu-shares` 選項。

針對使用 Fargate 啟動類型的任務，此欄位為選用。為任務中所有容器預訂的 CPU 總量必須低於任務階層的 `cpu` 值。

> **Note**
>
> 您可以決定每個 Amazon EC2 執行個體類型可用的 CPU 單位數。若要執行此操作，請將 Amazon EC2 執行個體詳細資訊頁面上為執行個體類型列出的 vCPU 數乘上 1,024。

Linux 容器會按照與其配置數量相同的比例，與容器執行個體上的其他容器共用未配置的 CPU 單位。例如，若您在單一核心執行個體類型上執行單一容器任務，為該容器指定 512 個 CPU 單位。此外，該任務是在容器執行個體上運行的唯一任務。在此範例中，該容器便可在任何指定的時間內使用完整 1,024 個 CPU 單位。但是，假設您在該容器執行個體上啟動相同任務的另一個複本。每個任務都可以保證在需要的時候取得最少 512 個 CPU 單位。同樣，如果其他容器沒有使用剩餘的 CPU，每個容器都可以浮動到更高的 CPU 用量。但若兩個任務都一直維持在 100% 作用中的狀態，兩者能夠取得的單位數便會限制在 512 個 CPU 單位。

在 Linux 容器執行個體上，容器執行個體的 Docker 常駐程式會使用 CPU 值計算執行中容器的相對 CPU 共用比例。如需詳細資訊，請參閱 Docker 文件中的 CPU share constraint。Linux 核心允許的最低有效 CPU 共用值為 2。但是，CPU 參數並非必要項目，且您可以在您的容器定義中使用低於 2 的 CPU 值。針對低於 2 的 CPU 值 (包含 null)，行為會隨您的 Amazon ECS 容器代理程式版本而異：

- 代理程式版本 <= 1.1.0：Null 和零 CPU 值會以 0 傳遞給 Docker。然後 Docker 將此值轉換成 1,024 個 CPU 共享。CPU 值若為 1，則會以 1 傳遞給 Docker，接著 Linux 核心便會轉換成兩個 CPU 共享。
- 代理程式版本 >= 1.2.0：Null、零和值為一的 CPU 值會以兩個 CPU 共享傳遞給 Docker。

在 Windows 容器執行個體上，CPU 配額會強制為絕對配額。Windows 容器只能存取任務定義中定義的指定 CPU 數量。Null 或零的 CPU 值會以 0 形式傳遞給 Docker。然後 Windows 將此值解讀為一個 CPU 的 1%。

如需其他範例，請參閱 Amazon ECS 如何管理 CPU 和記憶體資源。

`gpu`

類型：ResourceRequirement 物件

必要：否

Amazon ECS 容器代理程式保留給容器的實體 GPUs 數量。為任務中所有容器保留的 GPU 數量不得超過任務啟動所在之容器執行個體上可用的 GPU 數量。如需詳細資訊，請參閱 GPU 工作負載的 Amazon ECS 任務定義。

> **Note**
>
> Windows 容器或在 Fargate 上託管的容器不支援此參數。

`Elastic Inference accelerator`

類型：ResourceRequirement 物件

必要：否

對於 `InferenceAccelerator` 類型，`value` 符合任務定義中指定之 `InferenceAccelerator` 的 `deviceName`。如需詳細資訊，請參閱 the section called “Elastic Inference 加速器名稱”。

> **Note**
>
> 自 2023 年 4 月 15 日起，不 AWS 會將新客戶加入 Amazon Elastic Inference (EI)，並協助目前客戶將工作負載遷移到提供更優惠價格和效能的選項。2023 年 4 月 15 日之後，新客

戶將無法在 Amazon、Amazon ECS 或 Amazon Amazon EC2 使用亞馬遜 SageMaker EI 加速器啟動執行個體。但是，在過去 30 天內至少使用過一次 Amazon EI 的客戶將被視為目前客戶，並且可以繼續使用該服務。

> **Note**
>
> Windows 容器或在 Fargate 上託管的容器不支援此參數。

`essential`

類型：布林值

必要：否

假設容器的 `essential` 參數標記為 `true`，並且該容器因為任何理由失敗或停止。然後，會停止屬於該任務一部分的所有其他容器。若容器的 `essential` 參數標記為 `false`，則其失敗將不會影響任務中其餘的容器。若省略此參數，則容器會假設為「基本」。

所有任務都必須至少有一個基本容器。假設您有一個由多個容器組成的應用程式。然後，將用於一般用途的容器分組為元件，並把不同的元件分到多個任務定義。如需詳細資訊，請參閱 架構您的 Amazon ECS 應用程式。

```
"essential": true|false
```

`entryPoint`

> **Important**
>
> 舊版的 Amazon ECS 容器代理程式無法正確處理 `entryPoint` 參數。若您在使用 `entryPoint` 時發生任何問題，請更新您的容器代理，或改將您的命令和引數作為 `command` 陣列項目輸入。

類型：字串陣列

必要：否

傳遞至容器的進入點。此參數會映射到 Docker Remote API 的建立容器區段中的 Entrypoint 以及 docker run 的 --entrypoint 選項。如需 Docker ENTRYPOINT 參數的詳細資訊，請參閱 https://docs.docker.com/engine/reference/builder/#entrypoint。

```
"entryPoint": ["string", ...]
```

command

類型：字串陣列

必要：否

傳遞至容器的命令。此參數會映射至 Docker Remote API 的建立容器區段中的 Cmd 以及 docker run 的 COMMAND 參數。如需 Docker CMD 參數的詳細資訊，請參閱 https://docs.docker.com/engine/reference/builder/#cmd。如果有多個引數，確保每個引數在陣列中是分開的字串。

```
"command": ["string", ...]
```

workingDirectory

類型：字串

必要：否

容器內要執行命令的工作目錄。此參數會映射到 Docker Remote API 的建立容器區段中的 WorkingDir 以及 docker run 的 --workdir 選項。

```
"workingDirectory": "string"
```

environmentFiles

類型：物件陣列

必要：否

內含要傳遞至容器之環境變數的檔案清單。此參數會映射到 docker run 的 --env-file 選項。

這不適用於 Fargate 上的Windows容器和 Windows 容器

您最多可以指定 10 個環境檔案。檔案副檔名必須是 .env。環境檔案中的每一行都包含 VARIABLE=VALUE 格式的環境變數。以 # 開頭的行會被視為註解，而忽略。如需適合的環境變數檔案語法的詳細資訊，請參閱《在檔案中宣告預設環境變數》。

如果在容器定義中指定了個別環境變數，它們的優先順序高於環境檔案中包含的變數。如果指定了內含相同變數的多個環境檔案，則處理順序為由上而下。建議您使用唯一的變數名稱。如需詳細資訊，請參閱 將個別環境變數傳遞至 Amazon ECS 容器。

`value`

類型：字串

必要：是

包含環境變數檔案的 Amazon S3 物件的 Amazon Resource Name (ARN)。

`type`

類型：字串

必要：是

要使用的檔案類型。唯一支援的值為 `s3`。

`environment`

類型：物件陣列

必要：否

傳遞至容器的環境變數。此參數會映射到 Docker Remote API 的建立容器區段中的 `Env` 以及 docker run 的 `--env` 選項。

> **Important**
>
> 不建議在敏感資訊 (例如登入資料) 使用純文字環境變數。

`name`

類型：字串

必要：是 (當使用 `environment` 時)

環境變數的名稱。

`value`

類型：字串

必要：是 (當使用 environment 時)

環境變數的值。

```
"environment" : [
    { "name" : "string", "value" : "string" },
    { "name" : "string", "value" : "string" }
]
```

secrets

類型：物件陣列

必要：否

代表公開到容器之秘密的物件。如需詳細資訊，請參閱 將敏感資料傳遞至 Amazon ECS 容器。

name

類型：字串

必要：是

要設定為容器上環境變數的值。

valueFrom

類型：字串

必要：是

公開給容器的秘密。支援的值可以是 AWS Secrets Manager 秘密的完整 Amazon 資源名稱 (ARN)，或參數存放區中參數的完整 ARN。 AWS Systems Manager

> **Note**
>
> 如果 Systems Manager 參數存放區參數或 Secrets Manager 參數存在於與您正在啟動的工 AWS 區域 作相同，您可以使用完整的 ARN 或密碼名稱。如果參數存在於不同區域，則必須指定完整 ARN。

```
"secrets": [
    {
```

```
        "name": "environment_variable_name",
        "valueFrom": "arn:aws:ssm:region:aws_account_id:parameter/parameter_name"
    }
]
```

Network settings (網路設定)

`disableNetworking`

類型：布林值

必要：否

當此參數為 true 時，聯網便會在容器中關閉。此參數會映射到 Docker Remote API 建立容器區段中的 `NetworkDisabled`。

> **Note**
>
> Windows 容器或使用 `awsvpc` 網路模式的任務不支援此參數。

預設值為 `false`。

```
"disableNetworking": true|false
```

`links`

類型：字串陣列

必要：否

`link` 參數可讓容器彼此通訊，而無需連接埠映射。只有在任務定義的網路模式設定為 `bridge` 時才支援此參數。`name:internalName` 建構模組與 Docker 連結中的 `name:alias` 類似。可以包含最多可達 255 個字元 (大小寫)、數字、連字號和底線。如需連結 Docker 容器的詳細資訊，請前往 https://docs.docker.com/engine/userguide/networking/default_network/dockerlinks/。此參數會映射到 Docker Remote API 的建立容器區段中的 `Links` 以及 docker run 的 `--link` 選項。

> **Note**
>
> Windows 容器或使用 `awsvpc` 網路模式的任務不支援此參數。

> **Important**
>
> 在相同容器執行個體上配置的容器可和彼此通訊，而無需連結或主機連接埠映射。容器執行個體上的網路隔離是由安全群組和 VPC 設定所控制。

```
"links": ["name:internalName", ...]
```

`hostname`

類型：字串

必要：否

您容器要使用的主機名稱。此參數會映射到 Docker Remote API 的建立容器區段中的 `Hostname` 以及 docker run 的 `--hostname` 選項。

> **Note**
>
> 如果您使用 `awsvpc` 網路模式，則不支援 `hostname` 參數。

```
"hostname": "string"
```

`dnsServers`

類型：字串陣列

必要：否

提供給容器的 DNS 伺服器清單。此參數會映射到 Docker Remote API 的建立容器區段中的 `Dns` 以及 docker run 的 `--dns` 選項。

> **Note**
>
> Windows 容器或使用 `awsvpc` 網路模式的任務不支援此參數。

```
"dnsServers": ["string", ...]
```

`dnsSearchDomains`

類型：字串陣列

必要：否

模式：^[a-zA-Z0-9-.]{0,253}[a-zA-Z0-9]$

提供給容器的 DNS 搜尋網域清單。此參數會映射到 Docker Remote API 的建立容器區段中的 `DnsSearch` 以及 docker run 的 `--dns-search` 選項。

> **Note**
>
> Windows 容器或使用 `awsvpc` 網路模式的任務不支援此參數。

```
"dnsSearchDomains": ["string", ...]
```

`extraHosts`

類型：物件陣列

必要：否

要附加到容器上 `/etc/hosts` 檔案的主機名稱和 IP 地址映射清單。

此參數會映射到 Docker Remote API 的建立容器區段中的 `ExtraHosts` 以及 docker run 的 `--add-host` 選項。

> **Note**
>
> Windows 容器或使用 `awsvpc` 網路模式的任務不支援此參數。

```
"extraHosts": [
      {
        "hostname": "string",
        "ipAddress": "string"
      }
      ...
```

```
    ]
```

`hostname`

類型：字串

必要：是 (當使用 `extraHosts` 時)

要在 `/etc/hosts` 項目中使用的主機名稱。

`ipAddress`

類型：字串

必要：是 (當使用 `extraHosts` 時)

要在 `/etc/hosts` 項目中使用的 IP 地址。

儲存與記錄

`readonlyRootFilesystem`

類型：布林值

必要：否

此參數為 true 時，容器會取得根檔案系統的唯讀存取權。此參數會映射到 Docker Remote API 的建立容器區段中的 `ReadonlyRootfs` 以及 docker run 的 `--read-only` 選項。

> **Note**
> Windows 容器不支援此參數。

預設值為 `false`。

```
"readonlyRootFilesystem": true|false
```

`mountPoints`

類型：物件陣列

必要：否

容器中資料磁碟區的掛接點。此參數會映射到 Docker Remote API 的建立容器區段中的 Volumes 以及 docker run 的 --volume 選項。

Windows 容器可在 $env:ProgramData 所在的相同磁碟上掛載整個目錄。Windows 容器無法在不同的磁碟機上掛載目錄，而且無法跨磁碟機使用掛載點。您必須指定掛接點，才能將 Amazon EBS 磁碟區直接連接至 Amazon ECS 任務。

sourceVolume

類型：字串

必要：是 (當使用 mountPoints 時)

要掛載的磁碟區名稱。

containerPath

類型：字串

必要：是 (當使用 mountPoints 時)

要掛接磁碟區之容器中的路徑。

readOnly

類型：布林值

必要：否

如果此數值為 true，容器擁有磁碟區的唯讀存取權。如果此值為 false，則容器可寫入磁碟區。預設值為 false。

volumesFrom

類型：物件陣列

必要：否

要從其他容器掛載的資料磁碟區。此參數會映射到 Docker Remote API 的建立容器區段中的 VolumesFrom 以及 docker run 的 --volumes-from 選項。

sourceContainer

類型：字串

必要：是 (當使用 volumesFrom 時)

要從其中掛載磁碟區的容器名稱。

`readOnly`

類型：布林值

必要：否

如果此數值為 `true`，容器擁有磁碟區的唯讀存取權。如果此值為 `false`，則容器可寫入磁碟區。預設值為 `false`。

```
"volumesFrom": [
                {
                  "sourceContainer": "string",
                  "readOnly": true|false
                }
              ]
```

`logConfiguration`

類型：LogConfiguration物件

必要：否

容器的日誌組態規格。

例如，使用日誌組態的任務定義，請參閱Amazon ECS 任務定義範例。

此參數會映射到 Docker Remote API 的建立容器區段中的 `LogConfig` 以及 `docker run` 的 `--log-driver` 選項。根據預設，容器會和 Docker 常駐程式使用一樣的日誌記錄驅動程式。不過，容器可以在容器定義中使用此參數指定日誌驅動程式，和 Docker 常駐程式使用不同的日誌驅動程式。若要讓容器使用不同的日誌驅動程式，必須在容器執行個體上適當設定日誌系統 (或在不同的日誌伺服器上使用遠端記錄選項)。如需支援的不同日誌驅動程式選項的詳細資訊，請參閱 Docker 文件中的 Configure logging drivers (設定日誌驅動程式)。

指定容器的日誌組態時，請考量下列事項：

- Amazon ECS 支援 Docker 常駐程式可用的日誌驅動程式子集。未來的 Amazon ECS 容器代理程式版本可能會提供更多可用的其他日誌驅動程式。
- 在您的容器執行個體上，此參數需要 1.18 版或更新版本的 Docker Remote API。

- 對於使用 EC2 啟動類型的任務，在容器執行個體上執行的 Amazon ECS 容器代理程式，必須使用 ECS_AVAILABLE_LOGGING_DRIVERS 環境變數註冊在該執行個體上可用的日誌驅動程式，才能讓放置在該執行個體上的容器使用這些日誌組態選項。如需詳細資訊，請參閱 Amazon ECS 容器代理程式組態。
- 對於使用 Fargate 啟動類型的工作，您必須在工作之外安裝任何其他軟體。例如，Fluentd 輸出彙整工具或要傳送 Gelf 日誌執行 Logstash 的遠端主機。

```
"logConfiguration": {
      "logDriver": "awslogs","fluentd","gelf","json-
file","journald","logentries","splunk","syslog","awsfirelens",
      "options": {"string": "string"
        ...},
 "secretOptions": [{
  "name": "string",
  "valueFrom": "string"
 }]
}
```

logDriver

類型：字串

有效值："awslogs","fluentd","gelf","json-file","journald","logentries","splunk","syslog","awsfirelens"

必要：是 (當使用 logConfiguration 時)

容器要使用的日誌驅動程式。根據預設，先前列出的有效值是 Amazon ECS 容器代理程式可用來通訊的日誌驅動程式。

針對使用 Fargate 啟動類型的任務，支援的日誌驅動程式為 awslogs、splunk 和 awsfirelens。

針對使用 EC2 啟動類型的任務，支援的日誌驅動程式為 awslogs、fluentd、gelf、json-file、journald、logentries、syslog、splunk 和 awsfirelens。

如需如何在工作定義中使用記awslogs錄驅動程式將容器記錄檔傳送至 CloudWatch 記錄的詳細資訊，請參閱將 Amazon ECS 日誌傳送到 CloudWatch 。

如需使用 awsfirelens 日誌驅動程式的詳細資訊，請參閱自訂日誌路由。

> **Note**
>
> 如果您有未列出的自訂驅動程式，您可以分叉提供的 Amazon ECS 容器代理程式專案， GitHub並對其進行自訂以搭配該驅動程式使用。我們鼓勵您為想要進行的變更提交提取請求。但是，我們目前不支援執行此軟體經修改的複本。

在您的容器執行個體上，此參數需要 1.18 版或更新版本的 Docker Remote API。

`options`

類型：字串到字串映射

必要：否

要傳送到日誌驅動程式的索引鍵/值映射組態選項。

當您使用 FireLens 將記錄路由傳送至記錄檔儲存和分析的 AWS Partner Network 目的地 AWS 服務 或目的地時，您可以設定`log-driver-buffer-limit`選項，以限制記憶體中緩衝的事件數目，然後再傳送至記錄路由器容器。由於高輸送量可能會導致 Docker 內部緩衝區的記憶體不足，因此這可協助解決潛在的日誌受損問題。如需詳細資訊，請參閱 the section called “設定高輸送量的記錄”。

在您的容器執行個體上，此參數需要 1.19 版或更新版本的 Docker Remote API。

`secretOptions`

類型：物件陣列

必要：否

此物件代表要傳送至日誌組態的秘密。日誌組態中使用的秘密可能包括身分驗證權杖、憑證或加密金鑰。如需詳細資訊，請參閱 將敏感資料傳遞至 Amazon ECS 容器。

`name`

類型：字串

必要：是

要設定為容器上環境變數的值。

`valueFrom`

類型：字串

必要：是

公開到容器日誌組態的秘密。

```
"logConfiguration": {
 "logDriver": "splunk",
 "options": {
  "splunk-url": "https://cloud.splunk.com:8080",
  "splunk-token": "...",
  "tag": "...",
  ...
 },
 "secretOptions": [{
  "name": "splunk-token",
  "valueFrom": "/ecs/logconfig/splunkcred"
 }]
}
```

`firelensConfiguration`

類型：FirelensConfiguration物件

必要：否

容 FireLens 器的組態。這是用來指定及設定容器日誌的日誌路由器。如需詳細資訊，請參閱 將 Amazon ECS 日誌傳送到 AWS 服務或 AWS Partner。

```
{
    "firelensConfiguration": {
        "type": "fluentd",
        "options": {
            "KeyName": ""
        }
    }
}
```

`options`

類型：字串到字串映射

必要：否

設定日誌路由器時要使用的索引鍵/值映射選項。此欄位是選用欄位，可用於指定自訂組態檔案，或者將額外的中繼資料 (例如任務、任務定義、叢集和容器執行個體詳細資訊)

新增到日誌事件。如果已指定，則要使用的語法為 "options":{"enable-ecs-log-metadata":"true|false","config-file-type:"s3|file","config-file-value":"arn:aws:s3:::mybucket/fluent.conf|filepath"}。如需詳細資訊，請參閱 Amazon ECS 任務定義範例：將日誌路由到 FireLens。

type

類型：字串

必要：是

要使用的日誌路由器。有效值為 fluentd 或 fluentbit。

安全

如需有關容器安全的詳細資訊，請參閱《Amazon ECS 最佳實務指南》中的任務和容器安全。

credentialSpecs

類型：字串陣列

必要：否

SSM 或 Amazon S3 中的 ARN 清單，用於為 Active Directory 身分驗證設定容器的憑證規格 (CredSpec) 檔案。建議您使用此參數而不是 dockerSecurityOptions. ARN 的上限為 1。

每個 ARN 有兩種格式。

credentialspecdomainless：MyARN

您使用 credentialspecdomainless:MyARN 為 Secrets Manager 中的秘密提供具其他區段的 CredSpec。您在秘密中提供網域的登入憑證。

在任何容器執行個體上執行的每項任務都可加入不同的網域。

您可以使用此格式而無需將容器執行個體加入網域。

credentialspec：MyARN

您使用 credentialspec:MyARN 為單一網域提供 CredSpec。

您必須先將容器執行個體加入網域，才能啟動此任務定義的任何任務。

在這兩種格式中，以 SSM 或 Amazon S3 中的 ARN 替換 MyARN。

credspec 必須在 Secrets Manager 中提供 ARN，以取得包含使用者名稱、密碼和要連線之網域的秘密。為了提高安全性，執行個體不會加入網域以進行無網域驗證。執行個體上的其他應用程式無法使用無網域憑證。您可以使用此參數在相同的執行個體上執行任務，即使是任務需要加入不同的網域。如需詳細資訊，請參閱在使用 Windows 容器的 gMSA 和使用 Linux 容器的 gMSA。

privileged

類型：布林值

必要：否

此參數為 true 時，容器便會取得主機容器執行個體的更高權限 (類似 root 使用者)。建議不要使用 privileged 執行容器。在大多數情況下，可以使用特定參數，而不是使用 privileged 來指定所需的確切權限。

此參數會映射到 Docker Remote API 的建立容器區段中的 Privileged 以及 docker run 的 --privileged 選項。

> **Note**
>
> Windows 容器或使用 Fargate 啟動類型的任務不支援此參數。

預設值為 false。

```
"privileged": true|false
```

user

類型：字串

必要：否

要在容器內使用的使用者。此參數會映射到 Docker Remote API 的建立容器區段中的 User 以及 docker run 的 --user 選項。

> **Important**
>
> 執行使用 host 網路模式的任務時，不要使用根使用者 (UID 0) 執行容器。作為最佳安全實務，請一律使用非根使用者。

您可以使用下列格式指定 `user`。如果指定 UID 或 GID，您必須使用正整數指定它。

- `user`
- `user:group`
- `uid`
- `uid:gid`
- `user:gid`
- `uid:group`

> **Note**
>
> Windows 容器不支援此參數。

```
"user": "string"
```

`dockerSecurityOptions`

類型：字串陣列

有效值：「沒有新的權限」|「設備：配置文件」|「標籤：*#*」|「憑證規格：」*CredentialSpecFilePath*

必要：否

提供多個安全系統自訂組態的字串清單。如需有效值的詳細資訊，請參閱 Docker 執行安全性組態。此欄位對使用 Fargate 啟動類型之任務中的容器無效。

對於 EC2 上的 Linux 任務，此參數可用於參考 SELinux 和 AppArmor 多層級安全系統的自訂標籤。

對於 EC2 上的任何任務，此參數可在設定 Active Directory 身分驗證的容器時，用於參考憑證規格檔案。如需詳細資訊，請參閱 了解如何將 GMA 用於適用於 Amazon ECS 的 EC2 視窗容器 及 用 gMSA於 Amazon ECS 上的 EC2 Linux 容器。

此參數會映射到 Docker Remote API 的建立容器區段中的 `SecurityOpt` 以及 docker 的 `--security-opt` 選項。

```
"dockerSecurityOptions": ["string", ...]
```

> **Note**
>
> 在容器執行個體上執行的 Amazon ECS 容器代理程式必須先使用 ECS_SELINUX_CAPABLE=true 或 ECS_APPARMOR_CAPABLE=true 環境變數註冊，才能讓放置在該執行個體上的容器使用這些安全選項。如需詳細資訊，請參閱 Amazon ECS 容器代理程式組態。

資源限制

`ulimits`

類型：物件陣列

必要：否

要為容器定義的 ulimit 值清單。此值覆寫作業系統的預設資源配額設定。此參數會映射到 Docker Remote API 的建立容器區段中的 Ulimits 以及 docker run 的 --ulimit 選項。

在 Fargate 上託管的 Amazon ECS 任務會使用作業系統設定的預設資源限制值，但是 nofile 資源限制參數除外。nofile 資源限制會對容器可使用的開放檔案數量設限。在 Fargate 上，預設的 nofile 軟限制為 1024，硬限制為 65535。您可以將兩個限制的值設定為 1048576。如需詳細資訊，請參閱任務資源限制。

在您的容器執行個體上，此參數需要 1.18 版或更新版本的 Docker Remote API。

> **Note**
>
> Windows 容器不支援此參數。

```
"ulimits": [
      {
        "name":
 "core"|"cpu"|"data"|"fsize"|"locks"|"memlock"|"msgqueue"|"nice"|"nofile"|"nproc"|"rss"|"rtp
        "softLimit": integer,
        "hardLimit": integer
      }
      ...
    ]
```

name

類型：字串

有效值："core" | "cpu" | "data" | "fsize" | "locks" | "memlock" | "msgqueue" | "nice" | "nofile" | "nproc" | "rss" | "rtprio" | "rttime" | "sigpending" | "stack"

必要：是 (當使用 ulimits 時)

ulimit 的 type。

hardLimit

類型：整數

必要：是 (當使用 ulimits 時)

ulimit 類型的硬性限制。

softLimit

類型：整數

必要：是 (當使用 ulimits 時)

ulimit 類型的軟性限制。

Docker 標籤

dockerLabels

類型：字串到字串映射

必要：否

要新增到容器的標籤索引鍵/值映射。此參數會映射到 Docker Remote API 的建立容器區段中的 Labels 以及 docker run 的 --label 選項。

在您的容器執行個體上，此參數需要 1.18 版或更新版本的 Docker Remote API。

```
"dockerLabels": {"string": "string"
      ...}
```

其他容器定義參數

在 Amazon ECS 主控台中使用 Configure via JSON (透過 JSON 進行設定) 選項註冊任務定義時，可以使用下列容器定義參數。如需詳細資訊，請參閱 使用主控台建立 Amazon ECS 任務定義。

主題

- Linux 參數
- 容器相依性
- 容器逾時
- 系統控制
- 互動性
- 虛擬終端機

Linux 參數

`linuxParameters`

類型：LinuxParameters 物件

必要：否

Linux套用至容器的特定選項，例如KernelCapabilities。

> **Note**
>
> Windows 容器不支援此參數。

```
"linuxParameters": {
      "capabilities": {
        "add": ["string", ...],
        "drop": ["string", ...]
        }
      }
```

`capabilities`

類型：KernelCapabilities 物件

必要：否

從 Docker 提供的預設組態中新增或卸除的容器 Linux 功能。如需預設功能和其他可用功能的詳細資訊，請參閱《Docker 執行參考》中的執行期存取和 Linux 功能。如需這些 Linux 功能的詳細資訊，請參閱 Linux 手冊頁面的功能 (7)。

add

類型：字串陣列

有效值："ALL" | "AUDIT_CONTROL" | "AUDIT_READ" | "AUDIT_WRITE" | "BLOCK_SUSPEND" | "CHOWN" | "DAC_OVERRIDE" | "DAC_READ_SEARCH" | "FOWNER" | "FSETID" | "IPC_LOCK" | "IPC_OWNER" | "KILL" | "LEASE" | "LINUX_IMMUTABLE" | "MAC_ADMIN" | "MAC_OVERRIDE" | "MKNOD" | "NET_ADMIN" | "NET_BIND_SERVICE" | "NET_BROADCAST" | "NET_RAW" | "SETFCAP" | "SETGID" | "SETPCAP" | "SETUID" | "SYS_ADMIN" | "SYS_BOOT" | "SYS_CHROOT" | "SYS_MODULE" | "SYS_NICE" | "SYS_PACCT" | "SYS_PTRACE" | "SYS_RAWIO" | "SYS_RESOURCE" | "SYS_TIME" | "SYS_TTY_CONFIG" | "SYSLOG" | "WAKE_ALARM"

必要：否

要新增至 Docker 提供之預設組態容器的 Linux 功能。此參數對應到 Docker Remote API Create a container (建立容器) 一節中的 CapAdd，以及 Docker run 的 `--cap-add` 選項。

> **Note**
>
> 在 Fargate 上啟動的任務僅支援新增 SYS_PTRACE 核心功能。

add

類型：字串陣列

有效值："SYS_PTRACE"

必要：否

要新增至 Docker 提供之預設組態容器的 Linux 功能。此參數會映射到 Docker Remote API 建立容器區段中的 CapAdd 以及 Docker run 的 `--cap-add` 選項。

`drop`

類型：字串陣列

有效值："ALL" | "AUDIT_CONTROL" | "AUDIT_WRITE" | "BLOCK_SUSPEND" | "CHOWN" | "DAC_OVERRIDE" | "DAC_READ_SEARCH" | "FOWNER" | "FSETID" | "IPC_LOCK" | "IPC_OWNER" | "KILL" | "LEASE" | "LINUX_IMMUTABLE" | "MAC_ADMIN" | "MAC_OVERRIDE" | "MKNOD" | "NET_ADMIN" | "NET_BIND_SERVICE" | "NET_BROADCAST" | "NET_RAW" | "SETFCAP" | "SETGID" | "SETPCAP" | "SETUID" | "SYS_ADMIN" | "SYS_BOOT" | "SYS_CHROOT" | "SYS_MODULE" | "SYS_NICE" | "SYS_PACCT" | "SYS_PTRACE" | "SYS_RAWIO" | "SYS_RESOURCE" | "SYS_TIME" | "SYS_TTY_CONFIG" | "SYSLOG" | "WAKE_ALARM"

必要：否

要從 Docker 提供之預設組態中移除容器的 Linux 功能。此參數會映射到 Docker Remote API 建立容器區段中的 `CapDrop` 以及 Docker run 的 `--cap-drop` 選項。

`devices`

任何要公開給容器的主機裝置。此參數會映射到 Docker Remote API 建立容器區段中的 `Devices` 以及 Docker run 的 `--device` 選項。

> **Note**
>
> 當您使用 Fargate 啟動類型或 Windows 容器時，不支援 `devices` 參數。

類型：Device 物件的陣列

必要：否

`hostPath`

主機容器執行個體上裝置的路徑。

類型：字串

必要：是

`containerPath`

容器內的路徑，其為公開主機設備的目標路徑。

類型：字串

必要：否

`permissions`

要提供給裝置容器的明確許可。在預設情況下，容器在裝置上具有 `read`、`write` 及 `mknod` 許可。

類型：字串陣列

有效值: `read` | `write` | `mknod`

`initProcessEnabled`

在容器內執行 `init` 處理序，該處理序可轉寄訊號及獲得處理序。此參數會映射到 docker run 的 `--init` 選項。

在您的容器執行個體上，此參數需要 1.25 版或更新版本的 Docker Remote API。

`maxSwap`

容器可以使用的交換記憶體總量 (以 MiB 為單位)。此參數將會轉換為 docker run 的 `--memory-swap` 選項，其中值是容器記憶體與 `maxSwap` 值的總和。

如果將 `maxSwap` 值指定為 `0`，容器不會使用交換。接受的值為 `0` 或任何正整數。如果省略 `maxSwap` 參數，容器使用其執行所在的容器執行個體的交換組態。必須設定 `maxSwap` 值，才能使用 `swappiness` 參數。

> **Note**
>
> 如果您使用的是使用 Fargate 啟動類型的任務，則不支援 `maxSwap` 參數。

`sharedMemorySize`

`/dev/shm` 磁碟區的大小值 (以 MiB 為單位)。此參數會映射到 docker run 的 `--shm-size` 選項。

> **Note**
>
> 如果您使用的是使用 Fargate 啟動類型的任務，則不支援 `sharedMemorySize` 參數。

類型：整數

`swappiness`

您可使用此參數，調整容器的記憶體交換行為。`swappiness` 的值若為 `0` 將導致交換不會發生 (除非有需要)。`swappiness` 的值若為 `100` 將導致頻繁交換頁面。接受的值為介於 `0` 與 `100` 之間的整數。如果您未指定值，則會使用預設值 `60`。此外，如果您未對 `maxSwap` 指定值，則會忽略此參數。此參數會映射到 docker run 的 `--memory-swappiness` 選項。

> **Note**
>
> 如果您使用的是使用 Fargate 啟動類型的任務，則不支援 `swappiness` 參數。
> 如果您在 Amazon Linux 2023 上使用任務，則不支援 `swappiness` 參數。

`tmpfs`

tmpfs 掛載的容器路徑、掛載選項和大小上限 (以 MiB 為單位)。此參數會映射到 docker run 的 `--tmpfs` 選項。

> **Note**
>
> 如果您使用的是使用 Fargate 啟動類型的任務，則不支援 `tmpfs` 參數。

類型：Tmpfs 物件的陣列

必要：否

`containerPath`

要掛載 tmpfs 磁碟區的絕對檔案路徑。

類型：字串

必要：是

`mountOptions`

tmpfs 磁碟區掛載選項的清單。

類型：字串陣列

必要：否

有效值:"defaults" | "ro" | "rw" | "suid" | "nosuid" | "dev" | "nodev" | "exec" | "noexec" | "sync" | "async" | "dirsync" | "remount" | "mand" | "nomand" | "atime" | "noatime" | "diratime" | "nodiratime" | "bind" | "rbind" | "unbindable" | "runbindable" | "private" | "rprivate" | "shared" | "rshared" | "slave" | "rslave" | "relatime" | "norelatime" | "strictatime" | "nostrictatime" | "mode" | "uid" | "gid" | "nr_inodes" | "nr_blocks" | "mpol"

`size`

tmpfs 磁碟區大小上限 (以 MiB 為單位)。

類型：整數

必要：是

容器相依性

`dependsOn`

類型：ContainerDependency 物件陣列

必要：否

針對容器啟動和關閉而定義的相依性。容器可包含多個相依性。針對容器啟動而定義相依性時，它會保留給容器關閉。如需範例，請參閱容器相依性。

> **Note**
>
> 如果容器不符合相依性限制或在符合限制之前逾時，Amazon ECS 不會讓相依容器進入下一個狀態。

對於在 Amazon EC2 執行個體上託管的 Amazon ECS 任務，執行個體至少需要容器代理程式的 `1.26.0` 版本，才能啟用容器相依性。不過，我們建議您使用最新版的容器代理程式。如需檢查代理程式版本及更新至最新版本的資訊，請參閱「更新 Amazon ECS 容器代理程式」。如果您使用的是 Amazon ECS 最佳化 Amazon Linux AMI，您的執行個體至少需要 `1.26.0-1` 版的 `ecs-init` 套件。如果您的容器執行個體是從 `20190301` 版或更新版本啟動，它們會包含所需的容器代理程式和 `ecs-init` 版本。如需詳細資訊，請參閱 Amazon ECS 優化的 Linux AMI。

對於在 Fargate 上託管的 Amazon ECS 任務，此參數要求任務或服務使用平台版本 `1.3.0` 或更新版本 (Linux) 或 `1.0.0` (Windows)。

```
"dependsOn": [
    {
        "containerName": "string",
        "condition": "string"
    }
]
```

`containerName`

類型：字串

必要：是

必須符合指定條件的容器名稱。

`condition`

類型：字串

必要：是

容器的相依性條件。以下是可用的條件及其行為：

- START - 此條件會模擬連結和磁碟區目前的行為。該條件可驗證相依容器先啟動後，才允許其他容器啟動。
- COMPLETE - 此條件驗證相依容器執行到完成 (結束) 後，才允許其他容器啟動。這適合用於只是執行指令碼，然後就退出的非必要容器。無法在基本容器上設定此條件。
- SUCCESS - 此條件與 COMPLETE 相同，但還要求容器必須以 `zero` 狀態結束。無法在基本容器上設定此條件。
- HEALTHY - 此條件會在驗證相依容器傳遞了其容器運作狀態檢查後，才允許其他容器啟動。這會要求相依容器在任務定義中設定運作狀態檢查。僅在任務啟動時才會確認這個條件。

容器逾時

`startTimeout`

類型：整數

必要：否

範例值：120

解析容器的相依性時在放棄之前等待的持續時間 (以秒為單位)。

例如，您在任務定義中指定兩個容器，其中的 containerA 對達到 COMPLETE、SUCCESS 或 HEALTHY 狀態的 containerB 有相依性。如果為 containerB 指定 startTimeout 值，但在該時間內沒有達到所需狀態，則 containerA 不會啟動。

> **Note**
>
> 如果容器不符合相依性限制或在符合限制之前逾時，Amazon ECS 不會讓相依容器進入下一個狀態。

對於在 Fargate 上託管的 Amazon ECS 任務，此參數要求任務或服務使用平台版本 1.3.0 或更新版本 (Linux)。最高值為 120 秒。

stopTimeout

類型：整數

必要：否

範例值：120

當容器本身未正常結束時，強制終止容器之前的等待期間 (以秒為單位)。

對於在 Fargate 上託管的 Amazon ECS 任務，此參數要求任務或服務使用平台版本 1.3.0 或更新版本 (Linux)。如果未指定參數，則會使用預設值 30 秒。最高值為 120 秒。

對於使用 EC2 啟動類型的任務，如果未指定 stopTimeout 參數，會使用 Amazon ECS 容器代理程式組態變數 ECS_CONTAINER_STOP_TIMEOUT 設定的值。如果未設定 stopTimeout 參數或 ECS_CONTAINER_STOP_TIMEOUT 代理程式組態變數，則對於 Linux 容器，預設值為 30 秒，對於 Windows 容器會使用預設值 30 秒。容器執行個體至少需要 1.26.0 版的容器代理程式，才能啟用容器停止逾時值。不過，我們建議您使用最新版的容器代理程式。如需如何檢查代理程式版本及更新至最新版本的資訊，請參閱「更新 Amazon ECS 容器代理程式」。如果您使用的是 Amazon ECS 最佳化 Amazon Linux AMI，您的執行個體至少需要 1.26.0-1 版的 ecs-init 套件。如果您的容器執行個體是從 20190301 版或更新版本啟動，它們會包含所需的容器代理程式和 ecs-init 版本。如需詳細資訊，請參閱 Amazon ECS 優化的 Linux AMI。

系統控制

`systemControls`

類型：SystemControl 物件

必要：否

要在容器中設定的命名空間核心參數清單。此參數會映射到 Docker Remote API 的建立容器區段中的 `Sysctls` 以及 docker run 的 `--sysctl` 選項。例如，您可以規劃 `net.ipv4.tcp_keepalive_time` 設定以維持壽命較長的連線。

不建議您指定單一任務中多個容器的網路相關 `systemControls` 參數，而單一任務也使用 `awsvpc` 或 `host` 網路模式。執行此操作的缺點如下：

- 對於使用包括 Fargate `awsvpc` 網路模式的任務，如果您設定任何容器的 `systemControls`，則會套用至任務中的所有容器。如果您針對單一任務中的多個容器設定不同的 `systemControls`，則最後啟動的容器會判斷哪些 `systemControls` 生效。
- 對於使用 `host` 網路模式的任務，不支援網路命名空間 `systemControls`。

如果您為任務中的容器設定要使用的 IPC 資源命名空間，以下各項條件套用到您的系統控制。如需詳細資訊，請參閱 IPC 模式。

- 針對使用 `host` IPC 模式的任務，不支援 IPC 命名空間 `systemControls`。
- 針對使用 `task` IPC 模式的任務，IPC 命名空間 `systemControls` 值套用到任務內的所有容器。

> **Note**
>
> Windows 容器不支援此參數。

> **Note**
>
> 如果任務使用平台版本 `1.4.0` 或更高版本 (Linux)，只有託管於 AWS Fargate 的任務才支援此參數。Fargate 上的 Windows 容器不支援此參數。

```
"systemControls": [
```

```
    {
         "namespace":"string",
         "value":"string"
    }
]
```

namespace

類型：字串

必要：否

要設定的命名空間核心參數。value

有效的 IPC 命名空間值："kernel.msgmax" | "kernel.msgmnb" | "kernel.msgmni" | "kernel.sem" | "kernel.shmall" | "kernel.shmmax" | "kernel.shmmni" | "kernel.shm_rmid_forced"，以及開頭為 "fs.mqueue.*" 的 Sysctls

有效的網路命名空間值：開頭為 "net.*" 的 Sysctls

Fargate 支援所有這些值。

value

類型：字串

必要：否

在中指定的命名空間核心參數值namespace。

互動性

interactive

類型：布林值

必要：否

此參數為 true 時，您可部署需要配置 stdin 或 tty 的容器化應用程式。此參數會映射到 Docker Remote API 的建立容器區段中的 OpenStdin 以及 docker run 的 --interactive 選項。

預設值為 false。

虛擬終端機

`pseudoTerminal`

類型：布林值

必要：否

當此參數為 `true` 時，會配置 TTY。此參數會映射到 Docker Remote API 的建立容器區段中的 `Tty` 以及 docker run 的 `--tty` 選項。

預設值為 `false`。

Elastic Inference 加速器名稱

> **Note**
>
> 自 2023 年 4 月 15 日起，不 AWS 會將新客戶加入 Amazon Elastic Inference (EI)，並協助目前客戶將工作負載遷移到提供更優惠價格和效能的選項。2023 年 4 月 15 日之後，新客戶將無法在 Amazon、Amazon ECS 或 Amazon Amazon EC2 使用亞馬遜 SageMaker EI 加速器啟動執行個體。但是，在過去 30 天內至少使用過一次 Amazon EI 的客戶將被視為目前客戶，並且可以繼續使用該服務。

任務定義的 Elastic Inference 加速器資源要求。如需詳細資訊，請參閱什麼是 Amazon Elastic Inference？ 在 Amazon Elastic Inference 開發人員指南中。

任務定義允許使用以下參數：

`deviceName`

類型：字串

必要：是

彈性推論加速器裝置名稱。`deviceName` 也必須在容器定義中參照，請參閱 `Elastic Inference accelerator`。

`deviceType`

類型：字串

必要：是

要使用的彈性推論加速器。

任務置放限制條件

當您註冊任務定義時，您可以提供任務置放限制，自訂 Amazon ECS 放置任務的方式。

若您使用的是 Fargate 啟動類型，則不支援任務置放限制條件。根據預設，Fargate 任務會分散在可用區域中。

對於使用 EC2 啟動類型的任務，您可以使用限制，根據可用區域、執行個體類型或自訂屬性來放置任務。如需詳細資訊，請參閱 定義 Amazon ECS 用於執行任務的容器執行個體。

容器定義允許使用以下參數：

`expression`

類型：字串

必要：否

限制所套用的叢集查詢語言運算式。如需詳細資訊，請參閱 建立運算式以定義 Amazon ECS 任務的容器執行個體。

`type`

類型：字串

必要：是

限制類型。使用 `memberOf`，以將選擇限制於特定群組或有效的待選項目。

代理組態

`proxyConfiguration`

類型：ProxyConfiguration 物件

必要：否

App Mesh 代理的組態詳細資訊。

對於使用 EC2 啟動類型的任務，容器執行個體需要至少 1.26.0 版的容器代理程式和至少 1.26.0-1 版的 `ecs-init` 套裝服務，才能啟用代理組態。如果您的容器執行個體是從 Amazon ECS 最佳化 AMI 20190301 版或更新版本啟動，則它們會包含所需的容器代理程式和 `ecs-init` 版本。如需詳細資訊，請參閱 Amazon ECS 優化的 Linux AMI。

對於使用 Fargate 啟動類型的任務，此功能要求任務或服務必須使用平台版本 1.3.0 或更新版本。

> **Note**
>
> Windows 容器不支援此參數。

```
"proxyConfiguration": {
    "type": "APPMESH",
    "containerName": "string",
    "properties": [
        {
           "name": "string",
           "value": "string"
        }
    ]
}
```

`type`

類型：字串

有效值：APPMESH

必要：否

代理類型。唯一支援的值為 APPMESH。

`containerName`

類型：字串

必要：是

做為 App Mesh 代理的容器名稱。

properties

類型：KeyValue對對象的數組

必要：否

一組網路組態參數，用於提供指定為鍵值組的容器網路介面 (CNI) 外掛程式。

- IgnoredUID - (必要) user 參數在容器定義中定義之代理容器的使用者 ID (UID)。這是用於確保代理忽略自己的流量。如果指定了 IgnoredGID，這個欄位可以為空白。
- IgnoredGID - (必要) user 參數在容器定義中定義之代理容器的群組 ID (GID)。這是用於確保代理忽略自己的流量。如果指定了 IgnoredUID，這個欄位可以為空白。
- AppPorts - (必要) 應用程式所用連接埠的清單。這些連接埠的網路流量會轉發到 ProxyIngressPort 和 ProxyEgressPort。
- ProxyIngressPort - (必要) 指定傳至 AppPorts 的傳入流量所導向的連接埠。
- ProxyEgressPort - (必要) 指定來自 AppPorts 的傳出流量所導向的連接埠。
- EgressIgnoredPorts - (必要) 進入這些指定連接埠的傳出流量將忽略，不會重新引導至 ProxyEgressPort。它可以是空的清單。
- EgressIgnoredIPs - (必要) 進入這些指定 IP 位址的傳出流量將忽略，不會重新引導至 ProxyEgressPort。它可以是空的清單。

name

類型：字串

必要：否

鍵/值對的名稱。

value

類型：字串

必要：否

索引鍵/值對的值。

磁碟區

當您註冊工作定義時，您可以選擇性地指定要傳遞至容器執行個體上Docker精靈的磁碟區清單，然後可供相同容器執行個體上的其他容器存取。

以下為會用到的資料磁碟區類型：

- Amazon EBS 磁碟區 — 為資料密集型容器化工作負載提供符合成本效益、耐用且高效能的區塊儲存。執行獨立任務或建立或更新服務時，您可以為每個 Amazon ECS 任務附加 1 個 Amazon EBS 磁碟區。對於在 Fargate 或亞馬遜 EC2 執行個體上託管的 Linux 任務，支援 Amazon EBS 磁碟區。如需詳細資訊，請參閱 使用 Amazon EBS 卷與 Amazon ECS。
- Amazon EFS 磁碟區 — 提供簡單、可擴展且持久的檔案儲存，可與 Amazon ECS 任務搭配使用。利用 Amazon EFS，儲存容量即可有彈性。儲存容量會隨著您新增和移除檔案時自動擴展和縮減。應用程式可在需要時具備所需的儲存容量。在 Fargate 或 Amazon EC2 執行個體上託管的任務支援 Amazon EFS 磁碟區。如需詳細資訊，請參閱 搭配 Amazon ECS 使用 Amazon EFS 磁碟區。
- FSx for Windows File Server 磁碟區 — 提供全受管 Microsoft Windows 檔案伺服器。這些檔案伺服器由 Windows 檔案系統支援。搭配使用 FSx for Windows File Server 與 Amazon ECS 時，您可以使用持續、分散、共用、靜態的檔案儲存來佈建 Windows 任務。如需詳細資訊，請參閱 將 FSx 用於 FSx for Windows File Server 磁碟區與 Amazon ECS。

  Fargate 上的 Windows 容器不支援此選項。
- 碼頭磁碟區 — `/var/lib/docker/volumes` 在主機 Amazon EC2 執行個體上建立的由碼頭管理的磁碟區。Docker 磁碟區驅動程式 (也稱為外掛程式) 用外部儲存系統 (例如 Amazon EBS) 來整合磁碟區。可使用內建 `local` 磁碟區驅動程式或第三方磁碟區驅動程式。只有在 Amazon EC2 執行個體上執行任務時，才支援 Docker 磁碟區。Windows 容器僅支援`local`驅動程式的使用。若要使用 Docker 磁碟區，請在您的任務定義中指定 `dockerVolumeConfiguration`。如需詳細資訊，請參閱使用磁碟區。
- 繫結掛載 — 裝載至容器的主機上的檔案或目錄。在 AWS Fargate 或 Amazon EC2 執行個體上執行任務時，支援繫結掛載主機磁碟區。若要使用綁定掛載主機磁碟區，請在任務定義中指定 `host` 和選用的 `sourcePath` 值。如需詳細資訊，請參閱使用綁定掛載。

如需詳細資訊，請參閱 Amazon ECS 任務的儲存選項。

容器定義允許使用以下參數。

`name`

類型：字串

必要：否

磁碟區名稱。最多可使用 255 個字母 (大寫和小寫)、數字、連字號 (-) 和底線 (_)。此名稱會在容器定義`mountPoints`物件的`sourceVolume`參數中參照。

`host`

必要：否

host 參數用於將綁定掛載的生命週期綁定到主機 Amazon EC2 執行個體，而非任務，以及它的儲存位置。如果 host 參數是空的，則 Docker 常駐程式會為您的資料磁碟區指派主機路徑，但其相關聯的容器停止執行後，不保證會保留資料。

Windows 容器可在 `$env:ProgramData` 所在的相同磁碟上掛載整個目錄。

> **Note**
>
> 只有在使用 Amazon EC2 執行個體上託管的任務時，才支援此`sourcePath`參數。

`sourcePath`

類型：字串

必要：否

使用 host 參數時，指定 sourcePath 以宣告在主機 Amazon EC2 執行個體上提供給容器的路徑。如果此參數是空的，則 Docker 常駐程式會為您指派主機路徑。如果 host 參數包含 sourcePath 檔案位置，資料磁碟區將保留在主機 Amazon EC2 執行個體上的指定位置，直到您手動將其刪除為止。如果 sourcePath 值不存在於主機 Amazon EC2 執行個體，Docker 常駐程式將建立該值。如果位置存在，將匯出來源路徑資料夾的內容。

`configuredAtLaunch`

類型：布林值

必要：否

指定磁碟區是否可在啟動時進行設定。設定為時`true`，您可以在執行獨立工作或建立或更新服務時設定磁碟區。設定為時`true`，您將無法在工作定義中提供其他磁碟區組態。必須將此參數設定`true`為設定 Amazon EBS 磁碟區以附加到任務。設定`configuredAtLaunch`為`true`並將磁碟區組態延遲至啟動階段，可讓您建立不受磁碟區類型或特定磁碟區設定限制的工作定義。這樣做可讓您的任務定義在不同的執行環境中重複使用。如需詳細資訊，請參閱 Amazon EBS 磁碟區。

`dockerVolumeConfiguration`

類型：DockerVolume配置對象

必要：否

此參數只有使用 Docker 磁碟區時才會指定。只有在 EC2 執行個體上執行任務時，才支援 Docker 磁碟區。Windows 容器僅支援`local`驅動程式的使用。若要使用綁定掛載，請指定 `host`。

`scope`

類型：字串

有效值: `task` | `shared`

必要：否

決定生命週期的 Docker 磁碟區範圍。範圍受限於 `task` 的 Docker 磁碟區，會在任務啟動時自動佈建，以及在任務停止時銷毀。範圍為 `shared` 的 Docker 磁碟區會在任務停止之後保留。

`autoprovision`

類型：布林值

預設值：`false`

必要：否

若此數值為 `true`，Docker 磁碟區便得以建立 (若它尚不存在)。只有在是時才會使用此欄`scope`位`shared`。如果`scope`是`task`，則必須省略此參數或設定為`false`。

`driver`

類型：字串

必要：否

要使用的 Docker 磁碟區驅動程式。驅動程式值必須與 Docker 提供的驅動程式名稱相符，因為此名稱用於工作放置。如果驅動程式是使用 Docker 外掛程式 CLI 安裝的，請使用`docker plugin ls`從容器執行個體擷取驅動程式名稱。如果使用其他方法安裝驅動程式，請使用 Docker 外掛程式探索來擷取驅動程式名稱。如需詳細資訊，請參閱 Docker 外掛程式探索。此參數會映射到 Docker Remote API 的建立磁碟區區段中的 `Driver` 以及 `docker volume create` 的 `--driver` 選項。

`driverOpts`

類型：字串

必要：否

要通過的 Docker 驅動程序特定選項的地圖。此參數會映射到 Docker Remote API 的建立磁碟區區段中的 DriverOpts 以及 docker volume create 的 --opt 選項。

labels

類型：字串

必要：否

自訂中繼資料，新增到您的 Docker 磁碟區。此參數會映射到 Docker Remote API 的建立磁碟區區段中的 Labels 以及 docker volume create 的 --label 選項。

efsVolumeConfiguration

類型：EFS VolumeConfiguration 物件

必要：否

只有使用 Amazon EFS 磁碟區時才會指定此參數。

fileSystemId

類型：字串

必要：是

要使用的 Amazon EFS 檔案系統識別碼。

rootDirectory

類型：字串

必要：否

在 Amazon EFS 檔案系統中的目錄，其將掛載作為主機內的根目錄。如果省略此參數，將使用 Amazon EFS 磁碟區的根目錄。指定 / 的效果與忽略此參數的效果相同。

> **Important**
>
> 如果在中指定 EFS 存取點authorizationConfig，則必須省略根目錄參數或將其設定為/，這樣會強制在 EFS 存取點上設定的路徑。

transitEncryption

類型：字串

有效值：ENABLED | DISABLED

必要：否

指定是否要對在 Amazon ECS 主機和 Amazon EFS 伺服器之間傳輸中的 Amazon EFS 資料啟用加密功能。如果使用 Amazon EFS IAM 授權，則必須啟用傳輸加密。如果省略此參數，系統會使用 DISABLED 的預設值。如需詳細資訊，請參閱《Amazon Elastic File System 使用者指南》中的加密傳輸中的資料。

`transitEncryptionPort`

類型：整數

必要：否

在 Amazon ECS 主機和 Amazon EFS 伺服器之間傳送加密資料時所使用的連接埠。如果未指定傳輸加密連接埠，任務將使用 Amazon EFS 掛載協助程式使用的連接埠選取策略。如需詳細資訊，請參閱《Amazon Elastic File System 使用者指南》中的 EFS 掛載協助程式。

`authorizationConfig`

類型：EFS AuthorizationConfiguration 物件

必要：否

Amazon EFS 檔案系統的授權組態詳細資訊。

`accessPointId`

類型：字串

必要：否

要使用的存取點 ID。如果指定存取點，則`efsVolumeConfiguration`必須省略中的根目錄值或將其設定為`/`，這樣會強制在 EFS 存取點上設定的路徑。如果使用存取點，則必須在 `EFSVolumeConfiguration` 中啟用傳輸加密。如需詳細資訊，請參閱《Amazon Elastic File System 使用者指南》中的使用 Amazon EFS 存取點。

`iam`

類型：字串

有效值：ENABLED | DISABLED

必要：否

指定掛接 Amazon EFS 檔案系統時，是否要使用任務定義中定義的 Amazon ECS 任務 IAM 角色。如果已啟用，必須在 EFSVolumeConfiguration 中啟用傳輸加密。如果省略此參數，系統會使用 DISABLED 的預設值。如需詳細資訊，請參閱任務的 IAM 角色。

FSxWindowsFileServerVolumeConfiguration

類型：F SxWindows FileServer VolumeConfiguration 對象

必要：是

當您使用適用於 Windows 檔案伺服器的 Amazon FSx 檔案伺服器檔案系統進行任務儲存時，會指定此參數。

fileSystemId

類型：字串

必要：是

要使用的 FSx for Windows File Server 檔案系統 ID。

rootDirectory

類型：字串

必要：是

FSx for Windows File Server 檔案系統中的目錄，其將掛載做為主機內的根目錄。

authorizationConfig

credentialsParameter

類型：字串

必要：是

授權憑證選項。

選項：

- Amazon 資源名稱（ARN）的AWS Secrets Manager秘密。
- AWS Systems Manager參數的 ARN。

domain

類型：字串

必要：是

由 AWS Directory Service for Microsoft Active Directory(AWS Managed Microsoft AD) 目錄或自我託管的 EC2 活動目錄託管的完整網域名稱。

標籤

當您註冊任務定義時，您可以選擇性地指定套用至任務定義的中繼資料標籤。標籤可協助您分類和組織您的任務定義。每個標籤皆包含索引鍵與選用值。您可以兩個都定義。如需詳細資訊，請參閱 標記 Amazon ECS 資源。

> **Important**
>
> 請勿在標籤中加入個人識別資訊或其他機密或敏感資訊。許多 AWS 服務都可以使用標籤，包括帳單。標籤不適用於私有或敏感資料。

標籤物件允許使用以下參數。

`key`

類型：字串

必要：否

組成標籤的鍵值組的一部分。索引鍵是一般標籤，作用就像更特定標籤值的類別。

`value`

類型：字串

必要：否

組成標籤的鍵值組的選用部分。值就像標籤類別 (索引鍵) 內的描述項。

其他任務定義參數

在 Amazon ECS 主控台中使用 Configure via JSON (透過 JSON 進行設定) 選項註冊任務定義時，可以使用下列任務定義參數。如需詳細資訊，請參閱 使用主控台建立 Amazon ECS 任務定義。

主題

- 暫時性儲存
- IPC 模式
- PID 模式

暫時性儲存

`ephemeralStorage`

類型：EphemeralStorage 物件

必要：否

為任務配置的暫時性儲存量 (以 GB 為單位)。對於託管於 AWS Fargate的任務，此參數可用來擴充可用的暫時性儲存總量 (超過預設數量)。如需詳細資訊，請參閱 the section called “綁定掛載”。

> **Note**
>
> 只有託管於 AWS Fargate 且使用平台版本 `1.4.0` 或更新版本 (Linux) 或 `1.0.0` 或更新版本 (Windows) 的任務才支援此參數。

IPC 模式

`ipcMode`

類型：字串

必要：否

要用於任務中容器的 IPC 資源命名空間。有效值為 `host`、`task` 或 `none`。如果已指定 `host`，則任務內對相同容器執行個體指定 `host` IPC 模式的所有容器，會與主機 Amazon EC2 執行個體共用相同的 IPC 資源。如果已指定 `task`，則指定任務內的所有容器會共用相同的 IPC 資源。如果已指定 `none`，則任務內的容器內的 IPC 資源為私有，並且不會與任務中或容器執行個體上的其他容器共用。如果沒有指定值，則 IPC 資源命名空間共用取決於容器執行個體上 Docker 常駐程式的設定。如需詳細資訊，請參閱《Docker run 參考》https://docs.docker.com/engine/reference/run/#ipc-settings---ipc中的 IPC 設定。

如果是使用 `host` IPC 模式，這會提高將不需要的 IPC 命名空間公開的風險。如需詳細資訊，請參閱 Docker 安全性。

如果是使用 systemControls 來為任務中的容器設定命名空間核心參數，以下各項會套用到您的 IPC 資源命名空間。如需詳細資訊，請參閱 系統控制。

- 針對使用 host IPC 模式的任務，不支援與 systemControls 相關的 IPC 命名空間。
- 針對使用 task IPC 模式的任務，IPC 命名空間相關的 systemControls 套用到任務內的所有容器。

> **Note**
>
> Windows 容器或使用 Fargate 啟動類型的任務不支援此參數。

PID 模式

pidMode

類型：字串

有效值: host | task

必要：否

要用於任務中容器的程序命名空間。有效值為 host 或 task。在 Linux 容器的 Fargate 上，唯一有效的值為 task。例如，監控附屬可能需要 pidMode 存取相同任務中執行之其他容器的相關資訊。

如果已指定 host，任務內對相容器執行個體指定 host PID 模式的所有容器，會與主機 Amazon EC2 執行個體共用相同的程序命名空間。

如果已指定 task，則指定任務內的所有容器會共用相同的程序命名空間。

如果未指定任何值，每個容器的預設值會是私有命名空間。如需詳細資訊，請參閱《Docker run 參考》https://docs.docker.com/engine/reference/run/#pid-settings---pid中的 PID 設定。

如果是使用 host PID 模式，這會提高將不需要的程序命名空間公開的風險。如需詳細資訊，請參閱 Docker 安全性。

> **Note**
>
> Windows 容器不支援此參數。

> **Note**
>
> 如果任務使用平台版本 `1.4.0` 或更高版本 (Linux)，只有託管於 AWS Fargate 的任務才支援此參數。Fargate 上的 Windows 容器不支援此參數。

Amazon ECS 任務定義模板

空白的任務定義範本如下所示。您可以使用此範本建立工作定義，然後將其貼到主控台 JSON 輸入區域，或儲存至檔案並與 AWS CLI `--cli-input-json`選項搭配使用。如需詳細資訊，請參閱 Amazon ECS 任務定義參數。

Amazon EC2 啟動類型模板

```
{
    "family": "",
    "taskRoleArn": "",
    "executionRoleArn": "",
    "networkMode": "none",
    "containerDefinitions": [
        {
            "name": "",
            "image": "",
            "repositoryCredentials": {
                "credentialsParameter": ""
            },
            "cpu": 0,
            "memory": 0,
            "memoryReservation": 0,
            "links": [
                ""
            ],
            "portMappings": [
                {
                    "containerPort": 0,
                    "hostPort": 0,
                    "protocol": "tcp"
                }
            ],
            "essential": true,
            "entryPoint": [
                ""
```

```
            ],
            "command": [
                ""
            ],
            "environment": [
                {
                    "name": "",
                    "value": ""
                }
            ],
            "environmentFiles": [
                {
                    "value": "",
                    "type": "s3"
                }
            ],
            "mountPoints": [
                {
                    "sourceVolume": "",
                    "containerPath": "",
                    "readOnly": true
                }
            ],
            "volumesFrom": [
                {
                    "sourceContainer": "",
                    "readOnly": true
                }
            ],
            "linuxParameters": {
                "capabilities": {
                    "add": [
                        ""
                    ],
                    "drop": [
                        ""
                    ]
                },
                "devices": [
                    {
                        "hostPath": "",
                        "containerPath": "",
                        "permissions": [
                            "read"
```

```
                    ]
                }
            ],
            "initProcessEnabled": true,
            "sharedMemorySize": 0,
            "tmpfs": [
                {
                    "containerPath": "",
                    "size": 0,
                    "mountOptions": [
                        ""
                    ]
                }
            ],
            "maxSwap": 0,
            "swappiness": 0
        },
        "secrets": [
            {
                "name": "",
                "valueFrom": ""
            }
        ],
        "dependsOn": [
            {
                "containerName": "",
                "condition": "COMPLETE"
            }
        ],
        "startTimeout": 0,
        "stopTimeout": 0,
        "hostname": "",
        "user": "",
        "workingDirectory": "",
        "disableNetworking": true,
        "privileged": true,
        "readonlyRootFilesystem": true,
        "dnsServers": [
            ""
        ],
        "dnsSearchDomains": [
            ""
        ],
        "extraHosts": [
```

```
            {
                "hostname": "",
                "ipAddress": ""
            }
        ],
        "dockerSecurityOptions": [
            ""
        ],
        "interactive": true,
        "pseudoTerminal": true,
        "dockerLabels": {
            "KeyName": ""
        },
        "ulimits": [
            {
                "name": "nofile",
                "softLimit": 0,
                "hardLimit": 0
            }
        ],
        "logConfiguration": {
            "logDriver": "splunk",
            "options": {
                "KeyName": ""
            },
            "secretOptions": [
                {
                    "name": "",
                    "valueFrom": ""
                }
            ]
        },
        "healthCheck": {
            "command": [
                ""
            ],
            "interval": 0,
            "timeout": 0,
            "retries": 0,
            "startPeriod": 0
        },
        "systemControls": [
            {
                "namespace": "",
```

```
                    "value": ""
                }
            ],
            "resourceRequirements": [
                {
                    "value": "",
                    "type": "InferenceAccelerator"
                }
            ],
            "firelensConfiguration": {
                "type": "fluentbit",
                "options": {
                    "KeyName": ""
                }
            }
        }
    ],
    "volumes": [
        {
            "name": "",
            "host": {
                "sourcePath": ""
            },
            "configuredAtLaunch": true,
            "dockerVolumeConfiguration": {
                "scope": "shared",
                "autoprovision": true,
                "driver": "",
                "driverOpts": {
                    "KeyName": ""
                },
                "labels": {
                    "KeyName": ""
                }
            },
            "efsVolumeConfiguration": {
                "fileSystemId": "",
                "rootDirectory": "",
                "transitEncryption": "DISABLED",
                "transitEncryptionPort": 0,
                "authorizationConfig": {
                    "accessPointId": "",
                    "iam": "ENABLED"
                }
```

```
            },
            "fsxWindowsFileServerVolumeConfiguration": {
                "fileSystemId": "",
                "rootDirectory": "",
                "authorizationConfig": {
                    "credentialsParameter": "",
                    "domain": ""
                }
            }
        }
    ],
    "placementConstraints": [
        {
            "type": "memberOf",
            "expression": ""
        }
    ],
    "requiresCompatibilities": [
        "EC2"
    ],
    "cpu": "",
    "memory": "",
    "tags": [
        {
            "key": "",
            "value": ""
        }
    ],
    "pidMode": "task",
    "ipcMode": "task",
    "proxyConfiguration": {
        "type": "APPMESH",
        "containerName": "",
        "properties": [
            {
                "name": "",
                "value": ""
            }
        ]
    },
    "inferenceAccelerators": [
        {
            "deviceName": "",
            "deviceType": ""
```

```
        }
    ],
    "ephemeralStorage": {
        "sizeInGiB": 0
    },
    "runtimePlatform": {
        "cpuArchitecture": "X86_64",
        "operatingSystemFamily": "WINDOWS_SERVER_20H2_CORE"
    }
}
```

Fargate 啟動類型模板

Important

對於 Fargate 啟動類型，您必須包含具有下列其中一個值的`operatingSystemFamily`參數：

- `LINUX`

- `WINDOWS_SERVER_2019_FULL`
- `WINDOWS_SERVER_2019_CORE`
- `WINDOWS_SERVER_2022_FULL`
- `WINDOWS_SERVER_2022_CORE`

```
{
    "family": "",
    "runtimePlatform": {"operatingSystemFamily": ""},
    "taskRoleArn": "",
    "executionRoleArn": "",
    "networkMode": "awsvpc",
    "platformFamily": "",
    "containerDefinitions": [
        {
            "name": "",
            "image": "",
            "repositoryCredentials": {"credentialsParameter": ""},
            "cpu": 0,
            "memory": 0,
```

```
            "memoryReservation": 0,
            "links": [""],
            "portMappings": [
                {
                    "containerPort": 0,
                    "hostPort": 0,
                    "protocol": "tcp"
                }
            ],
            "essential": true,
            "entryPoint": [""],
            "command": [""],
            "environment": [
                {
                    "name": "",
                    "value": ""
                }
            ],
            "environmentFiles": [
                {
                    "value": "",
                    "type": "s3"
                }
            ],
            "mountPoints": [
                {
                    "sourceVolume": "",
                    "containerPath": "",
                    "readOnly": true
                }
            ],
            "volumesFrom": [
                {
                    "sourceContainer": "",
                    "readOnly": true
                }
            ],
            "linuxParameters": {
                "capabilities": {
                    "add": [""],
                    "drop": [""]
                },
                "devices": [
                    {
```

```
                    "hostPath": "",
                    "containerPath": "",
                    "permissions": ["read"]
                }
            ],
            "initProcessEnabled": true,
            "sharedMemorySize": 0,
            "tmpfs": [
                {
                    "containerPath": "",
                    "size": 0,
                    "mountOptions": [""]
                }
            ],
            "maxSwap": 0,
            "swappiness": 0
        },
        "secrets": [
            {
                "name": "",
                "valueFrom": ""
            }
        ],
        "dependsOn": [
            {
                "containerName": "",
                "condition": "HEALTHY"
            }
        ],
        "startTimeout": 0,
        "stopTimeout": 0,
        "hostname": "",
        "user": "",
        "workingDirectory": "",
        "disableNetworking": true,
        "privileged": true,
        "readonlyRootFilesystem": true,
        "dnsServers": [""],
        "dnsSearchDomains": [""],
        "extraHosts": [
            {
                "hostname": "",
                "ipAddress": ""
            }
```

```
            ],
            "dockerSecurityOptions": [""],
            "interactive": true,
            "pseudoTerminal": true,
            "dockerLabels": {"KeyName": ""},
            "ulimits": [
                {
                    "name": "msgqueue",
                    "softLimit": 0,
                    "hardLimit": 0
                }
            ],
            "logConfiguration": {
                "logDriver": "awslogs",
                "options": {"KeyName": ""},
                "secretOptions": [
                    {
                        "name": "",
                        "valueFrom": ""
                    }
                ]
            },
            "healthCheck": {
                "command": [""],
                "interval": 0,
                "timeout": 0,
                "retries": 0,
                "startPeriod": 0
            },
            "systemControls": [
                {
                    "namespace": "",
                    "value": ""
                }
            ],
            "resourceRequirements": [
                {
                    "value": "",
                    "type": "GPU"
                }
            ],
            "firelensConfiguration": {
                "type": "fluentd",
                "options": {"KeyName": ""}
```

```
            }
        }
    ],
    "volumes": [
        {
            "name": "",
            "host": {"sourcePath": ""},
            "configuredAtLaunch":true,
            "dockerVolumeConfiguration": {
                "scope": "task",
                "autoprovision": true,
                "driver": "",
                "driverOpts": {"KeyName": ""},
                "labels": {"KeyName": ""}
            },
            "efsVolumeConfiguration": {
                "fileSystemId": "",
                "rootDirectory": "",
                "transitEncryption": "ENABLED",
                "transitEncryptionPort": 0,
                "authorizationConfig": {
                    "accessPointId": "",
                    "iam": "ENABLED"
                }
            }
        }
    ],
    "requiresCompatibilities": ["FARGATE"],
    "cpu": "",
    "memory": "",
    "tags": [
        {
            "key": "",
            "value": ""
        }
    ],
    "ephemeralStorage": {"sizeInGiB": 0},
    "pidMode": "task",
    "ipcMode": "none",
    "proxyConfiguration": {
        "type": "APPMESH",
        "containerName": "",
        "properties": [
            {
```

```
                "name": "",
                "value": ""
            }
        ]
    },
    "inferenceAccelerators": [
        {
            "deviceName": "",
            "deviceType": ""
        }
    ]
}
```

您可以使用下列 AWS CLI 命令產生此任務定義範本。

```
aws ecs register-task-definition --generate-cli-skeleton
```

Amazon ECS 任務定義範例

您可以複製範例和片段，以開始建立自己的工作定義。

您可以複製範例，然後在使用主控台中的透過 JSON 設定選項時貼上。確保自定範例，例如使用您的帳戶 ID。您可以在任務定義 JSON 中包含這些程式碼片段。如需詳細資訊，請參閱 使用主控台建立 Amazon ECS 任務定義 及 Amazon ECS 任務定義參數。

如需更多作業定義範例，請參閱上的AWS 範例作業定義 GitHub。

主題

- 网络服务器
- splunk日誌驅動
- fluentd日誌驅動
- gelf日誌驅動
- 外部執行個體上的工
- Amazon ECR 映像和任務定義 IAM 角色
- 具有指令的入口點
- 容器相依性
- Windows 任務定義範例

网络服务器

以下是使用 Fargate 啟動類型上 Linux 容器並設定 Web 伺服器的範例任務定義：

```
{
   "containerDefinitions": [
      {
         "command": [
            "/bin/sh -c \"echo '<html> <head> <title>Amazon ECS Sample App</
title> <style>body {margin-top: 40px; background-color: #333;} </style> </
head><body> <div style=color:white;text-align:center> <h1>Amazon ECS Sample App</h1>
 <h2>Congratulations!</h2> <p>Your application is now running on a container in Amazon
 ECS.</p> </div></body></html>' >  /usr/local/apache2/htdocs/index.html && httpd-
foreground\""
         ],
         "entryPoint": [
            "sh",
            "-c"
         ],
         "essential": true,
         "image": "httpd:2.4",
         "logConfiguration": {
            "logDriver": "awslogs",
            "options": {
               "awslogs-group" : "/ecs/fargate-task-definition",
               "awslogs-region": "us-east-1",
               "awslogs-stream-prefix": "ecs"
            }
         },
         "name": "sample-fargate-app",
         "portMappings": [
            {
               "containerPort": 80,
               "hostPort": 80,
               "protocol": "tcp"
            }
         ]
      }
   ],
   "cpu": "256",
   "executionRoleArn": "arn:aws:iam::012345678910:role/ecsTaskExecutionRole",
   "family": "fargate-task-definition",
   "memory": "512",
```

```
   "networkMode": "awsvpc",
   "runtimePlatform": {
        "operatingSystemFamily": "LINUX"
    },
   "requiresCompatibilities": [
       "FARGATE"
    ]
}
```

以下是使用 Fargate 啟動類型上 Windows 容器並設定 Web 伺服器的範例任務定義：

```
{
    "containerDefinitions": [
        {
            "command": ["New-Item -Path C:\\inetpub\\wwwroot\\index.html -Type file
 -Value '<html> <head> <title>Amazon ECS Sample App</title> <style>body {margin-top:
 40px; background-color: #333;} </style> </head><body> <div style=color:white;text-
align:center> <h1>Amazon ECS Sample App</h1> <h2>Congratulations!</h2> <p>Your
 application is now running on a container in Amazon ECS.</p>'; C:\\ServiceMonitor.exe
 w3svc"],
            "entryPoint": [
                "powershell",
                "-Command"
            ],
            "essential": true,
            "cpu": 2048,
            "memory": 4096,
            "image": "mcr.microsoft.com/windows/servercore/iis:windowsservercore-
ltsc2019",
            "name": "sample_windows_app",
            "portMappings": [
                {
                    "hostPort": 80,
                    "containerPort": 80,
                    "protocol": "tcp"
                }
            ]
        }
    ],
    "memory": "4096",
    "cpu": "2048",
    "networkMode": "awsvpc",
    "family": "windows-simple-iis-2019-core",
```

```
    "executionRoleArn": "arn:aws:iam::012345678910:role/ecsTaskExecutionRole",
    "runtimePlatform": {"operatingSystemFamily": "WINDOWS_SERVER_2019_CORE"},
    "requiresCompatibilities": ["FARGATE"]
}
```

splunk日誌驅動

以下程式碼片段示範如何在任務定義中使用 splunk 日誌驅動程式將日誌傳送到遠端服務。Splunk 字符參數被指定為秘密選項，因為它可以視為敏感資料。如需詳細資訊，請參閱 [將敏感資料傳遞至 Amazon ECS 容器]。

```
"containerDefinitions": [{
  "logConfiguration": {
   "logDriver": "splunk",
   "options": {
    "splunk-url": "https://cloud.splunk.com:8080",
    "tag": "tag_name",
   },
   "secretOptions": [{
    "name": "splunk-token",
    "valueFrom": "arn:aws:secretsmanager:region:aws_account_id:secret:splunk-token-
KnrBkD"
}],
```

fluentd日誌驅動

以下程式碼片段示範如何在任務定義中使用 fluentd 日誌驅動程式將日誌傳送到遠端服務。fluentd-address 值被指定為秘密選項，因為它可以視為敏感資料。如需詳細資訊，請參閱 [將敏感資料傳遞至 Amazon ECS 容器]。

```
"containerDefinitions": [{
 "logConfiguration": {
  "logDriver": "fluentd",
  "options": {
   "tag": "fluentd demo"
  },
  "secretOptions": [{
   "name": "fluentd-address",
   "valueFrom": "arn:aws:secretsmanager:region:aws_account_id:secret:fluentd-address-
KnrBkD"
  }]
```

```
 },
 "entryPoint": [],
 "portMappings": [{
             "hostPort": 80,
             "protocol": "tcp",
             "containerPort": 80
             },
             {
  "hostPort": 24224,
  "protocol": "tcp",
  "containerPort": 24224
 }]
}],
```

gelf日誌驅動

以下程式碼片段示範如何在任務定義中使用 gelf 日誌驅動程式，將日誌傳送到執行 Logstash (以 Gelf 日誌做為輸入) 的遠端主機。如需詳細資訊，請參閱 logConfiguration。

```
"containerDefinitions": [{
 "logConfiguration": {
  "logDriver": "gelf",
  "options": {
   "gelf-address": "udp://logstash-service-address:5000",
   "tag": "gelf task demo"
  }
 },
 "entryPoint": [],
 "portMappings": [{
   "hostPort": 5000,
   "protocol": "udp",
   "containerPort": 5000
  },
  {
   "hostPort": 5000,
   "protocol": "tcp",
   "containerPort": 5000
  }
 ]
}],
```

外部執行個體上的工

註冊 Amazon ECS 任務定義時，請使用 `requiresCompatibilities` 參數並指定 `EXTERNAL`，這可驗證任務定義是相容的，可在外部執行個體上執行 Amazon ECS 工作負載時使用。如果您使用主控台註冊任務定義，則必須使用 JSON 編輯器。如需詳細資訊，請參閱 使用主控台建立 Amazon ECS 任務定義。

> **Important**
>
> 如果您的任務需要任務執行 IAM 角色，請確定在任務定義中指定它。

當您部署工作負載時，請在建立服務或執行獨立任務時使用 `EXTERNAL` 啟動類型。

以下是任務定義範例。

Linux

```
{
 "requiresCompatibilities": [
  "EXTERNAL"
 ],
 "containerDefinitions": [{
  "name": "nginx",
  "image": "public.ecr.aws/nginx/nginx:latest",
  "memory": 256,
  "cpu": 256,
  "essential": true,
  "portMappings": [{
   "containerPort": 80,
   "hostPort": 8080,
   "protocol": "tcp"
  }]
 }],
 "networkMode": "bridge",
 "family": "nginx"
}
```

Windows

```
{
```

```
 "requiresCompatibilities": [
  "EXTERNAL"
 ],
 "containerDefinitions": [{
  "name": "windows-container",
  "image": "mcr.microsoft.com/windows/servercore/iis:windowsservercore-ltsc2019",
  "memory": 256,
  "cpu": 512,
  "essential": true,
  "portMappings": [{
   "containerPort": 80,
   "hostPort": 8080,
   "protocol": "tcp"
  }]
 }],
 "networkMode": "bridge",
 "family": "windows-container"
}
```

Amazon ECR 映像和任務定義 IAM 角色

以下程式碼片段使用稱為 aws-nodejs-sample 的 Amazon ECR 映像，具有來自 123456789012.dkr.ecr.us-west-2.amazonaws.com 登錄檔的 v1 標籤。此任務中的容器會繼承 arn:aws:iam::123456789012:role/AmazonECSTaskS3BucketRole 角色的 IAM 許可。如需詳細資訊，請參閱 Amazon ECS 任務 IAM 角色。

```
{
    "containerDefinitions": [
        {
            "name": "sample-app",
            "image": "123456789012.dkr.ecr.us-west-2.amazonaws.com/aws-nodejs-
sample:v1",
            "memory": 200,
            "cpu": 10,
            "essential": true
        }
    ],
    "family": "example_task_3",
    "taskRoleArn": "arn:aws:iam::123456789012:role/AmazonECSTaskS3BucketRole"
}
```

具有指令的入口點

以下程式碼片段會示範使用進入點和命令引數的 Docker 容器語法。此容器 ping google.com 四次，然後結束。

```
{
    "containerDefinitions": [
        {
            "memory": 32,
            "essential": true,
            "entryPoint": ["ping"],
            "name": "alpine_ping",
            "readonlyRootFilesystem": true,
            "image": "alpine:3.4",
            "command": [
                "-c",
                "4",
                "example.com"
            ],
            "cpu": 16
        }
    ],
    "family": "example_task_2"
}
```

容器相依性

此程式碼片段示範任務定義的語法，其中有多個容器且指定容器相依性。在以下任務定義中，envoy 容器必須達到運作良好狀態 (由必要的容器運作狀態檢查參數來判斷)，app 容器才會啟動。如需詳細資訊，請參閱 容器相依性。

```
{
  "family": "appmesh-gateway",
  "runtimePlatform": {
        "operatingSystemFamily": "LINUX"
  },
  "proxyConfiguration":{
      "type": "APPMESH",
      "containerName": "envoy",
      "properties": [
          {
              "name": "IgnoredUID",
```

```
                "value": "1337"
            },
            {
                "name": "ProxyIngressPort",
                "value": "15000"
            },
            {
                "name": "ProxyEgressPort",
                "value": "15001"
            },
            {
                "name": "AppPorts",
                "value": "9080"
            },
            {
                "name": "EgressIgnoredIPs",
                "value": "169.254.170.2,169.254.169.254"
            }
        ]
    },
    "containerDefinitions": [
      {
        "name": "app",
        "image": "application_image",
        "portMappings": [
          {
            "containerPort": 9080,
            "hostPort": 9080,
            "protocol": "tcp"
          }
        ],
        "essential": true,
        "dependsOn": [
          {
            "containerName": "envoy",
            "condition": "HEALTHY"
          }
        ]
      },
      {
        "name": "envoy",
        "image": "840364872350.dkr.ecr.region-code.amazonaws.com/aws-appmesh-
envoy:v1.15.1.0-prod",
        "essential": true,
```

```
      "environment": [
        {
          "name": "APPMESH_VIRTUAL_NODE_NAME",
          "value": "mesh/meshName/virtualNode/virtualNodeName"
        },
        {
          "name": "ENVOY_LOG_LEVEL",
          "value": "info"
        }
      ],
      "healthCheck": {
        "command": [
          "CMD-SHELL",
          "echo hello"
        ],
        "interval": 5,
        "timeout": 2,
        "retries": 3
      }
    }
  ],
  "executionRoleArn": "arn:aws:iam::123456789012:role/ecsTaskExecutionRole",
  "networkMode": "awsvpc"
}
```

Windows 任務定義範例

以下任務定義範例可協助您在 Amazon ECS 上開始使用 Windows 容器。

Example 適用於 Windows 的 Amazon ECS 主控台範例應用程式

以下任務定義是在 Amazon ECS 的之初次執行精靈中產生的 Amazon ECS 主控台範例應用程式，其已移植來使用 `microsoft/iis` Windows 容器映像。

```
{
  "family": "windows-simple-iis",
  "containerDefinitions": [
    {
      "name": "windows_sample_app",
      "image": "mcr.microsoft.com/windows/servercore/iis",
      "cpu": 1024,
      "entryPoint":["powershell", "-Command"],
```

```
      "command":["New-Item -Path C:\\inetpub\\wwwroot\\index.html -Type file -
Value '<html> <head> <title>Amazon ECS Sample App</title> <style>body {margin-top:
 40px; background-color: #333;} </style> </head><body> <div style=color:white;text-
align:center> <h1>Amazon ECS Sample App</h1> <h2>Congratulations!</h2> <p>Your
 application is now running on a container in Amazon ECS.</p>'; C:\\ServiceMonitor.exe
 w3svc"],
      "portMappings": [
        {
          "protocol": "tcp",
          "containerPort": 80
        }
      ],
      "memory": 1024,
      "essential": true
    }
  ],
  "networkMode": "awsvpc",
  "memory": "1024",
  "cpu": "1024"
}
```

Amazon ECS 叢集

Amazon ECS 叢集是任務或服務的邏輯分組。除了任務和服務之外，叢集還包含下列資源：

- 基礎架構容量可以是下列各項的組合：
  - AWS 雲端中的 Amazon EC2 執行個體
  - 雲端中的 AWS 無伺服器 (AWS Fargate (Fargate))
  - 內部部署虛擬機器或伺服器
- 執行任務和服務的網路 (VPC 和子網路)

  當您將 Amazon EC2 執行個體用於容量時，子網路可以位於可用區域、Local Zones、Wavelength 區域或 AWS Outposts。
- 選用的命名空間

  命名空間用於與服務 Connect 進行 service-to-service 通信。
- 監控選項

  CloudWatch 容器深入解析需要額外付費，而且是完全受控的服務。此服務會自動收集、彙總及總結 Amazon ECS 指標和日誌。

以下是有關 Amazon ECS 叢集的一般概念。

- Amazon ECS 會建立預設叢集。您可以建立其他叢集來分隔資源。
- 叢集是 AWS 區域 特定的。
- 叢集可以處於下列任一狀態。

  ACTIVE

  叢集已準備好接受任務，如果適用，您可以向叢集註冊容器執行個體。

  佈建中

  叢集具有與其相關聯的容量提供者，並且將建立容量提供者所需的資源。

  取消佈建中

  叢集具有與其相關聯的容量提供者，並且將刪除容量提供者所需的資源。

失敗

叢集具有與其相關聯的容量提供者，並且無法建立容量提供者所需的資源。

非作用中

已刪除叢集。具有 INACTIVE 狀態的叢集可能會在您的帳戶中保持可探索一段時間。此行為 future 可能會發生變化，因此請確保您不要依賴INACTIVE叢集持續存在。

- 叢集可以包含託管於 AWS Fargate Amazon EC2 執行個體或外部執行個體的混合任務。任務可以作為啟動類型或容量提供者策略在 Fargate 或 EC2 基礎設施上執行。如果您使用 EC2 做為啟動類型，Amazon ECS 不會追蹤和擴展 Amazon EC2 Auto Scaling 群組的容量。如需啟動類型的詳細資訊，請參閱「Amazon ECS 啟動類型」。
- 叢集可以同時包含 Auto Scaling 群組容量提供者以及 Fargate 容量提供者。容量提供者策略只能包含 Auto Scaling 群組容量提供者或 Fargate 容量提供者。
- 您可以針對 EC2 啟動類型或 Auto Scaling 群組容量提供者使用不同的執行個體類型。一個執行個體一次只能註冊到一個叢集。
- 您可以透過建立自訂 IAM 政策來限制叢集的存取。如需資訊，請參閱中的〈〉一Amazon ECS 集群示例節Amazon Elastic Container Service 身分型政策範例。
- 您可以使用服務自動擴展來擴展 Fargate 任務。如需詳細資訊，請參閱 自動擴展您的 Amazon ECS 服務。
- 您可以為叢集設定預設的服務 Connect 命名空間。設定預設的 Service Connect 命名空間之後，藉由開啟 Service Connect，在叢集中建立的任何新服務都可新增為命名空間中的用戶端服務。不需任何其他設定。如需詳細資訊，請參閱 使用服務 Connect 以簡短名稱連接 Amazon ECS 服務。

適用於 Fargate 啟動類型的 Amazon ECS 叢集

Amazon ECS 容量提供者會管理叢集中任務的基礎設施擴展。每個叢集可以有一或多個容量提供者，以及選用的容量提供者策略。容量提供者策略會決定任務在叢集的容量提供者之間分散的方式。當您執行獨立任務或建立服務時，可以使用叢集的預設容量提供者策略，也可以使用能覆寫預設值的容量提供者策略。

在上執行工作時 AWS Fargate，不需要建立或管理容量。您只需要將下列任何預先定義的容量提供者與叢集建立關聯：

- Fargate
- Fargate Spot

使用 AWS Fargate 容量供應商上的 Amazon ECS，您可以將 Fargate 和 Fargate 競價型容量與 Amazon ECS 任務一起使用。

透過 Fargate Spot，您能以折扣價 (與 Fargate 價格相比) 執行可接受中斷的 Amazon ECS 任務。Fargate Spot 在備用運算容量上執行任務。當 AWS 需要恢復容量時，您的任務會在兩分鐘的警告中斷。Fargate 競價型僅支援平台 1.3.0 或更新版本上具有 X86_64 架構的 Linux 工作。

當使用 Fargate 和 Fargate Spot 容量提供者的任務停止時，任務狀態變更事件會傳送至 Amazon。EventBridge停止原因說明其原因。如需詳細資訊，請參閱 Amazon ECS 任務狀態變更事件。

叢集可以同時包含 Fargate 和 Auto Scaling 群組容量提供者。不過，容量提供者策略只能包含 Fargate 或 Auto Scaling 群組容量提供者，不能同時包含兩者。如需詳細資訊，請參閱 Auto Scaling 群組容量提供者。

使用容量提供者時，請考慮下列事項：

- 您必須先將容量提供者與叢集建立關聯，才能將其與容量提供者策略產生關聯。
- 您最多可以為容量提供者策略指定 20 個容量提供者。
- 您無法將使用 Auto Scaling 群組容量提供者的服務更新為使用 Fargate 容量提供者。反之亦然。
- 在容量提供者策略中，如果沒有在主控台中對容量提供者指定 `weight` 值，則會使用預設值 1。如果使用 API 或 AWS CLI，則會使用的0預設值。
- 在容量提供者策略中指定多個容量提供者時，至少有一個容量提供者必須具有大於零的權重值。任何重量為零的容量提供者都不會用於放置工作。如果您在策略中指定多個容量提供者權重均為零，則使用容量提供者策略的任何 `RunTask` 或 `CreateService` 動作都會失敗。
- 在容量提供者策略中，只有一個容量提供者已定義基準值。如果未指定基準值，則會使用預設值零。
- 叢集可以同時包含 Auto Scaling 群組容量提供者以及 Fargate 容量提供者。不過，容量提供者策略只能包含 Auto Scaling 群組或 Fargate 容量提供者，不能同時包含兩者。
- 叢集可以同時包含使用容量提供者和啟動類型的各種服務和獨立任務。服務可以更新為使用容量提供者策略，而非啟動類型。不過，若要執行此操作，您必須強制執行新部署。

Fargate 現貨終止通知

在需求極高期間，Fargate Spot 容量可能無法使用。這可能會導致 Fargate Spot 任務延遲。發生這種情況時，Amazon ECS 服務會重試啟動任務，直到所需的容量可用為止。Fargate 不會以隨需容量取代 Spot 容量。

當使用 Fargate Spot 容量的任務因 Spot 中斷而停止時，在任務停止之前會傳送兩分鐘的警告。警告會以任務狀態變更事件的形式傳送至 Amazon， EventBridge 並以 SIGTERM 訊號傳送至執行中的任

務。如果您使用 Fargate Spot 作為服務的一部分，在此情況中，服務排程器會收到中斷訊號，並嘗試在容量可用時啟動其他任務。只有一項任務的服務會中斷，直到容量可用為止。如需正常關機的詳細資訊，請參閱使用 ECS 正常關機。

若要確保您的容器在任務停止之前正常結束，可設定下列項目：

- 您可以在任務所使用的容器定義中指定 120 秒數的 `stopTimeout` 值 (或更少)。預設 `stopTimeout` 值為 30 秒。您可以指定更長的 `stopTimeout` 值，讓您從收到任務狀態變更事件到容器強制停止之間有更多時間。如需詳細資訊，請參閱 容器逾時。
- SIGTERM 信號必須從容器內接收，以執行任何清理動作。無法處理此信號將導致任務在設定 `stopTimeout` 後收到 SIGKILL 信號，甚至可能導致資料遺失或毀損。

以下是任務狀態變更事件的程式碼片段。此程式碼片段顯示停止原因和 Fargate Spot 中斷的停止程式碼。

```
{
  "version": "0",
  "id": "9bcdac79-b31f-4d3d-9410-fbd727c29fab",
  "detail-type": "ECS Task State Change",
  "source": "aws.ecs",
  "account": "111122223333",
  "resources": [
    "arn:aws:ecs:us-east-1:111122223333:task/b99d40b3-5176-4f71-9a52-9dbd6f1cebef"
  ],
  "detail": {
    "clusterArn": "arn:aws:ecs:us-east-1:111122223333:cluster/default",
    "createdAt": "2016-12-06T16:41:05.702Z",
    "desiredStatus": "STOPPED",
    "lastStatus": "RUNNING",
    "stoppedReason": "Your Spot Task was interrupted.",
    "stopCode": "SpotInterruption",
    "taskArn": "arn:aws:ecs:us-east-1:111122223333:task/
b99d40b3-5176-4f71-9a52-9dbd6fEXAMPLE",
    ...
  }
}
```

以下是用來建立 Amazon ECS 任務狀態變更事件 EventBridge 規則的事件模式。您可以選擇在 `detail` 欄位中指定叢集。若指定，則表示您會收到該叢集的任務狀態變更事件。如需詳細資訊，請參閱 Amazon EventBridge 使用者指南中的建立 EventBridge 規則。

```
{
    "source": [
        "aws.ecs"
    ],
    "detail-type": [
        "ECS Task State Change"
    ],
    "detail": {
        "clusterArn": [
            "arn:aws:ecs:us-west-2:111122223333:cluster/default"
        ]
    }
}
```

為 Fargate 啟動類型建立 Amazon ECS 叢集

您可以使用 Amazon ECS 主控台建立 Amazon ECS 叢集。開始之前，請務必先完成 設定以使用 Amazon ECS。 中的步驟，並指派適當的 IAM 許可。如需詳細資訊，請參閱 the section called "Amazon ECS 集群示例"。Amazon ECS 主控台透過建立堆疊來建立 Amazon ECS 叢集所需的資源。AWS CloudFormation

主控台會自動將 Fargate 和 Fargate Spot 容量提供者與叢集建立關聯。

除了叢集之外，主控台還會自動建立下列資源：

- 其中 AWS Cloud Map 的預設命名空間與叢集名稱相同。命名空間可讓您在叢集中建立的服務連線到命名空間中的其他服務，而不需要額外的組態。

  如需詳細資訊，請參閱 互連 Amazon ECS 服務。

您可以修改下列選項：

- 變更與叢集相關聯的預設命名空間。
- 開啟 Container Insights。

  CloudWatch Container Insights 會從您的容器化應用程式和微服務收集、彙總和摘要指標和記錄。Container Insights 還提供診斷資訊，例如容器重新啟動故障，您可以使用這些資訊快速隔離和解決這些問題。如需詳細資訊，請參閱 the section called "使用容器洞察來監控 Amazon ECS 容器"。
- 新增標籤以協助您識別叢集。

程序

建立新叢集 (Amazon ECS 主控台)

1. 開啟主控台，網址為 https://console.aws.amazon.com/ecs/v2。
2. 從導覽列中選取要使用的「區域」。
3. 在導覽窗格中，選擇叢集。
4. 在 Clusters (叢集) 頁面上，選擇 Create cluster (建立叢集)。
5. 在叢集組態下，設定下列項目：
   - 在叢集名稱下輸入唯一的名稱。

     名稱可以包含最多 255 個字母 (大小寫)、數字與連字號。
   - (選用) 若要讓 Service Connect 使用的命名空間與叢集名稱不同，請在命名空間中輸入唯一的名稱。
6. (選用) 若要開啟 Container Insights，請展開 Monitoring (監控)，然後開啟 Use Container Insights (使用 Container Insights)。
7. (選用) 為協助識別您的叢集，請展開 Tags (標籤)，然後設定標籤。

   [新增標籤] 選擇新增標籤，並執行下列動作：
   - 對於 Key (金鑰)，輸入金鑰名稱。
   - 對於 Value (值)，進入金鑰值。

   [移除標籤] 選擇標籤「金鑰」和「值」右側的移除。
8. 選擇建立。

後續步驟

建立叢集之後，您可以為應用程式建立任務定義，然後將它們當做獨立任務或服務的一部分執行。如需詳細資訊，請參閱下列內容：

- Amazon ECS 任務定義
- 以 Amazon ECS 任務的形式執行應用程式
- 使用主控台建立 Amazon ECS 服務

適用於 EC2 啟動類型的 Amazon ECS 容量供應商

當您針對容量使用 Amazon EC2 執行個體時，可使用 Auto Scaling 群組來管理註冊到其叢集的 Amazon EC2 執行個體。Auto Scaling 有助於確保您擁有正確數量的 Amazon EC2 執行個體可用來處理應用程式負載。

您可以使用受管擴展功能讓 Amazon ECS 管理 Auto Scaling 群組的擴展和水平擴展動作，或者您也可以自行管理擴展動作。如需詳細資訊，請參閱 透過叢集自動擴展自動管理 Amazon ECS 容量。

建議您建立新的空白「Auto Scaling」群組。如果您使用現有的 Auto Scaling 群組，則任何與執行中群組相關聯以及註冊至 Amazon ECS 叢集的 Amazon EC2 執行個體在使用 Auto Scaling 群組建立容量提供者之前，可能無法正確的註冊容量提供者。在容量提供者策略中使用容量提供者時，這可能會造成問題。使用 `DescribeContainerInstances` 可以確認容器執行個體是否與容量提供者建立關聯。

> **Note**
>
> 若要建立空白 Auto Scaling 群組，請將所需的計數設定為零。建立容量提供者並將其與叢集關聯後，您就可以進行橫向擴展。
> 當您使用 Amazon ECS 主控台時，Amazon ECS 會代表您建立 Amazon EC2 啟動範本和 Auto Scaling 群組，做為堆疊的一部分。 AWS CloudFormation 它們前綴為. `EC2ContainerService-<ClusterName>` 您可以使用 Auto Scaling 群組作為該叢集的容量提供者。

我們建議您使用受管執行個體排空，以允許正常終止不會中斷工作負載的 Amazon EC2 執行個體。此功能預設為開啟。如需更多資訊，請參閱安全停止在 EC2 執行個體上執行的 Amazon ECS 工作負載

在主控台中使用 Auto Scaling 群組容量提供者時應考慮以下事項：

- Auto Scaling 群組的 `MaxSize` 必須大於零，才能水平擴展。
- Auto Scaling 群組不能具有執行個體權重設定。
- 如果 Auto Scaling 群組無法橫向擴展以容納執行的任務數目，則任務將無法轉換超出 `PROVISIONING` 狀態。
- 不要修改與容量提供者管理的 Auto Scaling 群組關聯的擴展政策資源。
- 如果在建立容量提供者時開啟受管擴展，則可將 Auto Scaling 群組所需的計數設定為 `0`。開啟受管擴展時，Amazon ECS 會管理 Auto Scaling 群組的縮減和橫向擴展動作。
- 您必須先將容量提供者與叢集建立關聯，才能將其與容量提供者策略產生關聯。

- 您最多可以為容量提供者策略指定 20 個容量提供者。
- 您無法將使用 Auto Scaling 群組容量提供者的服務更新為使用 Fargate 容量提供者。反之亦然。
- 在容量提供者策略中，如果沒有在主控台中對容量提供者指定 `weight` 值，則會使用預設值 1。如果使用 API 或 AWS CLI，則會使用的0預設值。
- 在容量提供者策略中指定多個容量提供者時，至少有一個容量提供者必須具有大於零的權重值。任何重量為零的容量提供者都不會用於放置工作。如果您在策略中指定多個容量提供者權重均為零，則使用容量提供者策略的任何 `RunTask` 或 `CreateService` 動作都會失敗。
- 在容量提供者策略中，只有一個容量提供者已定義基準值。如果未指定基準值，則會使用預設值零。
- 叢集可以同時包含 Auto Scaling 群組容量提供者以及 Fargate 容量提供者。不過，容量提供者策略只能包含 Auto Scaling 群組或 Fargate 容量提供者，不能同時包含兩者。
- 叢集可以同時包含使用容量提供者和啟動類型的各種服務和獨立任務。服務可以更新為使用容量提供者策略，而非啟動類型。不過，若要執行此操作，您必須強制執行新部署。
- Amazon ECS 支援 Amazon EC2 Auto Scaling 暖集區。暖集區是一組準備投入使用的預先初始化 Amazon EC2 執行個體。每當您的應用程式需要向外擴展時，Amazon EC2 Auto Scaling 都會使用暖池中預先初始化的執行個體，而不是啟動冷執行個體。這可讓任何最終初始化程序在執行個體投入服務之前執行。如需詳細資訊，請參閱 為您的 Amazon ECS Auto Scaling 群組設定預先初始化的執行個體。

如需有關建立 Amazon EC2 Auto Scaling 啟動範本的詳細資訊，請參閱《Amazon EC2 Auto Scaling 使用者指南》中的啟動範本。如需有關建立 Amazon EC2 Auto Scaling 群組的詳細資訊，請參閱《Amazon EC2 Auto Scaling 使用者指南》中的 Auto Scaling 群組。

Amazon ECS 的亞馬遜 EC2 容器執行個體安全考量

您應該考量單一容器執行個體及其在威脅模型中的存取。例如，單一受影響的任務可能在相同執行個體上利用未受感染任務的 IAM 許可。

建議您採用下列動作以協助防止此種情況：

- 執行任務時，請勿使用管理員權限。
- 為您的任務指派具有最低權限存取的任務角色。

  容器代理程式會自動建立具有唯一憑證 ID 的字符，此字符可用於存取 Amazon ECS 資源。
- 針對任務中採用 `awsvpc` 網路模式的容器，如欲避免其存取提供給 Amazon EC2 執行個體設定檔的憑證資訊 (同時仍然允許任務角色所提供的許可)，請在代理程式組態檔案中將 `ECS_AWSVPC_BLOCK_IMDS` 代理程式組態變數設定為 true，並重新啟動代理程式。

- 使用 Amazon GuardDuty 執行階段監控來偵測 AWS 環境中叢集和容器的威脅。執行階段監控使用 GuardDuty 安全代理程式，為個別 Amazon ECS 工作負載新增執行時期可見性，例如檔案存取、程序執行和網路連線。如需詳細資訊，請參閱GuardDuty 使用指南中的GuardDuty執行階段監視。

為 Amazon EC2 啟動類型建立亞馬遜 ECS 叢集

您可以使用主控台建立 Amazon ECS 叢集。開始之前，請務必先完成 設定以使用 Amazon ECS。 中的步驟，並指派適當的 IAM 許可。如需詳細資訊，請參閱 the section called “Amazon ECS 集群示例”。Amazon ECS 主控台提供一種簡單的方法，透過建立堆疊來建立 Amazon ECS 叢集所需的資源。 AWS CloudFormation

為了使叢集建立程序盡可能簡單，主控台提供了許多可供選擇的預設選項，我們將在下方加以說明。主控台的大多數區段還有說明面板，以提供更多上下文。

您可以在建立叢集時註冊 Amazon EC2 執行個體，或在建立叢集之後用叢集註冊其他執行個體。

您可以修改下列預設選項：

- 變更執行個體啟動的子網路
- 變更用來控制容器執行個體流量的安全群組
- 變更與叢集相關聯的預設命名空間。

  命名空間可讓您在叢集中建立的服務連線到命名空間中的其他服務，而不需要額外的組態。預設命名空間與叢集名稱相同。如需詳細資訊，請參閱 互連 Amazon ECS 服務。
- 開啟 Container Insights。

  CloudWatch Container Insights 會從您的容器化應用程式和微服務收集、彙總和摘要指標和記錄。Container Insights 還提供診斷資訊，例如容器重新啟動故障，您可以使用這些資訊快速隔離和解決這些問題。如需詳細資訊，請參閱 the section called “使用容器洞察來監控 Amazon ECS 容器”。
- 新增標籤以協助您識別叢集。

Auto Scaling 群組選項

如果使用 Amazon EC2 執行個體，則必須指定 Auto Scaling 群組來管理任務和服務執行所在的基礎設施。

當您選擇建立新的 Auto Scaling 群組時，系統會自動設定為以下行為：

- Amazon ECS 管理 Auto Scaling 群組的縮減和水平擴展動作。
- Amazon ECS 不會防止包含任務和位於 Auto Scaling 群組中的 Amazon EC2 執行個體在縮減動作期間被終止。如需詳細資訊，請參閱 AWS Auto Scaling 使用者指南中的執行個體保護。

您可以設定以下 Auto Scaling 群組屬性，以確定要為群組啟動的執行個體類型和數量：

- Amazon ECS 最佳化 AMI。
- 執行個體類型。
- 連線到執行個體時證明您身分的 SSH 金鑰對。如需如何建立安全殼層金鑰的詳細資訊，請參閱 Amazon EC2 使用者指南中的 Amazon EC2 金鑰配對和 Linux 執行個體。
- 要為 Auto Scaling 群組啟動的最小執行個體數量。
- 將為 Auto Scaling 群組啟動的最大執行個體數量。

  為了水平擴展群組，最大值必須大於 0。

作為 AWS CloudFormation 堆疊的一部分，Amazon ECS 代表您建立 Amazon EC2 Auto Scaling 啟動範本和 Auto Scaling 群組。您為 AMI、執行個體類型和 SSH 金鑰對指定的值為啟動範本的一部分。範本字首會加上 `EC2ContainerService-<ClusterName>`，這使得它們容易識別。Auto Scaling 群組的字首為 `<ClusterName>-ECS-Infra-ECSAutoScalingGroup`。

為 Auto Scaling 群組啟動的執行個體使用啟動範本。

網路選項

根據預設，執行個體會啟動至「區域」的預設子網路中。系統會使用目前與子網路相關聯的安全性群組 (控制容器執行個體的流量)。您可以變更執行個體的子網路和安全群組。

您可以選擇現有的子網路。您可以使用現有的安全群組，也可以建立新的安全群組。當您建立新的安全群組時，您需要指定至少一個傳入規則。

傳入規則會決定哪些流量可以連接到您的容器執行個體，並包含下列屬性：

- 要允許的通訊協定
- 要允許的連接埠範圍
- 入站流量（來源）

若要允許來自特定位址或 CIDR 區塊的傳入流量，請針對來源使用自訂選項，且具有允許的 CIDR。

若要允許來自所有目的地的傳入流量，請針對來源使用隨處。此選項會自動新增 0.0.0.0/0 IPv4 CIDR 區塊和 ::/0 IPv6 CIDR 區塊。

若要允許來自本機電腦的傳入流量，請針對來源使用來源群組。這會自動將您本機電腦目前的 IP 地址新增為允許的來源。

建立新叢集 (Amazon ECS 主控台)

在開始之前，請指派適當的 IAM 許可。如需詳細資訊，請參閱 the section called “Amazon ECS 集群示例”。

1. 開啟主控台，網址為 https://console.aws.amazon.com/ecs/v2。
2. 從導覽列中選取要使用的「區域」。
3. 在導覽窗格中，選擇叢集。
4. 在 Clusters (叢集) 頁面上，選擇 Create cluster (建立叢集)。
5. 在叢集組態下，設定下列項目：
   - 在叢集名稱下輸入唯一的名稱。

     名稱可以包含最多 255 個字母 (大小寫)、數字與連字號。
   - (選用) 若要讓 Service Connect 使用的命名空間與叢集名稱不同，請在命名空間中輸入唯一的名稱。
6. 將 Amazon EC2 執行個體新增至叢集，展開基礎設施，清除 AWS Fargate (無伺服器)，然後選取 Amazon EC2 執行個體。接下來，設定作為容量提供者的 Auto Scaling 群組：
   a. 要使用現有 Auto Scaling 群組，請從 Auto Scaling group (ASG) (Auto Scaling 群組 (ASG)) 中選取該群組。
   b. 若要建立 Auto Scaling 群組，請從 Auto Scaling group (ASG) (Auto Scaling 群組 (ASG)) 中選取 Create new group (建立新群組)，然後提供有關該群組的下列詳細資訊：
      - 對於佈建模式，請選擇是使用隨需執行個體還是 Spot 執行個體。
      - 如果您選擇使用 Spot 執行個體，請針對配置策略選擇要用於執行個體的 Spot 容量集區 (執行個體類型和可用區域)。

        對於大多數工作負載，您可以選擇價格容量最佳化。

        如需詳細資訊，請參閱《Amazon EC2 使用者指南》中的 Spot 執行個體的分配策略。

- 針對 Operating system/Architecture (作業系統/架構)，為 Auto Scaling 群組執行個體選擇 Amazon ECS 最佳化 AMI。
- 對於 EC2 instance type (EC2 執行個體類型)，選擇適合您工作負載的執行個體類型。

  如果 Auto Scaling 群組使用相同或類似的執行個體類型，則受管擴展效果最佳。
- 對於 EC2 執行個體角色，請選擇現有的容器執行個體角色，或者建立新的執行個體角色。

  如需詳細資訊，請參閱 Amazon ECS 容器執行個體 IAM 角色。
- 對於容量，輸入 Auto Scaling 群組中要啟動的最小執行個體數和最大執行個體數。
- 對於 SSH key pair (SSH 金鑰對)，選擇在連線到執行個體時證明您身分的金鑰對。
- 若要允許較大的影像和儲存空間，對於根 EBS 磁碟區大小，請以 GiB 為單位輸入值。

7. (選用) 若要變更 VPC 和子網路，請在 Amazon EC2 執行個體的聯網能力下，執行下列任一操作：

   - 若要移除子網路，請在 Subnets (子網路) 下，對您要移除之每一個子網路選擇 X。
   - 若要變更為非預設 VPC，請在 VPC 下，選擇現有的 VPC，然後在子網路下選擇子網路。
   - 選擇安全群組。在安全群組下，選擇以下其中一個選項：
     - 若要使用現有的安全群組，請選擇使用現有安全群組，然後選擇安全群組。
     - 若要建立安全群組，請選擇建立新的安全群組。然後，針對傳入規則選擇新增規則。

       如需傳入規則的資訊，請參閱 網路選項。
   - 若要自動將公有 IP 地址指派給 Amazon EC2 容器執行個體，針對自動指派公有 IP，請選擇下列其中一個選項：
     - 使用子網路設定：當執行個體啟動的子網路為公有子網路時，將公有 IP 地址指派給行個體。
     - 開啟：將公有 IP 位址指派給執行個體。
8. (選用) 若要開啟 Container Insights，請展開 Monitoring (監控)，然後開啟 Use Container Insights (使用 Container Insights)。
9. (選用)

   如果您搭配手動選項使用「程式實際執行監視」，而且您想要監督此叢集 GuardDuty，請選擇「新增標記」，然後執行下列動作：

   - 對於金鑰，請輸入 **`guardDutyRuntimeMonitoringManaged`**
   - 針對數值，輸入 **`true`**。
10. (選用) 若要管理叢集標籤，請展開標籤，然後執行下列其中一項操作：

[新增標籤] 選擇新增標籤，並執行下列動作：

- 對於 Key (金鑰)，輸入金鑰名稱。
- 對於 Value (值)，進入金鑰值。

[移除標籤] 選擇標籤「金鑰」和「值」右側的移除。

11. 選擇建立。

後續步驟

建立叢集之後，您可以為應用程式建立任務定義，然後將它們當做獨立任務或服務的一部分執行。如需詳細資訊，請參閱下列內容：

- Amazon ECS 任務定義
- 以 Amazon ECS 任務的形式執行應用程式
- 使用主控台建立 Amazon ECS 服務

透過叢集自動擴展自動管理 Amazon ECS 容量

Amazon ECS 可管理在您的叢集註冊的 Amazon EC2 執行個體的擴展。這稱為 Amazon ECS 叢集自動擴展。您可以在建立 Amazon ECS 自動擴展群組容量提供者時開啟受管擴展。然後，您可以在此 Auto Scaling 群組中設定執行個體使用率的目標百分比 (`targetCapacity`)。Amazon ECS 會為您的自動擴展群組建立兩個自訂 CloudWatch 指標和一個目標追蹤擴展政策。然後，Amazon ECS 會根據您的任務使用的資源使用率來管理擴展和水平擴充動作。

針對每個與叢集關聯的 Auto Scaling 群組容量提供者，Amazon ECS 會建立並管理以下資源：

- 低度量值 CloudWatch 警報
- 高度量值 CloudWatch 警報
- 目標追蹤擴展政策

> **Note**
>
> Amazon ECS 會建立目標追蹤擴展政策，並將其連接到 Auto Scaling 群組。若要更新目標追蹤擴展政策，請更新容量提供者受管擴展設定，而不是直接更新擴展政策。

當您關閉受管擴展或取消容量提供者與叢集的關聯時，Amazon ECS 會同時移除 CloudWatch 指標和目標追蹤擴展政策資源。

Amazon ECS 使用下列指標來決定要採取的動作：

`CapacityProviderReservation`

特定容量提供者使用的容器執行個體百分比。Amazon ECS 會產生此指標。

Amazon ECS 將 `CapacityProviderReservation` 值設定為 0-100 之間的數字。Amazon ECS 使用下列公式來表示 Auto Scaling 群組中剩餘容量的比率。然後，Amazon ECS 將指標發佈到 CloudWatch。如需有關如何計算指標的詳細資訊，請參閱 Amazon ECS 叢集 Auto Scaling 的深入瞭解。

```
CapacityProviderReservation = (number of instances needed) / (number of running
 instances) x 100
```

`DesiredCapacity`

Auto Scaling 群組的容量。此量度未發佈 CloudWatch。

Amazon ECS 會將`CapacityProviderReservation`指標發佈到`AWS/ECS/ManagedScaling`命名空間 CloudWatch 中。`CapacityProviderReservation` 指標會導致發生下列其中一個動作：

`CapacityProviderReservation` 值等於 `targetCapacity`

Auto Scaling 群組不需要橫向擴展或縮減。已達到目標使用率百分比。

`CapacityProviderReservation` 值大於 `targetCapacity`

有更多任務使用比您的 `targetCapacity` 百分比高的容量百分比。`CapacityProviderReservation`量度的值增加會導致相關聯的 CloudWatch 警示發生作用。此警示會更新 Auto Scaling 群組的 `DesiredCapacity` 值。Auto Scaling 群組會使用此值啟動 EC2 執行個體，然後向叢集註冊這些執行個體。

當 `targetCapacity` 的預設值為 100% 時，新任務會在橫向擴展期間處於 `PENDING` 狀態，因為執行個體上沒有可用容量來執行任務。新執行個體向 ECS 註冊後，這些任務將在新執行個體上開始執行。

`CapacityProviderReservation` 值小於 `targetCapacity`

使用容量百分比低於 `targetCapacity` 的百分比的任務更少，並且至少有一個可以終止的執行個體。`CapacityProviderReservation`量度的值減少會導致相關聯的 CloudWatch 警示發生作

用。此警示會更新 Auto Scaling 群組的 `DesiredCapacity` 值。Auto Scaling 群組會使用此值終止 EC2 容器執行個體，然後向叢集取消註冊這些執行個體。

Auto Scaling 群組遵循群組終止政策來確定在縮減事件期間首先終止哪些執行個體。此外，它還可以避免開啟執行個體縮減保護設定的執行個體。如果您開啟受管終止保護，叢集自動擴展可以管理哪些執行個體具有執行個體縮減保護設定。如需有關受管終止保護的詳細資訊，請參閱 控制 Amazon ECS 終止實例。如需 Auto Scaling 群組終止執行個體的詳細資訊，請參閱 Amazon EC2 Auto Scaling 使用者指南中的控制在縮減期間會終止的 Auto Scaling 執行個體。

使用叢集自動擴展時應考慮以下事項：

- 請勿以任何擴展政策變更或管理與容量提供者相關聯的 Auto Scaling 群組所需的容量，Amazon ECS 管理的政策除外。
- Amazon ECS 會針對代表您呼叫 AWS Auto Scaling 所需的許可，使用`AWSServiceRoleForECS`服務連結的 IAM 角色。如需詳細資訊，請參閱 使用 Amazon ECS 的服務連結角色。
- 將容量提供者與 Auto Scaling 群組搭配使用時，建立容量提供者的使用者、群組或角色需要 `autoscaling:CreateOrUpdateTags` 許可。這是因為當群組與容量提供者產生關聯時，Amazon ECS 會將標籤加入至 Auto Scaling 群組。

> **Important**
>
> 確保您使用的任何工具不會將 `AmazonECSManaged` 標籤從 Auto Scaling 群組中移除。如果移除此標籤，Amazon ECS 將無法管理擴展。

- 叢集 auto 調整不會修改群組MaximumCapacity的MinimumCapacity或。對於要向外延展的群組，的值MaximumCapacity必須大於零。
- 當自動擴展 (受管擴展) 開啟時，一個容量提供者一次只能連接到一個叢集。如果容量提供者已關閉受管擴展，您可以將其關聯到多個叢集。
- 當受管擴展關閉時，容量提供者不會縮減或橫向擴展。您可以使用容量提供者策略在容量提供者之間平衡您的任務。
- 就`binpack`能力而言，該策略是最有效的策略。
- 當目標容量低於 100% 時，配置策略需要使用`binpack`策略，而`spread`策略的順序不會高於`binpack`策略。這樣可防止容量提供者向外擴充，直到每個工作都有專用執行個體或達到限制為止。

優化 Amazon ECS 叢集 auto 擴展

在 Amazon EC2 上執行 Amazon ECS 的客戶可以利用叢集 auto 擴展來管理 Amazon EC2 自動擴展群組的擴展。透過叢集 auto 動擴展，您可以將 Amazon ECS 設定為自動擴展自動擴展群組，並專注於執行任務。Amazon ECS 將確保「Auto Scaling」群組可根據需要擴展和擴展，無需進一步介入。Amazon ECS 容量提供者可透過確保有足夠的容器執行個體來滿足應用程式的需求，來管理叢集中的基礎設施。若要了解叢集 auto 擴展如何運作，請參閱深入瞭解 Amazon ECS 叢集自動擴展。

叢集 auto 擴展依賴 CloudWatch 於與 Auto Scaling 群組的整合來調整叢集容量。因此，它具有與發佈 CloudWatch 指標相關的固有延遲、指標違規 CloudWatch 警示所花費的時間 (高和低)，以及新啟動的 Amazon EC2 執行個體預熱所花費的時間。`CapacityProviderReservation`您可以採取下列動作，使叢集 auto 調整更快的回應速度，以加快部署速度：

容量提供者步驟擴展大小

Amazon ECS 容量供應商最終會擴大/縮小容器執行個體，以滿足應用程式的需求。預設情況下，Amazon ECS 將啟動的執行個體數目下限設定為 1。如果需要數個執行個體來放置擱置的工作，這可能會增加部署的時間。您可以`minimumScalingStepSize`透過 Amazon ECS API 增加，以增加 Amazon ECS 一次擴展或擴展出的最小執行個體數量。太低`maximumScalingStepSize`的容器執行個體可以限制一次擴展或擴展多少個容器執行個體，這可能會降低部署速度。

> **Note**
>
> 此組態目前僅可透過`CreateCapacityProvider`或 `UpdateCapacityProvider`API 使用。

執行個體預熱期

執行個體預熱期是指新啟動的 Amazon EC2 執行個體可為 Auto Scaling 群組 CloudWatch 指標做出貢獻的一段時間。指定的預熱期到期後，執行個體就會計入 Auto Scaling 群組的彙總指標，叢集自動調整規模會繼續進行下一次計算，以估計所需的執行個體數量。

預設值`instanceWarmupPeriod`為 300 秒，您可以透過`CreateCapacityProvider`或 `UpdateCapacityProvider`API 將其設定為較低的值，以獲得更快的回應速度調整。

備用容量

如果您的容量提供者沒有可用於放置任務的容器執行個體，則需要透過即時啟動 Amazon EC2 執行個體來增加 (向外擴展) 叢集容量，然後等待它們啟動，然後才能在其上啟動容器。這可以顯著降低任務啟動率。您在這裡有兩個選擇。

在這種情況下，已啟動備用 Amazon EC2 容量並準備好執行任務將提高有效任務啟動率。您可以使用`Target Capacity`配置來指示您希望在叢集中維持備用容量。例如，若設定`Target Capacity`為 80%，即表示叢集始終需要 20% 的備用容量。此備用容量可讓任何獨立工作立即啟動，確保工作啟動不受限制。這種方法的權衡是保持備用叢集容量的潛在成本增加。

您可以考慮的另一種方法是為您的服務增加預留空間，而不是容量提供者。這表示您可以修改目標追蹤擴展測量結果或服務自動調整`Target Capacity`規模的步驟調整閾值，來增加服務中的複本數目，而不是減少啟 auto 備用容量的組態。請注意，此方法僅對尖峰的工作負載有幫助，但是當您部署新服務並首次從 0 到 N 個任務時不會產生影響。如需相關擴展政策的詳細資訊，請參閱 Amazon Elastic 容器服務開發人員指南中的目標追蹤擴展政策或步驟擴展政策。

控制 Amazon ECS 終止實例

> **Important**
>
> 您必須在 Auto Scaling 群組上開啟 Auto Scaling 執行個體縮減保護，才能使用叢集自動擴展的受管終止保護功能。

受管終止保護可讓叢集 auto 調整規模，以控制要終止的執行個體。當您使用受管終止保護時，Amazon ECS 只會終止沒有任何執行中 Amazon ECS 任務的 EC2 執行個體。系統會忽略使用 `DAEMON` 排程策略的服務執行的任務，而且即使執行個體正在執行這些任務，叢集自動擴展也可以終止執行個體。這是因為叢集中的所有執行個體都在執行這些任務。

Amazon ECS 首先為自動擴展群組中 EC2 執行個體開啟執行個體擴展保護選項。然後，Amazon ECS 會將任務放置在執行個體上。在執行個體上停止所有非常駐程式任務時，Amazon ECS 會啟動縮減程序並關閉 EC2 執行個體的縮減保護。之後，Auto Scaling 群組就能終止執行個體。

Auto Scaling 執行個體縮減保護可控制 Auto Scaling 要終止哪些 EC2 執行個體。在縮減過程中，無法終止已啟用縮減功能的執行個體。如需有關防止 Auto Scaling 執行個體縮減保護的詳細資訊，請參閱 Amazon EC2 Auto Scaling 使用者指南中的使用執行個體縮減保護。

您可以設定`targetCapacity`百分比，以便擁有備用容量。由於 Auto Scaling 群組不需要啟動更多執行個體，因此可協助 future 工作更快地啟動。Amazon ECS 使用目標容量值來管理服務建立的 CloudWatch 指標。Amazon ECS 管理 CloudWatch 指標。「自動縮放」(Auto Scaling) 群組會被視為穩定狀態，因此不需要縮放動作。值可以介於 0-100% 之間。例如，若要設定 Amazon ECS，使其在用於 Amazon ECS 任務時能保持 10% 的可用容量，請將該目標容量值設定為 90%。設定容量提供者的 `targetCapacity` 值時，請考慮以下事項。

- 小於 100% 的 `targetCapacity` 值，表示叢集中需要具備的可用容量數量 (Amazon EC2 執行個體)。可用容量代表沒有執行中的任務。
- 在沒有其他 `binpack` 的情況下，置放限制條件 (如可用區域) 會強制 Amazon ECS 最終為各執行個體執行一個任務，但這可能不是必要行為。

您必須在 Auto Scaling 群組上開啟 Auto Scaling 執行個體縮減保護，才能使用受管終止保護功能。如果您未開啟縮減保護，則開啟受管終止保護可能會產生不良行為。例如，您可能會讓執行個體停留在消耗狀態。如需詳細資訊，請參閱 Amazon EC2 Auto Scaling 使用者指南中的使用執行個體縮減保護。

當您搭配容量提供者使用終止保護時，請勿在與容量提供者相關聯的 Auto Scaling 群組上執行任何手動動作 (例如分離執行個體)。手動動作可能會中斷容量提供者的縮減作業。如果您從 Auto Scaling 群組中分離執行個體，您還需要從 Amazon ECS 叢集中取消註冊已分離的執行個體。

受管向外擴展行為

如果您有使用受管擴展的 Auto Scaling 群組容量提供者，Amazon ECS 會預估要新增至叢集的最佳執行個體數量，並使用該值來判斷要請求的執行個體數量。

Amazon ECS 會依照服務、獨立任務或叢集預設的容量提供者策略，為每個任務選取容量提供者。Amazon ECS 會針對單一容量供應商遵循這些步驟的剩餘步驟。

容量提供者會忽略沒有容量提供者策略的工作。沒有容量提供者策略的等待中任務不會導致任何容量提供者橫向擴展。如果任務或服務設定了啟動類型，則任務或服務無法設定容量提供者策略。

以下詳細說明向外擴展行為。

- 將此容量提供者的所有佈建任務分組，以便每個群組具有相同的精確資源需求。
- 當您在 Auto Scaling 群組中使用多個執行個體類型時，Auto Scaling 群組中的執行個體會依其參數排序。這些參數包含 vCPU、記憶體、彈性網路介面 (ENI)、連接埠和 GPU。為每個參數選取最小和最大的執行個體類型。如需如何選擇執行個體類型的詳細資訊，請參閱Amazon ECS 的亞馬遜 EC2 容器實例。

> **Important**
>
> 如果一組任務的資源需求大於 Auto Scaling 群組中最小執行個體類型，則該任務群組無法使用此容量提供者執行。容量提供者不會擴展 Auto Scaling 群組。任務會維持在 `PROVISIONING` 狀態。
>
> 為了防止任務停留在 `PROVISIONING` 狀態，建議您針對不同的最低資源需求，建立個別的 Auto Scaling 群組和容量提供者。當您執行任務或建立服務時，請僅將容量提供者新增至可

以在 Auto Scaling 群組中最小執行個體類型上執行任務的容量提供者策略。對於其他參數，您可以使用置放條件限制

- 針對每個任務群組，Amazon ECS 會計算執行未放置任務所需的執行個體數目。此計算會使用 binpack 策略。此策略會說明任務的 vCPU、記憶體、彈性網路介面 (ENI)、連接埠和 GPU 需求。其也會說明 Amazon EC2 執行個體的資源可用性。最大執行個體類型的值會被視為所計算執行個體數目的最大值。最小執行個體類型的值會作為保護使用。如果最小的執行個體類型無法執行任務至少一個執行個體，則計算會認為該任務不相容。因此，任務會排除在橫向擴展計算之外。當所有任務都與最小執行個體類型不相容時，叢集自動擴展將會停止，而且 CapacityProviderReservation 值仍為 targetCapacity 值。
- Amazon ECS 會針對下列其中minimumScalingStepSize一種情況發佈相關的CapacityProviderReservation指標。 CloudWatch
  - 計算的執行個體計數上限小於最小縮放步長大小。
  - 計算執行個體計數maximumScalingStepSize或最大計算例證計數的較低值。
- CloudWatch 警示會針對容量提供者使用量CapacityProviderReservation度。當 CapacityProviderReservation 指標大於 targetCapacity 值時，警示也會增加 Auto Scaling 群組的 DesiredCapacity。此targetCapacity值是在叢集自動調整規模啟動階段期間傳送至 CloudWatch 警示的容量提供者設定。

  預設值targetCapacity為 100%。
- Auto Scaling 群組會啟動其他 EC2 執行個體。為了防止過度佈建，Auto Scaling 可確保最近啟動的 EC2 執行個體容量在啟動新執行個體之前穩定下來。Auto Scaling 會檢查是否所有現有的執行個體都已超過 instanceWarmupPeriod (現在時間減去執行個體啟動時間)。對於位於中的執行個體，會封鎖向外延展。instanceWarmupPeriod

  新啟動執行個體的暖機預設秒數為 300 秒。

如需詳細資訊，請參閱 Deep dive on Amazon ECS cluster auto scaling (《深入瞭解 Amazon ECS 叢集自動擴展》)。

擴增注意事項

使用擴增程序時，請注意以下事項：

- 儘管存在多個置放限制條件，我們建議您只使用 distinctInstance 任務置放限制條件。此舉可以防止橫向擴展程序停止，因為您使用的置放限制條件與取樣的執行個體不相容。
- 如果 Auto Scaling 群組使用相同或類似的執行個體類型，則受管擴展效果最佳。

- 當需要橫向擴展程序且目前沒有正在執行的容器執行個體時，Amazon ECS 一開始會一律橫向擴展至兩個執行個體，再執行額外橫向擴展或縮減程序。任何額外的橫向擴展都會等待執行個體暖機期間。關於縮減流程，Amazon ECS 會在橫向擴展程序後等待 15 分鐘，然後隨時開始縮減程序。
- 第二個橫向擴展步驟需要等到 `instanceWarmupPeriod` 過期，這可能會影響整體擴展限制。如果您需要縮短此時間，請確`instanceWarmupPeriod`定 EC2 執行個體足夠大，以便啟動和啟動 Amazon ECS 代理程式 (防止過度佈建)。
- 叢集自動擴展支援容量提供者 Auto Scaling 群組中的啟動組態、啟動範本和多種執行個體類型。您還可以使用屬性型執行個體類型選取範圍，而不使用多種執行個體類型。
- 將 Auto Scaling 群組與隨需執行個體和多個執行個體類型或 Spot 執行個體搭配使用時，較大的執行個體類型的優先級更高，並且不要指定權重。目前不支援指定權重。如需詳細資訊，請參閱 AWS Auto Scaling 使用者指南中的具有多個執行個體類型的 Auto Scaling 群組。
- 然後，如果計算出的執行個體計數上限小於擴展步驟大小下限，或為 `maximumScalingStepSize` 或計算出的執行個體計數上限中的較小者，則 Amazon ECS 會啟動 `minimumScalingStepSize`。
- 如果 Amazon ECS 服務或`run-task`啟動任務，且容量供應商容器執行個體沒有足夠的資源來啟動任務，則 Amazon ECS 會針對每個叢集限制具有此狀態的任務數量，並防止任何任務超過此限制。如需詳細資訊，請參閱 Service Quotas。

受管縮減行為

Amazon ECS 會監控叢集中每個容量提供者的容器執行個體。當容器執行個體未執行任務時，容器執行個體會被視為空白，Amazon ECS 會開始縮減程序。

CloudWatch 縮放警報需要 15 個資料點 (15 分鐘)，才能啟動「自動調整比例」群組的縮放程序。在縮減程序開始後直到 Amazon ECS 需要減少已註冊的容器執行個體數量，Auto Scaling 群組將 `DesireCapacity` 值設定為大於一個執行個體且每分鐘小於 50%。

如果 Amazon ECS 在縮減程序進行期間請求橫向擴展 (`CapacityProviderReservation` 大於 100 時)，縮減程序將停止，並在需要時從頭開始。

以下內容會詳細介紹縮減行為：

1. Amazon ECS 會計算空白容器執行個體的數量。即使常駐程式任務正在執行，則會將容器執行個體視為空白。
2. Amazon ECS 會將 `CapacityProviderReservation` 值設定為介於 0-100 之間的數字，此數字會使用下列公式來表示 Auto Scaling 群組需要相對於其實際大小的比率，並以百分比表示。然

後，Amazon ECS 將指標發佈到 CloudWatch。如需指標計算方式的詳細資訊，請參閱深入了解 Amazon ECS 叢集自動擴展

```
CapacityProviderReservation = (number of instances needed) / (number of running
 instances) x 100
```

3. 指CapacityProviderReservation標會產生 CloudWatch 警示。此警示會更新 Auto Scaling 群組的 DesiredCapacity 值。然後，執行下列其中一個動作：
   - 如果您不使用容量提供者受管終止，Auto Scaling 群組會選擇使用 Auto Scaling 群組終止政策的 EC2 執行個體，並終止執行個體，直到 EC2 執行個體數量達到 DesiredCapacity。之後，從叢集中取消註冊容器執行個體。
   - 如果所有容器執行個體都使用受管終止保護，則 Amazon ECS 會移除空白容器執行個體的縮減保護。之後，Auto Scaling 群組能夠終止 EC2 執行個體。之後，從叢集中取消註冊容器執行個體。

開啟 Amazon ECS 叢集 auto 擴展

您可以使用 AWS CLI 來開啟叢集 auto 調整規模。

開始之前，請先建立 Auto Scaling 群組和容量提供者。如需詳細資訊，請參閱 the section called “EC2 啟動類型的容量提供者”。

若要開啟叢集 auto 調整規模，請將容量提供者與叢集建立關聯，然後開啟叢集 auto 調整規模。

1. 使用 put-cluster-capacity-providers 命令，將一或多個容量提供者關聯到叢集。

   若要新增 AWS Fargate 容量提供者，請在要求中包含FARGATE和FARGATE_SPOT容量提供者。如需詳細資訊，請參閱 AWS CLI 命令參考中的 put-cluster-capacity-providers。

   ```
   aws ecs put-cluster-capacity-providers \
     --cluster ClusterName \
     --capacity-providers CapacityProviderName FARGATE FARGATE_SPOT \
     --default-capacity-provider-strategy capacityProvider=CapacityProvider,weight=1
   ```

   若要為 EC2 啟動類型新增「自動調整」群組，請在請求中包含「Auto Scaling」群組名稱。如需詳細資訊，請參閱 AWS CLI 命令參考中的 put-cluster-capacity-providers。

   ```
   aws ecs put-cluster-capacity-providers \
     --cluster ClusterName \
     --capacity-providers CapacityProviderName \
   ```

```
  --default-capacity-provider-strategy capacityProvider=CapacityProvider,weight=1
```

2. 使用 describe-clusters 命令來確認關聯是否成功。如需詳細資訊，請參閱 AWS CLI 命令參考中的 describe-clusters。

```
aws ecs describe-clusters \
  --cluster ClusterName \
  --include ATTACHMENTS
```

3. 使用 update-capacity-provider 命令，啟用容量提供者的受管自動擴展。如需詳細資訊，請參閱 AWS CLI 命令參考中的 update-capacity-provider。

```
aws ecs update-capacity-provider \
  --capacity-providers CapacityProviderName \
  --auto-scaling-group-provider managedScaling=ENABLED
```

關閉 Amazon ECS 群集 auto 擴展

您可以使用 AWS CLI 來關閉叢集 auto 調整規模。

若要關閉叢集的叢集 auto 調整，您可以取消容量提供者與從叢集開啟受管擴展的關聯，或更新容量提供者以關閉受管擴展。

取消容量提供者的關聯

使用下列步驟解除容量提供者與叢集之間的關聯。

1. 使用 put-cluster-capacity-providers 命令解除 Auto Scaling 群組容量提供者與叢集之間的關聯。叢集可以保持與 AWS Fargate 容量提供者的關聯。如需詳細資訊，請參閱 AWS CLI 命令參考中的 put-cluster-capacity-providers。

```
aws ecs put-cluster-capacity-providers \
  --cluster ClusterName \
  --capacity-providers FARGATE FARGATE_SPOT \
  --default-capacity-provider-strategy '[]'
```

使用 put-cluster-capacity-providers 命令解除 Auto Scaling 群組容量提供者與叢集之間的關聯。如需詳細資訊，請參閱 AWS CLI 命令參考中的 put-cluster-capacity-providers。

```
aws ecs put-cluster-capacity-providers \
  --cluster ClusterName \
  --capacity-providers [] \
  --default-capacity-provider-strategy '[]'
```

2. 使用 describe-clusters 命令來確認是否成功解除關聯。如需詳細資訊，請參閱 AWS CLI 命令參考中的 describe-clusters。

```
aws ecs describe-clusters \
  --cluster ClusterName \
  --include ATTACHMENTS
```

關閉容量提供者的受管擴展

使用下列步驟關閉容量提供者的受管擴展。

- 使用 update-capacity-provider 命令，關閉容量提供者的受管自動擴展。如需詳細資訊，請參閱 AWS CLI 命令參考中的 update-capacity-provider。

```
aws ecs update-capacity-provider \
  --capacity-providers CapacityProviderName \
  --auto-scaling-group-provider managedScaling=DISABLED
```

安全停止在 EC2 執行個體上執行的 Amazon ECS 工作負載

受管執行個體排空有助於正常終止 Amazon EC2 執行個體。這可讓您的工作負載安全停止，並重新安排至非終止執行個體。執行基礎結構維護和更新，而不必擔心工作負載中斷。透過使用受管執行個體排除，您可以簡化需要更換 Amazon EC2 執行個體的基礎設施管理工作流程，同時確保應用程式的彈性和可用性。

Amazon ECS 受管執行個體排空可與 Auto Scaling 群組執行個體替換搭配使用。根據執行個體重新整理和最長執行個體存留期，客戶可以確保其容量符合最新的作業系統和安全性要求。

受管執行個體排出只能與 Amazon ECS 容量提供者搭配使用。您可以在使用 Amazon ECS 主控台或 SDK 建立或更新 Auto Scaling 群組容量提供者時， AWS CLI開啟代管執行個體排除功能。

Amazon ECS 受管執行個體排水涵蓋下列事件。

- Auto Scaling 群組執行個體重新整理-使用執行個體重新整理在 Auto Scaling 群組中執行 Amazon EC2 執行個體的滾動替換，而不是分批手動執行。當您需要取代大量執行個體時，這很有用。執行個體重新整理是透過 Amazon EC2 主控台或 `StartInstanceRefresh` API 啟動的。如果您使用的是受管理終止保護，請務必在撥打電`StartInstanceRefresh`話時選擇`Replace`擴充保護。
- 最大執行個體存留期-您可以在取代 Auto Scaling 群組執行個體時定義最長存留期。這對於根據內部安全性原則或合規性排程替換執行個體很有幫助。
- Auto Scaling 群組可根據擴展政策和排定的擴展動作進行擴展，Auto Scaling 群組支援自動擴展執行個體。透過使用 Auto Scaling 群組做為 Amazon ECS 容量提供者，您可以在群組執行個體中沒有任務執行時擴展 Auto Scaling 群組執行個體。
- Auto Scaling 群組健康狀態檢查-Auto Scaling 群組支援許多運作狀態檢查，以管理運作狀態不良的執
- AWS CloudFormation 堆棧更新-您可以將`UpdatePolicy`屬性添加到 AWS CloudFormation 堆棧中，以便在組更改時執行滾動更新。
- 競價型容量重新平衡-Auto Scaling 群組會根據 Amazon EC2 容量重新平衡通知，嘗試主動更換有中斷風險較高的競價型執行個體。「Auto Scaling 例」群組會在取代作業啟動且狀態良好時終止舊執行個體。Amazon ECS 受管執行個體排出 Spot 執行個體的排水方式與排除非競價型執行個體的方式相同。
- Spot 中斷-Spot 執行個體會在兩分鐘通知的情況下終止。Amazon ECS 代管執行個體排空會使執行個體進入耗盡狀態，以作出回應。

Amazon EC2 Auto Scaling 生命週期掛鉤搭配受管執行個體排水

Auto Scaling 群組生命週期勾點可讓客戶建立由執行個體生命週期中某些事件觸發的解決方案，並在特定事件發生時執行自訂動作。「Auto Scaling」群組允許多達 50 個掛接。可以存在多個終止掛接並並行執 parallel，「Auto Scaling 例」群組會等待所有掛接完成，然後才終止執行個體。

除了 Amazon ECS 管理的勾點終止之外，您也可以設定自己的生命週期終止勾點。生命週期掛鉤有一個`default action`，我們建議將其設定`continue`為預設值，以確保其他勾點 (例如 Amazon ECS 受管勾點) 不會受到自訂勾點的任何錯誤影響。

如果您已設定 Auto Scaling 群組終止生命週期勾點，並且也啟用了 Amazon ECS 受管執行個體排除功能，則會執行兩個生命週期勾點。但是，不能保證相對時間。生命週期掛接有一項`default action`設定，可指定逾時時要採取的動作。如果發生故障，我們建議在自定義鉤子中使用`continue`作為默認結果。這可確保其他掛鉤 (特別是 Amazon ECS 管理的掛鉤) 不會受到自訂生命週期掛鉤中的任何錯誤影響。的替代結果`abandon`會略過所有其他掛接，並且應該避免。如需有

關 Auto Scaling 群組生命週期勾點的詳細資訊，請參閱 Amazon EC2 Auto Scaling 使用者指南中的 Amazon EC2 Auto Scaling 生命週期掛鉤。

工作和代管執行個體排空

Amazon ECS 受管執行個體排空使用容器執行個體中現有的排空功能。容器執行個體排空功能會執行替換，並停止屬於 Amazon ECS 服務的複本任務。RUNNING處於PENDING或狀態的獨立工作 (如呼叫的工作) 不受影響。RunTask您必須等待這些才能手動完成或停止它們。容器執行個體會保持狀DRAINING態，直到所有工作都停止或過了 48 小時為止。在所有複本工作停止之後，協助程式工作是最後一個停止的工作。

代管執行個體排水和受管終止保護

即使管理終止已停用，代管執行個體排除仍可運作。如需有關受管理終止保護的資訊，請參閱控制 Amazon ECS 終止實例。

下表摘要了管理終止與管理排水的不同組合的行為。

管理終止	管理排水	Outcome
已啟用	已啟用	Amazon ECS 可保護正在執行任務的 Amazon EC2 執行個體，避免因擴充事件而終止。任何正在終止的執行個體 (例如沒有設定終止保護的執行個體、已收到 Spot 中斷或執行個體重新整理強制執行個

管理終止	管理排水	Outcome
		體) 都會正常排空。
已停用	已啟用	Amazon ECS 無法保護執行任務的 Amazon EC2 執行個體免於擴充。但是，任何正在終止的例證都會正常排空。

管理終止	管理排水	Outcome
已啟用	已停用	Amazon ECS 可保護正在執行任務的 Amazon EC2 執行個體，避免因擴充事件而終止。但是，執行個體仍然可能會因 Spot 中斷或強制執行個體重新整理而終止，或者如果它們未執行任何工作。Amazon ECS 不會對這些執行個體執行正常排空，而是在替換服務任務停止後啟動。

管理終止	管理排水	Outcome
已停用	已停用	Amazon EC2 執行個體可隨時擴充或終止，即使它們正在執行 Amazon ECS 任務也是如此。Amazon ECS 將在停止之後啟動更換服務任務。

代管執行個體排水和 Spot 執行個體排空

使用競價型執行個體排空，您可以在 Amazon ECS 代理程式`ECS_ENABLE_SPOT_INSTANCE_DRAINING`上設定環境變數，讓 Amazon ECS 能夠將執行個體置於耗盡狀態，以回應兩分鐘的 Spot 中斷。Amazon ECS 受管執行個體排出有助於正在終止的 Amazon EC2 執行個體因許多原因而正常關閉，而不僅僅是 Spot 中斷。例如，您可以使用 Amazon EC2 Auto Scaling 容量重新平衡功能，在提高中斷風險的情況下主動更換 Spot 執行個體，而受管執行個體排出則會執行正常關閉所要取代的 Spot 執行個體。使用代管執行個體排空時，您不需要單獨啟用 Spot 執行個體排空，因此 Auto Scaling 群組`ECS_ENABLE_SPOT_INSTANCE_DRAINING`中的使用者資料是多餘的。若要取得有關 Spot 例證排水的更多資訊，請參閱Spot 執行個體。

代管執行個體排水的運作方式 EventBridge

Amazon ECS 受管執行個體排除事件會發佈到 Amazon EventBridge，而 Amazon ECS 會在您帳戶的預設匯流排中建立 EventBridge 受管規則，以支援受管執行個體排空。您可以將這些事件篩選至其他 AWS 服務 (例如 Lambda、Amazon SNS 和 Amazon SQS)，以監控和疑難排解。

- Amazon EC2 Auto Scaling 會在叫用生命週期勾點 EventBridge 時傳送事件至。
- 現貨中斷通知已發佈至 EventBridge。
- Amazon ECS 會產生錯誤訊息，您可以透過 Amazon ECS 主控台和 API 擷取這些錯誤訊息。

- EventBridge 內建的重試機制做為暫時失敗的緩和措施。

設定 Amazon ECS 容量供應商以安全關閉執行個體

當您使用 Amazon ECS 主控台和建立或更新 Auto Scaling 群組容量提供者時，您可以開啟代管執行個體排空。 AWS CLI

> **Note**
>
> 根據預設，當您建立容量提供者時，代管執行個體排除處於開啟狀態。

下列範例使用建立已啟 AWS CLI 用代管執行個體排空的容量提供者，以及啟用叢集現有容量提供者的代管執行個體耗盡。

建立已啟用代管執行個體排空的容量提供者

若要建立已啟用代管執行個體排空的容量提供者，請使用指`create-capacity-provider`令。將 `managedDraining` 參數設為 `ENABLED`。

```
aws ecs create-capacity-provider \
--name capacity-provider \
--auto-scaling-group-provider '{
  "autoScalingGroupArn": "asg-arn",
  "managedScaling": {
    "status": "ENABLED",
    "targetCapacity": 100,
    "minimumScalingStepSize": 1,
    "maximumScalingStepSize": 1
  },
  "managedDraining": "ENABLED",
  "managedTerminationProtection": "ENABLED",
}'
```

回應：

```
{
    "capacityProvider": {
        "capacityProviderArn": "capacity-provider-arn",
        "name": "capacity-provider",
```

```
        "status": "ACTIVE",
        "autoScalingGroupProvider": {
            "autoScalingGroupArn": "asg-arn",
            "managedScaling": {
                "status": "ENABLED",
                "targetCapacity": 100,
                "minimumScalingStepSize": 1,
                "maximumScalingStepSize": 1
            },
            "managedTerminationProtection": "ENABLED"
            "managedDraining": "ENABLED"
        }
    }
}
```

為叢集的現有容量提供者啟用代管執行個體排除

使用update-capacity-provider命令為叢集的現有容量提供者啟用代管執行個體排除功能。你看到，managedDraining目前說，DISABLED並updateStatus說UPDATE_IN_PROGRESS。

```
aws ecs update-capacity-provider \
--name cp-draining \
--auto-scaling-group-provider '{
  "managedDraining": "ENABLED"
}
```

回應：

```
{
    "capacityProvider": {
        "capacityProviderArn": "cp-draining-arn",
        "name": "cp-draining",
        "status": "ACTIVE",
        "autoScalingGroupProvider": {
            "autoScalingGroupArn": "asg-draining-arn",
            "managedScaling": {
                "status": "ENABLED",
                "targetCapacity": 100,
                "minimumScalingStepSize": 1,
                "maximumScalingStepSize": 1,
                "instanceWarmupPeriod": 300
            },
```

```
            "managedTerminationProtection": "DISABLED",
            "managedDraining": "DISABLED" // before update
        },
        "updateStatus": "UPDATE_IN_PROGRESS", // in progress and need describe again to
 find out the result
        "tags": [
        ]
    }
}
```

使用命describe-clusters令並包括ATTACHMENTS。代管執行個體排除附件為PRECREATED，而整體attachmentsStatus為UPDATING。status

```
aws ecs describe-clusters --clusters cluster-name --include ATTACHMENTS
```

回應：

```
{
    "clusters": [
        {
            ...

            "capacityProviders": [
                "cp-draining"
            ],
            "defaultCapacityProviderStrategy": [],
            "attachments": [
                # new precreated managed draining attachment
                {
                    "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
                    "type": "managed_draining",
                    "status": "PRECREATED",
                    "details": [
                        {
                            "name": "capacityProviderName",
                            "value": "cp-draining"
                        },
                        {
                            "name": "autoScalingLifecycleHookName",
                            "value": "ecs-managed-draining-termination-hook"
                        }
                    ]
```

```
                },

                ...

            ],
            "attachmentsStatus": "UPDATING"
        }
    ],
    "failures": []
}
```

當更新完成後，使用describe-capacity-providers，你看到的managedDraining是現在ENABLED。

```
aws ecs describe-capacity-providers --capacity-providers cp-draining
```

回應：

```
{
    "capacityProviders": [
        {
            "capacityProviderArn": "cp-draining-arn",
            "name": "cp-draining",
            "status": "ACTIVE",
            "autoScalingGroupProvider": {
                "autoScalingGroupArn": "asg-draning-arn",
                "managedScaling": {
                    "status": "ENABLED",
                    "targetCapacity": 100,
                    "minimumScalingStepSize": 1,
                    "maximumScalingStepSize": 1,
                    "instanceWarmupPeriod": 300
                },
                "managedTerminationProtection": "DISABLED",
                "managedDraining": "ENABLED" // successfully update
            },
            "updateStatus": "UPDATE_COMPLETE",
            "tags": []
        }
    ]
}
```

Amazon ECS 受管實例排除故障排除

您可能需要疑難排解代管執行個體耗盡的問題。以下是您在使用時可能遇到的問題和解決方案的示例。

使用 auto 擴展時，執行個體不會在超過執行個體存留期上限後終止

如果您的執行個體在使用 auto 擴展群組時達到並超過最大執行個體生命週期後仍未終止，可能是因為它們受到保護，免於擴充。您可以關閉受管理終止，並允許管理排水處理執行個體回收。

建立 Amazon ECS 叢集的資源，使用 AWS Management Console

了解如何使用建立叢集 auto 調整資源 AWS Management Console。當資源需要名稱時，我們會使用前綴`ConsoleTutorial`來確保它們都具有唯一的名稱，並使其易於查找。

主題

- 必要條件
- 步驟 1：建立 Amazon ECS 叢集。
- 步驟 2：註冊任務定義
- 步驟 3：執行任務
- 步驟 4：驗證
- 步驟 5：清除

必要條件

本教學課程假設已完成下列先決條件：

- 已完成「設定以使用 Amazon ECS。」中的步驟。
- 您的 AWS 使用者具有 亞馬遜 FullAccess IAM 政策範例中指定的必要許可。
- 已建立 Amazon ECS 容器執行個體 IAM 角色。如需詳細資訊，請參閱 Amazon ECS 容器執行個體 IAM 角色。
- 已建立 Amazon ECS 服務連結 IAM 角色。如需詳細資訊，請參閱 使用 Amazon ECS 的服務連結角色。
- 已建立 Auto Scaling 服務連結 IAM 角色。如需詳細資訊，請參閱《Amazon EC2 Auto Scaling 使用者指南》中的 Amazon EC2 Auto Scaling 的服務連結角色。
- 您已建立 VPC 和安全群組。如需詳細資訊，請參閱 the section called “建立 Virtual Private Cloud”。

步驟 1：建立 Amazon ECS 叢集。

使用下列步驟來建立 Amazon ECS 叢集。

Amazon ECS 會代表您建立 Amazon EC2 自動擴展啟動範本和 Auto Scaling 群組，做為 AWS CloudFormation 堆疊的一部分。

1. 開啟主控台，網址為 https://console.aws.amazon.com/ecs/v2。
2. 在導覽窗格中，選擇叢集，然後選擇建立叢集。
3. 在叢集組態下的叢集名稱中，輸入 `ConsoleTutorial-cluster`。
4. 在基礎設施下，清除 AWS Fargate (無伺服器)，然後選取 Amazon EC2 執行個體。接下來，設定作為容量提供者的 Auto Scaling 群組。
    - 在 Auto Scaling 群組 (ASG) 下。選取建立新 ASG，然後提供有關該群組的下列詳細資訊：
        - 在作業系統/架構中，選擇 Amazon Linux 2。
        - 在 EC2 執行個體類型中，選擇 t3.nano。
        - 對於容量，輸入 Auto Scaling 群組中要啟動的最小執行個體數和最大執行個體數。
5. (選用) 若要管理叢集標籤，請展開標籤，然後執行下列其中一項操作：

    [新增標籤] 選擇新增標籤，並執行下列動作：

    - 對於 Key (金鑰)，輸入金鑰名稱。
    - 對於 Value (值)，進入金鑰值。

    [移除標籤] 選擇標籤「金鑰」和「值」右側的移除。
6. 選擇建立。

步驟 2：註冊任務定義

您必須先註冊任務定義，才能在您的叢集上執行任務。任務定義是分在一組的容器清單。以下範例是一種簡單的任務定義，使用 Docker Hub 的 `amazonlinux` 映像，且正好處於睡眠的狀態。如需可用之任務定義參數的詳細資訊，請參閱「Amazon ECS 任務定義」。

1. 開啟主控台，網址為 https://console.aws.amazon.com/ecs/v2。
2. 在導覽窗格中，選擇 Task Definitions (任務定義)。

3. 選擇 Create new task definitio (建立新任務定義)、Create new task definition with JSON (使用 JSON 建立新的任務定義)。
4. 在 JSON 編輯器方塊中貼上下列內容。

```
{
    "family": "ConsoleTutorial-taskdef",
    "containerDefinitions": [
        {
            "name": "sleep",
            "image": "amazonlinux:2",
            "memory": 20,
            "essential": true,
            "command": [
                "sh",
                "-c",
                "sleep infinity"
            ]
        }
    ],
    "requiresCompatibilities": [
        "EC2"
    ]
}
```

5. 選擇建立。

步驟 3：執行任務

註冊帳戶的任務定義後，您就可以在叢集中執行任務。在本教學課程中，您會在 ConsoleTutorial-cluster 叢集中執行 ConsoleTutorial-taskdef 任務定義的五個執行個體。

1. 開啟主控台，網址為 https://console.aws.amazon.com/ecs/v2。
2. 在 [叢集] 頁面上，選擇 [ConsoleTutorial叢集]。
3. 在任務下選擇執行新任務。
4. 在環境區段的運算選項下，選擇容量提供者策略。
5. 在部署組態下，針對應用程式類型選擇任務。
6. 從家庭下拉列表中選擇 ConsoleTutorial-taskdef。
7. 在所需的任務下，輸入 5。
8. 選擇建立。

步驟 4：驗證

在本教學課程的這個階段，您應該擁有一個正在執行五個任務的叢集，以及一個具有容量提供者的 Auto Scaling 群組。容量提供者已啟用 Amazon ECS 受管擴展。

我們可以檢視 CloudWatch 指標、Auto Scaling 群組設定，最後查看 Amazon ECS 叢集任務計數，以確認一切正常運作。

若要檢視叢集的 CloudWatch 指標

1. 開啟主 CloudWatch 控台，網址為 https://console.aws.amazon.com/cloudwatch/。
2. 在螢幕上方的導覽列上，選取區域。
3. 在導覽窗格的指標下方，選擇所有指標。
4. 在所有指標頁面的瀏覽索引標籤下，選擇 AWS/ECS/ManagedScaling。
5. 選擇「CapacityProvider名稱」 ClusterName。
6. 選取對應於的核取方塊ConsoleTutorial-cluster ClusterName。
7. 在圖表化指標標籤下，將期間變更為 30 秒，統計值變更為最大值。

   圖表中顯示的值會顯示容量提供者的目標容量值。此值應從 100 開始，也就是我們設定的目標容量百分比。您應該會看到此值擴展到 200，這會觸發目標追蹤擴展政策的警示。此警示接著會觸發 Auto Scaling 群組水平擴展。

使用下列步驟來檢視 Auto Scaling 群組詳細資訊，以確認水平擴展動作已發生。

確認 Auto Scaling 群組已水平擴展

1. 前往 https://console.aws.amazon.com/ec2/ 開啟 Amazon EC2 主控台。
2. 在螢幕上方的導覽列上，選取區域。
3. 在導覽窗格的 Auto Scaling 下，選擇 Auto Scaling Groups (Auto Scaling 群組)。
4. 選擇在此教學課程中建立的 ConsoleTutorial-cluster Auto Scaling 群組。檢視所需容量下的值，並檢視執行個體管理索引標籤下的執行個體，以確認您的群組是否橫向擴展至兩個執行個體。

使用下列步驟來檢視 Amazon ECS 叢集，以確認 Amazon EC2 執行個體已向叢集註冊，且您的任務已轉換為 RUNNING 狀態。

驗證 Auto Scaling 群組中的執行個體

1. 開啟主控台，網址為 https://console.aws.amazon.com/ecs/v2。
2. 在導覽窗格中，選擇叢集。
3. 在 Clusters (叢集) 頁面上，選擇 `ConsoleTutorial-cluster` 叢集。
4. 在任務標籤上，確認您看到五個處於 `RUNNING` 狀態的任務。

步驟 5：清除

完成此教學課程時，清除與其相關的資源，以免未使用的資源產生費用。不支援刪除容量提供者和任務定義，但是這些資源都沒有相關聯的成本。

清除教學課程資源

1. 開啟主控台，網址為 https://console.aws.amazon.com/ecs/v2。
2. 在導覽窗格中，選擇叢集。
3. 在 [叢集] 頁面上，選擇 [ConsoleTutorial叢集]。
4. 在 ConsoleTutorial-叢集頁面上，選擇 [作業] 索引標籤，然後選擇 [停止] > [全部停止]。
5. 在導覽窗格中，選擇叢集。
6. 在 [叢集] 頁面上，選擇 [ConsoleTutorial叢集]。
7. 在頁面右上角，選擇 [刪除叢集]。
8. 在確認方塊中，輸入刪除ConsoleTutorial叢集，然後選擇刪除。
9. 使用以下步驟刪除 Auto Scaling 群組。
    a. 前往 https://console.aws.amazon.com/ec2/ 開啟 Amazon EC2 主控台。
    b. 在螢幕上方的導覽列上，選取區域。
    c. 在導覽窗格的 Auto Scaling 下，選擇 Auto Scaling Groups (Auto Scaling 群組)。
    d. 選取 `ConsoleTutorial-cluster` Auto Scaling 群組，然後選擇動作。
    e. 在 Actions (動作) 選單中，選擇 Delete (刪除)。在確認方塊中，輸入 delete，然後選擇刪除。

Amazon ECS 的亞馬遜 EC2 容器實例

Amazon ECS 容器執行個體是執行 Amazon ECS 容器代理程式並註冊到叢集的 Amazon EC2 執行個體。當使用 EC2 啟動類型、外部啟動類型或 Auto Scaling 群組容量提供者來透過 Amazon ECS 執行任務時，您的任務會放置在作用中的容器執行個體。您必須負責容器執行個體的管理與維護。

雖然您可以建立自己的 Amazon EC2 執行個體 AMI，以符合在 Amazon ECS 上執行容器化工作負載所需的基本規格，但 Amazon ECS 最佳化 AMI 會由工程師在 Amazon ECS 上預先設定和測試。 AWS 這是讓您快速上手並在 AWS 上執行容器最簡單的方法。

使用主控台建立叢集時，Amazon ECS 會為您的執行個體建立啟動範本，其中包含與所選作業系統相關聯的最新 AMI。

當您使 AWS CloudFormation 用建立叢集時，SSM 參數是 Auto Scaling 群組執行個體之 Amazon EC2 啟動範本的一部分。您可以將範本設定為使用動態 Systems Manager 參數來決定要部署哪些 Amazon ECS 最佳化 AMI。此參數可確保您每次部署堆疊時，都會檢查是否有需要套用至 EC2 執行個體的可用更新。如需如何使用 Systems Manager 參數的範例，請參閱使用者指南中的使用 Amazon ECS 最佳化 Amazon Linux 2023 AMI 建立 Amazon ECS 叢集。AWS CloudFormation

- 擷取 Amazon EC 最佳化的 Linux AMI 中繼資料
- 擷取 Amazon ECS 優化 Bottlerocket AMI 中繼資料
- 擷取 Amazon EC 最佳化視窗 AMI 中繼資料

您可以從與應用程式相容的執行個體類型中選擇。使用容量較大的執行個體，您可以同時啟動更多任務。對於較小的執行個體，您可以以更精細的方式向外擴充，以節省成本。您不需要選擇適合叢集中所有應用程式的單一 Amazon EC2 執行個體類型。您可以改為建立多個 Auto Scaling 群組，其中每個群組都有不同的執行個體類型。然後，您可以為這些群組中的每一個群組建立 Amazon EC2 容量提供者。

請遵循下列準則來決定要使用的例證族群類型和例證類型：

- 排除不符合應用程式特定需求的執行個體類型或執行個體系列。例如，如果您的應用程式需要 GPU，您可以排除任何沒有 GPU 的執行個體類型。
- 考慮需求，包括網路輸送量和儲存。
- 考慮 CPU 和內存。一般而言，CPU 和記憶體必須足夠容納至少一個要執行任務的複本。

Spot 執行個體

相較於隨需執行個體，Spot 容量可大幅節省成本。Spot 容量是指超額容量，其價格明顯低於隨需或預留容量。Spot 容量適用於批次處理和機器學習工作負載，以及開發和預備環境。更一般來說，它適用於任何能容忍暫時停機的工作負載。

了解下列結果，因為 Spot 容量可能無法隨時可用。

- 在需求極高期間，Spot 容量可能無法使用。這可能會導致 Amazon EC2 Spot 執行個體啟動延遲。在這些事件中，Amazon ECS 服務會重試啟動任務，而 Amazon EC2 Auto Scaling 群組也會重試啟動執行個體，直到所需的容量可用為止。Amazon EC2 不會以隨需容量取代 Spot 容量。
- 當容量的整體需求增加時，Spot 執行個體和工作可能只會在兩分鐘的警告下終止。在傳送警告之後，任務應該在執行個體完全終止之前，視需要開始依序關機。這有助於將錯誤的可能性降至最低。如需正常關機的詳細資訊，請參閱使用 ECS 正常關機。

若要協助減少 Spot 容量短缺，請考慮下列建議：

- 使用多個區域和可用區域：Spot 容量會因區域和可用區域而異。您可以透過在多個區域和可用區域執行工作負載來提高 Spot 可用性。如果可能，請在您執行任務和執行個體的區域中，在所有可用區域中指定子網路。
- 使用多個 Amazon EC2 執行個體類型：當您將混合執行個體政策與 Amazon EC2 Auto Scaling 搭配使用時，會在您的 Auto Scaling 群組中啟動多個執行個體類型。這可確保在需要時能滿足 Spot 容量的請求。為了將可靠性最大化並將複雜性最小化，請在混合執行個體政策中使用 CPU 和記憶體數量大致相同的執行個體類型。這些執行個體可以來自不同世代，也可以是相同基礎執行個體類型的變體。請注意，它們可能擁有您可能不需要的其他功能。這類清單的範例可能包括 m4.large、m5.large、m5a.large、m5d.large、m5n.large、m5dn.large 和 m5ad.large。如需詳細資訊，請參閱《Amazon EC2 Auto Scaling 使用者指南》中的具備多個執行個體類型及購買選項的 Auto Scaling 群組。
- 使用容量最佳化的 Spot 配置策略：使用 Amazon EC2 Spot，您可以在容量和成本最佳化配置策略之間進行選擇。如果您在啟動新執行個體時選擇容量最佳化策略，Amazon EC2 Spot 會在所選可用區域中選取具有最高可用性的執行個體類型。這有助於降低執行個體在啟動後立即終止的可能性。

如需如何在容器執行個體上設定 Spot 終止通知的相關資訊，請參閱：

- 將 Amazon ECS Linux 容器執行個體設定為接收競價型執行個體通知
- 設定 Amazon ECS 視窗容器執行個體以接收競價型執行個體通知

Amazon ECS 優化的 Linux AMI

Amazon ECS 提供 Amazon ECS 最佳化的 AMI，這些 AMI 預先設定了執行容器工作負載的需求和建議。建議您為 Amazon EC2 執行個體使用 Amazon ECS 最佳化 Amazon Linux 2023 AMI，除非您的應用程式需要 Amazon EC2 GPU 型執行個體、特定作業系統或在該 AMI 中尚不可用的 Docker 版本。有關 Amazon Linux 2 和 Amazon Linux 2023 執行個體的資訊，請參閱 Amazon Linux 2023 使用者指南中的比較 Amazon Linux 2 和 Amazon Linux 2023。透過最新的 Amazon ECS 最佳化 AMI 啟動您的

容器執行個體時，可確保您接收到最新的安全性更新和容器代理程式版本。如需執行個體啟動方式的詳細資訊，請參閱 啟動 Amazon ECS Linux 容器執行個體。

使用主控台建立叢集時，Amazon ECS 會為您的執行個體建立啟動範本，其中包含與所選作業系統相關聯的最新 AMI。

當您使 AWS CloudFormation 用建立叢集時，SSM 參數是 Auto Scaling 群組執行個體之 Amazon EC2 啟動範本的一部分。您可以將範本設定為使用動態 Systems Manager 參數來決定要部署哪些 Amazon ECS 最佳化 AMI。此參數可確保您每次部署堆疊時，都會檢查是否有需要套用至 EC2 執行個體的可用更新。如需如何使用 Systems Manager 參數的範例，請參閱使用者指南中的使用 Amazon ECS 最佳化 Amazon Linux 2023 AMI 建立 Amazon ECS 叢集。AWS CloudFormation

如果您需要自訂 Amazon ECS 最佳化 AMI，請參閱上的 Amazon ECS 最佳化 AMI 建置收件人。GitHub

Amazon ECS 最佳化 AMI 的 Linux 變體使用 Amazon Linux 2 AMI 作為其基礎。此外，還提供 Amazon Linux 2 AMI 版本備註。如需詳細資訊，請參閱 Amazon Linux 2 版本備註。

我們建議您將 AMI 與 Linux 核心 5.10 搭配使用，因為 Linux 核心 4.14 已 end-of-life 於 2024 年 1 月 10 日達到。

下列 Amazon ECS 最佳化 AMI 的變體適用於您的 Amazon EC2 執行個體。

作業系統	AMI	描述	儲存組態
Amazon Linux 2023	Amazon ECS 最佳化 Amazon Linux 2023 AMI	Amazon Linux 2023 是 Amazon Linux 的下一代. AWS建議在大部分情況下為 Amazon ECS 工作負載啟動 Amazon EC2 執行個體。如需詳細資訊，請參閱 Amazon Linux 2023 使用者指南中的什麼是 Amazon Linux 2023。	根據預設，Amazon ECS 最佳化 Amazon Linux 2023 AMI 附有單個 30 GiB 的根磁碟區。您可以在修改啟動階段的這個 30 GiB 根磁碟區，增加或減少您容器執行個體的可用儲存體。此儲存體用於作業系統和 Docker 映像及中繼資料。
Amazon Linux 2023 (arm64)	Amazon ECS 最佳化 Amazon Linux 2023 (arm64) AMI	以 Amazon Linux 2023 為基礎，建議在啟動 Amazon EC2 執	Amazon ECS 最佳化 Amazon Linux 2023

作業系統	AMI	描述	儲存組態
		行個體 (採用 ARM 型 AWS Graviton/Graviton 2 處理器的執行個體) 時為 Amazon ECS 工作負載使用此 AMI。如需詳細資訊，請參閱 Amazon EC2 使用者指南中的一般用途執行個體。 Amazon ECS 優化的 Amazon Linux 2023 (ARM64) AMI 不附帶預裝。 AWS CLI	AMI 的預設檔案系統為 `xfs`，並且 Docker 使用 `overlay2` 儲存驅動程式。如需詳細資訊，請參閱 Docker 文件中的使用 OverlayFS 儲存體驅動程式。

作業系統	AMI	描述	儲存組態
Amazon Linux 2023 (Neuron)	Amazon Linux 2023 (Neuron)	以 Amazon Linux 2023 為基礎，此 AMI 適用於 Amazon EC2 Inf1、TRN1 或 In2 執行個體。它已預先配置了 AWS 推論和 AWS Trainium 驅動程式，以及 Docker 的 AWS 神經元執行階段，讓 Amazon ECS 上的機器學習推論工作負載更容易執行。如需詳細資訊，請參閱 適用於 AWS 神經元機器學習工作負載的 Amazon ECS 任務定義。 Amazon ECS 優化的 Amazon Linux 2023（神經元）AMI 不附帶預裝。 AWS CLI	

作業系統	AMI	描述	儲存組態
Amazon Linux 2	Amazon ECS 最佳化 Amazon Linux 2 核心 5.10 AMI	以 Amazon Linux 2 為基礎，在啟動 Amazon EC2 執行個體時使用此 AMI，而針對 Amazon ECS 工作負載，則使用 Linux 核心 5.10 而不是核心 4.14。Amazon ECS 最佳化 Amazon Linux 2 核心 5.10 AMI 未預先安裝 AWS CLI 。	依預設，基於 Amazon Linux 2 的 Amazon ECS 最佳化 AMI (Amazon ECS 最佳化 Amazon Linux 2 AMI、Amazon ECS 最佳化 Amazon Linux 2 (arm64) AMI 和 Amazon ECS GPU 最佳化 AMI) 隨附有一個 30-GiB 的根磁碟區。您可以在修改啟動階段的這個 30 GiB 根磁碟區，增加或減少您容器執行個體的可用儲存體。此儲存體用於作業系統和 Docker 映像及中繼資料。 Amazon ECS 最佳化 Amazon Linux 2 AMI 的預設檔案系統為 `xfs`，並且 Docker 使用 `overlay2` 儲存驅動程式。如需詳細資訊，請參閱 Docker 文件中的使用 OverlayFS 儲存體驅動程式。
	Amazon ECS 最佳化 Amazon Linux 2 AMI	這適用於您的 Amazon ECS 工作負載。Amazon ECS 最佳化 Amazon Linux 2 AMI 不提供 AWS CLI 預先安裝。	
Amazon Linux 2 (arm64)	Amazon ECS 最佳化 Amazon Linux 2 核心 5.10 (arm64) AMI	此 AMI 以 Amazon Linux 2 為基礎，適用於您的 Amazon EC2 執行個體，這些執行個體由 ARM 式重力 AWS /引力 2 處理器提供支援，而您希望針對 Amazon ECS 工作負載使用 Linux 核心 5.10 而非 Linux 核心 4.14。如需詳細資訊，請參閱 Amazon EC2 使用者指南中的一般用途執行個體。	

作業系統	AMI	描述	儲存組態
		Amazon ECS 優化的 Amazon Linux 2（arm64）AMI 不附帶預裝。 AWS CLI	
	Amazon ECS 最佳化 Amazon Linux 2 (arm64) AMI	此 AMI 以亞馬遜 Linux 2 為基礎，適用於啟動 Amazon EC2 執行個體，這些執行個體由 ARM 式重力 AWS /重力子 2 處理器提供支援，適用於您的 Amazon ECS 工作負載。 Amazon ECS 優化的 Amazon Linux 2（arm64）AMI 不附帶預裝。 AWS CLI	

作業系統	AMI	描述	儲存組態
Amazon Linux 2 (GPU)	Amazon ECS GPU 優化內核 5.10 AMI	以 Amazon Linux 2 為基礎，建議您在為 Amazon ECS 工作負載啟動具有 Linux 核心 5.10 的亞馬遜 EC2 GPU 型執行個體時使用此 AMI。它配備有已預先設定的 NVIDIA 核心驅動程式和 Docker GPU 執行時間，可讓您在 Amazon ECS 上執行利用 GPU 的工作負載。如需詳細資訊，請參閱 GPU 工作負載的 Amazon ECS 任務定義。	

作業系統	AMI	描述	儲存組態
	Amazon ECS GPU 最佳化 AMI	以 Amazon Linux 2 為基礎，建議您在針對 Amazon ECS 工作負載啟動具有 Linux 核心 4.14 的亞馬遜 EC2 GPU 型執行個體時使用此 AMI。它配備有已預先設定的 NVIDIA 核心驅動程式和 Docker GPU 執行時間，可讓您在 Amazon ECS 上執行利用 GPU 的工作負載。如需詳細資訊，請參閱 GPU 工作負載的 Amazon ECS 任務定義。	

作業系統	AMI	描述	儲存組態
Amazon Linux 2 (Neuron)	Amazon ECS 優化 Amazon Linux 2（神經元）內核 5.10 AMI	以 Amazon Linux 2 為基礎，此 AMI 適用於 Amazon EC2 Inf1、Trn1 或 Inf2 執行個體。它預先配置了 AWS 推論與 Linux 內核 5.10 和 AWS Trainium 驅動程序和碼頭工人的 AWS 神經元運行時，這使得在 Amazon ECS 上運行機器學習推論工作負載更容易。如需詳細資訊，請參閱適用於 AWS 神經元機器學習工作負載的 Amazon ECS 任務定義。Amazon ECS 優化 Amazon Linux 2（神經元）AMI 不附帶預裝。 AWS CLI	

作業系統	AMI	描述	儲存組態
	Amazon ECS 最佳化 Amazon Linux 2 (Neuron) AMI	以 Amazon Linux 2 為基礎，此 AMI 適用於 Amazon EC2 Inf1、Trn1 或 Inf2 執行個體。它已預先配置了 AWS 推論和 AWS Trainium 驅動程式，以及 Docker 的 AWS 神經元執行階段，讓 Amazon ECS 上的機器學習推論工作負載更容易執行。如需詳細資訊，請參閱 適用於 AWS 神經元機器學習工作負載的 Amazon ECS 任務定義。Amazon ECS 優化 Amazon Linux 2（神經元）AMI 不附帶預裝。 AWS CLI	

Amazon ECS 為 Amazon ECS 優化 AMI 的 Linux 變體提供了一個更改日誌。 GitHub如需詳細資訊，請參閱變更記錄。

Amazon ECS 最佳化 AMI 的 Linux 變體使用 Amazon Linux 2 AMI 或 Amazon Linux 2023 AMI 作為其基礎。透過查詢 Systems Manager 參數存放區 API，可以擷取每個變體的 Amazon Linux 2 來源 AMI 名稱或 Amazon Linux 2023 AMI 名稱。如需詳細資訊，請參閱 擷取 Amazon EC 最佳化的 Linux AMI 中繼資料。此外，還提供 Amazon Linux 2 AMI 版本備註。如需詳細資訊，請參閱 Amazon Linux 2 版本備註。此外，還提供 Amazon Linux 2023 版本備註。如需詳細資訊，請參閱 Amazon Linux 2023 版本備註。

下列頁面提供變更的其他資訊：

- 來源 AMI 發行說明 GitHub
- Docker 文件中的 Docker 引擎版本備註

- NVIDIA 文件中的 NVIDIA 驅動程式文件
- Amazon ECS 代理更新日誌 GitHub

  ecs-init 應用程式的原始程式碼，以及用來封裝代理程式的指令碼和組態，現在已成為代理程式儲存庫的一部分。對於舊版本ecs-init和包裝，請參閱 Amazon ECS-init 更改日誌 GitHub

將安全性更新套用至 Amazon ECS 最佳化 AMI

以 Amazon Linux 為基礎的 Amazon ECS 最佳化 AMI 包含的自訂版本。cloud-init Cloud-init是用於在雲端運算環境中啟動 Linux 映像檔，並在啟動執行個體時執行所需動作的套件。根據預設，在 2024 年 6 月 12 日之前發行的所有以 Amazon Linux 為基礎的 Amazon ECS 最佳化 AMI 都會在執行個體啟動時套用所有「重大」和「重要」安全性更新。

從 2024 年 6 月 12 日發行的 Amazon ECS 最佳化 AMI 以 Amazon Linux 2 為基礎，預設行為將不再包含在啟動時更新套件。相反地，我們建議您在發行版本時更新為新的 Amazon ECS 最佳化 AMI。當有可用的安全性更新或基本 AMI 變更時，會發行 Amazon ECS 最佳化的 AMI。這樣可以確保您收到最新的套件版本和安全性更新，而且套件版本在執行個體啟動時不可變。如需擷取最新 Amazon ECS 最佳化 AMI 的詳細資訊，請參閱。擷取 Amazon EC 最佳化的 Linux AMI 中繼資料

我們建議您將環境自動化，以便在可用時更新為新的 AMI。如需可用選項的相關資訊，請參閱 Amazon ECS 透過受管執行個體排出功能，讓 EC2 容量管理更輕鬆。

若要繼續在 AMI 版本上手動套用「重大」和「重要」安全更新，您可以在 Amazon EC2 執行個體上執行下列命令。

```
yum update --security
```

如果您想要在啟動時重新啟用安全更新，可以在啟動 Amazon EC2 執行個體時，將以下行新增至雲端初始化使用者資料的#cloud-config區段。如需詳細資訊，請參閱 Amazon Linux 使用者指南中的在 Amazon Linux 2 上使用雲端初始化。

```
#cloud-config
repo_upgrade: security
```

擷取 Amazon EC 最佳化的 Linux AMI 中繼資料

您可以以程式設計方式擷取 Amazon ECS 最佳化 AMI 中繼資料。中繼資料包括 AMI 名稱、Amazon ECS 容器代理程式版本，以及包含 Docker 版本的 Amazon ECS 執行階段版本。

使用主控台建立叢集時，Amazon ECS 會為您的執行個體建立啟動範本，其中包含與所選作業系統相關聯的最新 AMI。

當您使 AWS CloudFormation 用建立叢集時，SSM 參數是 Auto Scaling 群組執行個體之 Amazon EC2 啟動範本的一部分。您可以將範本設定為使用動態 Systems Manager 參數來決定要部署哪些 Amazon ECS 最佳化 AMI。此參數可確保您每次部署堆疊時，都會檢查是否有需要套用至 EC2 執行個體的可用更新。如需如何使用 Systems Manager 參數的範例，請參閱使用者指南中的使用 Amazon ECS 最佳化 Amazon Linux 2023 AMI 建立 Amazon ECS 叢集。AWS CloudFormation

透過查詢 Systems Manager 參數存放區 API，可以程式設計方式擷取 Amazon ECS 最佳化 AMI 的每個變體的 AMI ID、映像名稱、作業系統、容器代理程式版本、來源映像名稱以及執行時間版本。如需有關 Systems Manager 參數存放區 API 的詳細資訊，請參閱GetParameters和GetParametersByPath。

> **Note**
>
> 您的管理使用者必須擁有以下 IAM 許可，才能擷取 Amazon ECS 最佳化 AMI 中繼資料。已將這些權限新增至 `AmazonECS_FullAccess` IAM 政策。
>
> - SSM：GetParameters
> - SSM：GetParameter
> - SSM：GetParametersByPath

Systems Manager 參數存放區參數格式。

以下是每個 Amazon ECS 最佳化 AMI 變體的參數名稱格式。

Linux Amazon ECS 最佳化 AMI

- Amazon Linux 2023 AMI 中繼資料：

```
/aws/service/ecs/optimized-ami/amazon-linux-2023/<version>
```

- Amazon Linux 2023 (arm64) AMI 中繼資料：

```
/aws/service/ecs/optimized-ami/amazon-linux-2023/arm64/<version>
```

- Amazon Linux 2023 (Neuron) AMI 中繼資料：

```
/aws/service/ecs/optimized-ami/amazon-linux-2023/neuron/<version>
```

- Amazon Linux 2 AMI 中繼資料：

```
/aws/service/ecs/optimized-ami/amazon-linux-2/<version>
```

- Amazon Linux 2 核心 5.10 AMI 中繼資料：

```
/aws/service/ecs/optimized-ami/amazon-linux-2/kernel-5.10/<version>
```

- Amazon Linux 2 (arm64) AMI 中繼資料：

```
/aws/service/ecs/optimized-ami/amazal2023neuronamion-linux-2/arm64/<version>
```

- Amazon Linux 2 核心 5.10 (arm64) AMI 中繼資料：

```
/aws/service/ecs/optimized-ami/amazon-linux-2/kernel-5.10/arm64/<version>
```

- Amazon ECS GPU 優化內核 5.10 AMI 元數據：

```
/aws/service/ecs/optimized-ami/amazon-linux-2/kernel-5.10/gpu/<version>
```

- Amazon Linux 2 (GPU) AMI 中繼資料：

```
/aws/service/ecs/optimized-ami/amazon-linux-2/gpu/<version>
```

- Amazon ECS 優化了 Amazon Linux 2（神經元）內核 5.10 AMI 元數據：

```
/aws/service/ecs/optimized-ami/amazon-linux-2/kernel-5.10/inf/<version>
```

- Amazon Linux 2 (Neuron) AMI 中繼資料：

```
/aws/service/ecs/optimized-ami/amazon-linux-2/inf/<version>
```

以下參數名稱格式使用子參數 `image_id` 來擷取最新的穩定 Amazon ECS 最佳化 Amazon Linux 2 AMI 的映像 ID。

```
/aws/service/ecs/optimized-ami/amazon-linux-2/recommended/image_id
```

以下參數名稱格式透過指定 AMI 名稱來擷取特定 Amazon ECS 最佳化 AMI 版本的中繼資料。

- Amazon ECS 最佳化 Amazon Linux 2 AMI 中繼資料：

```
/aws/service/ecs/optimized-ami/amazon-linux-2/amzn2-ami-ecs-hvm-2.0.20181112-x86_64-ebs
```

> **Note**
>
> 可擷取 Amazon ECS 最佳化 Amazon Linux 2 AMI 的所有版本。只能擷取 Amazon ECS 最佳化 AMI 版本 `amzn-ami-2017.09.l-amazon-ecs-optimized` (Linux) 及更新版本。

範例

下列範例顯示您可擷取每個 Amazon ECS 最佳化 AMI 變體中繼資料的方法。

擷取最新的穩定 Amazon ECS 最佳化 AMI 的中繼資料

您可以使用下 AWS CLI 列命令擷取最新穩定的 Amazon ECS 最 AWS CLI 佳化 AMI。

Linux Amazon ECS 最佳化 AMI

- 對於 Amazon ECS 最佳化 Amazon Linux 2023 AMI：

```
aws ssm get-parameters --names /aws/service/ecs/optimized-ami/amazon-linux-2023/recommended --region us-east-1
```

- 對於 Amazon ECS 最佳化 Amazon Linux 2023 (arm64) AMI：

```
aws ssm get-parameters --names /aws/service/ecs/optimized-ami/amazon-linux-2023/arm64/recommended --region us-east-1
```

- 對於 Amazon ECS 優化的 Amazon Linux 2023（神經元）AMI：

```
aws ssm get-parameters --names /aws/service/ecs/optimized-ami/amazon-linux-2023/neuron/recommended --region us-east-1
```

- 對於 Amazon ECS 最佳化 Amazon Linux 2 核心 5.10 AMI：

```
aws ssm get-parameters --names /aws/service/ecs/optimized-ami/amazon-linux-2/
kernel-5.10/recommended --region us-east-1
```

- 對於 Amazon ECS 最佳化 Amazon Linux 2 AMI：

```
aws ssm get-parameters --names /aws/service/ecs/optimized-ami/amazon-linux-2/
recommended --region us-east-1
```

- 對於 Amazon ECS 最佳化 Amazon Linux 2 核心 5.10 (arm64) AMI：

```
aws ssm get-parameters --names /aws/service/ecs/optimized-ami/amazon-linux-2/
kernel-5.10/arm64/recommended --region us-east-1
```

- 對於 Amazon ECS 最佳化 Amazon Linux 2 (arm64) AMI：

```
aws ssm get-parameters --names /aws/service/ecs/optimized-ami/amazal2023neuronamion-
linux-2/arm64/recommended --region us-east-1
```

- 針對 Amazon ECS GPU 最佳化核心 5.10 AMI：

```
aws ssm get-parameters --names /aws/service/ecs/optimized-ami/amazon-linux-2/
kernel-5.10/gpu/recommended --region us-east-1
```

- 對於 Amazon ECS GPU 最佳化 AMI：

```
aws ssm get-parameters --names /aws/service/ecs/optimized-ami/amazon-linux-2/gpu/
recommended --region us-east-1
```

- 對於 Amazon ECS 優化 Amazon Linux 2（神經元）內核 5.10 AMI：

```
aws ssm get-parameters --names /aws/service/ecs/optimized-ami/amazon-linux-2/
kernel-5.10/inf/recommended --region us-east-1
```

- 對於 Amazon ECS 最佳化 Amazon Linux 2 (Neuron) AMI：

```
aws ssm get-parameters --names /aws/service/ecs/optimized-ami/amazon-linux-2/inf/
recommended --region us-east-1
```

擷取最新建議的 Amazon ECS 最佳化 Amazon Linux 2023 AMI 的映像 ID

您可以使用子參數 `image_id` 來擷取最新建議的 Amazon ECS 最佳化 Amazon Linux 2023 AMI ID 的映像 ID。

```
aws ssm get-parameters --names /aws/service/ecs/optimized-ami/amazon-
linux-2023/recommended/image_id --region us-east-1
```

若只要擷取 `image_id` 值，您可以查詢特定的參數值，例如：

```
aws ssm get-parameters --names /aws/service/ecs/optimized-ami/amazon-linux-2023/
recommended/image_id --region us-east-1 --query "Parameters[0].Value"
```

擷取特定 Amazon ECS 最佳化 Amazon Linux 2 AMI 版本的中繼資料

使用下 AWS CLI 列命令擷取特定 Amazon ECS 最佳化 Amazon Linux AMI 版本 AWS CLI 的中繼資料。將 AMI 名稱替換為要擷取的 Amazon ECS 最佳化 Amazon Linux AMI 的名稱。

```
aws ssm get-parameters --names /aws/service/ecs/optimized-ami/amazon-linux-2/amzn2-ami-
ecs-hvm-2.0.20200928-x86_64-ebs --region us-east-1
```

使用 Systems Manager API 擷取 Amazon ECS 最佳化的 Amazon Linux 2 核心 5.10 AMI 中繼資料 GetParametersByPath

使用下列命令，使用 Systems Manager GetParametersByPath API 擷取 Amazon ECS 最佳化的 Amazon Linux 2 AMI 中繼資料。 AWS CLI

```
aws ssm get-parameters-by-path --path /aws/service/ecs/optimized-ami/amazon-linux-2/
kernel-5.10/ --region us-east-1
```

檢索最新推薦的 Amazon ECS 優化 Amazon Linux 2 內核 5.10 AMI 的圖像 ID

您可以使用子參數擷取最新推薦的 Amazon ECS 最佳化 Amazon Linux 2 核心 5.10 AMI ID 的映像識別碼。`image_id`

```
aws ssm get-parameters --names /aws/service/ecs/optimized-ami/amazon-linux-2/
kernel-5.10/recommended/image_id --region us-east-1
```

若只要擷取 `image_id` 值，您可以查詢特定的參數值，例如：

```
aws ssm get-parameters --names /aws/service/ecs/optimized-ami/amazon-linux-2/
recommended/image_id --region us-east-1 --query "Parameters[0].Value"
```

在模板中使用最新推薦的 Amazon ECS 優化 AMI AWS CloudFormation

透過參考 Systems Manager 參數存放區名稱，可以在 AWS CloudFormation 範本中參考最新建議的 Amazon ECS 最佳化 AMI。

Linux 範例

```
Parameters:kernel-5.10
  LatestECSOptimizedAMI:
    Description: AMI ID
    Type: AWS::SSM::Parameter::Value<AWS::EC2::Image::Id>
    Default: /aws/service/ecs/optimized-ami/amazon-linux-2/kernel-5.10/recommended/
image_id
```

Amazon ECS 最佳化 Linux AMI 建置指令碼

Amazon ECS 已採用開放原始碼的形式提供建置指令碼，可用於建置 Amazon ECS 最佳化 AMI 的 Linux 變體。這些建置指令碼現在可在上使用 GitHub。有關更多信息，請參閱亞馬遜 EC-ami 上的。GitHub

如果您需要自訂 Amazon ECS 最佳化 AMI，請參閱上的 Amazon ECS 最佳化 AMI 建置收件人。GitHub

建置指令碼存放庫包含HashiCorp封裝程式範本和建置指令碼，可產生 Amazon ECS 最佳化 AMI 的每個 Linux 變體。這些指令碼是 Amazon ECS 最佳化 AMI 組建的事實來源，因此您可以依照 GitHub 儲存庫監控 AMI 的變更。例如，您可能希望自己的 AMI 使用的 Docker 版本，與 Amazon ECS 團隊用於官方 AMI 的版本相同。

有關更多信息，請參閱上的 Amazon ECS AMI 存儲庫。 GitHub

建置 Amazon ECS 最佳化 Linux AMI

1. 克隆aws/amazon-ecs-ami GitHub 回購。

   ```
   git clone https://github.com/aws/amazon-ecs-ami.git
   ```

2. 為建立 AMI 時要使用的 AWS 區域新增環境變數。將 us-west-2 值替換為要使用的區域。

```
export REGION=us-west-2
```

3. 提供 Makefile 以用於建置 AMI。在已複製儲存庫的根目錄中，使用以下其中一項與您要建置的 Amazon ECS 最佳化 AMI 的 Linux 變數相對應的命令。

   - Amazon ECS 最佳化 Amazon Linux 2 AMI

     ```
     make al2
     ```

   - Amazon ECS 最佳化 Amazon Linux 2 (arm64) AMI

     ```
     make al2arm
     ```

   - Amazon ECS GPU 最佳化 AMI

     ```
     make al2gpu
     ```

   - Amazon ECS 最佳化 Amazon Linux 2 (Neuron) AMI

     ```
     make al2inf
     ```

   - Amazon ECS 最佳化 Amazon Linux 2023 AMI

     ```
     make al2023
     ```

   - Amazon ECS 最佳化 Amazon Linux 2023 (arm64) AMI

     ```
     make al2023arm
     ```

   - Amazon ECS 最佳化 Amazon Linux 2023 (Neuron) AMI

     ```
     make al2023neu
     ```

Amazon ECS 最佳化 Bottlerocket AMI

Bottlerocket是一個以開放原始碼為Linux基礎的作業系統，專 AWS 為在虛擬機器或裸機主機上執行容器而建置。Amazon ECS 最佳化 Bottlerocket AMI 非常安全，只包括執行容器所需的最少數量套件。這可改善資源使用率、減少安全性受攻擊面，並有助於降低管理負荷。Bottlerocket AMI 也與 Amazon ECS 整合，有助於減少更新叢集中容器執行個體所涉及的營運開銷。

Bottlerocket 在以下方面不同於 Amazon Linux：

- Bottlerocket 不包括套件管理工具，其軟體僅可作為容器執行。Bottlerocket 的更新一律套用，而且可以在單一步驟中復原，以減少發生更新錯誤的可能性。
- 管理 Bottlerocket 主機的主要機制是使用容器排程器。與 Amazon Linux 不同的是，登入個別 Bottlerocket 執行個體的用意只是為了進階偵錯和故障排除目的，而非經常執行的操作。

如需有關 Bottlerocket 的詳細資訊，請參閱 GitHub 上的文件和版本。

有針對內核 6.1 和內核 5.10 的 Amazon ECS 優化 Bottlerocket AMI 的變體。

下列變體使用核心 6.1：

- `aws-ecs-2`
- `aws-ecs-2-nvidia`

下列變體使用核心 5.1.10：

- `aws-ecs-1`
- `aws-ecs-1-nvidia`

  如需有關 `aws-ecs-1-nvidia` 變體的詳細資訊，請參閱宣布 NVIDIA GPU 支援 Amazon ECS 上的 Bottlerocket。

考量事項

搭配使用 Bottlerocket AMI 與 Amazon ECS 時，應考慮以下事項。

- Bottlerocket 支援具有 `x86_64` 與 `arm64` 處理器的 Amazon EC2 執行個體。Bottlerocket AMI 不建議與採用 Inferentia 晶片的 Amazon EC2 執行個體搭配使用。
- Bottlerocket 映像不包含 SSH 伺服器或 Shell。不過，您可以使用 out-of-band 管理工具取得 SSH 管理員存取權並執行啟動載入。如需詳細資訊，請參閱 GitHub 上 bottlerocket README.md 中的這些章節：
  - 探勘
  - 管理員容器

- 在預設情況下，Bottlerocket 具有已啟用的控制容器。此容器會執行 AWS Systems Manager 代理程式，可以讓您在 Amazon EC2 Bottlerocket 執行個體上執行命令或啟動 Shell 工作階段。如需詳細資訊，請參閱《AWS Systems Manager 使用者指南》中的設定工作階段管理員。

- Bottlerocket 已針對容器工作負載進行了最佳化，並著重於安全性。Bottlerocket 不包括套件管理工具，且不可變。如需有關安全性功能和指導方針的資訊，請參閱上的安全性功能和安全性指引 GitHub。
- Bottlerocket AMI 版本 `1.1.0` 或更新版本支援 `awsvpc` 網路模式。
- Bottlerocket AMI 版本 `1.15.0` 或更新版本支援任務定義中的 App Mesh。
- `initProcessEnabled`任務定義參數支持保安機器 AMI 版本或更高版本`1.19.0`。
- Bottlerocket AMI 也不支援下列服務和功能：
  - ECS Anywhere
  - Service Connect
  - 處於加密模式和 `awsvpc` 網路模式的 Amazon EFS
  - Elastic Inference 加速器

擷取 Amazon ECS 優化 Bottlerocket AMI 中繼資料

您可以透過查詢 AWS Systems Manager 參數存放區 API，擷取 Amazon ECS 最佳化 AMI 的 Amazon 機器映像 (AMI) 識別碼。若使用此參數，您無需手動查詢 Amazon ECS 最佳化 AMI ID。如需有關 Systems Manager 參數存放區 API 的詳細資訊，請參閱GetParameter。您使用的使用者必須擁有 `ssm:GetParameter` IAM 許可，才能擷取 Amazon ECS 最佳化 AMI 中繼資料。

`aws-ecs-2`瓶裝機 AMI 變種

您可以使用或通 AWS 區域 過架構檢索最新的穩定`aws-ecs-2`保安密封裝箱 AMI 變體。 AWS CLI AWS Management Console

- AWS CLI— 您可以使用子參數，使用下列 AWS CLI 命令擷取最新推薦的 Amazon ECS 最佳化 Bottlerocket AMI 的映像 ID。`image_id`將 *`region`* 替換為您需要 AMI ID 的區域代碼。如需支援的相關資訊 AWS 區域，請參閱在上尋找 AMI GitHub。若要擷取最新版本以外的版本，請以版本號碼取代 `latest`。
  - 對於 64 位元 (x86_64) 架構：

```
aws ssm get-parameter --region us-east-2 --name "/aws/service/bottlerocket/aws-
ecs-2/x86_64/latest/image_id" --query Parameter.Value --output text
```

- 對於 64 位元 Arm (arm64) 架構：

```
aws ssm get-parameter --region us-east-2 --name "/aws/service/bottlerocket/aws-ecs-2/arm64/latest/image_id" --query Parameter.Value --output text
```

- AWS Management Console：您可以使用 AWS Management Console中的 URL 查詢建議的 Amazon ECS 最佳化 AMI ID。此 URL 會以參數的 ID 值開啟 Amazon EC2 Systems Manager 主控台。在下列 URL 中，將 *region* 替換為您需要 AMI ID 的區域代碼。如需支援的相關資訊 AWS 區域，請參閱在上尋找 AMI GitHub。
  - 對於 64 位元 (x86_64) 架構：

```
https://console.aws.amazon.com/systems-manager/parameters/aws/service/bottlerocket/aws-ecs-2/x86_64/latest/image_id/description?region=region#
```

  - 對於 64 位元 Arm (arm64) 架構：

```
https://console.aws.amazon.com/systems-manager/parameters/aws/service/bottlerocket/aws-ecs-2/arm64/latest/image_id/description?region=region#
```

aws-ecs-2-nvidia瓶裝機 AMI 變種

您可以使用或按區域和架構檢索最新的穩定aws-ecs-2-nvdia瓶裝機器 AMI 變體。 AWS CLI AWS Management Console

- AWS CLI— 您可以使用子參數，使用下列 AWS CLI 命令擷取最新推薦的 Amazon ECS 最佳化 Bottlerocket AMI 的映像 ID。image_id將 *region* 替換為您需要 AMI ID 的區域代碼。如需支援的相關資訊 AWS 區域，請參閱在上尋找 AMI GitHub。若要擷取最新版本以外的版本，請以版本號碼取代 latest。
  - 對於 64 位元 (x86_64) 架構：

```
aws ssm get-parameter --region us-east-1 --name "/aws/service/bottlerocket/aws-ecs-2-nvidia/x86_64/latest/image_id" --query Parameter.Value --output text
```

  - 對於 64 位元 Arm (arm64) 架構：

```
aws ssm get-parameter --region us-east-1 --name "/aws/service/bottlerocket/aws-ecs-2-nvidia/arm64/latest/image_id" --query Parameter.Value --output text
```

- AWS Management Console：您可以使用 AWS Management Console中的 URL 查詢建議的 Amazon ECS 最佳化 AMI ID。此 URL 會以參數的 ID 值開啟 Amazon EC2 Systems Manager 主控台。在下列 URL 中，將 *region* 替換為您需要 AMI ID 的區域代碼。如需支援的相關資訊 AWS 區域，請參閱在上尋找 AMI GitHub。
  - 對於 64 位元 (x86_64) 架構：

```
https://regionconsole.aws.amazon.com/systems-manager/parameters/aws/service/
bottlerocket/aws-ecs-2-nvidia/x86_64/latest/image_id/description?region=region#
```

  - 對於 64 位元 Arm (arm64) 架構：

```
https://regionconsole.aws.amazon.com/systems-manager/parameters/aws/service/
bottlerocket/aws-ecs-2-nvidia/arm64/latest/image_id/description?region=region#
```

aws-ecs-1瓶裝機 AMI 變種

您可以使用或通 AWS 區域 過架構檢索最新的穩定aws-ecs-1保安密封裝箱 AMI 變體。 AWS CLI AWS Management Console

- AWS CLI— 您可以使用子參數，使用下列 AWS CLI 命令擷取最新推薦的 Amazon ECS 最佳化 Bottlerocket AMI 的映像 ID。image_id將 *region* 替換為您需要 AMI ID 的區域代碼。如需支援的相關資訊 AWS 區域，請參閱在上尋找 AMI GitHub。若要擷取最新版本以外的版本，請以版本號碼取代 latest。
  - 對於 64 位元 (x86_64) 架構：

```
aws ssm get-parameter --region us-east-1 --name "/aws/service/bottlerocket/aws-
ecs-1/x86_64/latest/image_id" --query Parameter.Value --output text
```

  - 對於 64 位元 Arm (arm64) 架構：

```
aws ssm get-parameter --region us-east-1 --name "/aws/service/bottlerocket/aws-
ecs-1/arm64/latest/image_id" --query Parameter.Value --output text
```

- AWS Management Console：您可以使用 AWS Management Console中的 URL 查詢建議的 Amazon ECS 最佳化 AMI ID。此 URL 會以參數的 ID 值開啟 Amazon EC2 Systems Manager 主控台。在下列 URL 中，將 *region* 替換為您需要 AMI ID 的區域代碼。如需支援的相關資訊 AWS 區域，請參閱在上尋找 AMI GitHub。
  - 對於 64 位元 (x86_64) 架構：

```
https://region.console.aws.amazon.com/systems-manager/parameters/aws/service/
bottlerocket/aws-ecs-1/x86_64/latest/image_id/description
```

- 對於 64 位元 Arm (arm64) 架構：

```
https://region.console.aws.amazon.com/systems-manager/parameters/aws/service/
bottlerocket/aws-ecs-1/arm64/latest/image_id/description
```

aws-ecs-1-nvidia瓶裝機 AMI 變種

您可以使用或按區域和架構檢索最新的穩定aws-ecs-1-nvdia瓶裝機器 AMI 變體。 AWS CLI AWS Management Console

- AWS CLI— 您可以使用子參數，使用下列 AWS CLI 命令擷取最新推薦的 Amazon ECS 最佳化 Bottlerocket AMI 的映像 ID。image_id將 *region* 替換為您需要 AMI ID 的區域代碼。如需支援的相關資訊 AWS 區域，請參閱在上尋找 AMI GitHub。若要擷取最新版本以外的版本，請以版本號碼取代 latest。
  - 對於 64 位元 (x86_64) 架構：

```
aws ssm get-parameter --region us-east-1 --name "/aws/service/bottlerocket/aws-
ecs-1-nvidia/x86_64/latest/image_id" --query Parameter.Value --output text
```

  - 對於 64 位元 Arm (arm64) 架構：

```
aws ssm get-parameter --region us-east-1 --name "/aws/service/bottlerocket/aws-
ecs-1-nvidia/arm64/latest/image_id" --query Parameter.Value --output text
```

- AWS Management Console：您可以使用 AWS Management Console中的 URL 查詢建議的 Amazon ECS 最佳化 AMI ID。此 URL 會以參數的 ID 值開啟 Amazon EC2 Systems Manager 主控台。在下列 URL 中，將 *region* 替換為您需要 AMI ID 的區域代碼。如需支援的相關資訊 AWS 區域，請參閱在上尋找 AMI GitHub。
  - 對於 64 位元 (x86_64) 架構：

```
https://console.aws.amazon.com/systems-manager/parameters/aws/service/bottlerocket/
aws-ecs-1-nvidia/x86_64/latest/image_id/description?region=region#
```

  - 對於 64 位元 Arm (arm64) 架構：

```
https://console.aws.amazon.com/systems-manager/parameters/aws/service/bottlerocket/
aws-ecs-1-nvidia/arm64/latest/image_id/description?region=region#
```

後續步驟

如需有關如何在 Amazon ECS 上開始使用Bottlerocket作業系統的詳細教學，請參閱在部落格網站上使用 Amazon ECS GitHub Bottlerocket 和 Amazon ECS 入門指南。 AWS

如需如何啟動瓶裝機執行個體的詳細資訊，請參閱 為 Amazon ECS 啟動一個Bottlerocket實例

為 Amazon ECS 啟動一個Bottlerocket實例

您可以啟動執行個Bottlerocket體，以便執行容器工作負載。

您可以使用啟 AWS CLI 動Bottlerocket執行個體。

1. 建立稱為 `userdata.toml` 的檔案。此檔案會用於執行個體使用者資料。以您的叢集名稱取代 *cluster-name*。

```
[settings.ecs]
cluster = "cluster-name"
```

2. 使用 the section called “擷取 Amazon ECS 優化 Bottlerocket AMI 中繼資料” 中包含的其中一個命令來取得 Bottlerocket AMI ID。您會在以下步驟中使用此 ID。
3. 執行下列命令以啟動 Bottlerocket 執行個體。請記得替換以下參數：
   - 將*###*替換為執行個體將在其中啟動的私有或公有子網路的 ID。
   - 將 *bottlerocket_ami* 替換為上一個步驟中取得的 AMI ID。
   - 將 *t3.large* 替換為您要使用的執行個體類型。
   - 將 *region* 替換為區域代碼。

```
aws ec2 run-instances --key-name ecs-bottlerocket-example \
   --subnet-id subnet \
   --image-id bottlerocket_ami \
   --instance-type t3.large \
   --region region \
   --tag-specifications
 'ResourceType=instance,Tags=[{Key=bottlerocket,Value=example}]' \
```

```
    --user-data file://userdata.toml \
    --iam-instance-profile Name=ecsInstanceRole
```

4. 執行下列命令來驗證容器執行個體是否已註冊至叢集。當您執行此命令時，請記得替代下列參數：

   - 將 *cluster* 替代為叢集名稱。
   - 將 *region* 替換為您的區域代碼。

```
aws ecs list-container-instances --cluster cluster-name --region region
```

如需如何在 Amazon ECS 上開始使用Bottlerocket作業系統的詳細逐步解說，請參閱在部落格網站上使用 Amazon ECS GitHub Bottlerocket和 Amazon EC S 入門指南。 AWS

Amazon ECS Linux 容器執行個體管理

當您將 EC2 執行個體用於 Amazon ECS 工作負載時，您必須負責維護執行個體

管理程序

- 啟動 Amazon ECS Linux 容器執行個體
- 引導 Amazon ECS Linux 容器執行個體以傳遞資料
- 將 Amazon ECS Linux 容器執行個體設定為接收競價型執行個體通知
- 當您啟動 Amazon ECS Linux 容器執行個體時執行指令碼
- 增加 Amazon ECS Linux 容器實例網絡界面
- 保留 Amazon ECS Linux 容器執行個體記憶體
- 使用遠端管理 Amazon ECS 容器執行個體 AWS Systems Manager
- 針對 Amazon ECS 容器執行個體使用 HTTP 代理伺服器
- 為您的 Amazon ECS Auto Scaling 群組設定預先初始化的執行個體
- 更新 Amazon ECS 容器代理程式

每個 Amazon ECS 容器代理程式版本都支援不同的功能集，並提供先前版本的錯誤修復。可能的話，我們建議您一律使用最新版本的 Amazon ECS 容器代理程式。若要將您的容器代理更新到最新版本，請參閱「更新 Amazon ECS 容器代理程式」。

若要查看每個代理版本中包含哪些功能和強化功能，請參閱 https://github.com/aws/amazon-ecs-agent/releases。

> **Important**
>
> 可靠指標的最低 Docker 版本是 Docker 版本 `v20.10.13` 及更新版本，該版本隨附於 Amazon ECS 最佳化 AMI `20220607` 及更新版本中。
> Amazon ECS 代理程式版本 `1.20.0` 和更新版本已不支援 `1.9.0` 之前的 Docker 版本。

啟動 Amazon ECS Linux 容器執行個體

您可以使用 Amazon EC2 主控台建立亞馬遜 ECS 容器執行個體。

您可以透過各種方法啟動執行個體，包括 Amazon EC2 主控 AWS CLI台和開發套件。如需啟動執行個體的其他方法的詳細資訊，請參閱 Amazon EC2 使用者指南中的啟動執行個體。

如需有關啟動精靈的詳細資訊，請參閱 Amazon EC2 使用者指南中的使用新啟動執行個體精靈啟動執行個體。

開始之前，請完成 設定以使用 Amazon ECS。 中的步驟。

您可以使用新的 Amazon EC2 精靈啟動執行個體。啟動執行個體精靈會指定啟動執行個體所需的所有啟動參數。

執行個體組態的參數

- 程序
- 名稱和標籤
- 應用程式和作業系統映像 (Amazon Machine Image)
- 執行個體類型
- 金鑰對 (登入)
- Network settings (網路設定)
- 設定儲存
- 進階詳細資訊

程序

開始之前，請完成 設定以使用 Amazon ECS。 中的步驟。

1. 前往 https://console.aws.amazon.com/ec2/ 開啟 Amazon EC2 主控台。

2. 在畫面頂端的導覽列中，會顯示目前的「 AWS 區域」(例如，美國東部 (俄亥俄))。選取要在其中啟動執行個體的區域。
3. 從 Amazon EC2 主控台儀表板選擇 Launch Instance (啟動執行個體)。

名稱和標籤

執行個體名稱是一個標籤，其中鍵是 Name (名稱)，而值是您指定的名稱。您可以標記執行個體、磁碟區和彈性圖形。對於 Spot 執行個體，您只能標記 Spot 執行個體請求。

指定執行個體名稱和其他標籤是選用的。

- 對於 Name (名稱)，輸入執行個體的描述性名稱。如果您未指定名稱，則可以透過其 ID 來標識執行個體，該 ID 將在您啟動執行個體時自動產生。
- 若要新增其他標籤，請選擇 Add additional tags (新增其他標籤)。選取 Add tag (新增標籤)，然後輸入鍵和值，然後選取要標記的資源類型。為每個要新增的其他標籤重新選擇 Add tag (新增標籤)。

應用程式和作業系統映像 (Amazon Machine Image)

Amazon Machine Image (AMI) 包含建立執行個體所需的資訊。例如，AMI 可能包含作為 Web 伺服器所必需的軟體，例如 Apache 和您的網站。

使用搜尋列尋找由發佈的適合 Amazon ECS 最佳化 AMI。 AWS

1. 在 Search (搜尋) 列中輸入以下其中一個詞語。
   - **`ami-ecs`**
   - Amazon ECS 最佳化 AMI 的 Value (值)。

     如需最新的 Amazon ECS 最佳化 AMI 及其值，請參閱《Linux Amazon ECS 最佳化 AMI》。
2. 按 Enter。
3. 在 Choose an Amazon Machine Image (AMI) (選擇 Amazon Machine Image (AMI)) 頁面上，選取 AWS Marketplace AMIs 索引標籤。
4. 在左側的 Refine results (細化結果) 窗格中，選取 Amazon Web Services 作為 Publisher (發佈者)。
5. 在您要使用的 AMI 的列上選擇 Select (選取)。

   或者，選擇右上方的 Cancel (取消)，以返回啟動執行個體精靈，而不選擇 AMI。將會選取預設 AMI。確保 AMI 符合《Linux 執行個體》中所示之要求。

執行個體類型

執行個體類型定義執行個體的硬體組態和大小。較大的執行個體類型具有較多的 CPU 和記憶體。如需詳細資訊，請參閱執行個體類型。

- 針對 Instance type (執行個體類型)，選取執行個體的執行個體類型。

  您選取的執行個體類型，會決定您任務執行所在的可用資源。

金鑰對 (登入)

針對 Key pair name (金鑰對名稱)，選擇現有的金鑰對，或選擇 Create new key pair (建立新的金鑰對) 以建立新的金鑰對。

> **Important**
>
> 如果您選擇 Proceed without key pair (Not recommended) (繼續而不使用金鑰對 (不建議)) 選項，您將無法連線到執行個體，除非您選擇已設定為允許使用者透過其他方式登入的 AMI。

Network settings (網路設定)

視需要設定網路設定。

- Networking platform (聯網平台)：選擇 Virtual Private Cloud (VPC) (虛擬私有雲端 (VPC))，然後在 Network interfaces (網路介面) 區段指定子網路。
- VPC：選取要在其中建立安全群組的現有 VPC。
- Subnet (子網)：您可以在與可用區域、Local Zone、Wavelength 區域或 Outpost 相關聯的子網中啟動執行個體。

  若要在可用區域中啟動執行個體，請選取要在當中啟動執行個體的子網。若要建立新的子網，請選擇 Create new subnet (建立新的子網)，以前往 Amazon VPC 主控台。完成後，請返回啟動執行個體精靈並選擇 Refresh (重新整理) 圖示，載入清單中的子網。

  在 Local Zone 中啟動執行個體，選取您在 Local Zone 中建立的子網。

  若要在 Outpost 中啟動執行個體，請在與 Outpost 相關聯的 VPC 中選取子網。
- Auto-assign Public IP (自動指派公有 IP)：如果您的執行個體應從網際網路存取，請確認 Auto-assign Public IP (自動指派公有 IP) 欄位設為 Enable (啟用)。如果不是，請將此欄位設為 Disable (停用)。

> **Note**
>
> 容器執行個體需要存取，才可以與 Amazon ECS 服務端點通訊。可透過介面 VPC 端點或透過具備公有 IP 位址的容器執行個體來實現。
> 如需介面 VPC 端點的詳細資訊，請參閱 Amazon ECS 介面 VPC 端點 (AWS PrivateLink)
> 如果您沒有設定介面 VPC 端點，且容器執行個體沒有公有 IP 位址，則它們必須使用網路位址轉譯 (NAT) 來提供此存取。如需詳細資訊，請參閱《Amazon VPC 使用者指南》中的 NAT 閘道和本指南中的 針對 Amazon ECS 容器執行個體使用 HTTP 代理伺服器。

- Firewall (security groups) (防火牆 (安全群組))：使用安全群組定義容器執行個體的防火牆規則。這些規則指定傳遞至容器執行個體的傳入網路流量。所有其他流量都會遭到忽略。
  - 若要選取現有的安全群組，請選擇 Select existing security group (選取現有安全群組)，然後從您在 設定以使用 Amazon ECS。 建立的安全群組選取您的安全群組。

設定儲存

您選取的 AMI 包含一或多個儲存體磁碟區，包括根磁碟區。您可以指定要連接至執行個體的其他磁碟區。

您可以使用 Simple (簡單) 檢視。

- Storage type (儲存類型)：為您的容器執行個體設定儲存。

  如果您使用的是 Amazon ECS 最佳化 Amazon Linux 2 AMI，則您的執行個體已設定單一 30 GiB 磁碟區 (在作業系統和 Docker 之間共用)。

  如果您使用的是 Amazon ECS 最佳化 AMI，則您的執行個體已設定兩個磁碟區。Root (根目錄) 磁碟區供作業系統使用，而第二個 Amazon EBS 磁碟區 (連接到 `/dev/xvdcz`) 則供 Docker 使用。

  您可以選擇性地增加或減少您執行個體的磁碟區大小，使其符合您的應用程式需求。

進階詳細資訊

針對 Advanced Details (進階詳細資訊)，展開此區段來檢視欄位，指定執行個體的其他參數。

- Purchasing option (購買選項)：選擇 Request Spot instances (請求 Spot 執行個體) 以請求 Spot 執行個體。您也需要設定與 Spot 執行個體相關的其他欄位。如需詳細資訊，請參閱「Spot 執行個體要求」。

> **Note**
>
> 如果您使用的是 Spot 執行個體，而且看到 `Not available` 訊息，您可能需要選擇不同的執行個體類型。

.

- IAM instance profile (IAM 執行個體設定檔)：選取您的容器執行個體 IAM 角色。這通常命名為 `ecsInstanceRole`。

> **Important**
>
> 如果您未使用適當的 IAM 許可啟動容器執行個體，Amazon ECS 代理程式就無法連線到叢集。如需詳細資訊，請參閱 Amazon ECS 容器執行個體 IAM 角色。

- (選用) User data (使用者資料)：利用使用者資料設定您的 Amazon ECS 容器執行個體，例如 Amazon ECS 容器代理程式組態中的代理程式環境變數。Amazon EC2 使用者資料指令碼僅於執行個體初次啟動時執行一次。以下是使用者資料用途的常見範例：
  - 您的容器執行個體預設會在您的預設叢集中啟動。若要在非預設的叢集中啟動，請選擇 Advanced Details (進階詳細資訊) 清單。然後，將以下指令碼貼入 User data (使用者資料) 欄位，以您的叢集名稱取代 *`your_cluster_name`*。

```
#!/bin/bash
echo ECS_CLUSTER=your_cluster_name >> /etc/ecs/ecs.config
```

  - 如果您在 Amazon S3 中有一個 `ecs.config` 檔案，並已為您的容器執行個體角色啟用 Amazon S3 唯讀存取，請選擇 Advanced Details (進階詳細資訊) 清單。然後，將以下腳本粘貼到「用戶數據」字段中，將 *`your_bucket_name`* 替換為存儲桶的名稱，以便在啟動時安裝 AWS CLI 並寫入配置文件。

> **Note**
>
> 如需此組態的詳細資訊，請參閱「在 Amazon S3 中存儲亞馬遜 ECS 容器實例配置」。

```
#!/bin/bash
yum install -y aws-cli
```

```
aws s3 cp s3://your_bucket_name/ecs.config /etc/ecs/ecs.config
```

- 使用 ECS_CONTAINER_INSTANCE_TAGS 組態參數為您的容器執行個體指定標籤。這會建立只與 Amazon ECS 相關聯的標籤，所以無法使用 Amazon EC2 API 列出這些標籤。

  > **Important**
  >
  > 如果您使用 Amazon EC2 Auto Scaling 群組啟動容器執行個體，則應使用 ECS_CONTAINER_INSTANCE_TAGS 代理程式組態參數來新增標籤。這是由於標籤新增到了使用 Auto Scaling 群組啟動的 Amazon EC2 執行個體。

  ```
  #!/bin/bash
  cat <<'EOF' >> /etc/ecs/ecs.config
  ECS_CLUSTER=your_cluster_name
  ECS_CONTAINER_INSTANCE_TAGS={"tag_key": "tag_value"}
  EOF
  ```

- 為您的容器執行個體指定標籤，然後使用 ECS_CONTAINER_INSTANCE_PROPAGATE_TAGS_FROM 組態參數將它們從 Amazon EC2 傳播到 Amazon ECS

  以下的使用者資料指令碼範例會傳播與容器執行個體相關聯的標籤，還會向名為 your_cluster_name 的叢集註冊容器執行個體：

  ```
  #!/bin/bash
  cat <<'EOF' >> /etc/ecs/ecs.config
  ECS_CLUSTER=your_cluster_name
  ECS_CONTAINER_INSTANCE_PROPAGATE_TAGS_FROM=ec2_instance
  EOF
  ```

如需詳細資訊，請參閱 引導 Amazon ECS Linux 容器執行個體以傳遞資料。

引導 Amazon ECS Linux 容器執行個體以傳遞資料

當您啟動 Amazon EC2 執行個體時，您可以將使用者資料傳遞至 EC2 執行個體。此資料可用來執行常見的自動化組態任務，甚至在執行個體啟動時，執行指令碼。對於 Amazon ECS，使用者資料的最常用案例是將組態資訊傳送到 Docker 常駐程式和 Amazon ECS 容器代理程式。

您可以將多種類型的使用者資料傳遞給 Amazon EC2，包含雲端 boothook、shell 指令碼和 cloud-init 指令。如需這些和其他格式類型的詳細資訊，請參閱 Cloud-Init 文件。

若要在使用 Amazon EC2 啟動精靈時傳遞使用者資料，請參閱啟動 Amazon ECS Linux 容器執行個體。

您可以將容器執行個體設定為在容器代理程式組態或 Docker 精靈組態中傳遞資料。

Amazon ECS 容器代理程式

Amazon ECS 最佳化 AMI 的 Linux 變體會在容器代理程式啟動時，於 /etc/ecs/ecs.config 檔案中尋找代理程式組態資料。您可以使用 Amazon EC2 使用者資料在啟動時指定此組態資料。如需可用 Amazon ECS 容器代理程式組態變數的詳細資訊，請參閱 Amazon ECS 容器代理程式組態。

若只要設定單一代理程式組態變數 (例如叢集名稱)，請使用 echo 將變數複製至組態檔案：

```
#!/bin/bash
echo "ECS_CLUSTER=MyCluster" >> /etc/ecs/ecs.config
```

如有多個變數要寫入 /etc/ecs/ecs.config，請使用以下 heredoc 格式。此格式會將開頭為 cat 和 EOF 之行間的所有項目寫入組態檔案。

```
#!/bin/bash
cat <<'EOF' >> /etc/ecs/ecs.config
ECS_CLUSTER=MyCluster
ECS_ENGINE_AUTH_TYPE=docker
ECS_ENGINE_AUTH_DATA={"https://index.docker.io/v1/":
{"username":"my_name","password":"my_password","email":"email@example.com"}}
ECS_LOGLEVEL=debug
ECS_WARM_POOLS_CHECK=true
EOF
```

若要設定自訂執行個體屬性，請設定 ECS_INSTANCE_ATTRIBUTES 環境變數。

```
#!/bin/bash
cat <<'EOF' >> ecs.config
ECS_INSTANCE_ATTRIBUTES={"envtype":"prod"}
EOF
```

Docker 常駐程式

您可以使用 Amazon EC2 使用者資料指定 Docker 常駐程式組態資訊。如需組態選項的詳細資訊，請參閱 Docker 常駐程式文件。

在下例中，自訂選項會新增到 Docker 常駐程式組態檔案，/etc/docker/daemon.json，然後在執行個體啟動時在使用者資料中指定它。

```
#!/bin/bash
cat <<EOF >/etc/docker/daemon.json
{"debug": true}
EOF
systemctl restart docker --no-block
```

在下例中，自訂選項會新增到 Docker 常駐程式組態檔案，/etc/docker/daemon.json，然後在執行個體啟動時在使用者資料中指定它。此範例顯示如何停用 Docker 常駐程式組態檔案中的 Docker 代理。

```
#!/bin/bash
cat <<EOF >/etc/docker/daemon.json
{"userland-proxy": false}
EOF
systemctl restart docker --no-block
```

將 Amazon ECS Linux 容器執行個體設定為接收競價型執行個體通知

當 Spot 價格超過請求的最高價或容量不再可用時，Amazon EC2 會終止、停止或休眠您的 Spot 執行個體。Amazon EC2 會針對終止和停止動作提前兩分鐘發出 Spot 執行個體中斷通知。其不會針對休眠動作提前兩分鐘發出通知。如果執行個體已開啟 Amazon ECS 競價型執行個體排除功能，Amazon ECS 會收到競價型執行個體中斷通知，並將執行個體置於狀態。DRAINING

> **Important**
>
> 當透過 Auto Scaling 容量重新平衡移除執行個體時，Amazon ECS 不會收到 Amazon EC2 發出的通知。如需詳細資訊，請參閱 Amazon EC2 Auto Scaling 容量重新平衡。

將容器執行個體設定為 DRAINING 時，Amazon ECS 會避免在容器執行個體中放置新的任務排程。PENDING 狀態下即將耗盡的容器執行個體服務任務會立即停止。如果叢集有可用的容器執行個體，則會在這些容器執行個體上啟動替代服務任務。

依預設關閉定點例證排水。

您可以在啟動執行個體時開啟 Spot 執行個體排空。將下列指令碼新增至 [使用者資料] 欄位。取代*MyCluster*為要向其註冊容器執行個體的叢集名稱。

```
#!/bin/bash
cat <<'EOF' >> /etc/ecs/ecs.config
ECS_CLUSTER=MyCluster
ECS_ENABLE_SPOT_INSTANCE_DRAINING=true
EOF
```

如需詳細資訊，請參閱 啟動 Amazon ECS Linux 容器執行個體。

針對現有的容器執行個體開啟 Spot 執行個體耗盡

1. 透過 SSH 連接到 Spot 執行個體。
2. 編輯 `/etc/ecs/ecs.config` 檔案並新增以下內容：

   ```
   ECS_ENABLE_SPOT_INSTANCE_DRAINING=true
   ```

3. 重新啟動 `ecs` 服務。

   - 對於 Amazon ECS 最佳化 Amazon Linux 2 AMI：

     ```
     sudo systemctl restart ecs
     ```

4. (選用) 您可以驗證代理已在執行中，並透過查詢代理自我檢查 API 操作，查看您新的容器執行個體的一些資訊。如需詳細資訊，請參閱 the section called “容器內省”。

   ```
   curl http://localhost:51678/v1/metadata
   ```

當您啟動 Amazon ECS Linux 容器執行個體時執行指令碼

您可能需要在每個容器執行個體上執行特定的容器，以處理作業或安全性問題，例如監視、安全性、指標、服務探索或記錄。

若要執行此作業，您可以將您的容器執行個體設定為在啟動時或在某些 init 系統中 (例如 Upstart 或 systemd)，以使用者資料指令碼呼叫 docker run 命令。雖然此方法可行，但有一些缺點，因為 Amazon ECS 對容器一無所知，也無法監控 CPU、記憶體、連接埠或任何其他使用的資源。為確保

Amazon ECS 可正確說明所有任務資源，請為在您容器執行個體上執行的容器建立任務定義。然後，使用 Amazon ECS，利用 Amazon EC2 使用者資料在啟動時放置任務。

下列程序中的 Amazon EC2 使用者資料指令碼會使用 Amazon ECS 自我檢查 API 來識別容器執行個體。然後，它會使用 AWS CLI 和start-task命令在啟動期間自行執行指定的工作。

在容器執行個體啟動階段啟動任務

1. 修改您的 `ecsInstanceRole` IAM 角色，新增 `StartTask` API 操作的許可。如需更多資訊，請參閱 AWS Identity and Access Management IAM 使用者指南中的修改角色。
2. 使用 Amazon ECS 優化的 Amazon Linux 2 AMI 啟動一個或多個容器執行個體。啟動新的容器執行個體，並在 EC2 使用者資料中使用下列範例指令碼。將 *your_cluster_name* 取代為要註冊的容器執行個體的叢集，並將 *my_task_def* 取代為要在啟動時在執行個體上執行的任務定義。

   如需詳細資訊，請參閱 啟動 Amazon ECS Linux 容器執行個體。

   > **Note**
   >
   > 以下的 MIME 多分段內容使用 shell 指令碼來設定組態值並安裝套件。在 ecs 服務已執行，並可使用自我檢查 API 之後，還會使用 systemd 工作啟動任務。

```
Content-Type: multipart/mixed; boundary="==BOUNDARY=="
MIME-Version: 1.0

--==BOUNDARY==
Content-Type: text/x-shellscript; charset="us-ascii"

#!/bin/bash
# Specify the cluster that the container instance should register into
cluster=your_cluster_name

# Write the cluster configuration variable to the ecs.config file
# (add any other configuration variables here also)
echo ECS_CLUSTER=$cluster >> /etc/ecs/ecs.config

START_TASK_SCRIPT_FILE="/etc/ecs/ecs-start-task.sh"
cat <<- 'EOF' > ${START_TASK_SCRIPT_FILE}
 exec 2>>/var/log/ecs/ecs-start-task.log
 set -x
```

```
 # Install prerequisite tools
 yum install -y jq aws-cli

 # Wait for the ECS service to be responsive
 until curl -s http://localhost:51678/v1/metadata
 do
  sleep 1
 done

 # Grab the container instance ARN and AWS Region from instance metadata
 instance_arn=$(curl -s http://localhost:51678/v1/metadata | jq -r '.
 | .ContainerInstanceArn' | awk -F/ '{print $NF}' )
 cluster=$(curl -s http://localhost:51678/v1/metadata | jq -r '. | .Cluster' | awk
 -F/ '{print $NF}' )
 region=$(curl -s http://localhost:51678/v1/metadata | jq -r '.
 | .ContainerInstanceArn' | awk -F: '{print $4}')

 # Specify the task definition to run at launch
 task_definition=my_task_def

 # Run the AWS CLI start-task command to start your task on this container instance
 aws ecs start-task --cluster $cluster --task-definition $task_definition --
container-instances $instance_arn --started-by $instance_arn --region $region
EOF

# Write systemd unit file
UNIT="ecs-start-task.service"
cat <<- EOF > /etc/systemd/system/${UNIT}
      [Unit]
      Description=ECS Start Task
      Requires=ecs.service
      After=ecs.service

      [Service]
      Restart=on-failure
      RestartSec=30
      ExecStart=/usr/bin/bash ${START_TASK_SCRIPT_FILE}

      [Install]
      WantedBy=default.target
EOF

# Enable our ecs.service dependent service with `--no-block` to prevent systemd
 deadlock
```

```
# See https://github.com/aws/amazon-ecs-agent/issues/1707
systemctl enable --now --no-block "${UNIT}"
--==BOUNDARY==--
```

3. 確認您的容器執行個體在正確的叢集中啟動，而且您的任務已啟動。

   a. 開啟主控台，網址為 https://console.aws.amazon.com/ecs/v2。

   b. 從導覽列選擇您叢集所在的區域。

   c. 在導覽窗格中選擇 Clusters (叢集)，並選取託管您容器執行個體的叢集。

   d. 在叢集頁面上，選擇任務，然後選擇您的任務。

      您啟動的每個容器執行個體都應該有任務在正在其中執行。

      如果您沒有看到您的任務，您可以使用 SSH 登入您的容器執行個體，檢查 `/var/log/ecs/ecs-start-task.log` 檔案的偵錯資訊。

增加 Amazon ECS Linux 容器實例網絡界面

> **Note**
>
> 此功能不適用於 Fargate。

使用awsvpc網路模式的每個 Amazon ECS 任務都會收到自己的 elastic network interface (ENI)，該界面會連接到託管它的容器執行個體。Amazon EC2 執行個體可連接的網路介面數量有預設限制，主要網路介面計算在內。例如，根據預設，`c5.large` 執行個體最多可連接三個 ENI。執行個體的主要網路界面計算在內，所以您可以再連接兩個 ENI 到執行個體。由於使用awsvpc網路模式的每項工作都需要ENI，因此您通常只能在此執行個體類型上執行兩項此類工作。

Amazon ECS 支援使用支援的 Amazon EC2 執行個體類型，啟動ENI密度更高的容器執行個體。當您使用這些執行個體類型並開啟awsvpcTrunking帳戶設定時，新啟動的容器執行個體會提供額外的ENI。此組態可讓您在每個容器執行個體中安排更多任務。如需有關awsvpcTrunking帳戶設定的資訊，請參閱透過帳戶設定存取 Amazon ECS 功能。

例如，增加了 12 awsvpcTrunking 個c5.large執行個體的ENI限制。容器執行個體會有主要網路介面，而 Amazon ECS 會建立「幹線」網路介面，並將它連接到容器執行個體。因此，此組態可讓您在容器執行個體中啟動十項任務，而不是目前的兩項任務。

幹線網路介面由 Amazon ECS 全受管，當您在叢集中終止或取消註冊您的容器執行個體時，會將其刪除。如需詳細資訊，請參閱 適用於 EC2 啟動類型的 Amazon ECS 任務聯網選項。

考量事項

使用ENI幹線功能時，請考慮下列事項。

- 只有 Amazon ECS 最佳化 AMI 的 Linux 變體，或其他具有容器代理程式版本`1.28.1`或更新版本和 ecs-init 套件版本`1.28.1-2`或更新版本的 Amazon Linux 版本，才能支援增加的限制。ENI如果您使用 Amazon ECS 最佳化 AMI 的最新 Linux 變體，將會滿足這些要求。目前不支援 Windows 容器。
- 只有在啟用後啟動的新 Amazon EC2 執行個體才`awsvpcTrunking`會收到增加的ENI限制和中繼網路界面。無論採取何種動作，先前啟動的執行個體都不會收到這些功能。
- Amazon EC2 執行個體必須關閉資源型 IPv4 DNS 請求。若要停用此選項，在使用 Amazon EC2 主控台建立新執行個體時，請務必取消選取 Enable resource-based IPV4 (A record) DNS requests (啟用資源型 IPV4 (A 記錄) DNS 請求) 選項。若要使用停用此選項 AWS CLI，請使用下列命令。

```
aws ec2 modify-private-dns-name-options --instance-id i-xxxxxxx --no-enable-resource-name-dns-a-record --no-dry-run
```

- 不支援共用子網路中的 Amazon EC2 執行個體。就算使用也無法註冊到叢集。
- 您的 Amazon ECS 任務必須使用 `awsvpc` 網路模式和 EC2 啟動類型。ENI無論啟動多少次，使用 Fargate 啟動類型的任務總是會收到專用的，因此不需要此功能。
- 您的 Amazon ECS 任務必須與容器執行個體在相同的 Amazon VPC 中啟動。如果您的任務不在同一 VPC 中，則任務會因為屬性錯誤而失敗。
- 啟動新的容器執行個體時，執行個體將轉換為 `REGISTERING` 狀態，同時為執行個體佈建幹線彈性網路介面。如果註冊失敗，執行個體會轉換到 `REGISTRATION_FAILED` 狀態。透過說明容器執行個體以檢視說明失敗原因的 `statusReason` 欄位，可對失敗註冊進行故障排除。然後可以手動解除註冊或終止容器執行個體。成功取消註冊或終止容器執行個體後，Amazon ECS 將刪除幹線。ENI

> **Note**
>
> Amazon ECS 會發出容器執行個體狀態變更事件，您可以監控執行個體是否轉換為 `REGISTRATION_FAILED` 狀態。如需詳細資訊，請參閱 Amazon ECS 容器執行個體狀態變更事件。

- 容器執行個體終止後，執行個體將轉換為 `DEREGISTERING` 狀態，同時解除佈建幹線彈性網路介面。然後，執行個體會轉換為 `INACTIVE` 狀態。

- 如果公用子網路中有增加ENI限制的容器執行個體停止然後重新啟動，則執行個體會遺失其公用 IP 位址，且容器代理程式會失去連線。
- 啟用後`awsvpcTrunking`，容器執行個體會收到一ENI個使用 VPC 預設安全群組的額外資訊，並由 Amazon ECS 管理。

必要條件

啟動具有增加ENI限制的容器執行個體之前，必須先完成下列先決條件。

- 必須先建立 Amazon ECS 的服務連結角色。Amazon ECS 服務連結角色為 Amazon ECS 提供了代表您撥打其他 AWS 服務電話的許可。這個角色會在您建立叢集，或在 AWS Management Console 中建立或更新服務時，自動為您建立。如需詳細資訊，請參閱 使用 Amazon ECS 的服務連結角色。您也可以使用下列 AWS CLI 命令建立服務連結角色。

```
aws iam create-service-linked-role --aws-service-name ecs.amazonaws.com
```

- 您的帳戶或容器執行個體 IAM 角色必須啟用 `awsvpcTrunking` 帳戶設定。建議您建立 2 個容器執行個體角色 (`ecsInstanceRole`)。然後，您可以為一個角色啟用`awsvpcTrunking`帳戶設定，並將該角色用於需要 ENI 中繼的工作。如需容器執行個體角色的相關資訊，請參閱Amazon ECS 容器執行個體 IAM 角色。

符合先決條件後，您可以使用其中一種支援的 Amazon EC2 執行個體類型啟動新的容器執行個體，而且執行個體將有更高的ENI限制。如需支援的執行個體類型清單，請參閱 支援增加 Amazon ECS 容器網路界面的執行個體。容器執行個體必須具有容器代理程式的 `1.28.1` 版或更新版本，以及 ecs-init 套件的 `1.28.1-2` 版或更新版本。如果您使用 Amazon ECS 最佳化 AMI 的最新 Linux 變體，將會滿足這些要求。如需詳細資訊，請參閱 啟動 Amazon ECS Linux 容器執行個體。

> **Important**
>
> Amazon EC2 執行個體必須關閉資源型 IPv4 DNS 請求。若要停用此選項，在使用 Amazon EC2 主控台建立新執行個體時，請務必取消選取 Enable resource-based IPV4 (A record) DNS requests (啟用資源型 IPV4 (A 記錄) DNS 請求) 選項。若要使用停用此選項 AWS CLI，請使用下列命令。
>
> ```
> aws ec2 modify-private-dns-name-options --instance-id i-xxxxxxx --no-enable-resource-name-dns-a-record --no-dry-run
> ```

若要檢視增加ENI限制的容器執行個體 AWS CLI

每個容器執行個體都有預設的網路界面，稱為幹線網路界面。使用下列命令，透過查詢屬性 (表示該ecs.awsvpc-trunk-id屬性具有主幹網路介面) 來列出具有增加ENI限制的容器執行個體。

- list-attributes (AWS CLI)

```
aws ecs list-attributes \
      --target-type container-instance \
      --attribute-name ecs.awsvpc-trunk-id \
      --cluster cluster_name \
      --region us-east-1
```

- 取得 ECs () AttributeListAWS Tools for Windows PowerShell

```
Get-ECSAttributeList -TargetType container-instance -AttributeName ecs.awsvpc-trunk-
id -Region us-east-1
```

支援增加 Amazon ECS 容器網路界面的執行個體

下列顯示受支援的 Amazon EC2 執行個體類型，以及在啟用 awsvpcTrunking 帳戶設定的前後，可以在每個執行個體類型上啟動使用 awsvpc 網路模式的任務數量。針對每個執行個體類型的 elastic network interface (ENI) 限制，請在目前的工作限制中新增一個，因為主要網路介面會計入限制，並在新的工作限制中增加兩個，因為主要網路介面和主幹網路介面都會再次計入限制。

> **Important**
>
> 雖然同一執行個體系列支援其他執行個體類型，但不支援 a1.metal、c5.metal、c5a.8xlarge、c5ad.8xlarge、c5d.metal、m5.metal、p3dn.24xla 和 r5d.metal 執行個體類型。
> 不支援 c5n、d3、d3en、g3、g3s、g4dn、i3、i3en、inf1、m5dn、m5n、m5zn、mac1、r5b、r5n、r5 和 z1d 執行個體系列。

主題

- 一般用途
- 運算最佳化

- 記憶體最佳化
- 儲存最佳化
- 加速運算
- 高效能運算

一般用途

執行個體類型	沒有中ENI繼的任務限制	具有中ENI繼功能的任務限制
a1.medium	1	10
a1.large	2	10
a1.xlarge	3	20
a1.2xlarge	3	40
a1.4xlarge	7	60
m5.large	2	10
m5.xlarge	3	20
m5.2xlarge	3	40
m5.4xlarge	7	60
m5.8xlarge	7	60
m5.12xlarge	7	60
m5.16xlarge	14	120
m5.24xlarge	14	120
m5a.large	2	10
m5a.xlarge	3	20
m5a.2xlarge	3	40

執行個體類型	沒有中ENI繼的任務限制	具有中ENI繼功能的任務限制
m5a.4xlarge	7	60
m5a.8xlarge	7	60
m5a.12xlarge	7	60
m5a.16xlarge	14	120
m5a.24xlarge	14	120
m5ad.large	2	10
m5ad.xlarge	3	20
m5ad.2xlarge	3	40
m5ad.4xlarge	7	60
m5ad.8xlarge	7	60
m5ad.12xlarge	7	60
m5ad.16xlarge	14	120
m5ad.24xlarge	14	120
m5d.large	2	10
m5d.xlarge	3	20
m5d.2xlarge	3	40
m5d.4xlarge	7	60
m5d.8xlarge	7	60
m5d.12xlarge	7	60
m5d.16xlarge	14	120

執行個體類型	沒有中ENI繼的任務限制	具有中ENI繼功能的任務限制
m5d.24xlarge	14	120
m5d.metal	14	120
m5n.large	2	10
m5n.xlarge	3	20
m5n.2xlarge	3	40
m5n.4xlarge	7	60
m5n.8xlarge	7	60
m5n.12xlarge	7	60
m5n.16xlarge	14	120
m5zn.large	2	14
m5zn.xlarge	3	31
m5zn.2xlarge	3	64
m5zn.3xlarge	7	98
m5zn.6xlarge	7	120
m6a.large	2	10
m6a.xlarge	3	20
m6a.2xlarge	3	40
m6a.4xlarge	7	60
m6a.8xlarge	7	90
m6a.12xlarge	7	120

執行個體類型	沒有中ENI繼的任務限制	具有中ENI繼功能的任務限制
m6a.16xlarge	14	120
m6a.24xlarge	14	120
m6a.32xlarge	14	120
m6a.48xlarge	14	120
m6a.metal	14	120
m6g.medium	1	4
m6g.large	2	10
m6g.xlarge	3	20
m6g.2xlarge	3	40
m6g.4xlarge	7	60
m6g.8xlarge	7	60
m6g.12xlarge	7	60
m6g.16xlarge	14	120
m6g.metal	14	120
m6gd.medium	1	4
m6gd.large	2	10
m6gd.xlarge	3	20
m6gd.2xlarge	3	40
m6gd.4xlarge	7	60
m6gd.8xlarge	7	60

執行個體類型	沒有中ENI繼的任務限制	具有中ENI繼功能的任務限制
m6gd.12xlarge	7	60
m6gd.16xlarge	14	120
m6gd.metal	14	120
m6i.large	2	10
m6i.xlarge	3	20
m6i.2xlarge	3	40
m6i.4xlarge	7	60
m6i.8xlarge	7	90
m6i.12xlarge	7	120
m6i.16xlarge	14	120
m6i.24xlarge	14	120
m6i.32xlarge	14	120
m6i.metal	14	120
m6id.large	2	10
m6id.xlarge	3	20
m6id.2xlarge	3	40
m6id.4xlarge	7	60
m6id.8xlarge	7	90
m6id.12xlarge	7	120
m6id.16xlarge	14	120

執行個體類型	沒有中ENI繼的任務限制	具有中ENI繼功能的任務限制
m6id.24xlarge	14	120
m6id.32xlarge	14	120
m6id.metal	14	120
m6idn.large	2	10
m6idn.xlarge	3	20
m6idn.2xlarge	3	40
m6idn.4xlarge	7	60
m6idn.8xlarge	7	90
m6idn.12xlarge	7	120
m6idn.16xlarge	14	120
m6idn.24xlarge	14	120
m6idn.32xlarge	15	120
m6idn.metal	15	120
m6in.large	2	10
m6in.xlarge	3	20
m6in.2xlarge	3	40
m6in.4xlarge	7	60
m6in.8xlarge	7	90
m6in.12xlarge	7	120
m6in.16xlarge	14	120

執行個體類型	沒有中ENI繼的任務限制	具有中ENI繼功能的任務限制
m6in.24xlarge	14	120
m6in.32xlarge	15	120
m6in.metal	15	120
m7a.medium	1	4
m7a.large	2	10
m7a.xlarge	3	20
m7a.2xlarge	3	40
m7a.4xlarge	7	60
m7a.8xlarge	7	90
m7a.12xlarge	7	120
m7a.16xlarge	14	120
m7a.24xlarge	14	120
m7a.32xlarge	14	120
m7a.48xlarge	14	120
m7a.metal-48xl	14	120
m7g.medium	1	4
m7g.large	2	10
m7g.xlarge	3	20
m7g.2xlarge	3	40
m7g.4xlarge	7	60

執行個體類型	沒有中ENI繼的任務限制	具有中ENI繼功能的任務限制
m7g.8xlarge	7	60
m7g.12xlarge	7	60
m7g.16xlarge	14	120
m7g.metal	14	120
m7gd.medium	1	4
m7gd.large	2	10
m7gd.xlarge	3	20
m7gd.2xlarge	3	40
m7gd.4xlarge	7	60
m7gd.8xlarge	7	60
m7gd.12xlarge	7	60
m7gd.16xlarge	14	120
金屬	14	120
m7i.large	2	10
m7i.xlarge	3	20
m7i.2xlarge	3	40
m7i.4xlarge	7	60
m7i.8xlarge	7	90
m7i.12xlarge	7	120
m7i.16xlarge	14	120

執行個體類型	沒有中ENI繼的任務限制	具有中ENI繼功能的任務限制
m7i.24xlarge	14	120
m7i.48xlarge	14	120
m7i.metal-24xl	14	120
m7i.metal-48xl	14	120
m7i-flex.large	2	4
m7i-flex.xlarge	3	10
m7i-flex.2xlarge	3	20
m7i-flex.4xlarge	7	40
m7i-flex.8xlarge	7	60
mac2.metal	7	12
mac2-m2.metal	7	12
mac2-m2pro.metal	7	12

運算最佳化

執行個體類型	沒有中ENI繼的任務限制	具有中ENI繼功能的任務限制
c5.large	2	10
c5.xlarge	3	20
c5.2xlarge	3	40
c5.4xlarge	7	60
c5.9xlarge	7	60

執行個體類型	沒有中ENI繼的任務限制	具有中ENI繼功能的任務限制
c5.12xlarge	7	60
c5.18xlarge	14	120
c5.24xlarge	14	120
c5a.large	2	10
c5a.xlarge	3	20
c5a.2xlarge	3	40
c5a.4xlarge	7	60
c5a.12xlarge	7	60
c5a.16xlarge	14	120
c5a.24xlarge	14	120
c5ad.large	2	10
c5ad.xlarge	3	20
c5ad.2xlarge	3	40
c5ad.4xlarge	7	60
c5ad.12xlarge	7	60
c5ad.16xlarge	14	120
c5ad.24xlarge	14	120
c5d.large	2	10
c5d.xlarge	3	20
c5d.2xlarge	3	40

執行個體類型	沒有中ENI繼的任務限制	具有中ENI繼功能的任務限制
c5d.4xlarge	7	60
c5d.9xlarge	7	60
c5d.12xlarge	7	60
c5d.18xlarge	14	120
c5d.24xlarge	14	120
c6a.large	2	10
c6a.xlarge	3	20
c6a.2xlarge	3	40
c6a.4xlarge	7	60
c6a.8xlarge	7	90
c6a.12xlarge	7	120
c6a.16xlarge	14	120
c6a.24xlarge	14	120
c6a.32xlarge	14	120
c6a.48xlarge	14	120
c6a.metal	14	120
c6g.medium	1	4
c6g.large	2	10
c6g.xlarge	3	20
c6g.2xlarge	3	40

執行個體類型	沒有中ENI繼的任務限制	具有中ENI繼功能的任務限制
c6g.4xlarge	7	60
c6g.8xlarge	7	60
c6g.12xlarge	7	60
c6g.16xlarge	14	120
c6g.metal	14	120
c6gd.medium	1	4
c6gd.large	2	10
c6gd.xlarge	3	20
c6gd.2xlarge	3	40
c6gd.4xlarge	7	60
c6gd.8xlarge	7	60
c6gd.12xlarge	7	60
c6gd.16xlarge	14	120
c6gd.metal	14	120
c6gn.medium	1	4
c6gn.large	2	10
c6gn.xlarge	3	20
c6gn.2xlarge	3	40
c6gn.4xlarge	7	60
c6gn.8xlarge	7	60

執行個體類型	沒有中ENI繼的任務限制	具有中ENI繼功能的任務限制
c6gn.12xlarge	7	60
c6gn.16xlarge	14	120
c6i.large	2	10
c6i.xlarge	3	20
c6i.2xlarge	3	40
c6i.4xlarge	7	60
c6i.8xlarge	7	90
c6i.12xlarge	7	120
c6i.16xlarge	14	120
c6i.24xlarge	14	120
c6i.32xlarge	14	120
c6i.metal	14	120
c6id.large	2	10
c6id.xlarge	3	20
c6id.2xlarge	3	40
c6id.4xlarge	7	60
c6id.8xlarge	7	90
c6id.12xlarge	7	120
c6id.16xlarge	14	120
c6id.24xlarge	14	120

執行個體類型	沒有中ENI繼的任務限制	具有中ENI繼功能的任務限制
c6id.32xlarge	14	120
c6id.metal	14	120
c6in.large	2	10
c6in.xlarge	3	20
c6in.2xlarge	3	40
c6in.4xlarge	7	60
c6in.8xlarge	7	90
c6in.12xlarge	7	120
c6in.16xlarge	14	120
c6in.24xlarge	14	120
c6in.32xlarge	15	120
c6in.metal	15	120
c7a.medium	1	4
c7a.large	2	10
c7a.xlarge	3	20
c7a.2xlarge	3	40
c7a.4xlarge	7	60
c7a.8xlarge	7	90
c7a.12xlarge	7	120
c7a.16xlarge	14	120

執行個體類型	沒有中ENI繼的任務限制	具有中ENI繼功能的任務限制
c7a.24xlarge	14	120
c7a.32xlarge	14	120
c7a.48xlarge	14	120
c7a.metal-48xl	14	120
c7g.medium	1	4
c7g.large	2	10
c7g.xlarge	3	20
c7g.2xlarge	3	40
c7g.4xlarge	7	60
c7g.8xlarge	7	60
c7g.12xlarge	7	60
c7g.16xlarge	14	120
c7g.metal	14	120
c7gd.medium	1	4
c7gd.large	2	10
c7gd.xlarge	3	20
c7gd.2xlarge	3	40
c7gd.4xlarge	7	60
c7gd.8xlarge	7	60
c7gd.12xlarge	7	60

執行個體類型	沒有中ENI繼的任務限制	具有中ENI繼功能的任務限制
c7gd.16xlarge	14	120
金屬	14	120
c7gn.medium	1	4
c7gn.large	2	10
c7gn.xlarge	3	20
c7gn.2xlarge	3	40
c7gn.4xlarge	7	60
c7gn.8xlarge	7	60
c7gn.12xlarge	7	60
c7gn.16xlarge	14	120
金屬	14	120
c7i.large	2	10
c7i.xlarge	3	20
c7i.2xlarge	3	40
c7i.4xlarge	7	60
c7i.8xlarge	7	90
c7i.12xlarge	7	120
c7i.16xlarge	14	120
c7i.24xlarge	14	120
c7i.48xlarge	14	120

執行個體類型	沒有中ENI繼的任務限制	具有中ENI繼功能的任務限制
c7i.metal-24xl	14	120
c7i.metal-48xl	14	120
C7-柔性. 大	2	4
C7-柔性. 大	3	10
C7-柔性 .2XL	3	20
C7-柔性 .4 倍大	7	40
C7-柔性 .8 倍大	7	60

記憶體最佳化

執行個體類型	沒有中ENI繼的任務限制	具有中ENI繼功能的任務限制
r5.large	2	10
r5.xlarge	3	20
r5.2xlarge	3	40
r5.4xlarge	7	60
r5.12xlarge	7	60
r5.16xlarge	14	120
r5.24xlarge	14	120
r5a.large	2	10
r5a.xlarge	3	20
r5a.2xlarge	3	40

執行個體類型	沒有中ENI繼的任務限制	具有中ENI繼功能的任務限制
r5a.4xlarge	7	60
r5a.8xlarge	7	60
r5a.12xlarge	7	60
r5a.16xlarge	14	120
r5a.24xlarge	14	120
r5ad.large	2	10
r5ad.xlarge	3	20
r5ad.2xlarge	3	40
r5ad.4xlarge	7	60
r5ad.8xlarge	7	60
r5ad.12xlarge	7	60
r5ad.16xlarge	14	120
r5ad.24xlarge	14	120
r5b.16xlarge	14	120
r5d.large	2	10
r5d.xlarge	3	20
r5d.2xlarge	3	40
r5d.4xlarge	7	60
r5d.8xlarge	7	60
r5d.12xlarge	7	60

執行個體類型	沒有中ENI繼的任務限制	具有中ENI繼功能的任務限制
r5d.16xlarge	14	120
r5d.24xlarge	14	120
r5dn.16xlarge	14	120
r6a.large	2	10
r6a.xlarge	3	20
r6a.2xlarge	3	40
r6a.4xlarge	7	60
r6a.8xlarge	7	90
r6a.12xlarge	7	120
r6a.16xlarge	14	120
r6a.24xlarge	14	120
r6a.32xlarge	14	120
r6a.48xlarge	14	120
r6a.metal	14	120
r6g.medium	1	4
r6g.large	2	10
r6g.xlarge	3	20
r6g.2xlarge	3	40
r6g.4xlarge	7	60
r6g.8xlarge	7	60

執行個體類型	沒有中ENI繼的任務限制	具有中ENI繼功能的任務限制
r6g.12xlarge	7	60
r6g.16xlarge	14	120
r6g.metal	14	120
r6gd.medium	1	4
r6gd.large	2	10
r6gd.xlarge	3	20
r6gd.2xlarge	3	40
r6gd.4xlarge	7	60
r6gd.8xlarge	7	60
r6gd.12xlarge	7	60
r6gd.16xlarge	14	120
r6gd.metal	14	120
r6i.large	2	10
r6i.xlarge	3	20
r6i.2xlarge	3	40
r6i.4xlarge	7	60
r6i.8xlarge	7	90
r6i.12xlarge	7	120
r6i.16xlarge	14	120
r6i.24xlarge	14	120

執行個體類型	沒有中ENI繼的任務限制	具有中ENI繼功能的任務限制
r6i.32xlarge	14	120
r6i.metal	14	120
r6idn.large	2	10
r6idn.xlarge	3	20
r6idn.2xlarge	3	40
r6idn.4xlarge	7	60
r6idn.8xlarge	7	90
r6idn.12xlarge	7	120
r6idn.16xlarge	14	120
r6idn.24xlarge	14	120
r6idn.32xlarge	15	120
r6idn.metal	15	120
r6in.large	2	10
r6in.xlarge	3	20
r6in.2xlarge	3	40
r6in.4xlarge	7	60
r6in.8xlarge	7	90
r6in.12xlarge	7	120
r6in.16xlarge	14	120
r6in.24xlarge	14	120

執行個體類型	沒有中ENI繼的任務限制	具有中ENI繼功能的任務限制
r6in.32xlarge	15	120
r6in.metal	15	120
r6id.large	2	10
r6id.xlarge	3	20
r6id.2xlarge	3	40
r6id.4xlarge	7	60
r6id.8xlarge	7	90
r6id.12xlarge	7	120
r6id.16xlarge	14	120
r6id.24xlarge	14	120
r6id.32xlarge	14	120
r6id.metal	14	120
r7a.medium	1	4
r7a.large	2	10
r7a.xlarge	3	20
r7a.2xlarge	3	40
r7a.4xlarge	7	60
r7a.8xlarge	7	90
r7a.12xlarge	7	120
r7a.16xlarge	14	120

執行個體類型	沒有中ENI繼的任務限制	具有中ENI繼功能的任務限制
r7a.24xlarge	14	120
r7a.32xlarge	14	120
r7a.48xlarge	14	120
r7a.metal-48xl	14	120
r7g.medium	1	4
r7g.large	2	10
r7g.xlarge	3	20
r7g.2xlarge	3	40
r7g.4xlarge	7	60
r7g.8xlarge	7	60
r7g.12xlarge	7	60
r7g.16xlarge	14	120
r7g.metal	14	120
r7gd.medium	1	4
r7gd.large	2	10
r7gd.xlarge	3	20
r7gd.2xlarge	3	40
r7gd.4xlarge	7	60
r7gd.8xlarge	7	60
r7gd.12xlarge	7	60

執行個體類型	沒有中ENI繼的任務限制	具有中ENI繼功能的任務限制
r7gd.16xlarge	14	120
金屬	14	120
r7i.large	2	10
r7i.xlarge	3	20
r7i.2xlarge	3	40
r7i.4xlarge	7	60
r7i.8xlarge	7	90
r7i.12xlarge	7	120
r7i.16xlarge	14	120
r7i.24xlarge	14	120
r7i.48xlarge	14	120
r7i.metal-24xl	14	120
r7i.metal-48xl	14	120
大	2	10
大	3	20
r7iz.2xlarge	3	40
r7iz.4xlarge	7	60
r7iz.8xlarge	7	90
r7iz.12xlarge	7	120
r7iz.16xlarge	14	120

執行個體類型	沒有中ENI繼的任務限制	具有中ENI繼功能的任務限制
r7iz.32xlarge	14	120
r7iz.metal-16xl	14	120
r7iz.metal-32xl	14	120
u-3tb1.56xlarge	7	12
u-6tb1.56xlarge	14	12
u-18tb1.112xlarge	14	12
u-18tb1.metal	14	12
u-24tb1.112xlarge	14	12
u-24tb1.metal	14	12
大尺寸	14	120
超大	15	120
超大 24 英寸	15	120
超大二十二分之二十四	15	120
x2gd.medium	1	10
x2gd.large	2	10
x2gd.xlarge	3	20
x2gd.2xlarge	3	40
x2gd.4xlarge	7	60
x2gd.8xlarge	7	60
x2gd.12xlarge	7	60

執行個體類型	沒有中ENI繼的任務限制	具有中ENI繼功能的任務限制
x2gd.16xlarge	14	120
x2gd.metal	14	120
x2idn.16xlarge	14	120
x2idn.24xlarge	14	120
x2idn.32xlarge	14	120
x2idn.metal	14	120
x2iedn.xlarge	3	13
x2iedn.2xlarge	3	29
x2iedn.4xlarge	7	60
x2iedn.8xlarge	7	120
x2iedn.16xlarge	14	120
x2iedn.24xlarge	14	120
x2iedn.32xlarge	14	120
x2iedn.metal	14	120
x2iezn.2xlarge	3	64
x2iezn.4xlarge	7	120
x2iezn.6xlarge	7	120
x2iezn.8xlarge	7	120
x2iezn.12xlarge	14	120
x2iezn.metal	14	120

儲存最佳化

執行個體類型	沒有中ENI繼的任務限制	具有中ENI繼功能的任務限制
i4g.large	2	10
i4g.xlarge	3	20
i4g.2xlarge	3	40
i4g.4xlarge	7	60
i4g.8xlarge	7	60
i4g.16xlarge	14	120
i4i.xlarge	3	8
i4i.2xlarge	3	28
i4i.4xlarge	7	58
i4i.8xlarge	7	118
i4i.12xlarge	7	118
i4i.16xlarge	14	248
i4i.24xlarge	14	118
i4i.32xlarge	14	498
i4i.metal	14	498
im4gn.large	2	10
im4gn.xlarge	3	20
im4gn.2xlarge	3	40
im4gn.4xlarge	7	60

執行個體類型	沒有中ENI繼的任務限制	具有中ENI繼功能的任務限制
im4gn.8xlarge	7	60
im4gn.16xlarge	14	120
is4gen.medium	1	4
is4gen.large	2	10
is4gen.xlarge	3	20
is4gen.2xlarge	3	40
is4gen.4xlarge	7	60
is4gen.8xlarge	7	60

加速運算

執行個體類型	沒有中ENI繼的任務限制	具有中ENI繼功能的任務限制
dl1.24xlarge	59	120
dl2q.24xlarge	14	120
g4ad.xlarge	1	12
g4ad.2xlarge	1	12
g4ad.4xlarge	2	12
g4ad.8xlarge	3	12
g4ad.16xlarge	7	12
g5.xlarge	3	6
g5.2xlarge	3	19

執行個體類型	沒有中ENI繼的任務限制	具有中ENI繼功能的任務限制
g5.4xlarge	7	40
g5.8xlarge	7	90
g5.12xlarge	14	120
g5.16xlarge	7	120
g5.24xlarge	14	120
g5.48xlarge	6	120
g5g.xlarge	3	20
g5g.2xlarge	3	40
g5g.4xlarge	7	60
g5g.8xlarge	7	60
g5g.16xlarge	14	120
g5g.metal	14	120
大	3	20
大	3	40
大	7	60
大	7	90
大	7	120
大	14	120
大	14	120
大	14	120

執行個體類型	沒有中ENI繼的任務限制	具有中ENI繼功能的任務限制
GR6.4 x 大	7	60
克 6.8 倍大	7	90
inf2.xlarge	3	20
inf2.8xlarge	7	90
inf2.24xlarge	14	120
inf2.48xlarge	14	120
p4d.24xlarge	59	120
p4de.24xlarge	59	120
p5.48xlarge	63	242
trn1.2xlarge	3	19
trn1.32xlarge	39	120
trn1n.32xlarge	79	242
vt1.3xlarge	3	40
vt1.6xlarge	7	60
vt1.24xlarge	14	120

高效能運算

執行個體類型	沒有中ENI繼的任務限制	具有中ENI繼功能的任務限制
hpc6a.48xlarge	1	120
hpc6id.32xlarge	1	120

執行個體類型	沒有中ENI繼的任務限制	具有中ENI繼功能的任務限制
hpc7g.4xlarge	3	120
hpc7g.8xlarge	3	120
hpc7g.16xlarge	3	120

保留 Amazon ECS Linux 容器執行個體記憶體

Amazon ECS 容器代理程式將容器執行個體註冊到叢集時，代理程式必須判斷容器執行個體可用於為您的任務保留多少記憶體。由於存在平台記憶體額外負荷和由系統核心佔用的記憶體，這數字會與 Amazon EC2 執行個體公告的安裝記憶體數量不同。舉例而言，`m4.large` 執行個體安裝了 8 GiB 的記憶體。但是，當容器實例註冊時，這並不總是轉換為可用於任務的 8192 MiB 內存。

Amazon ECS 容器代理程式提供名為的組態變數`ECS_RESERVED_MEMORY`，您可以使用該變數從分配給任務的集區中移除指定數量的 MiB 記憶體。這可為重要系統程序有效地預留記憶體。

如果您的工作佔用了容器執行個體上的所有記憶體，則您的工作可能會與重要的系統處理程序爭奪記憶體，並可能啟動系統故障。

舉例而言，若在容器代理程式檔案中指定 `ECS_RESERVED_MEMORY=256`，則代理程式會將記憶體總量減去 256 MiB 再註冊給該執行個體，而 256 MiB 的記憶體就無法由 ECS 任務配置。如需代理程式組態變數的詳細資訊及設定方式，請參閱 Amazon ECS 容器代理程式組態 和 引導 Amazon ECS Linux 容器執行個體以傳遞資料。

如果您為工作指定 8192 MiB，且容器執行個體都沒有 8192 MiB 或更大的記憶體可用來滿足此需求，則無法將工作放置在叢集中。如果您使用的是受管運算環境，則 AWS Batch 必須啟動較大的執行個體類型以容納要求。

也應預留一些記憶體供 Amazon ECS 容器代理程式以及容器執行個體上其他重要的系統程序使用，任務的容器才不會彼此爭奪相同的記憶體，而導致引發系統故障的可能。

Amazon ECS 容器代理程式會使用 Docker `ReadMemInfo()` 函數來查詢作業系統可用的記憶體總量。Linux 和 Windows 都提供了命令行實用程序來確定總內存。

Example - 判定 Linux 記憶體總量

free 命令會傳回作業系統辨識出的記憶體總量。

```
$ free -b
```

執行 Amazon ECS 最佳化 Amazon Linux AMI 的 m4.large 執行個體的輸出範例。

```
             total       used       free     shared    buffers     cached
Mem:    8373026816  348180480 8024846336      90112   25534464  205418496
-/+ buffers/cache:  117227520 8255799296
```

執行個體的記憶體總量有 8373026816 位元組，轉為任務可使用的記憶體則有 7985 MiB。

Example - 判定 Windows 記憶體總量

wmic 命令會傳回作業系統辨識出的記憶體總量。

```
C:\> wmic ComputerSystem get TotalPhysicalMemory
```

執行 Amazon ECS 最佳化視窗伺服器 AMI 的m4.large執行個體輸出範例。

```
TotalPhysicalMemory
8589524992
```

執行個體的記憶體總量有 8589524992 位元組，轉為任務可使用的記憶體則有 8191 MiB。

查看容器實例記憶體

您可以在 Amazon ECS 主控台 (或透過執行個DescribeContainer體 API 作業) 檢視容器執行個體註冊多少記憶體。如果您嘗試透過為特定執行個體類型提供任務盡可能多的記憶體，以最大限度地提高資源使用率，則可以觀察該容器執行個體可用的記憶體，然後為工作指派那麼多記憶體。

檢視容器執行個體記憶體

1. 開啟主控台，網址為 https://console.aws.amazon.com/ecs/v2。
2. 在瀏覽窗格中，選擇 [叢集]，然後選擇託管容器執行個體的叢集。
3. 選擇基礎結構，然後在容器執行個體下選擇容器執行個體。
4. 「資源」段落顯示容器執行處理的已註冊記憶體和可用記憶體。

   「已註冊的記憶體」值是容器執行個體的值；首次啟動 Amazon ECS 時向 Amazon ECS 註冊，而「可用記憶體」值則是尚未分配給任務的值。

使用遠端管理 Amazon ECS 容器執行個體 AWS Systems Manager

您可以使用 AWS Systems Manager (Systems Manager) 中的執行命令功能，以安全且遠端的方式管理 Amazon ECS 容器執行個體的組態。Run Command 提供執行常見管理任務的簡單方法，而不需在本機登入執行個體。您可以同時對多個容器執行個體執行命令，跨叢集管理組態變更。Run Command 會報告每個命令的狀態和結果。

以下是您可以使用 Run Command 執行的一些任務類型範例：

- 安裝或解除安裝套件。
- 執行安全性更新。
- 清除 Docker 映像。
- 停止或啟動服務。
- 檢視系統資源。
- 檢視日誌檔。
- 執行檔案操作。

如需執行命令的詳細資訊，請參閱 AWS Systems Manager 使用者指南中的 AWS Systems Manager 執行命令。

以下是將 Systems Manager 與 Amazon ECS 搭配使用的準則。

1. 您必須授與容器執行個體角色 (ecs InstanceRole) 權限，才能存取 Systems Manager API。您可以通過將 AmazonSSM ManagedInstance 核心分配給角色來執行此操作。`ecsInstanceRole`如需如何將原則附加至角色的詳細資訊，請參閱AWS Identity and Access Management 使用指南中的修改角色權限原則 (主控台)
2. 確認 SSM Agent 安裝於您的容器執行個體上。如需詳細資訊，請參閱在 Linux 的 EC2 執行個體上手動安裝 SSM Agent。

將 Systems Manager 受管理的原則附加至您的，`ecsInstanceRole`並確認 AWS Systems Manager 代理程式 (SSM 代理程式) 已安裝在容器執行個體上之後，您就可以開始使用執行命令將命令傳送至容器執行個體。如需有關在執行個體上執行命令和 shell 指令碼並檢視結果輸出的詳細資訊，請參閱 AWS Systems Manager 使用者指南中的使用 Systems Manager Run Command 執行命令和執行命令演練。

常見的使用案例是使用執行命令更新容器執行個體軟體。您可以使用下列參數遵循《 AWS Systems Manager 使用指南》中的程序。

參數	Value
命令文件	AWS-RunShellScript
命令	`$ yum update -y`
目標實例	容器執行個體

針對 Amazon ECS 容器執行個體使用 HTTP 代理伺服器

您可以設定您的 Amazon ECS 容器執行個體，針對 Amazon ECS 容器代理程式和 Docker 常駐程式使用 HTTP 代理。這在您的容器執行個體沒有透過 Amazon VPC 網際網路閘道、NAT 閘道或執行個體進行外部網路存取時會很有用。

若要設定您的 Amazon ECS Linux 容器執行個體以使用 HTTP 代理，請在啟動時 (使用 Amazon EC2 使用者資料) 設定相關檔案中的下列變數。您也可以手動編輯組態檔，然後重新啟動代理程式。

`/etc/ecs/ecs.config` (Amazon Linux 2和 AmazonLinux AMI)

`HTTP_PROXY=10.0.0.131:3128`

將此數值設為要讓 Amazon ECS 代理程式連線到網際網路的 HTTP 代理之主機名稱 (或 IP 地址) 及連接埠號碼。例如，您的容器執行個體可能沒有透過 Amazon VPC 網際網路閘道、NAT 閘道或執行個體的外部網路存取。

`NO_PROXY=169.254.169.254,169.254.170.2,/var/run/docker.sock`

將此值設為 `169.254.169.254,169.254.170.2,/var/run/docker.sock`，篩選 EC2 執行個體中繼資料、任務的 IAM 角色，以及來自代理的 Docker 常駐程式流量。

`/etc/systemd/system/ecs.service.d/http-proxy.conf` (僅限 Amazon Linux 2)

`Environment="HTTP_PROXY=10.0.0.131:3128/"`

將此數值設為要讓 `ecs-init` 連線到網際網路的 HTTP 代理之主機名稱 (或 IP 地址) 及連接埠號碼。例如，您的容器執行個體可能沒有透過 Amazon VPC 網際網路閘道、NAT 閘道或執行個體的外部網路存取。

```
Environment="NO_PROXY=169.254.169.254,169.254.170.2,/var/run/
docker.sock"
```

將此值設為 169.254.169.254,169.254.170.2,/var/run/docker.sock，篩選 EC2 執行個體中繼資料、任務的 IAM 角色，以及來自代理的 Docker 常駐程式流量。

/etc/init/ecs.override (僅限 Amazon Linux AMI)

```
env HTTP_PROXY=10.0.0.131:3128
```

將此數值設為要讓 ecs-init 連線到網際網路的 HTTP 代理之主機名稱 (或 IP 地址) 及連接埠號碼。例如，您的容器執行個體可能沒有透過 Amazon VPC 網際網路閘道、NAT 閘道或執行個體的外部網路存取。

```
env NO_PROXY=169.254.169.254,169.254.170.2,/var/run/docker.sock
```

將此值設為 169.254.169.254,169.254.170.2,/var/run/docker.sock，篩選 EC2 執行個體中繼資料、任務的 IAM 角色，以及來自代理的 Docker 常駐程式流量。

/etc/systemd/system/docker.service.d/http-proxy.conf (僅限 Amazon Linux 2)

```
Environment="HTTP_PROXY=http://10.0.0.131:3128"
```

將此數值設為要讓 Docker 常駐程式連線到網際網路的 HTTP 代理之主機名稱 (或 IP 地址) 及連接埠號碼。例如，您的容器執行個體可能沒有透過 Amazon VPC 網際網路閘道、NAT 閘道或執行個體的外部網路存取。

```
Environment="NO_PROXY=169.254.169.254"
```

將此數值設為 169.254.169.254，篩選來自代理的 EC2 執行個體中繼資料。

/etc/sysconfig/docker (僅限 Amazon Linux AMI 和 Amazon Linux 2)

```
export HTTP_PROXY=http://10.0.0.131:3128
```

將此數值設為要讓 Docker 常駐程式連線到網際網路的 HTTP 代理之主機名稱 (或 IP 地址) 及連接埠號碼。例如，您的容器執行個體可能沒有透過 Amazon VPC 網際網路閘道、NAT 閘道或執行個體的外部網路存取。

```
export NO_PROXY=169.254.169.254,169.254.170.2
```

將此數值設為 169.254.169.254，篩選來自代理的 EC2 執行個體中繼資料。

設定上述檔案中的這些環境變數只會影響 Amazon ECS 容器代理、ecs-init 及 Docker 常駐程式。他們不會設定任何其他服務 (例如 yum) 使用代理。

如需如何設定代理伺服器的相關資訊，請參閱如何在 Amazon Linux 2 或 AL2023 中為碼頭設定 HTTP 代理和 Amazon ECS 容器代理程式。

為您的 Amazon ECS Auto Scaling 群組設定預先初始化的執行個體

Amazon ECS 支援 Amazon EC2 Auto Scaling 暖集區。暖集區是一組準備投入使用的預先初始化 Amazon EC2 執行個體。每當應用程式需要水平擴展時，Amazon EC2 Auto Scaling 都會使用暖集區中的預初始化執行個體 (而不是啟動冷執行個體)，讓任何最終初始化程序執行，然後將執行個體投入使用。

若要瞭解有關暖集區以及如何將暖集區新增到 Auto Scaling 群組中的詳細資訊，請參閱《Amazon EC2 Auto Scaling 使用者指南》中的 Amazon EC2 Auto Scaling 的暖集區。

當您為 Amazon ECS 的 Auto Scaling 群組建立或更新暖集區時，無法設定將執行個體傳回縮減暖集區的選項 (`ReuseOnScaleIn`)。如需詳細資訊，請參閱 AWS Command Line Interface 參考中的 put-warm-pool。

若要將暖集區與 Amazon ECS 叢集配合使用，請在 Amazon EC2 Auto Scaling 群組啟動範本 User data (使用者資料) 欄位中將 `ECS_WARM_POOLS_CHECK` 代理程式組態變數設定為 `true`。

下列示範如何在 Amazon EC2 啟動範本的 User data (使用者資料) 欄位中指定代理程式組態變數。*`MyCluster`*用我們的集群的名稱替換。

```
#!/bin/bash
cat <<'EOF' >> /etc/ecs/ecs.config
ECS_CLUSTER=MyCluster
ECS_WARM_POOLS_CHECK=true
EOF
```

僅代理程式版本 `1.59.0` 和更新版本支援 `ECS_WARM_POOLS_CHECK` 變數。如需變數的詳細資訊，請參閱 Amazon ECS 容器代理程式組態。

更新 Amazon ECS 容器代理程式

有時候，您可能需要更新 Amazon ECS 容器代理程式，以取得錯誤修正和新功能。更新 Amazon ECS 容器代理不會中斷容器執行個體上正在執行中的任務或服務。更新代理程式的程序有所不同，這取決於您是否使用 Amazon ECS 最佳化 AMI 來啟動容器執行個體，或是否是在其他作業系統上。

> **Note**
>
> 代理更新不適用於 Windows 容器執行個體。我們建議您啟動新的容器執行個體，以更新您 Windows 叢集中的代理版本。

檢查 Amazon ECS 容器代理程式版本

您可以檢查在您的容器執行個體上執行的容器代理版本，以查看是否需要更新它。Amazon ECS 主控台中的容器執行個體檢視可提供代理程式版本。請使用下列步驟來檢查您的代理版本。

Amazon ECS console

1. 開啟主控台，網址為 https://console.aws.amazon.com/ecs/v2。
2. 從導覽列中選擇註冊外部執行個體所在的區域。
3. 在導覽窗格中選擇 Clusters (叢集)，並選取託管外部執行個體的叢集。
4. 在 Cluster : ***name*** (叢集：名稱) 頁面上，選擇 Infrastructure (基礎基礎設施) 索引標籤。
5. 在 Container instances (容器執行個體) 下，注意您容器執行個體的 Agent version (代理程式版本) 資料欄。如果容器執行個體不包含最新版的容器代理，主控台會使用訊息和標記提醒您過時的代理版本。

   若您的代理版本已過時，可以使用以下程序更新容器代理程式：

   - 若您的容器執行個體執行的是 Amazon ECS 最佳化 AMI，請參閱 在 Amazon ECS 最佳化 AMI 上更新 Amazon ECS 容器代理。
   - 若您的容器執行個體執行的不是 Amazon ECS 最佳化 AMI，請參閱 手動更新 Amazon ECS 容器代理程式 (適用於非 Amazon ECS 最佳化 AMI)。

   > **Important**
   >
   > 若要在您的 Amazon ECS 最佳化 AMI 上，更新 v1.0.0 版本之前的 Amazon ECS 代理程式版本，我們建議您終止目前的容器執行個體，並使用最近的 AMI 版本啟動新的執行個體。任何使用預覽版本的容器執行個體都應進行淘汰，並使用最近的 AMI 取代。如需詳細資訊，請參閱 啟動 Amazon ECS Linux 容器執行個體。

Amazon ECS container agent introspection API

您也可以使用 Amazon ECS 容器代理程式從容器執行個體自我檢查 API 版本。如需詳細資訊，請參閱 Amazon ECS 容器內省。

若要使用自我檢查 API 檢查您的 Amazon ECS 容器代理是否執行最新版本

1. 透過 SSH 登入您的容器執行個體。
2. 查詢自我檢查 API。

```
[ec2-user ~]$ curl -s 127.0.0.1:51678/v1/metadata | python3 -mjson.tool
```

> **Note**
>
> 自我檢查 API 在 Amazon ECS 容器代理版本 v1.0.0 中新增 `Version` 資訊。若在查詢自我檢查 API 時沒有看見 `Version`，或是您的代理中甚至沒有自我檢查 API，則您執行的版本便是 v0.0.3 及更早版本。您應更新您的版本。

在 Amazon ECS 最佳化 AMI 上更新 Amazon ECS 容器代理

若您使用的是 Amazon ECS 最佳化 AMI，您有幾個選項可取得最新版本的 Amazon ECS 容器代理程式 (以下顯示的順序為建議順序)：

- 終止容器執行個體，並啟動最新版本的 Amazon ECS 最佳化 Amazon Linux 2 AMI (以手動方式或使用最新 AMI 更新您的 Auto Scaling 啟動組態)。這可提供全新的容器執行個體，以及最新已測試和驗證的 Amazon Linux、Docker、`ecs-init` 和 Amazon ECS 容器代理程式的版本。如需詳細資訊，請參閱 Amazon ECS 優化的 Linux AMI。
- 使用 SSH 連線到執行個體，將 `ecs-init` 套裝服務 (及其相依性) 更新到最新版本。此操作可提供 Amazon Linux 儲存庫中目前經過測試及驗證的 Docker 和 `ecs-init` 版本，以及最新版 Amazon ECS 容器代理程式。如需詳細資訊，請參閱 若要更新 Amazon ECS 最佳化 AMI 上的 `ecs-init` 套件。
- 透過主控台或使用或 AWS SDK，透過 `UpdateContainerAgent` API 作業更新容器代理程式。AWS CLI 如需詳細資訊，請參閱 使用 `UpdateContainerAgent` API 操作更新 Amazon ECS 容器代理程式。

> **Note**
>
> 代理更新不適用於 Windows 容器執行個體。我們建議您啟動新的容器執行個體，以更新您 Windows 叢集中的代理版本。

若要更新 Amazon ECS 最佳化 AMI 上的 **ecs-init** 套件

1. 透過 SSH 登入您的容器執行個體。
2. 使用以下命令更新 `ecs-init` 套裝服務。

```
sudo yum update -y ecs-init
```

> **Note**
>
> `ecs-init` 套件和 Amazon ECS 容器代理程式會立即更新。但是，直到 Docker 常駐程式重新啟動前，都不會載入較新版本的 Docker。將執行個體重新開機，或在執行個體上執行下列命令，以重新啟動：
>
> - Amazon ECS 最佳化 Amazon Linux 2 AMI：
>
> ```
> sudo systemctl restart docker
> ```
>
> - Amazon ECS 最佳化 Amazon Linux AMI：
>
> ```
> sudo service docker restart && sudo start ecs
> ```

使用 **UpdateContainerAgent** API 操作更新 Amazon ECS 容器代理程式

> **Important**
>
> 僅在 Amazon ECS 最佳化 AMI 的 Linux 變體上支援 `UpdateContainerAgent` API，但 Amazon ECS 最佳化 Amazon Linux 2 (arm64) AMI 除外。對於使用 Amazon ECS 最佳化 Amazon Linux 2 (arm64) AMI 的容器執行個體，請更新 `ecs-init` 套件以更新代理程式。針對在其他作業系統上執行的容器執行個體，請參閱「手動更新 Amazon ECS 容器代理程式 (適

用於非 Amazon ECS 最佳化 AMI)」。如果您正在使用 Windows 容器執行個體，我們建議您啟動新的容器執行個體，以更新您 Windows 叢集中的代理程式版本。

當您透過主控台或使用或 AWS SDK 要求代理程式更新時，UpdateContainerAgentAPI 程序便會開始。 AWS CLI Amazon ECS 會根據最新可用的代理程式版本檢查您目前的代理程式版本，以及是否可以進行更新。若無法取得更新 (例如若代理已在執行最近的版本)，便會傳回 NoUpdateAvailableException。

上圖顯示更新程序中的階段如下：

PENDING

有可用的代理更新，並已啟動更新程序。

STAGING

代理已開始下載代理更新。若代理無法下載更新，或更新的內容不正確或已損毀，則代理會傳送失敗的通知，且更新的狀態會轉換成 FAILED 狀態。

STAGED

代理下載已完成並已驗證代理內容。

UPDATING

ecs-init 服務已重新啟動，並使用最新版本的代理。若代理程式因為某些原因無法重新啟動，則更新會轉換為 FAILED 狀態；否則，代理程式會通知 Amazon ECS 更新已完成。

Note

代理更新不適用於 Windows 容器執行個體。我們建議您啟動新的容器執行個體，以更新您 Windows 叢集中的代理版本。

若要在主控台的 Amazon ECS 最佳化 AMI 上更新 Amazon ECS 容器代理程式

1. 開啟主控台，網址為 https://console.aws.amazon.com/ecs/v2。
2. 從導覽列中選擇註冊外部執行個體所在的區域。
3. 在導覽窗格中，選擇 Clusters (叢集)，然後選取叢集。
4. 在 Cluster : ***name*** (叢集：名稱) 頁面上，選擇 Infrastructure (基礎基礎設施) 索引標籤。

5. 在容器執行個體下，選取要更新的執行個體，然後選擇動作、更新代理程式。

手動更新 Amazon ECS 容器代理程式 (適用於非 Amazon ECS 最佳化 AMI)

若要手動更新 Amazon ECS 容器代理程式 (適用於非 Amazon ECS 最佳化 AMI)

> **Note**
>
> 代理更新不適用於 Windows 容器執行個體。我們建議您啟動新的容器執行個體，以更新您 Windows 叢集中的代理版本。

1. 透過 SSH 登入您的容器執行個體。
2. 檢查以查看您的代理是否是使用 ECS_DATADIR 環境變數儲存其狀態。

   ```
   ubuntu:~$ docker inspect ecs-agent | grep ECS_DATADIR
   ```

   輸出：

   ```
   "ECS_DATADIR=/data",
   ```

   > **Important**
   >
   > 若先前的命令並未傳回 ECS_DATADIR 環境變數，您必須停止任何在此容器執行個體上執行的任務，才能更新您的代理。較新的代理會使用 ECS_DATADIR 環境變數儲存其狀態，讓您可以在任務執行中時更新它們，而不會有任何問題。

3. 停用 Amazon ECS 容器代理程式。

   ```
   ubuntu:~$ docker stop ecs-agent
   ```

4. 刪除代理容器。

   ```
   ubuntu:~$ docker rm ecs-agent
   ```

5. 確認 /etc/ecs 目錄和 Amazon ECS 容器代理程式組態檔案存在於 /etc/ecs/ecs.config。

   ```
   ubuntu:~$ sudo mkdir -p /etc/ecs && sudo touch /etc/ecs/ecs.config
   ```

6. 編輯 /etc/ecs/ecs.config 檔案，並確保其至少包含以下變數宣告。若您不希望您的容器執行個體使用預設叢集註冊，請將叢集名稱指定為 ECS_CLUSTER 的值。

```
ECS_DATADIR=/data
ECS_ENABLE_TASK_IAM_ROLE=true
ECS_ENABLE_TASK_IAM_ROLE_NETWORK_HOST=true
ECS_LOGFILE=/log/ecs-agent.log
ECS_AVAILABLE_LOGGING_DRIVERS=["json-file","awslogs"]
ECS_LOGLEVEL=info
ECS_CLUSTER=default
```

如需這些和其他代理執行時間選項的詳細資訊，請參閱「Amazon ECS 容器代理程式組態」。

> **Note**
>
> 您可以選擇性地將您的代理程式環境變數存放在 Amazon S3 中 (可在啟動時使用 Amazon EC2 使用者資料將其下載到您的容器執行個體)。針對敏感性資訊 (例如私有存放庫的身分驗證登入資料)，此為建議選項。如需詳細資訊，請參閱在 Amazon S3 中存儲亞馬遜 ECS 容器實例配置及在 Amazon ECS 中使用非AWS 容器映像。

7. 從 Amazon Elastic Container Registry Public 取出最新的 Amazon ECS 容器代理程式映像。

```
ubuntu:~$ docker pull public.ecr.aws/ecs/amazon-ecs-agent:latest
```

輸出：

```
Pulling repository amazon/amazon-ecs-agent
a5a56a5e13dc: Download complete
511136ea3c5a: Download complete
9950b5d678a1: Download complete
c48ddcf21b63: Download complete
Status: Image is up to date for amazon/amazon-ecs-agent:latest
```

8. 在您的容器執行個體上執行最新的 Amazon ECS 容器代理程式。

> **Note**
>
> 使用 Docker 重新啟動政策或處理序管理員 (例如 upstart 或 systemd) 將容器代理程式做為服務或常駐程式處理，並確保在結束後重新啟動它。如需詳細資訊，請參閱 Docker 文

件中的 Automatically start containers 和 Restart policies。Amazon ECS 最佳化 AMI 會針對此目的使用 ecs-init RPM，而且您可以在上檢視此 RPM 的原始程式碼。 GitHub

以下代理程式執行命令範例分成獨立的各行來顯示每個選項。如需這些和其他代理執行時間選項的詳細資訊，請參閱「Amazon ECS 容器代理程式組態」。

> **Important**
>
> 針對啟用 SELinux 的作業系統，需要在您的 docker run 命令中包含 --privileged 選項。此外，針對啟用 SELinux 的容器執行個體，我們建議您為 /log 和 /data 磁碟區掛載新增 :Z 選項。但是，這些磁碟區的主機掛載必須在您執行命令前存在，否則您會接收到 no such file or directory 錯誤。若您在啟用 SELinux 的容器執行個體上執行 Amazon ECS 代理程式時遭遇困難，請採取以下動作：
>
> - 在您的容器執行個體上建立主機磁碟區掛載點。
>
>   ```
>   ubuntu:~$ sudo mkdir -p /var/log/ecs /var/lib/ecs/data
>   ```
>
> - 為以下 docker run 命令新增 --privileged 選項。
> - 為以下 docker run 命令的 /log 和 /data 容器磁碟區掛載附加 :Z 選項 (例如，--volume=/var/log/ecs/:/log:Z)。

```
ubuntu:~$ sudo docker run --name ecs-agent \
--detach=true \
--restart=on-failure:10 \
--volume=/var/run:/var/run \
--volume=/var/log/ecs/:/log \
--volume=/var/lib/ecs/data:/data \
--volume=/etc/ecs:/etc/ecs \
--volume=/etc/ecs:/etc/ecs/pki \
--net=host \
--env-file=/etc/ecs/ecs.config \
amazon/amazon-ecs-agent:latest
```

> **Note**
>
> 若您接收到 `Error response from daemon: Cannot start container` 訊息，您可以使用 sudo docker rm ecs-agent 命令刪除失敗的容器，並再度嘗試執行代理程式。

Amazon ECS 優化視窗 AMI

Amazon ECS 最佳化 AMI 已預先設定好執行 Amazon ECS 工作負載所需的必要元件。雖然您可以建立自己的容器執行個體 AMI，以符合在 Amazon ECS 上執行容器化工作負載所需的基本規格，但 Amazon ECS 最佳化 AMI 會由工程師在 Amazon ECS 上預先設定和測試。 AWS 這是讓您快速上手並在 AWS 上執行容器最簡單的方法。

可以透過程式設計方式擷取每個變體的 Amazon ECS 最佳化 AMI 中繼資料，包括 AMI 名稱、Amazon ECS 容器代理程式版本，以及包含 Docker 版本的 Amazon ECS 執行時間版本。如需詳細資訊，請參閱 the section called “擷取 Amazon EC 最佳化視窗 AMI 中繼資料”。

您可以訂閱 Windows AMI Amazon SNS 主題，以便在發行新的 AMI 或 AMI 版本被標記為私有時收到通知。如需詳細資訊，請參閱 訂閱 Amazon ECS 最佳化視窗 AMI 更新通知。

> **Important**
>
> 8 月之後生產的所有 ECS 最佳化 AMI 變體，將從 Docker EE (Mirantis) 遷移至 Docker CE (Moby 專案)。
> 為確保客戶預設擁有最新的安全更新，Amazon ECS 至少維護最後三個 Windows Amazon ECS 最佳化 AMI。在發佈新的 Windows Amazon ECS 最佳化 AMI 之後，Amazon ECS 使舊的 Windows Amazon ECS 最佳化 AMI 變為私有。如果有您需要存取的私有 AMI，請向 Cloud Support 提出票證，從而通知我們。

Amazon ECS 最佳化 AMI 變體

下列 Amazon ECS 最佳化 AMI 的 Windows Server 變體適用於您的 Amazon EC2 執行個體。

> **Important**
>
> 8 月之後生產的所有 ECS 最佳化 AMI 變體，將從 Docker EE (Mirantis) 遷移至 Docker CE (Moby 專案)。

- Amazon ECS 最佳化 Windows Server 2022 Full AMI
- Amazon ECS 最佳化 Windows Server 2022 Core AMI
- Amazon ECS 最佳化 Windows Server 2019 Full AMI
- Amazon ECS 最佳化 Windows Server 2019 Core AMI
- Amazon ECS 最佳化 Windows Server 2016 Full AMI

> **Important**
>
> 視窗伺服器 2016 不支援最新的泊塢視窗版本，例如 25.x.x。因此 Windows 伺服器 2016 年完整 AMI 將不會收到泊塢視窗執行階段的安全性或錯誤修補程式。我們建議您移至下列其中一個 Windows 平台：
>
> - Windows Server 2022 Full
> - Windows Server 2022 Core
> - Windows Server 2019 Full
> - Windows Server 2019 Core

在 2022 年 8 月 9 日，Amazon ECS 最佳化的 Windows Server 20H2 Core AMI 已屆支援終止日期。不再發佈此 AMI 的新版本。如需詳細資訊，請參閱 Windows Server 發行版本資訊。

Windows Server 2022、Windows Server 2019 和 Windows Server 2016 是長期維護通道 (LTSC) 的發行版本。Windows Server 20H2 是半年通道 (SAC) 的發行版本。如需詳細資訊，請參閱 Windows Server 發行版本資訊。

考量事項

以下是一些您應該知道的 Amazon EC2 Windows 容器和 Amazon ECS 有關事項。

- Windows 容器無法在 Linux 容器執行個體上執行，反之亦然。為更合理放置 Windows 和 Linux 任務，將 Windows 和 Linux 容器執行個體放置在不同的叢集中，而且只在 Windows 叢集中放置 Windows 任務。您可以設定下列置放限制：`memberOf(ecs.os-type=='windows')`，以確保 Windows 任務定義只放置在 Windows 執行個體。
- Windows 容器支援使用 EC2 和 Fargate 啟動類型的任務。
- Windows 容器和容器執行個體無法支援所有可供 Linux 容器和容器執行個體使用的任務定義參數。有些參數完全不予以支援，有些參數在 Windows 的行為表現和在 Linux 上不同。如需詳細資訊，請參閱 執行 Windows 的 EC2 執行個體的 Amazon ECS 任務定義差異。

- 針對任務功能的 IAM 角色，您需要設定您的 Windows 容器執行個體，在啟動時允許該功能。您的容器在使用此功能時必須執行某些提供的 PowerShell 程式碼。如需詳細資訊，請參閱 Amazon EC2 窗口實例附加配置。
- 任務的 IAM 角色功能會使用登入資料代理將登入資料提供給容器。此登入資料代理佔用了容器執行個體的連接埠 80，所以如果您使用任務的 IAM 角色，連接埠 80 就不提供任務使用。對於 Web 服務容器，您可以使用 Application Load Balancer 和動態連接埠映射，向您的容器提供標準的 HTTP 連接埠 80 連線。如需詳細資訊，請參閱 使用負載平衡來分配 Amazon ECS 服務流量。
- Windows 伺服器 Docker 映像很大 (9 GiB)。所以您的 Windows 容器執行個體需要比 Linux 容器執行個體還多的儲存空間。
- 若要在 Windows 伺服器上執行 Windows 容器，容器的基本映像作業系統版本必須與主機的版本相符。如需詳細資訊，請參閱 Microsoft 文件網站上的 Windows 容器版本相容性。如果您的叢集執行多個 Windows 版本，您可以使用置放條件限制：`memberOf(attribute:ecs.os-family == WINDOWS_SERVER_<OS_Release>_<FULL or CORE>)`，確保將任務置放在執行相同版本的 EC2 執行個體上。如需詳細資訊，請參閱 the section called “擷取 Amazon EC 最佳化視窗 AMI 中繼資料”。

擷取 Amazon EC 最佳化視窗 AMI 中繼資料

透過查詢 Systems Manager 參數存放區 API，可以程式設計方式擷取 Amazon ECS 最佳化 AMI 的每個變體的 AMI ID、映像名稱、作業系統、容器代理程式版本以及執行時間版本。如需有關 Systems Manager 參數存放區 API 的詳細資訊，請參閱GetParameters和GetParametersByPath。

> **Note**
>
> 您的管理使用者必須擁有以下 IAM 許可，才能擷取 Amazon ECS 最佳化 AMI 中繼資料。已將這些權限新增至 `AmazonECS_FullAccess` IAM 政策。
>
> - SSM：GetParameters
> - SSM：GetParameter
> - SSM：GetParametersByPath

Systems Manager 參數存放區參數格式。

> **Note**
>
> 下列 Systems Manager 參數存放區 API 參數已淘汰，不應該用來擷取最新的 Windows AMI：

- /aws/service/ecs/optimized-ami/windows_server/2016/english/full/recommended/image_id
- /aws/service/ecs/optimized-ami/windows_server/2019/english/full/recommended/image_id

以下是每個 Amazon ECS 最佳化 AMI 變體的參數名稱格式。

- Windows Server 2022 Full AMI 中繼資料：

```
/aws/service/ami-windows-latest/Windows_Server-2022-English-Full-ECS_Optimized
```

- Windows Server 2022 Core AMI 中繼資料：

```
/aws/service/ami-windows-latest/Windows_Server-2022-English-Core-ECS_Optimized
```

- Windows Server 2019 Full AMI 中繼資料：

```
/aws/service/ami-windows-latest/Windows_Server-2019-English-Full-ECS_Optimized
```

- Windows Server 2019 Core AMI 中繼資料：

```
/aws/service/ami-windows-latest/Windows_Server-2019-English-Core-ECS_Optimized
```

- Windows Server 2016 Full AMI 中繼資料：

```
/aws/service/ami-windows-latest/Windows_Server-2016-English-Full-ECS_Optimized
```

下列參數名稱格式擷取最新穩定 Windows Server 2019 Full AMI 的中繼資料。

```
aws ssm get-parameters --names /aws/service/ami-windows-latest/Windows_Server-2019-English-Full-ECS_Optimized
```

以下是傳回參數值的 JSON 物件範例。

```
{
    "Parameters": [
        {
```

```
            "Name": "/aws/service/ami-windows-latest/Windows_Server-2019-English-Full-
ECS_Optimized",
            "Type": "String",
            "Value": "{\"image_name\":\"Windows_Server-2019-English-Full-
ECS_Optimized-2023.06.13\",\"image_id\":\"ami-0debc1fb48e4aee16\",\"ecs_runtime_version
\":\"Docker (CE) version 20.10.21\",\"ecs_agent_version\":\"1.72.0\"}",
            "Version": 58,
            "LastModifiedDate": "2023-06-22T19:37:37.841000-04:00",
            "ARN": "arn:aws:ssm:us-east-1::parameter/aws/service/ami-windows-latest/
Windows_Server-2019-English-Full-ECS_Optimized",
            "DataType": "text"
        }
    ],
    "InvalidParameters": []
}
```


輸出中的每個欄位可做為子參數查詢。透過將子參數名稱附加至所選 AMI 路徑來建構子參數的參數路徑。下列子參數可供使用：

- `schema_version`
- `image_id`
- `image_name`
- `os`
- `ecs_agent_version`
- `ecs_runtime_version`

範例

下列範例顯示您可擷取每個 Amazon ECS 最佳化 AMI 變體中繼資料的方法。

擷取最新的穩定 Amazon ECS 最佳化 AMI 的中繼資料

您可以使用下 AWS CLI 列命令擷取最新穩定的 Amazon ECS 最 AWS CLI 佳化 AMI。

- 對於 Amazon ECS 最佳化 Windows Server 2022 Full AMI：

```
aws ssm get-parameters --names /aws/service/ami-windows-latest/Windows_Server-2022-
English-Full-ECS_Optimized --region us-east-1
```

- 對於 Amazon ECS 最佳化 Windows Server 2022 Core AMI：

```
aws ssm get-parameters --names /aws/service/ami-windows-latest/Windows_Server-2022-English-Core-ECS_Optimized --region us-east-1
```

- 對於 Amazon ECS 最佳化 Windows Server 2019 Full AMI：

```
aws ssm get-parameters --names /aws/service/ami-windows-latest/Windows_Server-2019-English-Full-ECS_Optimized --region us-east-1
```

- 對於 Amazon ECS 最佳化 Windows Server 2019 Core AMI：

```
aws ssm get-parameters --names /aws/service/ami-windows-latest/Windows_Server-2019-English-Core-ECS_Optimized --region us-east-1
```

- 對於 Amazon ECS 最佳化 Windows Server 2016 Full AMI：

```
aws ssm get-parameters --names /aws/service/ami-windows-latest/Windows_Server-2016-English-Full-ECS_Optimized --region us-east-1
```

在模板中使用最新推薦的 Amazon ECS 優化 AMI AWS CloudFormation

透過參考 Systems Manager 參數存放區名稱，可以在 AWS CloudFormation 範本中參考最新建議的 Amazon ECS 最佳化 AMI。

```
Parameters:
  LatestECSOptimizedAMI:
    Description: AMI ID
    Type: AWS::SSM::Parameter::Value<AWS::EC2::Image::Id>
    Default: /aws/service/ami-windows-latest/Windows_Server-2019-English-Full-ECS_Optimized/image_id
```

訂閱 Amazon ECS 最佳化視窗 AMI 更新通知

AWS 針對與視窗伺服器 AMI 相關的通知提供兩個 Amazon SNS 主題 ARN。發行新的 Windows Server AMI 時，其中一個主題會傳送更新通知。先前發行的 Windows Server AMI 設為私有時，另一個主題會傳送通知。雖然這些主題並非針對 Amazon ECS 最佳化 Windows AMI，因為 Amazon ECS 最佳化 Windows AMI 遵循相同的發行排程，但是您可以使用這些通知來指示新的 Amazon ECS 最佳化 Windows AMI 何時更新。如需訂閱 Windows AMI 通知的詳細資訊，請參閱 Amazon EC2 使用者指南中的訂閱視窗 AMI 通知。

> **Note**
>
> 您的使用者或連接至使用者的角色必須擁有 `sns::subscribe` IAM 許可，才可訂閱 Amazon SNS 主題。

Amazon ECS 優化視窗 AMI 版本

檢視 Amazon ECS 最佳化 AMI 的目前和先前版本，以及其對應的 Amazon ECS 容器代理程式、泊塢視窗和套件版本。`ecs-init`

可透過程式設計方式擷取每個變體的 Amazon ECS 最佳化 AMI 中繼資料 (包括 AMI ID)。如需詳細資訊，請參閱 the section called “擷取 Amazon EC 最佳化視窗 AMI 中繼資料”。

以下標籤顯示 Windows Amazon ECS 最佳化 AMI 版本清單。如需在 AWS CloudFormation 範本中參考「Systems Manager 參數存放區」參數的詳細資訊，請參閱在模板中使用最新推薦的 Amazon ECS 優化 AMI AWS CloudFormation。

> **Important**
>
> 為確保客戶預設擁有最新的安全更新，Amazon ECS 至少維護最後三個 Windows Amazon ECS 最佳化 AMI。在發佈新的 Windows Amazon ECS 最佳化 AMI 之後，Amazon ECS 使舊的 Windows Amazon ECS 最佳化 AMI 變為私有。如果有您需要存取的私有 AMI，請向 Cloud Support 提出票證，從而通知我們。
> 視窗伺服器 2016 不支援最新的泊塢視窗版本，例如 25.x.x。因此 Windows 伺服器 2016 年完整 AMI 將不會收到泊塢視窗執行階段的安全性或錯誤修補程式。我們建議您移至下列其中一個 Windows 平台：
>
> - Windows Server 2022 Full
> - Windows Server 2022 Core
> - Windows Server 2019 Full
> - Windows Server 2019 Core

Windows Server 2022 Full AMI versions

下表列出目前和舊版的 Amazon ECS 最佳化 Windows Server 2022 Full AMI 的完整清單及其對應的 Amazon ECS 容器代理程式和 Docker 版本。

Amazon ECS 最佳化 Windows Server 2022 Full AMI	Amazon ECS 容器代理程式版本	Docker 版本	Visibility
視窗伺服器 2022-英文完整版最佳化	1.82.3	25.0.3 (Docker CE)	公有
視窗伺服器 2022-英文完整版最佳化	1.82.2	25.0.3 (Docker CE)	公有
視窗伺服器 2022-英文完整版最佳化	1.82.0	20.10.23 (Docker CE)	公有
視窗伺服器 2022-英文完整版最佳化	1.81.0	20.10.23 (Docker CE)	公有
視窗伺服器 2022-英文完整版最佳化	1.79.2	20.10.23 (Docker CE)	公有
視窗伺服器 2022-英文完整版最佳化	1.79.1	20.10.23 (Docker CE)	私有
視窗伺服器 2022-英文完整版最佳化	1.79.0	20.10.23 (Docker CE)	私有
Windows_Server-2022-English-Full-ECS_Optimized-2023.10.11	1.77.0	20.10.21 (Docker CE)	私有
Windows_Server-2022-English-Full-ECS_Optimized-2023.09.15	1.75.3	20.10.21 (Docker CE)	私有
Windows_Server-2022-English-Full-ECS	1.74.1	20.10.21 (Docker CE)	私有

Amazon ECS 最佳化 Windows Server 2022 Full AMI	Amazon ECS 容器代理程式版本	Docker 版本	Visibility
_Optimized-2023.08.09			
Windows_Server-2022-English-Full-ECS_Optimized-2023.07.11	1.73.1	20.10.21 (Docker CE)	私有
Windows_Server-2022-English-Full-ECS_Optimized-2023.06.13	1.72.0	20.10.21 (Docker CE)	私有
Windows_Server-2022-English-Full-ECS_Optimized-2023.05.18	1.71.1	20.10.21 (Docker CE)	私有
Windows_Server-2022-English-Full-ECS_Optimized-2023.04.18	1.70.2	20.10.21 (Docker CE)	私有
Windows_Server-2022-English-Full-ECS_Optimized-2023.03.21	1.69.0	20.10.21 (Docker CE)	私有
Windows_Server-2022-English-Full-ECS_Optimized-2023.02.21	1.68.2	20.10.21 (Docker CE)	私有

Amazon ECS 最佳化 Windows Server 2022 Full AMI	Amazon ECS 容器代理程式版本	Docker 版本	Visibility
Windows_Server-2022-English-Full-ECS_Optimized-2023.01.11	1.68.0	20.10.21 (Docker CE)	私有
Windows_Server-2022-English-Full-ECS_Optimized-2022.12.14	1.67.2	20.10.21 (Docker CE)	私有
Windows_Server-2022-English-Full-ECS_Optimized-2022.11.09	1.65.1	20.10.21 (Docker CE)	私有
Windows_Server-2022-English-Full-ECS_Optimized-2022.10.12	1.64.0	20.10.17 (Docker CE)	私有
Windows_Server-2022-English-Full-ECS_Optimized-2022.09.22	1.63.1	20.10.17 (Docker CE)	私有
Windows_Server-2022-English-Full-ECS_Optimized-2022.09.14	1.62.2	20.10.17 (Docker CE)	私有
Windows_Server-2022-English-Full-ECS_Optimized-2022.08.15	1.62.1	20.10.9	私有

Amazon ECS 最佳化 Windows Server 2022 Full AMI	Amazon ECS 容器代理程式版本	Docker 版本	Visibility
Windows_Server-2022-English-Full-ECS_Optimized-2022.07.13	1.61.3	20.10.9	私有
Windows_Server-2022-English-Full-ECS_Optimized-2022.06.15	1.61.2	20.10.9	私有
Windows_Server-2022-English-Full-ECS_Optimized-2022.01.18	1.57.1	20.10.9	私有
Windows_Server-2022-English-Full-ECS_Optimized-2021.12.16	1.57.1	20.10.7	私有
Windows_Server-2022-English-Full-ECS_Optimized-2021.11.11	1.57.0	20.10.7	私有
Windows_Server-2022-English-Full-ECS_Optimized-2021.009.23	1.55.3	20.10.7	私有

使用下列 AWS CLI 命令擷取目前的 Amazon ECS 最佳化視窗伺服器 2022 年完整 AMI。

```
aws ssm get-parameters --names /aws/service/ami-windows-latest/Windows_Server-2022-English-Full-ECS_Optimized
```

Windows Server 2022 Core AMI versions

下表列出目前和舊版的 Amazon ECS 最佳化 Windows Server 2022 Core AMI 的完整清單及其對應的 Amazon ECS 容器代理程式和 Docker 版本。

Amazon ECS 最佳化 Windows Server 2022 Core AMI	Amazon ECS 容器代理程式版本	Docker 版本	Visibility
視窗伺服器 2022-英文核心 EC_ 最佳化	1.82.3	25.0.3 (Docker CE)	公有
視窗伺服器 2022-英文核心 EC_ 最佳化	1.82.2	25.0.3 (Docker CE)	公有
視窗伺服器 2022-英文核心 EC_ 最佳化	1.82.0	20.10.23 (Docker CE)	公有
視窗伺服器 2022-英文核心 EC_ 最佳化	1.81.0	20.10.23 (Docker CE)	公有
視窗伺服器 2022-英文核心電腦 _ 最佳化	1.79.2	20.10.23 (Docker CE)	公有
視窗伺服器 2022-英文核心 EC_ 最佳化	1.79.1	20.10.23 (Docker CE)	私有
視窗伺服器 2022-英文核心 EC_ 最佳化	1.79.0	20.10.23 (Docker CE)	私有
Windows_Server-2022-English-Core-ECS_Optimized-2023.10.11	1.77.0	20.10.21 (Docker CE)	私有

Amazon ECS 最佳化 Windows Server 2022 Core AMI	Amazon ECS 容器代理程式版本	Docker 版本	Visibility
Windows_Server-2022-English-Core-ECS_Optimized-2023.09.15	1.75.3	20.10.21 (Docker CE)	私有
Windows_Server-2022-English-Core-ECS_Optimized-2023.08.09	1.74.1	20.10.21 (Docker CE)	私有
Windows_Server-2022-English-Core-ECS_Optimized-2023.07.11	1.73.1	20.10.21 (Docker CE)	私有
Windows_Server-2022-English-Core-ECS_Optimized-2023.06.13	1.72.0	20.10.21 (Docker CE)	私有
Windows_Server-2022-English-Core-ECS_Optimized-2023.05.18	1.71.1	20.10.21 (Docker CE)	私有
Windows_Server-2022-English-Core-ECS_Optimized-2023.04.18	1.70.2	20.10.21 (Docker CE)	私有
Windows_Server-2022-English-Core-ECS_Optimized-2023.03.21	1.69.0	20.10.21 (Docker CE)	私有

Amazon ECS 最佳化 Windows Server 2022 Core AMI	Amazon ECS 容器代理程式版本	Docker 版本	Visibility
Windows_Server-2022-English-Core-ECS_Optimized-2023.02.21	1.68.2	20.10.21 (Docker CE)	私有
Windows_Server-2022-English-Core-ECS_Optimized-2023.01.11	1.68.0	20.10.21 (Docker CE)	私有
Windows_Server-2022-English-Core-ECS_Optimized-2022.12.14	1.67.2	20.10.21 (Docker CE)	私有
Windows_Server-2022-English-Core-ECS_Optimized-2022.11.09	1.65.1	20.10.21 (Docker CE)	私有
Windows_Server-2022-English-Core-ECS_Optimized-2022.10.12	1.64.0	20.10.17 (Docker CE)	私有
Windows_Server-2022-English-Core-ECS_Optimized-2022.09.22	1.63.1	20.10.17 (Docker CE)	私有
Windows_Server-2022-English-Core-ECS_Optimized-2022.09.14	1.62.2	20.10.17 (Docker CE)	私有

Amazon ECS 最佳化 Windows Server 2022 Core AMI	Amazon ECS 容器代理程式版本	Docker 版本	Visibility
Windows_Server-2022-English-Core-ECS_Optimized-2022.08.15	1.62.1	20.10.9	私有
Windows_Server-2022-English-Core-ECS_Optimized-2022.07.13	1.61.3	20.10.9	私有
Windows_Server-2022-English-Core-ECS_Optimized-2022.06.15	1.61.2	20.10.9	私有
Windows_Server-2022-English-Core-ECS_Optimized-2021.12.16	1.57.1	20.10.7	私有
Windows_Server-2022-English-Core-ECS_Optimized-2021.11.11	1.57.0	20.10.7	私有
Windows_Server-2022-English-Core-ECS_Optimized-2021.009.23	1.55.3	20.10.7	私有

使用下列 AWS CLI 命令擷取目前的 Amazon ECS 最佳化視窗伺服器 2022 年完整 AMI。

```
aws ssm get-parameters --names /aws/service/ami-windows-latest/Windows_Server-2022-English-Core-ECS_Optimized
```

Windows Server 2019 Full AMI versions

下表列出目前和舊版的 Amazon ECS 最佳化 Windows Server 2019 Full AMI 的完整清單及其對應的 Amazon ECS 容器代理程式和 Docker 版本。

Amazon ECS 最佳化 Windows Server 2019 Full AMI	Amazon ECS 容器代理程式版本	Docker 版本	Visibility
視窗伺服器 2019 年英文全文版最佳化	1.82.3	25.0.3 (Docker CE)	公有
視窗伺服器 2019-英文完整版優化	1.82.2	25.0.3 (Docker CE)	公有
視窗伺服器-英文全文版最佳化	1.82.0	20.10.23 (Docker CE)	公有
視窗伺服器 2019 年英文全文版最佳化	1.81.0	20.10.23 (Docker CE)	公有
視窗伺服器 2019 年全文版最佳化	1.79.2	20.10.23 (Docker CE)	公有
視窗伺服器 2019 年全文版最佳化	1.79.1	20.10.23 (Docker CE)	私有
視窗伺服器 2019 年英文全文版最佳化	1.79.0	20.10.23 (Docker CE)	私有
Windows_Server-2019-English-Full-ECS_Optimized-2023.10.11	1.77.0	20.10.21 (Docker CE)	私有

Amazon ECS 最佳化 Windows Server 2019 Full AMI	Amazon ECS 容器代理程式版本	Docker 版本	Visibility
Windows_Server-2019-English-Full-ECS_Optimized-2023.09.15	1.75.3	20.10.21 (Docker CE)	私有
Windows_Server-2019-English-Full-ECS_Optimized-2023.08.09	1.74.1	20.10.21 (Docker CE)	私有
Windows_Server-2019-English-Full-ECS_Optimized-2023.07.11	1.73.1	20.10.21 (Docker CE)	私有
Windows_Server-2019-English-Full-ECS_Optimized-2023.06.13	1.72.0	20.10.21 (Docker CE)	私有
Windows_Server-2019-English-Full-ECS_Optimized-2023.05.18	1.71.1	20.10.21 (Docker CE)	私有
Windows_Server-2019-English-Full-ECS_Optimized-2023.04.18	1.70.2	20.10.21 (Docker CE)	私有
Windows_Server-2019-English-Full-ECS_Optimized-2023.03.21	1.69.0	20.10.21 (Docker CE)	私有

Amazon ECS 最佳化 Windows Server 2019 Full AMI	Amazon ECS 容器代理程式版本	Docker 版本	Visibility
Windows_Server-2019-English-Full-ECS_Optimized-2023.02.21	1.68.2	20.10.21 (Docker CE)	私有
Windows_Server-2019-English-Full-ECS_Optimized-2023.01.11	1.68.0	20.10.21 (Docker CE)	私有
Windows_Server-2019-English-Full-ECS_Optimized-2022.12.14	1.67.2	20.10.21 (Docker CE)	私有
Windows_Server-2019-English-Full-ECS_Optimized-2022.11.09	1.65.1	20.10.21 (Docker CE)	私有
Windows_Server-2019-English-Full-ECS_Optimized-2022.10.12	1.64.0	20.10.17 (Docker CE)	私有
Windows_Server-2019-English-Full-ECS_Optimized-2022.09.22	1.63.1	20.10.17 (Docker CE)	私有
Windows_Server-2019-English-Full-ECS_Optimized-2022.09.14	1.62.2	20.10.17 (Docker CE)	私有

Amazon ECS 最佳化 Windows Server 2019 Full AMI	Amazon ECS 容器代理程式版本	Docker 版本	Visibility
Windows_Server-2019-English-Full-ECS_Optimized-2022.08.15	1.62.1	20.10.9	私有
Windows_Server-2019-English-Full-ECS_Optimized-2022.07.13	1.61.3	20.10.9	私有
Windows_Server-2019-English-Full-ECS_Optimized-2022.06.15	1.61.2	20.10.9	私有
Windows_Server-2019-English-Full-ECS_Optimized-2022.01.18	1.57.1	20.10.9	私有
Windows_Server-2019-English-Full-ECS_Optimized-2021.12.16	1.57.1	20.10.7	私有
Windows_Server-2019-English-Full-ECS_Optimized-2021.11.11	1.57.0	20.10.7	私有
Windows_Server-2019-English-Full-ECS_Optimized-2021.09.23	1.55.3	20.10.7	私有

Amazon ECS 最佳化 Windows Server 2019 Full AMI	Amazon ECS 容器代理程式版本	Docker 版本	Visibility
Windows_Server-2019-English-Full-ECS_Optimized-2021.08.12	1.55.0	20.10.6	公有
Windows_Server-2019-English-Full-ECS_Optimized-2021.07.13	1.54.02	20.10.6	私有
Windows_Server-2019-English-Full-ECS_Optimized-2021.07.08	1.54.0	20.10.5	私有
Windows_Server-2019-English-Full-ECS_Optimized-2021.06.11	1.53.0	20.10.5	私有
Windows_Server-2019-English-Full-ECS_Optimized-2021.05.21	1.52.2	20.10.4	私有
Windows_Server-2019-English-Full-ECS_Optimized-2021.04.14	1.51.0	20.10.0	私有
Windows_Server-2019-English-Full-ECS_Optimized-2021.03.11	1.50.2	19.03.14	私有

Amazon ECS 最佳化 Windows Server 2019 Full AMI	Amazon ECS 容器代理程式版本	Docker 版本	Visibility
Windows_Server-2019-English-Full-ECS_Optimized-2021.02.10	1.50.0	19.03.14	私有
Windows_Server-2019-English-Full-ECS_Optimized-2021.01.13	1.49.0	19.03.14	私有
Windows_Server-2019-English-Full-ECS_Optimized-2020.11.18	1.48.0	19.03.13	私有
Windows_Server-2019-English-Full-ECS_Optimized-2020.11.06	1.47.0	19.03.11	私有
Windows_Server-2019-English-Full-ECS_Optimized-2020.10.14	1.45.0	19.03.11	私有
Windows_Server-2019-English-Full-ECS_Optimized-2020.08.12	1.43.0	19.03.11	私有
Windows_Server-2019-English-Full-ECS_Optimized-2020.07.15	1.41.1	19.03.5	私有

Amazon ECS 最佳化 Windows Server 2019 Full AMI	Amazon ECS 容器代理程式版本	Docker 版本	Visibility
Windows_Server-2019-English-Full-ECS_Optimized-2020.06.11	1.40.0	19.03.5	私有
Windows_Server-2019-English-Full-ECS_Optimized-2020.05.14	1.39.0	19.03.5	私有
Windows_Server-2019-English-Full-ECS_Optimized-2020.01.15	1.35.0	19.03.5	私有
Windows_Server-2019-English-Full-ECS_Optimized-2019.12.16	1.34.0	19.03.5	私有
Windows_Server-2019-English-Full-ECS_Optimized-2019.11.25	1.34.0	19.03.4	私有
Windows_Server-2019-English-Full-ECS_Optimized-2019.11.13	1.32.1	19.03.4	私有
Windows_Server-2019-English-Full-ECS_Optimized-2019.10.09	1.32.0	19.03.2	私有

Amazon ECS 最佳化 Windows Server 2019 Full AMI	Amazon ECS 容器代理程式版本	Docker 版本	Visibility
Windows_Server-2019-English-Full-ECS_Optimized-2019.09.11	1.30.0	19.03.1	私有
Windows_Server-2019-English-Full-ECS_Optimized-2019.08.16	1.29.1	19.03.1	私有
Windows_Server-2019-English-Full-ECS_Optimized-2019.07.19	1.29.0	18.09.8	私有
Windows_Server-2019-English-Full-ECS_Optimized-2019.05.10	1.27.0	18.09.4	私有

使用以下 AWS CLI 命令來檢索當前 Amazon ECS 優化視窗服務器 2019 年完整 AMI。

```
aws ssm get-parameters --names /aws/service/ami-windows-latest/Windows_Server-2019-English-Full-ECS_Optimized
```

Windows Server 2019 Core AMI versions

> **Important**
>
> 下表列出目前和舊版的 Amazon ECS 最佳化 Windows Server 2019 Core AMI 的完整清單及其對應的 Amazon ECS 容器代理程式和 Docker 版本。

Amazon ECS 最佳化 Windows Server 2019 Core AMI	Amazon ECS 容器代理程式版本	Docker 版本	Visibility
視窗伺服器-英文核心 EC_ 最佳化	1.82.3	25.0.3 (Docker CE)	公有
視窗伺服器-英文核心 EC_ 最佳化	1.82.2	25.0.3 (Docker CE)	公有
視窗伺服器-英文核心 EC_ 最佳化	1.82.0	20.10.23 (Docker CE)	公有
視窗伺服器-英文核心 EC_ 最佳化	1.81.0	20.10.23 (Docker CE)	公有
視窗伺服器 -2019 年英文核心網路最佳化	1.79.2	20.10.23 (Docker CE)	公有
視窗伺服器-英文核心 EC_ 最佳化	1.79.1	20.10.23 (Docker CE)	私有
視窗伺服器-英文核心 EC_ 最佳化	1.79.0	20.10.23 (Docker CE)	私有
Windows_Server-2019-English-Core-ECS_Optimized-2023.10.11	1.77.0	20.10.21 (Docker CE)	私有
Windows_Server-2019-English-Core-ECS_Optimized-2023.09.15	1.75.3	20.10.21 (Docker CE)	私有

Amazon ECS 最佳化 Windows Server 2019 Core AMI	Amazon ECS 容器代理程式版本	Docker 版本	Visibility
Windows_Server-2019-English-Core-ECS_Optimized-2023.08.09	1.74.1	20.10.21 (Docker CE)	私有
Windows_Server-2019-English-Core-ECS_Optimized-2023.07.11	1.73.1	20.10.21 (Docker CE)	私有
Windows_Server-2019-English-Core-ECS_Optimized-2023.06.13	1.72.0	20.10.21 (Docker CE)	私有
Windows_Server-2019-English-Core-ECS_Optimized-2023.05.18	1.71.1	20.10.21 (Docker CE)	私有
Windows_Server-2019-English-Core-ECS_Optimized-2023.04.18	1.70.2	20.10.21 (Docker CE)	私有

Amazon ECS 最佳化 Windows Server 2019 Core AMI	Amazon ECS 容器代理程式版本	Docker 版本	Visibility
Windows_Server-2019-English-Core-ECS_Optimized-2023.03.21	1.69.0	20.10.21 (Docker CE)	私有
Windows_Server-2019-English-Core-ECS_Optimized-2023.02.21	1.68.2	20.10.21 (Docker CE)	私有
Windows_Server-2019-English-Core-ECS_Optimized-2023.01.11	1.68.0	20.10.21 (Docker CE)	私有
Windows_Server-2019-English-Core-ECS_Optimized-2022.12.14	1.67.2	20.10.21 (Docker CE)	私有
Windows_Server-2019-English-Core-ECS_Optimized-2022.11.09	1.65.1	20.10.21 (Docker CE)	私有

Amazon ECS 最佳化 Windows Server 2019 Core AMI	Amazon ECS 容器代理程式版本	Docker 版本	Visibility
Windows_Server-2019-English-Core-ECS_Optimized-2022.10.12	1.64.0	20.10.17 (Docker CE)	私有
Windows_Server-2019-English-Core-ECS_Optimized-2022.09.22	1.63.1	20.10.17 (Docker CE)	私有
Windows_Server-2019-English-Core-ECS_Optimized-2022.09.14	1.62.2	20.10.17 (Docker CE)	私有
Windows_Server-2019-English-Core-ECS_Optimized-2022.08.15	1.62.1	20.10.9	私有
Windows_Server-2019-English-Core-ECS_Optimized-2022.07.13	1.61.3	20.10.9	私有

Amazon ECS 最佳化 Windows Server 2019 Core AMI	Amazon ECS 容器代理程式版本	Docker 版本	Visibility
Windows_Server-2019-English-Core-ECS_Optimized-2022.06.15	1.61.2	20.10.9	私有
Windows_Server-2019-English-Core-ECS_Optimized-2022.01.18	1.57.1	20.10.9	私有
Windows_Server-2019-English-Core-ECS_Optimized-2021.12.16	1.57.1	20.10.7	私有
Windows_Server-2019-English-Core-ECS_Optimized-2021.11.11	1.57.0	20.10.7	私有
Windows_Server-2019-English-Core-ECS_Optimized-2021.09.23	1.55.3	20.10.7	私有

Amazon ECS 最佳化 Windows Server 2019 Core AMI	Amazon ECS 容器代理程式版本	Docker 版本	Visibility
Windows_Server-2019-English-Core-ECS_Optimized-2021.08.12	1.55.0	20.10.6	私有
Windows_Server-2019-English-Core-ECS_Optimized-2021.07.13	1.54.02	20.10.6	私有
Windows_Server-2019-English-Core-ECS_Optimized-2021.07.08	1.54.0	20.10.6	私有
Windows_Server-2019-English-Core-ECS_Optimized-2021.06.11	1.53.0	20.10.5	私有
Windows_Server-2019-English-Core-ECS_Optimized-2021.05.21	1.52.2	20.10.4	私有

Amazon ECS 最佳化 Windows Server 2019 Core AMI	Amazon ECS 容器代理程式版本	Docker 版本	Visibility
Windows_Server-2019-English-Core-ECS_Optimized-2021.04.14	1.51.0	20.10.0	私有
Windows_Server-2019-English-Core-ECS_Optimized-2021.03.11	1.50.2	19.03.14	私有
Windows_Server-2019-English-Core-ECS_Optimized-2021.02.10	1.50.0	19.03.14	私有
Windows_Server-2019-English-Core-ECS_Optimized-2021.01.13	1.49.0	19.03.14	私有
Windows_Server-2019-English-Core-ECS_Optimized-2020.11.18	1.48.0	19.03.13	私有

Amazon ECS 最佳化 Windows Server 2019 Core AMI	Amazon ECS 容器代理程式版本	Docker 版本	Visibility
Windows_Server-2019-English-Core-ECS_Optimized-2020.11.06	1.47.0	19.03.11	私有
Windows_Server-2019-English-Core-ECS_Optimized-2020.10.14	1.45.0	19.03.11	私有
Windows_Server-2019-English-Core-ECS_Optimized-2020.09.09	1.44.3	19.03.11	私有
Windows_Server-2019-English-Core-ECS_Optimized-2020.08.12	1.43.0	19.03.11	私有
Windows_Server-2019-English-Core-ECS_Optimized-2020.07.15	1.41.1	19.03.5	私有

Amazon ECS 最佳化 Windows Server 2019 Core AMI	Amazon ECS 容器代理程式版本	Docker 版本	Visibility
Windows_Server-2019-English-Core-ECS_Optimized-2020.06.11	1.40.0	19.03.5	私有
Windows_Server-2019-English-Core-ECS_Optimized-2020.05.14	1.39.0	19.03.5	私有
Windows_Server-2019-English-Core-ECS_Optimized-2020.01.15	1.35.0	19.03.5	私有
Windows_Server-2019-English-Core-ECS_Optimized-2019.12.16	1.34.0	19.03.5	私有
Windows_Server-2019-English-Core-ECS_Optimized-2019.11.25	1.34.0	19.03.4	私有

Amazon ECS 最佳化 Windows Server 2019 Core AMI	Amazon ECS 容器代理程式版本	Docker 版本	Visibility
Windows_Server-2019-English-Core-ECS_Optimized-2019.11.13	1.32.1	19.03.4	私有
Windows_Server-2019-English-Core-ECS_Optimized-2019.10.09	1.32.0	19.03.2	私有

使用以下 AWS CLI 命令來檢索當前 Amazon ECS 優化視窗服務器 2019 年完整 AMI。

```
aws ssm get-parameters --names /aws/service/ami-windows-latest/Windows_Server-2019-English-Core-ECS_Optimized
```

Windows Server 2016 Full AMI versions

Important

視窗伺服器 2016 不支援最新的泊塢視窗版本，例如 25.x.x。因此 Windows 伺服器 2016 年完整 AMI 將不會收到泊塢視窗執行階段的安全性或錯誤修補程式。我們建議您移至下列其中一個 Windows 平台：

- Windows Server 2022 Full
- Windows Server 2022 Core
- Windows Server 2019 Full
- Windows Server 2019 Core

下表列出目前和舊版的 Amazon ECS 最佳化 Windows Server 2016 Full AMI 的完整清單及其對應的 Amazon ECS 容器代理程式和 Docker 版本。

Amazon ECS 最佳化 Windows Server 2016 Full AMI	Amazon ECS 容器代理程式版本	Docker 版本	Visibility
視窗伺服器 2016 年英文完整版最佳化	1.82.0	20.10.23 (Docker CE)	公有
視窗伺服器 2016 年英文完整版最佳化	1.81.0	20.10.23 (Docker CE)	公有
視窗伺服器 2016 年中文全文版最佳化	1.79.2	20.10.23 (Docker CE)	公有
視窗伺服器 2016 年英文完整版最佳化	1.79.1	20.10.23 (Docker CE)	公有
視窗伺服器 2016 年英文完整版優化	1.79.0	20.10.23 (Docker CE)	公有
Windows_Server-2016-English-Full-ECS_Optimized-2023.10.11	1.77.0	20.10.21 (Docker CE)	私有
Windows_Server-2016-English-Full-ECS_Optimized-2023.09.15	1.75.3	20.10.21 (Docker CE)	私有
Windows_Server-2016-English-Full-ECS_Optimized-2023.08.09	1.74.1	20.10.21 (Docker CE)	私有
Windows_Server-2016-English-Full-ECS	1.73.1	20.10.21 (Docker CE)	私有

Amazon ECS 最佳化 Windows Server 2016 Full AMI	Amazon ECS 容器代理程式版本	Docker 版本	Visibility
_Optimized-2023.07.11			
Windows_Server-2016-English-Full-ECS_Optimized-2023.06.13	1.72.0	20.10.21 (Docker CE)	私有
Windows_Server-2016-English-Full-ECS_Optimized-2023.05.18	1.71.1	20.10.21 (Docker CE)	私有
Windows_Server-2016-English-Full-ECS_Optimized-2023.04.18	1.70.2	20.10.21 (Docker CE)	私有
Windows_Server-2016-English-Full-ECS_Optimized-2023.03.21	1.69.0	20.10.21 (Docker CE)	私有
Windows_Server-2016-English-Full-ECS_Optimized-2023.02.21	1.68.2	20.10.21 (Docker CE)	私有
Windows_Server-2016-English-Full-ECS_Optimized-2023.01.11	1.68.0	20.10.21 (Docker CE)	私有

Amazon ECS 最佳化 Windows Server 2016 Full AMI	Amazon ECS 容器代理程式版本	Docker 版本	Visibility
Windows_Server-2016-English-Full-ECS_Optimized-2022.12.14	1.67.2	20.10.21 (Docker CE)	私有
Windows_Server-2016-English-Full-ECS_Optimized-2022.11.09	1.65.1	20.10.21 (Docker CE)	私有
Windows_Server-2016-English-Full-ECS_Optimized-2022.10.12	1.64.0	20.10.17 (Docker CE)	私有
Windows_Server-2016-English-Full-ECS_Optimized-2022.09.22	1.63.1	20.10.17 (Docker CE)	私有
Windows_Server-2016-English-Full-ECS_Optimized-2022.09.14	1.62.2	20.10.17 (Docker CE)	私有
Windows_Server-2016-English-Full-ECS_Optimized-2022.08.15	1.62.1	20.10.9	私有
Windows_Server-2016-English-Full-ECS_Optimized-2022.07.13	1.61.3	20.10.9	私有

Amazon ECS 最佳化 Windows Server 2016 Full AMI	Amazon ECS 容器代理程式版本	Docker 版本	Visibility
Windows_Server-2016-English-Full-ECS_Optimized-2022.06.15	1.61.2	20.10.9	私有
Windows_Server-2016-English-Full-ECS_Optimized-2022.01.18	1.57.1	20.10.9	私有
Windows_Server-2016-English-Full-ECS_Optimized-2021.12.16	1.57.1	20.10.7	私有
Windows_Server-2016-English-Full-ECS_Optimized-2021.11.11	1.57.0	20.10.7	私有
Windows_Server-2016-English-Full-ECS_Optimized-2021.09.23	1.55.3	20.10.7	私有
Windows_Server-2016-English-Full-ECS_Optimized-2021.08.12	1.55.0	20.10.6	私有
Windows_Server-2016-English-Full-ECS_Optimized-2021.07.13	1.54.02	20.10.6	私有

Amazon ECS 最佳化 Windows Server 2016 Full AMI	Amazon ECS 容器代理程式版本	Docker 版本	Visibility
Windows_Server-2016-English-Full-ECS_Optimized-2021.07.08	1.54.0	20.10.5	私有
Windows_Server-2016-English-Full-ECS_Optimized-2021.06.11	1.53.0	20.10.5	私有
Windows_Server-2016-English-Full-ECS_Optimized-2021.05.21	1.52.2	20.10.4	私有
Windows_Server-2016-English-Full-ECS_Optimized-2021.04.14	1.51.0	20.10.0	私有
Windows_Server-2016-English-Full-ECS_Optimized-2021.03.11	1.50.2	19.03.14	私有
Windows_Server-2016-English-Full-ECS_Optimized-2021.02.10	1.50.0	19.03.14	私有
Windows_Server-2016-English-Full-ECS_Optimized-2021.01.13	1.49.0	19.03.14	私有

Amazon ECS 最佳化 Windows Server 2016 Full AMI	Amazon ECS 容器代理程式版本	Docker 版本	Visibility
Windows_Server-2016-English-Full-ECS_Optimized-2020.11.18	1.48.0	19.03.13	私有
Windows_Server-2016-English-Full-ECS_Optimized-2020.11.06	1.47.0	19.03.11	私有
Windows_Server-2016-English-Full-ECS_Optimized-2020.10.14	1.45.0	19.03.12	私有
Windows_Server-2016-English-Full-ECS_Optimized-2020.09.09	1.44.3	19.03.11	私有
Windows_Server-2016-English-Full-ECS_Optimized-2020.08.12	1.43.0	19.03.11	私有
Windows_Server-2016-English-Full-ECS_Optimized-2020.07.15	1.41.1	19.03.5	私有
Windows_Server-2016-English-Full-ECS_Optimized-2020.06.11	1.40.0	19.03.5	私有

Amazon ECS 最佳化 Windows Server 2016 Full AMI	Amazon ECS 容器代理程式版本	Docker 版本	Visibility
Windows_Server-2016-English-Full-ECS_Optimized-2020.05.14	1.39.0	19.03.5	私有
Windows_Server-2016-English-Full-ECS_Optimized-2020.01.15	1.35.0	19.03.5	私有
Windows_Server-2016-English-Full-ECS_Optimized-2019.12.16	1.34.0	19.03.5	私有
Windows_Server-2016-English-Full-ECS_Optimized-2019.11.25	1.34.0	19.03.4	私有
Windows_Server-2016-English-Full-ECS_Optimized-2019.11.13	1.32.1	19.03.4	私有
Windows_Server-2016-English-Full-ECS_Optimized-2019.10.09	1.32.0	19.03.2	私有
Windows_Server-2016-English-Full-ECS_Optimized-2019.09.11	1.30.0	19.03.1	私有

Amazon ECS 最佳化 Windows Server 2016 Full AMI	Amazon ECS 容器代理程式版本	Docker 版本	Visibility
Windows_Server-2016-English-Full-ECS_Optimized-2019.08.16	1.29.1	19.03.1	私有
Windows_Server-2016-English-Full-ECS_Optimized-2019.07.19	1.29.0	18.09.8	私有
Windows_Server-2016-English-Full-ECS_Optimized-2019.03.07	1.26.0	18.03.1	私有

使用以下 AWS CLI Amazon ECS 優化視窗服務器 2016 年全 AMI。

```
aws ssm get-parameters --names /aws/service/ami-windows-latest/Windows_Server-2016-English-Full-ECS_Optimized
```

建立自己的 Amazon ECS 最佳化 Windows AMI

使用 EC2 Image Builder 建立您自己的自訂 Amazon ECS 最佳化視窗 AMI。這使得在 Amazon ECS 上使用您自己的授權 Windows AMI 變得更簡單。Amazon ECS 提供受管的 Image Builder 元件，由該元件提供執行 Windows 執行個體所需的系統組態，以託管您的容器。每個 Amazon ECS 受管元件包含特定的容器代理程式和 Docker 版本。您可以自訂映像以使用最新的 Amazon ECS 受管元件，或者如果需要較舊的容器代理程式或 Docker 版本，您可以指定不同的元件。

如需使用 EC2 Image Builder 的完整逐步解說，請參閱 EC2 Image Builder 使用者指南中的 EC2 Image Builder 入門。

在使用 EC2 Image Builder 建立您自己的 Amazon ECS 最佳化 Windows AMI 時，您將建立映像配方。您的映像配方必須符合以下要求：

- 源圖像應基於視窗服務器 2019 核心, 視窗服務器 2019 完整版, 視窗服務器 2022 核心, 或視窗服務器 2022 完整版. 任何其他 Windows 作業系統不受支援，而且可能與元件不相容。
- 指定建置元件時需要 `ecs-optimized-ami-windows` 元件。為確保映像包含最新的安全性更新，建議使用 `update-windows` 元件。

  若要指定不同的元件版本，展開 Versioning options (版本控制選項) 選單，然後指定您要使用的元件版本。如需詳細資訊，請參閱 列出 `ecs-optimized-ami-windows` 元件版本。

列出 **`ecs-optimized-ami-windows`** 元件版本

在建立 EC2 Image Builder 配方並指定 `ecs-optimized-ami-windows` 元件時，您可以使用預設選項，或者您也可以指定特定的元件版本。要判定可用的元件版本，以及元件所包含的 Amazon ECS 容器代理程式和 Docker 版本，您可以使用 AWS Management Console。

列出可用的 **`ecs-optimized-ami-windows`** 元件版本

1. 開啟 EC2 Image Builder 主控台，位於 https://console.aws.amazon.com/imagebuilder/。
2. 在導覽列上選取您在哪個區域建立您的映像。
3. 在導覽窗格的 Saved configurations (已儲存組態) 選單的下方，選擇 Components (元件)。
4. 在 Components (元件) 頁面的搜尋列鍵入 `ecs-optimized-ami-windows`，然後向下拉資格選單並選取 Quick start (Amazon-managed) (Quick start (Amazon 受管))。
5. 使用 Description (描述) 欄來判定元件版本，以及您的映像所需的 Amazon ECS 容器代理程序和 Docker 版本。

Amazon ECS 視窗容器執行個體管理

當您將 EC2 執行個體用於 Amazon ECS 工作負載時，您必須負責維護執行個體。

代理更新不適用於 Windows 容器執行個體。我們建議您啟動新的容器執行個體，以更新您 Windows 叢集中的代理版本。

管理程序

- 啟動 Amazon ECS Windows 容器執行個體
- 引導 Amazon ECS 視窗容器執行個體以傳遞資料
- 針對 Amazon ECS 視窗容器執行個體使用 HTTP 代理伺服器
- 設定 Amazon ECS 視窗容器執行個體以接收競價型執行個體通知

啟動 Amazon ECS Windows 容器執行個體

您的 Amazon ECS 容器執行個體是使用 Amazon EC2 主控台建立的。開始之前，請務必先完成 設定以使用 Amazon ECS。 中的步驟。

如需有關啟動精靈的詳細資訊，請參閱 Amazon EC2 使用者指南中的使用新啟動執行個體精靈啟動執行個體。

您可以使用新的 Amazon EC2 精靈啟動執行個體。您可以使用以下參數列表，並將參數保留不作為預設列出。下列指示會引導您完成每個參數群組。

程序

開始之前，請完成 設定以使用 Amazon ECS。 中的步驟。

1. 前往 https://console.aws.amazon.com/ec2/ 開啟 Amazon EC2 主控台。
2. 在畫面頂端的導覽列中，會顯示目前的「 AWS 區域」(例如，美國東部 (俄亥俄))。選取要在其中啟動執行個體的區域。這項選擇非常重要，因為有些 Amazon EC2 資源可在不同區域間共享，其他則無法。
3. 從 Amazon EC2 主控台儀表板選擇 Launch Instance (啟動執行個體)。

名稱和標籤

執行個體名稱是一個標籤，其中鍵是 Name (名稱)，而值是您指定的名稱。您可以標記執行個體、磁碟區和彈性圖形。對於 Spot 執行個體，您只能標記 Spot 執行個體請求。

指定執行個體名稱和其他標籤是選用的。

- 對於 Name (名稱)，輸入執行個體的描述性名稱。如果您未指定名稱，則可以透過其 ID 來標識執行個體，該 ID 將在您啟動執行個體時自動產生。
- 若要新增其他標籤，請選擇 Add additional tags (新增其他標籤)。選取 Add tag (新增標籤)，然後輸入鍵和值，然後選取要標記的資源類型。為每個要新增的其他標籤重新選擇 Add tag (新增標籤)。

應用程式和作業系統映像 (Amazon Machine Image)

Amazon Machine Image (AMI) 包含建立執行個體所需的資訊。例如，AMI 可能包含作為 Web 伺服器所必需的軟體，例如 Apache 和您的網站。

如需最新的 Amazon ECS 最佳化 AMI 及其值，請參閱《Windows Amazon ECS 最佳化 AMI》。

使用搜尋列尋找由發佈的適合 Amazon ECS 最佳化 AMI。 AWS

1. 根據您的要求，在 Search (搜尋) 列中輸入以下其中一個 AMI，並按下 Enter。
   - Windows_Server-2022-English-Full-ECS_Optimized
   - Windows_Server-2022-English-Core-ECS_Optimized
   - Windows_Server-2019-English-Full-ECS_Optimized
   - Windows_Server-2019-English-Core-ECS_Optimized
   - Windows_Server-2016-English-Full-ECS_Optimized
2. 在 Choose an Amazon Machine Image (AMI) (選擇 Amazon Machine Image (AMI)) 頁面上，選取 Community AMIs (社群 AMI) 索引標籤。
3. 從出現的清單中，選擇具有最新發佈日期的 Microsoft 驗證 AMI，然後按一下 Select (選取)。

執行個體類型

執行個體類型定義執行個體的硬體組態和大小。較大的執行個體類型具有較多的 CPU 和記憶體。如需詳細資訊，請參閱執行個體類型。

- 針對 Instance type (執行個體類型)，選取執行個體的執行個體類型。

  您選取的執行個體類型，會決定您任務執行所在的可用資源。

金鑰對 (登入)

針對 Key pair name (金鑰對名稱)，選擇現有的金鑰對，或選擇 Create new key pair (建立新的金鑰對) 以建立新的金鑰對。

> **Important**
>
> 如果您選擇 Proceed without key pair (Not recommended) (繼續而不使用金鑰對 (不建議)) 選項，您將無法連線到執行個體，除非您選擇已設定為允許使用者透過其他方式登入的 AMI。

Network settings (網路設定)

視需要設定網路設定。

- Networking platform (聯網平台)：選擇 Virtual Private Cloud (VPC) (虛擬私有雲端 (VPC))，然後在 Network interfaces (網路介面) 區段指定子網路。
- VPC：選取要在其中建立安全群組的現有 VPC。

- Subnet (子網)：您可以在與可用區域、Local Zone、Wavelength 區域或 Outpost 相關聯的子網中啟動執行個體。

  若要在可用區域中啟動執行個體，請選取要在當中啟動執行個體的子網。若要建立新的子網，請選擇 Create new subnet (建立新的子網)，以前往 Amazon VPC 主控台。完成後，請返回啟動執行個體精靈並選擇 Refresh (重新整理) 圖示，載入清單中的子網。

  在 Local Zone 中啟動執行個體，選取您在 Local Zone 中建立的子網。

  若要在 Outpost 中啟動執行個體，請在與 Outpost 相關聯的 VPC 中選取子網。
- Auto-assign Public IP (自動指派公有 IP)：如果您的執行個體應從網際網路存取，請確認 Auto-assign Public IP (自動指派公有 IP) 欄位設為 Enable (啟用)。如果不是，請將此欄位設為 Disable (停用)。

  > **Note**
  >
  > 容器執行個體需要存取，才可以與 Amazon ECS 服務端點通訊。可透過介面 VPC 端點或透過具備公有 IP 位址的容器執行個體來實現。
  > 如需介面 VPC 端點的詳細資訊，請參閱 Amazon ECS 介面 VPC 端點 (AWS PrivateLink)
  > 如果您沒有設定介面 VPC 端點，且容器執行個體沒有公有 IP 位址，則它們必須使用網路位址轉譯 (NAT) 來提供此存取。如需詳細資訊，請參閱《Amazon VPC 使用者指南》中的 NAT 閘道和本指南中的 針對 Amazon ECS 容器執行個體使用 HTTP 代理伺服器。
- Firewall (security groups) (防火牆 (安全群組))：使用安全群組定義容器執行個體的防火牆規則。這些規則指定傳遞至容器執行個體的傳入網路流量。所有其他流量都會遭到忽略。
  - 若要選取現有的安全群組，請選擇 Select existing security group (選取現有安全群組)，然後從您在設定以使用 Amazon ECS。建立的安全群組選取您的安全群組。

設定儲存

您選取的 AMI 包含一或多個儲存體磁碟區，包括根磁碟區。您可以指定要連接至執行個體的其他磁碟區。

您可以使用 Simple (簡單) 檢視。

- Storage type (儲存類型)：為您的容器執行個體設定儲存。

  如果您使用的是 Amazon ECS 最佳化 Amazon Linux 2 AMI，則您的執行個體已設定單一 30 GiB 磁碟區 (在作業系統和 Docker 之間共用)。

如果您使用的是 Amazon ECS 最佳化 AMI，則您的執行個體已設定兩個磁碟區。Root (根目錄) 磁碟區供作業系統使用，而第二個 Amazon EBS 磁碟區 (連接到 `/dev/xvdcz`) 則供 Docker 使用。

您可以選擇性地增加或減少您執行個體的磁碟區大小，使其符合您的應用程式需求。

進階詳細資訊

針對 Advanced Details (進階詳細資訊)，展開此區段來檢視欄位，指定執行個體的其他參數。

- Purchasing option (購買選項)：選擇 Request Spot instances (請求 Spot 執行個體) 以請求 Spot 執行個體。您也需要設定與 Spot 執行個體相關的其他欄位。如需詳細資訊，請參閱「Spot 執行個體要求」。

  > **Note**
  >
  > 如果您使用的是 Spot 執行個體，而且看到 `Not available` 訊息，您可能需要選擇不同的執行個體類型。

  .

- IAM instance profile (IAM 執行個體設定檔)：選取您的容器執行個體 IAM 角色。這通常命名為 `ecsInstanceRole`。

  > **Important**
  >
  > 如果您未使用適當的 IAM 許可啟動容器執行個體，Amazon ECS 代理程式就無法連線到叢集。如需詳細資訊，請參閱 Amazon ECS 容器執行個體 IAM 角色。

- (選用) User data (使用者資料)：利用使用者資料設定您的 Amazon ECS 容器執行個體，例如 Amazon ECS 容器代理程式組態中的代理程式環境變數。Amazon EC2 使用者資料指令碼僅於執行個體初次啟動時執行一次。以下是使用者資料用途的常見範例：
  - 您的容器執行個體預設會在您的預設叢集中啟動。若要在非預設的叢集中啟動，請選擇 Advanced Details (進階詳細資訊) 清單。然後，將以下指令碼貼入 User data (使用者資料) 欄位，以您的叢集名稱取代 *`your_cluster_name`*。

    `EnableTaskIAMRole` 會開啟任務的任務 IAM 角色功能。

    此外，當您使用 `awsvpc` 網路模式時，可使用以下選項。

- EnableTaskENI：此標記會開啟任務聯網，並且當您使用 awsvpc 網路模式時需要它。
- AwsvpcBlockIMDS：此選擇性標記會封鎖在 awsvpc 網路模式中執行的任務容器的 IMDS 存取。
- AwsvpcAdditionalLocalRoutes：此選擇性標記可讓您在任務命名空間中擁有其他路由。

  將 ip-address 替換為其他路由的 IP 地址，例如 172.31.42.23/32。

```
<powershell>
Import-Module ECSTools
Initialize-ECSAgent -Cluster your_cluster_name -EnableTaskIAMRole -EnableTaskENI -
AwsvpcBlockIMDS -AwsvpcAdditionalLocalRoutes
'["ip-address"]'
</powershell>
```

引導 Amazon ECS 視窗容器執行個體以傳遞資料

當您啟動 Amazon EC2 執行個體時，您可以將使用者資料傳遞至 EC2 執行個體。此資料可用來執行常見的自動化組態任務，甚至在執行個體啟動時，執行指令碼。對於 Amazon ECS，使用者資料的最常用案例是將組態資訊傳送到 Docker 常駐程式和 Amazon ECS 容器代理程式。

您可以將多種類型的使用者資料傳遞給 Amazon EC2，包含雲端 boothook、shell 指令碼和 cloud-init 指令。如需這些和其他格式類型的詳細資訊，請參閱 Cloud-Init 文件。

您可以在使用 Amazon EC2 啟動精靈時傳遞此使用者資料。如需詳細資訊，請參閱 啟動 Amazon ECS Linux 容器執行個體。

預設 Windows 使用者資料

此範例使用者資料指令碼顯示您使用主控台時，Windows 容器執行個體所收到的預設使用者資料。下列指令碼會執行下列動作：

- 將叢集名稱設定為您輸入的名稱。
- 設定任務的 IAM 角色。
- 將 json-file 和 awslogs 設定為可用的記錄驅動程式。

此外，當您使用 awsvpc 網路模式時，可使用以下選項。

- EnableTaskENI：此標記會開啟任務聯網，並且當您使用 awsvpc 網路模式時需要它。

- `AwsvpcBlockIMDS`：此選擇性標記會封鎖在 awsvpc 網路模式中執行的任務容器的 IMDS 存取。
- `AwsvpcAdditionalLocalRoutes`：此選擇性標記可讓您擁有其他路由。

  將 `ip-address` 替換為其他路由的 IP 地址，例如 172.31.42.23/32。

您可以將此指令碼用於您自己的容器執行個體 (前提是執行個體是從 Amazon ECS 最佳化 Windows Server AMI 啟動)。

替換 `-Cluster` *`cluster-name`* 行以指定您自己的叢集名稱。

```
<powershell>
Initialize-ECSAgent -Cluster cluster-name -EnableTaskIAMRole -LoggingDrivers '["json-
file","awslogs"]' -EnableTaskENI -AwsvpcBlockIMDS -AwsvpcAdditionalLocalRoutes
'["ip-address"]'
</powershell>
```

對於設定為使用 awslogs 日誌記錄驅動程式的 Windows 任務，您也必須在容器執行個體上設定 ECS_ENABLE_AWSLOGS_EXECUTIONROLE_OVERRIDE 環境變數。請使用下列語法。

替換 `-Cluster` *`cluster-name`* 行以指定您自己的叢集名稱。

```
<powershell>
[Environment]::SetEnvironmentVariable("ECS_ENABLE_AWSLOGS_EXECUTIONROLE_OVERRIDE",
 $TRUE, "Machine")
Initialize-ECSAgent -Cluster cluster-name -EnableTaskIAMRole -LoggingDrivers '["json-
file","awslogs"]'
</powershell>
```

Windows 代理程式安裝使用者資料

此範例使用者資料指令碼會將 Amazon ECS 容器代理程式安裝在以 Windows_Server-2016-English-Full-Containers AMI 啟動的執行個體上。已根據 Amazon ECS 容器代理程式 GitHub儲存庫讀我頁面上的代理程式安裝說明進行改編。

> **Note**
>
> 此指令碼會基於舉例用途而共享。使用 Amazon ECS 最佳化 Windows Server AMI，更容易開始使用 Windows 容器。如需詳細資訊，請參閱 為 Fargate 啟動類型建立 Amazon ECS 叢集。

您可以對您自己的容器執行個體使用此指令碼 (假設是使用 Windows_Server-2016-English-Full-Containers AMI 啟動它們)。務必取代 *windows* 一行，以指定您自己的叢集名稱 (如果您使用的叢集名稱不是 windows)。

```
<powershell>
# Set up directories the agent uses
New-Item -Type directory -Path ${env:ProgramFiles}\Amazon\ECS -Force
New-Item -Type directory -Path ${env:ProgramData}\Amazon\ECS -Force
New-Item -Type directory -Path ${env:ProgramData}\Amazon\ECS\data -Force
# Set up configuration
$ecsExeDir = "${env:ProgramFiles}\Amazon\ECS"
[Environment]::SetEnvironmentVariable("ECS_CLUSTER", "windows", "Machine")
[Environment]::SetEnvironmentVariable("ECS_LOGFILE", "${env:ProgramData}\Amazon\ECS\log
\ecs-agent.log", "Machine")
[Environment]::SetEnvironmentVariable("ECS_DATADIR", "${env:ProgramData}\Amazon\ECS
\data", "Machine")
# Download the agent
$agentVersion = "latest"
$agentZipUri = "https://s3.amazonaws.com/amazon-ecs-agent/ecs-agent-windows-
$agentVersion.zip"
$zipFile = "${env:TEMP}\ecs-agent.zip"
Invoke-RestMethod -OutFile $zipFile -Uri $agentZipUri
# Put the executables in the executable directory.
Expand-Archive -Path $zipFile -DestinationPath $ecsExeDir -Force
Set-Location ${ecsExeDir}
# Set $EnableTaskIAMRoles to $true to enable task IAM roles
# Note that enabling IAM roles will make port 80 unavailable for tasks.
[bool]$EnableTaskIAMRoles = $false
if (${EnableTaskIAMRoles}) {
  $HostSetupScript = Invoke-WebRequest https://raw.githubusercontent.com/aws/amazon-
ecs-agent/master/misc/windows-deploy/hostsetup.ps1
  Invoke-Expression $($HostSetupScript.Content)
}
# Install the agent service
New-Service -Name "AmazonECS" `
        -BinaryPathName "$ecsExeDir\amazon-ecs-agent.exe -windows-service" `
        -DisplayName "Amazon ECS" `
        -Description "Amazon ECS service runs the Amazon ECS agent" `
        -DependsOn Docker `
        -StartupType Manual
sc.exe failure AmazonECS reset=300 actions=restart/5000/restart/30000/restart/60000
sc.exe failureflag AmazonECS 1
Start-Service AmazonECS
```

```
</powershell>
```

針對 Amazon ECS 視窗容器執行個體使用 HTTP 代理伺服器

您可以設定您的 Amazon ECS 容器執行個體，針對 Amazon ECS 容器代理程式和 Docker 常駐程式使用 HTTP 代理。這在您的容器執行個體沒有透過 Amazon VPC 網際網路閘道、NAT 閘道或執行個體進行外部網路存取時會很有用。

若要設定您的 Amazon ECS Windows 容器執行個體以使用 HTTP 代理，請在啟動時 (使用 Amazon EC2 使用者資料) 設定下列變數。

```
[Environment]::SetEnvironmentVariable("HTTP_PROXY",
"http://proxy.mydomain:port", "Machine")
```

將 HTTP_PROXY 設為要讓 Amazon ECS 代理連線到網際網路的 HTTP 代理之主機名稱 (或 IP 地址) 及連接埠號碼。例如，您的容器執行個體可能沒有透過 Amazon VPC 網際網路閘道、NAT 閘道或執行個體的外部網路存取。

```
[Environment]::SetEnvironmentVariable("NO_PROXY",
"169.254.169.254,169.254.170.2,\\.\pipe\docker_engine", "Machine")
```

將 NO_PROXY 設為 169.254.169.254,169.254.170.2,\\.\pipe\docker_engine，篩選 EC2 執行個體中繼資料、任務的 IAM 角色，以及來自代理的 Docker 常駐程式流量。

Example Windows HTTP 代理使用者資料指令碼

下面的使用者資料 PowerShell 指令碼範例會將 Amazon ECS 容器代理程式和 Docker 精靈設定為使用您指定的 HTTP 代理伺服器。您也可以指定容器執行個體自行註冊的叢集。

若要在您啟動容器執行個體時使用此指令碼，請遵循「the section called “啟動容器執行個體”」中的步驟進行。只需將下面的 PowerShell腳本複製並粘貼到用戶數據字段中（請確保用您自己的代理和集群信息替換紅色示例值）。

> **Note**
>
> -EnableTaskIAMRole 選項是啟用任務的 IAM 角色的必要項目。如需詳細資訊，請參閱 Amazon EC2 窗口實例附加配置。

```
<powershell>
```

```
Import-Module ECSTools

$proxy = "http://proxy.mydomain:port"
[Environment]::SetEnvironmentVariable("HTTP_PROXY", $proxy, "Machine")
[Environment]::SetEnvironmentVariable("NO_PROXY", "169.254.169.254,169.254.170.2,\\.
\pipe\docker_engine", "Machine")

Restart-Service Docker
Initialize-ECSAgent -Cluster MyCluster -EnableTaskIAMRole
</powershell>
```

設定 Amazon ECS 視窗容器執行個體以接收競價型執行個體通知

當 Spot 價格超過請求的最高價或容量不再可用時，Amazon EC2 會終止、停止或休眠您的 Spot 執行個體。Amazon EC2 會提供 Spot 執行個體中斷通知，在執行個體中斷前會向執行個體發出兩分鐘的警告。如果在執行個體上啟用 Amazon ECS Spot 執行個體耗盡，則 ECS 會收到 Spot 執行個體中斷通知，並將執行個體置於 DRAINING 狀態。

> **Important**
>
> Amazon ECS 會監控具有 `terminate` 和 `stop` 執行個體動作的 Spot 執行個體中斷通知。如果您在請求 Spot 執行個體或 Spot 機群時指定了 `hibernate` 執行個體中斷行為，則這些執行個體不支援 Amazon ECS Spot 執行個體耗盡。

將容器執行個體設定為 DRAINING 時，Amazon ECS 會避免在容器執行個體中放置新的任務排程。PENDING 狀態下即將耗盡的容器執行個體服務任務會立即停止。如果叢集有可用的容器執行個體，則會在這些容器執行個體上啟動替代服務任務。

您可以在啟動執行個體時開啟 Spot 執行個體排空。您必須先設定 ECS_ENABLE_SPOT_INSTANCE_DRAINING 參數，然後才能啟動容器代理程式。使用您叢集的名稱取代 *my-cluster*。

```
[Environment]::SetEnvironmentVariable("ECS_ENABLE_SPOT_INSTANCE_DRAINING", "true",
 "Machine")

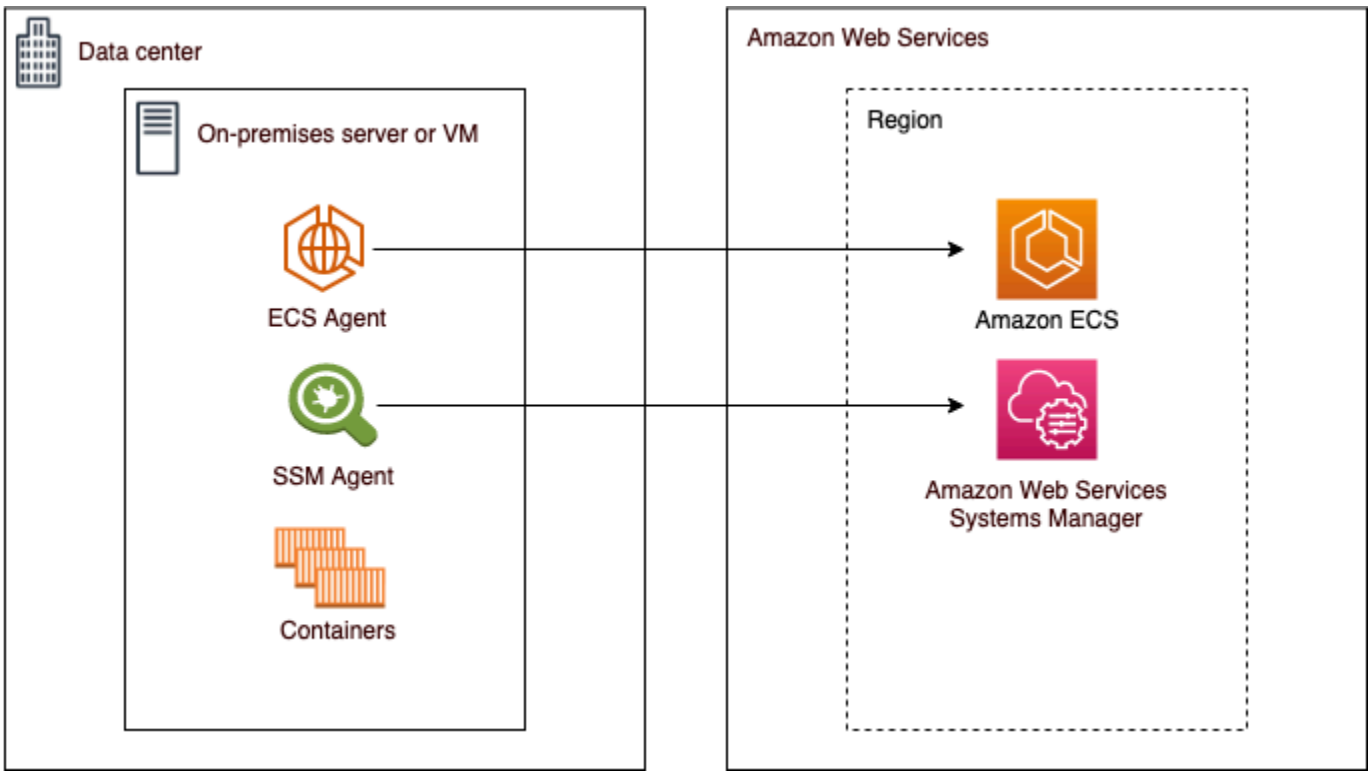
# Initialize the agent
Initialize-ECSAgent -Cluster my-cluster
```

如需詳細資訊，請參閱 the section called “啟動容器執行個體”。

適用於外部啟動類型的 Amazon ECS 叢集

Amazon ECS Anywhere 支援將外部執行個體 (例如內部部署伺服器或虛擬機器 (VM)) 註冊到 Amazon ECS 叢集。外部執行個體已進行優化，可執行產生傳出流量或處理資料的應用程式。如果您的應用程式需要傳入流量，缺乏 Elastic Load Balancing 支援會使這些工作負載的執行效率降低。Amazon ECS 新增了一個新的 EXTERNAL 啟動類型，可用來在外部執行個體上建立服務或執行任務。

以下提供 Amazon ECS Anywhere 的高階系統架構概觀。您的現場部署伺服器同時安裝了 Amazon ECS 代理程式和 SSM 代理程式。



支援的作業系統和系統架構

以下是支援的作業系統和系統架構清單。

- Amazon Linux 2
- CentOS 7
- CentOS Stream 8
- RHEL 7、RHEL 8：Docker 或 RHEL 的開放套件儲存庫都不支援在 RHEL 上原生安裝 Docker。您必須先確定已安裝 Docker，然後再執行本文件中所述的安裝指令碼。
- 軟呢帽 32 軟呢帽，軟呢帽 33

- openSUSE Tumbleweed
- Ubuntu 18, Ubuntu 20, Ubuntu 22
- Debian 10

> **Important**
>
> Debian 9 長期支援 (LTS 支援) 於 2022 年 6 月 30 日結束，並不再受 Amazon ECS Anywhere 支援。

- Debian
- Debian 12 — 目前在 Debian 12 上不支援該NVIDIA容器工具組。您將無法在 Debian 12 個執行個體上執行 GPU。
- SUSE Enterprise Server 15
- 支援 x86_64 和 ARM64 CPU 架構。
- 支援以下 Windows 作業系統版本：
  - Windows Server 2022
  - Windows Server 2019
  - Windows Server 2016
  - Windows Server 20H2

考量事項

在您開始使用外部執行個體之前，請注意下列考量事項。

- 您可以一次向一個叢集註冊外部執行個體。如需有關如何向其他叢集註冊外部執行個體的說明，請參閱 取消註冊 Amazon ECS 外部執行個體。
- 您的外部執行個體需要能夠與 AWS API 通訊的 IAM 角色。如需詳細資訊，請參閱 Amazon ECS Anywhere IAM 角色。
- 您的外部執行個體不應該在本機定義預先設定的執行個體憑證鏈結，因為這會干擾註冊指令碼。
- 若要將容器記錄傳送至 CloudWatch 記錄，請務必在任務定義中建立並指定任務執行 IAM 角色。
- 當外部執行個體註冊到叢集時，`ecs.capability.external` 屬性會與執行個體相關聯。此屬性會將執行個體視為外部執行個體。您可以將自訂屬性新增至外部執行個體，以作為任務置放條件限制。如需詳細資訊，請參閱 自訂屬性。
- 您可以將資源標籤新增至外部執行個體。如需詳細資訊，請參閱 外部容器實例。

- 外部執行個體支援 ECS Exec。如需詳細資訊，請參閱 使用 ECS 執行監控 Amazon ECS 容器。
- 以下是與外部執行個體聯網時特有的其他考量。如需詳細資訊，請參閱 聯網 。
  - 不支援服務負載平衡。
  - 不支援服務探索。
  - 在外部執行個體上執行的任務必須使用 `bridge`、`host` 或none 網路模式。不支援 `awsvpc` 網路模式。
  - 每個 AWS 區域都有 Amazon ECS 服務網域。必須允許這些服務網域將流量傳送到您的外部執行個體。
  - 安裝在外部執行個體上的 SSM Agent 會維護 IAM 憑證，會使用硬體指紋每 30 分鐘輪換這些憑證。如果您的外部執行個體中斷與的連線 AWS，SSM 代理程式會在重新建立連線後自動重新整理認證。如需詳細資訊，請參閱 AWS Systems Manager 使用者指南中的使用硬體指紋驗證內部部署伺服器和虛擬機器。
- 不支援 `UpdateContainerAgent` API。如需有關如何更新外部執行個體上的 SSM Agent 或 Amazon ECS 代理程式的說明，請參閱 更新外部執行個體上的 AWS Systems Manager 代理程式和 Amazon ECS 容器代理程式。
- 不支援 Amazon ECS 容量提供者。若要在外部執行個體上建立服務或執行獨立任務，請使用 `EXTERNAL` 啟動類型。
- 不支援 SELinux。
- 不支援使用 Amazon EFS 磁碟區或指定 `EFSVolumeConfiguration`。
- 不支援與 App Mesh 整合。
- 如果您使用主控台建立外部執行個體任務定義，則必須使用主控台 JSON 編輯器建立任務定義。
- 在 Windows 上執行 ECS Anywhere 時，必須在內部部署基礎設施上使用自己的 Windows 授權。
- 當您使用非 Amazon ECS 最佳化 AMI 時，請在外部容器執行個體上執行下列命令，以設定規則將 IAM 角色用於任務。如需詳細資訊，請參閱 外部執行個體的其他。

```
$ sysctl -w net.ipv4.conf.all.route_localnet=1
$ iptables -t nat -A PREROUTING -p tcp -d 169.254.170.2 --dport 80 -j DNAT --to-
destination 127.0.0.1:51679
$ iptables -t nat -A OUTPUT -d 169.254.170.2 -p tcp -m tcp --dport 80 -j REDIRECT --
to-ports 51679
```

聯網

Amazon ECS 外部執行個體已進行優化，可執行產生傳出流量或處理資料的應用程式。如果您的應用程式需要傳入流量 (例如 Web 服務)，則缺少 Elastic Load Balancing 支援會使執行這些工作負載效率降低，因為不支援將這些工作負載置於負載平衡器之後。

以下是與外部執行個體聯網時特有的其他考量。

- 不支援服務負載平衡。
- 不支援服務探索。
- 在外部執行個體上執行的 Linux 任務必須使用 `bridge`、`host` 或none 網路模式。不支援 `awsvpc` 網路模式。

  如需每個網路模式的詳細資訊，請參閱 Amazon ECS 最佳實務指南中的選擇網路模式。
- 在外部執行個體上執行的 Windows 任務必須使用 `default` 網路模式。
- 每個區域中有 Amazon ECS 服務網域，且必須允許將流量傳送到您的外部執行個體。
- 安裝在外部執行個體上的 SSM Agent 會維護 IAM 憑證，會使用硬體指紋每 30 分鐘輪換這些憑證。如果您的外部執行個體中斷與的連線 AWS，SSM 代理程式會在重新建立連線後自動重新整理認證。如需詳細資訊，請參閱 AWS Systems Manager 使用者指南中的使用硬體指紋驗證內部部署伺服器和虛擬機器。

下列網域用於 Amazon ECS 服務與外部執行個體上安裝的 Amazon ECS 代理程式之間的通訊。請確定允許流量，而且 DNS 解析可以運作。對於每個端點，*##*表示 Amazon ECS 支援的 AWS 區域的區域識別符，例如美國東部 (俄亥俄) 區域的 `us-east-2`。應允許您使用的所有區域的端點。對於 `ecs-a` 和 `ecs-t` 端點，應該包含星號 (例如 `ecs-a-*`)。

- `ecs-a-*.`*`region`*`.amazonaws.com` - 管理任務時會使用此端點。
- `ecs-t-*.`*`region`*`.amazonaws.com` - 此端點可用來管理任務和容器指標。
- `ecs.`*`region`*`.amazonaws.com` - 這是適用於 Amazon ECS 的服務端點。
- `ssm.`*`region`*`.amazonaws.com` — 這是的服務端點 AWS Systems Manager。
- `ec2messages.`*`region`*`.amazonaws.com`— 這是 AWS Systems Manager 用來在系統管理員代理程式與雲端中的 Systems Manager 服務之間進行通訊的服務端點。
- `ssmmessages.`*`region`*`.amazonaws.com` — 必須使用這個服務端點建立和刪除連往雲端工作階段管理員服務的工作階段管道。

- 如果您的工作需要與任何其他 AWS 服務通訊，請確定允許這些服務端點。範例應用程式包括使用 Amazon ECR 提取容器映像或用 CloudWatch 於 CloudWatch 日誌。如需詳細資訊，請參閱 AWS 一般參考中的服務端點。

Amazon FSx for Windows File Server 隨時隨地使用 ECS

若要 Amazon FSx for Windows File Server 搭配 Amazon ECS 外部執行個體使用，您必須在現場部署資料中心和. AWS 雲端如需關於將您的網路連線至 VPC 連線選項的資訊，請參閱 Amazon Virtual Private Cloud 連線能力選項。

gMSA 搭配 ECS Anywhere

支援下列 ECS Anywhere 使用案例。

- Active Directory 位於 AWS 雲端 -對於此組態，您可以在內部部署網路與 AWS 雲端 使用連線之間建立 AWS Direct Connect 連線。如需有關如何建立連線的資訊，請參閱 Amazon Virtual Private Cloud 連線選項。請在 AWS 雲端建立一個 Active Directory。如需如何開始使用的相關資訊 AWS Directory Service，請參閱《AWS Directory Service 管理指南》 AWS Directory Service中的〈設定〉。然後，您可以使用 AWS Direct Connect 連線將外部執行個體加入網域。如需搭配 Amazon ECS 使用 gMSA 的詳細資訊，請參閱the section called “了解如何在 EC2 視窗容器上使用 gMSAs A”。
- Active Directory 位於內部部署資料中心。- 對於此組態，您將外部執行個體加入到內部部署 Active Directory。然後，您可以在執行 Amazon ECS 任務時使用本機可用的憑證。

為外部啟動類型建立 Amazon ECS 叢集

您可以使用 Amazon ECS 主控台建立 Amazon ECS 叢集。開始之前，請務必先完成 設定以使用 Amazon ECS。 中的步驟，並指派適當的 IAM 許可。如需詳細資訊，請參閱 the section called “Amazon ECS 集群示例”。Amazon ECS 主控台提供一種簡單的方法，透過建立堆疊來建立 Amazon ECS 叢集所需的資源。 AWS CloudFormation

為了使叢集建立程序盡可能簡單，主控台提供了許多可供選擇的預設選項，我們將在下方加以說明。主控台的大多數區段還有說明面板，以提供更多上下文。

- 在其中 AWS Cloud Map 建立與叢集相同名稱的預設命名空間。命名空間可讓您在叢集中建立的服務連線到命名空間中的其他服務，而不需要額外的組態。

  如需詳細資訊，請參閱 互連 Amazon ECS 服務。

您可以修改下列選項：

- 變更與叢集相關聯的預設命名空間。

  命名空間可讓您在叢集中建立的服務連線到命名空間中的其他服務，而不需要額外的組態。預設命名空間與叢集名稱相同。如需詳細資訊，請參閱 互連 Amazon ECS 服務。
- 設定外部執行個體的叢集
- 開啟 Container Insights。

  CloudWatch Container Insights 會從您的容器化應用程式和微服務收集、彙總和摘要指標和記錄。Container Insights 還提供診斷資訊，例如容器重新啟動故障，您可以使用這些資訊快速隔離和解決這些問題。如需詳細資訊，請參閱 the section called “使用容器洞察來監控 Amazon ECS 容器”。
- 新增標籤以協助您識別叢集。

建立新叢集 (Amazon ECS 主控台)

1. 開啟主控台，網址為 https://console.aws.amazon.com/ecs/v2。
2. 從導覽列中選取要使用的「區域」。
3. 在導覽窗格中，選擇叢集。
4. 在 Clusters (叢集) 頁面上，選擇 Create cluster (建立叢集)。
5. 在叢集組態下，設定下列項目：

   - 在叢集名稱下輸入唯一的名稱。

     名稱可以包含最多 255 個字母 (大小寫)、數字與連字號。
   - (選用) 若要讓 Service Connect 使用的命名空間與叢集名稱不同，請在命名空間中輸入唯一的名稱。
6. 展開基礎結構，選取 AWS Fargate (無伺服器)。
7. (選用) 若要開啟 Container Insights，請展開 Monitoring (監控)，然後開啟 Use Container Insights (使用 Container Insights)。
8. (選用) 為協助識別您的叢集，請展開 Tags (標籤)，然後設定標籤。

   [新增標籤] 選擇新增標籤，並執行下列動作：

   - 對於 Key (金鑰)，輸入金鑰名稱。

- 在值中，進入索引鍵值。

9. 選擇建立。

後續步驟

您必須在叢集中註冊執行個體。如需詳細資訊，請參閱 將外部執行個體註冊到 Amazon ECS 叢集。

建立叢集之後，您可以為應用程式建立任務定義，然後將它們當做獨立任務或服務的一部分執行。如需詳細資訊，請參閱下列內容：

- Amazon ECS 任務定義
- 以 Amazon ECS 任務的形式執行應用程式
- 使用主控台建立 Amazon ECS 服務

將外部執行個體註冊到 Amazon ECS 叢集

對於您向 Amazon ECS 叢集註冊的每個外部執行個體，它必須安裝 SSM Agent、Amazon ECS 容器代理程式和 Docker。若要將外部執行個體註冊到 Amazon ECS 叢集，必須先將其註冊為 AWS Systems Manager 受管執行個體。在 Amazon ECS 主控台上按幾下，可建立安裝指令碼。安裝指令碼包含 Systems Manager 啟用金鑰，以及安裝每個必要的代理程式和 Docker 的命令。必須在內部部署伺服器或虛擬機器上執行安裝指令碼，才能完成安裝和註冊步驟。

> **Note**
>
> 在向叢集註冊 Linux 外部執行個體之前，請先在外部執行個體上建立 `/etc/ecs/ecs.config` 檔案，並新增您想要的任何容器代理程式組態參數。將外部執行個體註冊到叢集之後，您無法執行此操作。如需詳細資訊，請參閱 Amazon ECS 容器代理程式組態。

AWS Management Console

1. 開啟主控台，網址為 https://console.aws.amazon.com/ecs/v2。
2. 從導覽列中選取要使用的「區域」。
3. 在導覽窗格中，選擇叢集。
4. 在 Clusters (叢集) 頁面上，選擇要向其註冊外部執行個體的叢集。
5. 在 Cluster : ***name*** (叢集：名稱) 頁面上，選擇 Infrastructure (基礎基礎設施) 索引標籤。

6. 在 Register external instances (註冊外部執行個體) 頁面上，完成下列步驟。

   a. 對於 Activation key duration (in days) (啟用金鑰持續時間 (以天為單位))，輸入啟用金鑰保持作用中的天數。在您輸入的天數到期之後，當註冊外部執行個體時，金鑰就不再有效。

   b. 對於 Number of instances (執行個體數目)，輸入要使用啟用金鑰向叢集註冊的外部執行個體數。

   c. 對於 Instance role (執行個體角色)，選擇要與外部執行個體建立關聯的 IAM 角色。如果尚未建立角色，請選擇 Create new role (建立新角色)，讓 Amazon ECS 代表您建立角色。如需外部執行個體需要哪些 IAM 許可的詳細資訊，請參閱 Amazon ECS Anywhere IAM 角色。

   d. 複製註冊命令。應該在您要將其註冊到叢集的每個外部執行個體上執行此命令。

      > **Important**
      >
      > 指令碼的 bash 部分必須以 root 身分執行。如果命令不是以 root 身分執行，則會傳回錯誤。

   e. 選擇關閉。

AWS CLI for Linux operating systems

1. 建立 Systems Manager 啟用對。這是用於 Systems Manager 受管執行個體啟用。輸出包含 `ActivationId` 和 `ActivationCode`。在後續步驟中會使用它們。請確定指定您已建立的 ECS Anywhere IAM 角色。如需詳細資訊，請參閱 Amazon ECS Anywhere IAM 角色。

   ```
   aws ssm create-activation --iam-role ecsAnywhereRole | tee ssm-activation.json
   ```

2. 在您的內部部署伺服器或虛擬機器 (VM) 上，下載安裝指令碼。

   ```
   curl --proto "https" -o "/tmp/ecs-anywhere-install.sh" "https://amazon-ecs-agent.s3.amazonaws.com/ecs-anywhere-install-latest.sh"
   ```

3. (選用) 在您的內部部署伺服器或虛擬機器 (VM) 上，使用指令碼簽章檔案透過以下步驟來驗證安裝指令碼。

   a. 下載並安裝 GnuPG。如需 GNUpg 的詳細資訊，請參閱 GnuPG 網站。若您的系統為 Linux，請使用套件軟體管理工具在 Linux 上安裝 gpg。

   b. 擷取Amazon ECS PGP 公有金鑰。

```
gpg --keyserver hkp://keys.gnupg.net:80 --recv BCE9D9A42D51784F
```

c. 下載安裝指令碼簽章。簽章是 ASCII 分離的 PGP 簽章，存放於副檔名為 .asc 的檔案中。

```
curl --proto "https" -o "/tmp/ecs-anywhere-install.sh.asc" "https://amazon-ecs-agent.s3.amazonaws.com/ecs-anywhere-install-latest.sh.asc"
```

d. 使用金鑰驗證安裝指令碼檔案。

```
gpg --verify /tmp/ecs-anywhere-install.sh.asc /tmp/ecs-anywhere-install.sh
```

預期的輸出如下：

```
gpg: Signature made Tue 25 May 2021 07:16:29 PM UTC
gpg:                using RSA key 50DECCC4710E61AF
gpg: Good signature from "Amazon ECS <ecs-security@amazon.com>" [unknown]
gpg: WARNING: This key is not certified with a trusted signature!
gpg:          There is no indication that the signature belongs to the
 owner.
Primary key fingerprint: F34C 3DDA E729 26B0 79BE  AEC6 BCE9 D9A4 2D51 784F
     Subkey fingerprint: D64B B6F9 0CF3 77E9 B5FB  346F 50DE CCC4 710E 61AF
```

4. 在您的內部部署伺服器或虛擬機器 (VM) 上，執行安裝指令碼。在第一步中指定叢集名稱、區域和 Systems Manager 啟用 ID 和啟用代碼。

```
sudo bash /tmp/ecs-anywhere-install.sh \
    --region $REGION \
    --cluster $CLUSTER_NAME \
    --activation-id $ACTIVATION_ID \
    --activation-code $ACTIVATION_CODE
```

對於內部部署伺服器，或已為 GPU 工作負載安裝 NVIDIA 驅動程式的虛擬機器 (VM)，您必須將 --enable-gpu 旗標新增至安裝指令碼。指定此旗標時，安裝指令碼會驗證 NVIDIA 驅動程式是否正在執行，然後新增所需的組態變數來執行您的 Amazon ECS 任務。如需關於執行 GPU 工作負載和在任務定義中指定 GPU 要求的詳細資訊，請參閱 在 Amazon ECS 任務定義中指定 GPU。

```
sudo bash /tmp/ecs-anywhere-install.sh \
```

```
    --region $REGION \
    --cluster $CLUSTER_NAME \
    --activation-id $ACTIVATION_ID \
    --activation-code $ACTIVATION_CODE \
    --enable-gpu
```

進行以下步驟向其他叢集註冊現有的外部執行個體

若要向其他叢集註冊現有的外部執行個體

1. 停用 Amazon ECS 容器代理程式。

```
sudo systemctl stop ecs.service
```

2. 編輯 `/etc/ecs/ecs.config` 檔案，在 `ECS_CLUSTER` 行中，確保叢集名稱與要註冊外部執行個體的叢集名稱相符。
3. 移除現有的 Amazon ECS 代理程式資料。

```
sudo rm /var/lib/ecs/data/agent.db
```

4. 啟動 Amazon ECS 容器代理程式。

```
sudo systemctl start ecs.service
```

AWS CLI for Windows operating systems

1. 建立 Systems Manager 啟用對。這是用於 Systems Manager 受管執行個體啟用。輸出包含 `ActivationId` 和 `ActivationCode`。在後續步驟中會使用它們。請確定指定您已建立的 ECS Anywhere IAM 角色。如需詳細資訊，請參閱 Amazon ECS Anywhere IAM 角色。

```
aws ssm create-activation --iam-role ecsAnywhereRole | tee ssm-activation.json
```

2. 在您的內部部署伺服器或虛擬機器 (VM) 上，下載安裝指令碼。

```
Invoke-RestMethod -URI "https://amazon-ecs-agent.s3.amazonaws.com/ecs-anywhere-
install.ps1" -OutFile “ecs-anywhere-install.ps1”
```

3. (選用) Powershell 指令碼由 Amazon 簽署，因此 Windows 會自動對該指令碼執行相同的憑證驗證。您不需要執行任何手動驗證。

若要手動驗證憑證，請用右鍵按一下該檔案，導覽到屬性，然後使用 Digital Signatures (數位簽署) 索引標籤取得更多詳細資料。

僅在主機在憑證存放區中具有憑證時，此選項才可用。

驗證應會傳回類似以下內容的資訊：

```
# Verification (PowerShell)
Get-AuthenticodeSignature -FilePath .\ecs-anywhere-install.ps1

SignerCertificate                         Status      Path
-----------------                         ------      ----
EXAMPLECERTIFICATE                        Valid       ecs-anywhere-install.ps1

...

Subject              : CN="Amazon Web Services, Inc.",...

----
```

4. 在您的內部部署伺服器或虛擬機器 (VM) 上，執行安裝指令碼。在第一步中指定叢集名稱、區域和 Systems Manager 啟用 ID 和啟用代碼。

```
.\ecs-anywhere-install.ps1 -Region $Region -Cluster $Cluster -
ActivationID $ActivationID -ActivationCode $ActivationCode
```

5. 驗證 Amazon ECS 容器代理程式是否正在執行。

```
Get-Service AmazonECS

Status   Name               DisplayName
------   ----               -----------
Running  AmazonECS          Amazon ECS
```

進行以下步驟向其他叢集註冊現有的外部執行個體

若要向其他叢集註冊現有的外部執行個體

1. 停用 Amazon ECS 容器代理程式。

```
Stop-Service AmazonECS
```

2. 修改 ECS_CLUSTER 參數，以便叢集名稱與要註冊外部執行個體的叢集名稱相符。

```
[Environment]::SetEnvironmentVariable("ECS_CLUSTER", $ECSCluster,
 [System.EnvironmentVariableTarget]::Machine)
```

3. 移除現有的 Amazon ECS 代理程式資料。

```
Remove-Item -Recurse -Force $env:ProgramData\Amazon\ECS\data\*
```

4. 啟動 Amazon ECS 容器代理程式。

```
Start-Service AmazonECS
```

在執行安裝指令碼 AWS CLI 以完成外部執行個體註冊程序之前，可用來建立 Systems Manager 啟動。

取消註冊 Amazon ECS 外部執行個體

我們建議您從 Amazon ECS 和完成執行個體 AWS Systems Manager 之後取消註冊執行個體。取消註冊後，外部執行個體即不再接受新的任務。

如果在您取消註冊容器執行個體時，還有執行中的任務，則這些任務會繼續執行，直到透過一些其他方式停用為止。不過，Amazon ECS 不再監控或負責這些任務。如果外部執行個體中的這些任務是 Amazon ECS 服務的一部分，則服務排程器可能會在不同的執行個體上啟動該任務的另一個副本。

取消註冊執行個體後，請清除執行個體上的剩餘 AWS 資源。然後，您可以將其註冊到新的叢集。

程序

AWS Management Console

1. 開啟主控台，網址為 https://console.aws.amazon.com/ecs/v2。
2. 從導覽列中選擇註冊外部執行個體所在的區域。
3. 在導覽窗格中選擇 Clusters (叢集)，並選取託管外部執行個體的叢集。
4. 在 Cluster : ***name*** (叢集：名稱) 頁面上，選擇 Infrastructure (基礎基礎設施) 索引標籤。

5. 在 Container instances (容器執行個體) 中，選擇要取消註冊的外部執行個體 ID。系統會將您重新導向至容器執行個體詳細資訊頁面。
6. 在 Container Instance : ***id*** (容器執行個體：ID) 頁面中選擇 Deregister (取消註冊)。
7. 檢閱取消註冊訊息。選取 Deregister from AWS Systems Manager (從 AWS Systems Manager中取消註冊)，也將作為 Systems Manager 受管執行個體來取消註冊外部執行個體。選擇 Deregister (取消註冊)。

> **Note**
>
> 在 Systems Manager 主控台中，可作為 Systems Manager 受管執行個體來取消註冊外部執行個體。如需說明，請參閱 AWS Systems Manager 使用者指南中的取消註冊受管執行個體。

8. 取消註冊執行個體後，請清除內部部署伺服器或 VM 上的 AWS 資源。

作業系統	步驟	

Linux	a. 停用執行個體上的 Amazon ECS 容器代理程式和 SSM Agent 服務。
	`sudo systemctl stop ecs amazon-ssm-agent`
	b. 移除 Amazon ECS 和 Systems Manager 套件。
	對於 CentOS 7、CentOS 8 和 RHEL 7
	`sudo yum remove -y amazon-ecs-init amazon-ssm-agent`
	對於 SUSE Enterprise Server 15	

作業系統	步驟	

	`sudo zypper remove -y amazon-ecs-init amazon-ssm-agent`
	
	對於 Debian 和 Ubuntu
	
	`sudo apt remove -y amazon-ecs-init amazon-ssm-agent`
	
	c. 移除剩餘的目錄。
	
	`sudo rm -rf /var/lib/ecs /etc/ecs / var/lib/amazon/ssm /var/log/ecs / var/log/amazon/ssm`	

Windows	a. 停用執行個體上的 Amazon ECS 容器代理程式和 SSM Agent 服務。
	
	`Stop-Service AmazonECS`
	
	`Stop-Service AmazonSSMAgent`
	
	b. 移除 Amazon ECS 套件。
	
	`.\ecs-anywhere-install.ps1 -Uninstall`	

AWS CLI

1. 您需要執行個體 ID 和容器執行個體 ARN，才能取消註冊容器執行個體。如果您沒有這些值，請執行下列命令

   執行下列命令以取得執行個體 ID。

   您可以使用執行個體 ID (`instanceID`) 取得容器執行個體 ARN (`containerInstanceARN`)。

   ```
   instanceId=$(aws ssm describe-instance-information --region "{{ region }}" |
    jq ".InstanceInformationList[] |select(.IPAddress==\"{{ IPv4 Address }}\")
    | .InstanceId" | tr -d'"'
   ```

   執行下列命令。

   您可以在命令中使用 `containerInstanceArn` 做為參數來取消註冊執行個體 (`deregister-container-instance`)。

   ```
   instances=$(aws ecs list-container-instances --cluster "{{ cluster }}" --region
    "{{ region }}" | jq -c '.containerInstanceArns')
   containerInstanceArn=$(aws ecs describe-container-instances --cluster
    "{{ cluster }}" --region "{{ region }}" --container-instances $instances
    | jq ".containerInstances[] | select(.ec2InstanceId==\"{{ instanceId }}\")
    | .containerInstanceArn" | tr -d '"')
   ```

2. 執行下列命令以消耗執行個體。

   ```
   aws ecs update-container-instances-state --cluster "{{ cluster }}" --region
    "{{ region }}" --container-instances "{{ containerInstanceArn }}" --status
    DRAINING
   ```

3. 容器執行個體完成消耗後，執行下列命令以取消註冊執行個體。

   ```
   aws ecs deregister-container-instance --cluster "{{ cluster }}" --region
    "{{ region }}" --container-instance "{{ containerInstanceArn }}"
   ```

4. 使用下列命令從 SSM 移除容器執行個體。

   ```
   aws ssm deregister-managed-instance --region "{{ region }}" --instance-id
    "{{ instanceId }}"
   ```

5. 取消註冊執行個體後，請清除內部部署伺服器或 VM 上的 AWS 資源。

作業系統	步驟	

Linux	a. 停用執行個體上的 Amazon ECS 容器代理程式和 SSM Agent 服務。
	`sudo systemctl stop ecs amazon-ssm-agent`
	b. 移除 Amazon ECS 和 Systems Manager 套件。
	`sudo (yum/apt/zypper) remove amazon-ecs-init amazon-ssm-agent`
	c. 移除剩餘的目錄。
	`sudo rm -rf /var/lib/ecs /etc/ecs /var/lib/amazon/ssm /var/log/ecs /var/log/amazon/ssm`	

作業系統	步驟	

Windows	a. 停用執行個體上的 Amazon ECS 容器代理程式和 SSM Agent 服務。
	`Stop-Service AmazonECS`
	`Stop-Service AmazonSSMAgent`
	b. 移除 Amazon ECS 套件。
	`.\ecs-anywhere-install.ps1 -Uninstall`	

更新外部執行個體上的 AWS Systems Manager 代理程式和 Amazon ECS 容器代理程式

執行 Amazon ECS 工作負載時，您的現場部署伺服器或虛擬機器必須同時執行代理程式 (SSM 代理程式) 和 Amazon ECS 容器代理程式。 AWS Systems Manager AWS 新增或更新任何功能時，會發行這些代理程式的新版本。如果您的外部執行個體使用舊版代理程式，您可以使用下列程序來更新它們。

在外部執行個體上更新 SSM Agent

AWS Systems Manager 建議您將執行個體上的 SSM 代理程式更新程序自動化。它們提供多種方法來自動化更新。如需詳細資訊，請參閱 AWS Systems Manager 使用者指南中的自動化 SSM Agent 更新。

在外部執行個體上更新 Amazon ECS 代理程式

在外部執行個體上，Amazon ECS 容器代理程式會透過升級 `ecs-init` 套件進行更新。更新 Amazon ECS 代理程式不會中斷執行中的任務或服務。Amazon ECS 在每個區域的 Amazon S3 儲存貯體中提供 `ecs-init` 套件和簽章檔案。從 `ecs-init` 版本 `1.52.1-1` 開始，Amazon ECS 提供單獨的 `ecs-init` 套件以供使用，這取決於外部執行個體所使用的作業系統和系統架構。

依據外部執行個體使用的作業系統和系統架構，使用下表來確定您應下載的 ecs-init 套件。

> **Note**
>
> 透過使用下列命令來確定外部執行個體所使用的作業系統和系統架構。
>
> ```
> cat /etc/os-release
> uname -m
> ```

作業系統 (架構)	ecs-init 套件
CentOS 7 (x86_64) CentOS 8 (x86_64) SUSE Enterprise Server 15 (x86_64) RHEL 7 (x86_64) RHEL 8 (x86_64)	amazon-ecs-init-latest.x86_64.rpm
CentOS 7 (aarch64) CentOS 8 (aarch64) RHEL 7 (aarch64)	amazon-ecs-init-latest.aarch64.rpm
Debian 9 (x86_64) Debian 10 (x86_64) Debian Debian Ubuntu 18 (x86_64) Ubuntu 20 (x86_64)	amazon-ecs-init-latest.amd64.deb
Debian 9 (aarch64)	amazon-ecs-init-latest.arm64.deb

作業系統 (架構)	ecs-init 套件
Debian 10 (aarch64) Debian 11 (阿爾查 64) Debian 12 (阿爾查 64) Ubuntu 18 (aarch64) Ubuntu 20 (aarch64)	

請依照以下步驟更新 Amazon ECS 代理程式。

若要更新 Amazon ECS 代理程式

1. 確認您正在執行的 Amazon ECS 代理程式版本。

```
curl -s 127.0.0.1:51678/v1/metadata | python3 -mjson.tool
```

2. 下載適用於您的作業系統和系統架構的 `ecs-init` 套件。Amazon ECS 在每個區域的 Amazon S3 儲存貯體中提供 `ecs-init` 套件檔案。請務必將命令中的 *<region>* 識別符替換為與您的地理位置最接近的區域名稱 (例如，`us-west-2`)。

   amazon-ecs-init-latest.x86_64.rpm

```
curl -o amazon-ecs-init.rpm https://s3.<region>.amazonaws.com/amazon-ecs-
agent-<region>/amazon-ecs-init-latest.x86_64.rpm
```

   amazon-ecs-init-latest.aarch64.rpm

```
curl -o amazon-ecs-init.rpm https://s3.<region>.amazonaws.com/amazon-ecs-
agent-<region>/amazon-ecs-init-latest.aarch64.rpm
```

   amazon-ecs-init-latest.amd64.deb

```
curl -o amazon-ecs-init.deb https://s3.<region>.amazonaws.com/amazon-ecs-
agent-<region>/amazon-ecs-init-latest.amd64.deb
```

amazon-ecs-init-latest.arm64.deb

```
curl -o amazon-ecs-init.deb https://s3.<region>.amazonaws.com/amazon-ecs-agent-<region>/amazon-ecs-init-latest.arm64.deb
```

3. (選用) 使用 PGP 簽章確認 ecs-init 套件檔案的有效性。

   a. 下載並安裝 GnuPG。如需 GNUpg 的詳細資訊，請參閱 GnuPG 網站。若您的系統為 Linux，請使用套件軟體管理工具在 Linux 上安裝 gpg。

   b. 擷取Amazon ECS PGP 公有金鑰。

   ```
   gpg --keyserver hkp://keys.gnupg.net:80 --recv BCE9D9A42D51784F
   ```

   c. 下載 ecs-init 套件簽章。簽章是 ASCII 分離的 PGP 簽章，存放於副檔名為 .asc 的檔案中。Amazon ECS 在每個區域的 Amazon S3 儲存貯體中提供簽章檔案。請務必將命令中的 *<region>* 識別符替換為與您的地理位置最接近的區域名稱 (例如，us-west-2)。

   amazon-ecs-init-latest.x86_64.rpm

   ```
   curl -o amazon-ecs-init.rpm.asc https://s3.<region>.amazonaws.com/amazon-ecs-agent-<region>/amazon-ecs-init-latest.x86_64.rpm.asc
   ```

   amazon-ecs-init-latest.aarch64.rpm

   ```
   curl -o amazon-ecs-init.rpm.asc https://s3.<region>.amazonaws.com/amazon-ecs-agent-<region>/amazon-ecs-init-latest.aarch64.rpm.asc
   ```

   amazon-ecs-init-latest.amd64.deb

   ```
   curl -o amazon-ecs-init.deb.asc https://s3.<region>.amazonaws.com/amazon-ecs-agent-<region>/amazon-ecs-init-latest.amd64.deb.asc
   ```

   amazon-ecs-init-latest.arm64.deb

   ```
   curl -o amazon-ecs-init.deb.asc https://s3.<region>.amazonaws.com/amazon-ecs-agent-<region>/amazon-ecs-init-latest.arm64.deb.asc
   ```

   d. 使用金鑰驗證 ecs-init 套件檔案。

對於 **rpm** 套件

```
gpg --verify amazon-ecs-init.rpm.asc ./amazon-ecs-init.rpm
```

對於 **deb** 套件

```
gpg --verify amazon-ecs-init.deb.asc ./amazon-ecs-init.deb
```

預期的輸出如下：

```
gpg: Signature made Fri 14 May 2021 09:31:36 PM UTC
gpg:                using RSA key 50DECCC4710E61AF
gpg: Good signature from "Amazon ECS <ecs-security@amazon.com>" [unknown]
gpg: WARNING: This key is not certified with a trusted signature!
gpg:          There is no indication that the signature belongs to the owner.
Primary key fingerprint: F34C 3DDA E729 26B0 79BE  AEC6 BCE9 D9A4 2D51 784F
     Subkey fingerprint: D64B B6F9 0CF3 77E9 B5FB  346F 50DE CCC4 710E 61AF
```

4. 安裝 `ecs-init` 套裝服務。

   對於 CentOS 7、CentOS 8 以及 RHEL 7 中的 **rpm** 套件

   ```
   sudo yum install -y ./amazon-ecs-init.rpm
   ```

   對於 SUSE Enterprise Server 15 中的 **rpm** 套件

   ```
   sudo zypper install -y --allow-unsigned-rpm ./amazon-ecs-init.rpm
   ```

   對於 **deb** 套件

   ```
   sudo dpkg -i ./amazon-ecs-init.deb
   ```

5. 重新啟動 `ecs` 服務。

   ```
   sudo systemctl restart ecs
   ```

6. 確認 Amazon ECS 代理程式版本已更新。

```
curl -s 127.0.0.1:51678/v1/metadata | python3 -mjson.tool
```

更新 Amazon ECS 叢集

您可以修改下列叢集屬性：

- 設定預設容量提供者

  每個叢集可以有一或多個容量提供者，以及選用的容量提供者策略。容量提供者策略會決定任務在叢集的容量提供者之間分散的方式。當您執行獨立任務或建立服務時，可以使用叢集的預設容量提供者策略，也可以使用能覆寫預設值的容量提供者策略。
- 開啟 Container Insights。

  CloudWatch Container Insights 會從您的容器化應用程式和微服務收集、彙總和摘要指標和記錄。Container Insights 還提供診斷資訊，例如容器重新啟動故障，您可以使用這些資訊快速隔離和解決這些問題。如需詳細資訊，請參閱 the section called “使用容器洞察來監控 Amazon ECS 容器”。
- 新增標籤以協助您識別叢集。

程序

1. 開啟主控台，網址為 https://console.aws.amazon.com/ecs/v2。
2. 在導覽窗格中，選擇叢集。
3. 在叢集頁面上，選擇叢集。
4. 在叢集：**##**頁面上，選擇更新叢集。
5. 若要設定預設容量提供者，請在預設容量提供者策略下選擇新增更多。
   a. 對於容量提供者，選擇容量提供者。
   b. (選用) 在基礎中，輸入在容量提供者上執行的任務的最小數目。

      您只能為一個容量提供者設定基礎值。
   c. (選用) 在加權中，輸入使用指定容量提供者的任務在已啟動任務的總數中所佔的相對百分比。
   d. (選用) 針對任何其他容量提供者重複上述步驟。
6. 若要開啟或關閉 Container Insights，請展開監控，然後開啟使用 Container Insights。

7. 為協助識別您的叢集，請展開標籤，然後設定標籤。

   [新增標籤] 選擇新增標籤，並執行下列動作：

   - 對於 Key (金鑰)，輸入金鑰名稱。
   - 對於 Value (值)，進入金鑰值。

   [移除標籤] 選擇標籤「金鑰」和「值」右側的移除。

8. 選擇更新。

刪除 Amazon ECS 叢集

如果叢集已使用完畢，即可將其刪除。刪除叢集後，其會轉換到 INACTIVE 狀態。具有 INACTIVE 狀態的叢集可能會在您的帳戶中保持可探索一段時間。不過，此行為未來可能變動，因此不建議依賴 INACTIVE 叢集得以保存。

刪除叢集之前，您必須執行下列操作：

- 刪除叢集中的所有服務。如需詳細資訊，請參閱 the section called “刪除服務”。
- 停止所有當前正在執行的任務。如需詳細資訊，請參閱 the section called “停止工作”。
- 取消註冊叢集中的所有已註冊的容器執行個體。如需詳細資訊，請參閱 the section called “取消註冊容器執行個體”。
- 刪除 命名空間。如需詳細資訊，請參閱 AWS Cloud Map 開發人員指南中的刪除命名空間。

程序

1. 開啟主控台，網址為 https://console.aws.amazon.com/ecs/v2。
2. 從導覽列中選取要使用的「區域」。
3. 在導覽窗格中，選擇叢集。
4. 在 Clusters (叢集) 頁面上，選取要刪除的叢集。
5. 在頁面右上角，請選擇 Delete Cluster (刪除叢集)。

   當您沒有刪除與叢集關聯的所有資源時，將顯示一則訊息。

6. 在確認方塊中，輸入 delete ***cluster name***。

為 Amazon ECS 建立容量提供者

叢集建立完成後，即可為 EC2 啟動類型建立新的容量提供者 (Auto Scaling 群組)。

建立容量提供者前，您需要建立 Auto Scaling 群組。如需詳細資訊，請參閱《Amazon EC2 Auto Scaling 使用者指南》中的 Auto Scaling 群組。

建立叢集的容量提供者 (Amazon ECS 主控台)

1. 開啟主控台，網址為 https://console.aws.amazon.com/ecs/v2。
2. 在導覽窗格中，選擇叢集。
3. 在叢集頁面上，選擇叢集。
4. 在 Cluster:***name*** (叢集：名稱) 頁面中，選擇 Infrastructure (基礎設施)，然後選擇 Create (建立)。
5. 在 Create capacity providers (建立容量提供者) 頁面中，設定下列選項。
   a. 在 Basic details (基本詳細資訊) 下的 Capacity provider name (容量提供者名稱) 中，輸入容量提供者名稱。
   b. 在 Auto Scaling group (Auto Scaling 群組) 下的 Use an existing Auto Scaling group (使用現有 Auto Scaling 群組) 欄位中，選擇 Auto Scaling 群組。
   c. (選用) 若要設定擴展政策，請在 Scaling policies (擴展政策) 下設定下列選項。
      - 若要讓 Amazon ECS 管理縮減和橫向擴展動作，請選取 Turn on managed scaling (開啟受管擴展)。
      - 若要防止具有執行中 Amazon ECS 任務的 EC2 執行個體遭到終止，請選取 Turn on scaling protection (開啟擴展保護)。
      - 在設定目標容量中，輸入 Amazon ECS 管理的目標追蹤擴展政策中使用的 CloudWatch 指標的目標值。
6. 選擇建立。

更新 Amazon ECS 容量供應商

使用 Auto Scaling 群組作為容量提供者時，您可以修改群組的擴展政策。

更新叢集的容量提供者 (Amazon ECS 主控台)

1. 開啟主控台，網址為 https://console.aws.amazon.com/ecs/v2。

2. 在導覽窗格中，選擇叢集。
3. 在叢集頁面上，選擇叢集。
4. 在 Cluster : ***name*** (叢集：名稱) 頁面中，選擇 Infrastructure (基礎設施)，然後選擇 Update (更新)。
5. 在 Create capacity providers (建立容量提供者) 頁面中，設定下列選項。
   - 在 Auto Scaling 群組下的擴展政策下，設定下列選項。
     - 若要讓 Amazon ECS 管理縮減和橫向擴展動作，請選取 Turn on managed scaling (開啟受管擴展)。
     - 若要防止具有執行中 Amazon ECS 任務的 EC2 執行個體遭到終止，請選取開啟擴展保護。
     - 在設定目標容量中，輸入 Amazon ECS 管理的目標追蹤擴展政策中使用的 CloudWatch 指標的目標值。
6. 選擇更新。

刪除 Amazon ECS 容量提供者

如果 Auto Scaling 群組容量提供者已使用完畢，則可將其刪除。刪除群組後，Auto Scaling 群組容量提供者會轉換為`INACTIVE`狀態。具有 `INACTIVE` 狀態的容量提供者可能會在您的帳戶中保持可探索一段時間。不過，此行為未來可能變動，因此不建議依賴 `INACTIVE` 容量提供者得以保存。刪除 Auto Scaling 群組容量提供者之前，必須從所有服務的容量提供者策略中移除容量提供者。您可以使用 `UpdateService` API 或 Amazon ECS 主控台中的更新服務工作流程，從服務的容量提供者策略中移除容量提供者。使用強制新部署選項可確保使用容量提供者提供之 Amazon EC2 執行個體容量的任務都會轉換為使用剩餘容量供應商的容量。

刪除叢集的容量提供者 (Amazon ECS 主控台)

1. 開啟主控台，網址為 https://console.aws.amazon.com/ecs/v2。
2. 在導覽窗格中，選擇叢集。
3. 在叢集頁面上，選擇叢集。
4. 在 Cluster : ***name*** (叢集：名稱) 頁面中，選擇 Infrastructure (基礎設施)、Auto Scaling 群組，然後選擇 Delete (刪除)。
5. 在確認方塊中，輸入 delete ***Auto Scaling ####***
6. 選擇刪除。

取消註冊 Amazon ECS 容器執行個體

> **Important**
>
> 本主題僅適用於在 Amazon EC2 中建立的容器執行個體。如需取消註冊外部執行個體的詳細資訊，請參閱 取消註冊 Amazon ECS 外部執行個體。

完成 Amazon EC2 支援的容器執行個體後，應該可以從您的叢集中取消註冊它。取消註冊後，容器執行個體即不再接受新的任務。

如果在您取消註冊容器執行個體時，還有執行中的任務，這些任務會繼續執行，直到您終止執行個體或任務因某些緣由而停止為止。不過，這些任務是孤立的，這意味著 Amazon ECS 不再監控或負責它們。如果容器執行個體中的孤立任務是 Amazon ECS 服務的一部分，該服務排程器有可能會在不同的容器執行個體上啟動該任務的另一個副本。會取消註冊已註冊有 Application Load Balancer 目標群組的孤立服務任務中的所有容器。他們會根據負載平衡器或目標群組的設定，開始進行連接耗盡。如果孤立任務正在使用 `awsvpc` 網路模式，則會刪除其彈性網路介面。

如果在取消註冊後，您打算將容器執行個體另作他用，您應該先停止該容器執行個體上執行的所有任務，再取消註冊。這可防止任何遺棄的任務耗費資源。

取消註冊容器執行個體時，請注意下列考量事項。

- 因為每個容器執行個體都有唯一的狀態資訊，所以不應從一個叢集取消註冊，又在另一個叢集中重新註冊。若要重新定位容器執行個體資源，我們建議您從一個叢集終止容器執行個體，再於新叢集中啟動新的容器執行個體。如需詳細資訊，請參閱 Amazon EC2 使用者指南中的終止執行個體和啟動 Amazon ECS Linux 容器執行個體。
- 如果容器執行個體是由 Auto Scaling 群組或 AWS CloudFormation 堆疊所管理，請透過更新 Auto Scaling 群組或 AWS CloudFormation 堆疊來終止執行個體。否則，Auto Scaling 群組或 AWS CloudFormation 會在您終止執行個體後建立新執行個體。
- 如果您終止的是具有連線的 Amazon ECS 容器代理的執行中容器執行個體，代理程式會自動從您的叢集取消註冊該執行個體。如果是已停止的容器執行個體或具有中斷連線代理的執行個體，則不會在終止時自動取消註冊。
- 取消註冊容器執行個體會從叢集中移除執行個體，但不會終止 Amazon EC2 執行個體。如果您已完成使用執行個體，請務必終止它以停止計費。如需詳細資訊，請參閱《Amazon EC2 使用者指南》中的終止您的執行個體。

程序

1. 開啟主控台，網址為 https://console.aws.amazon.com/ecs/v2。
2. 從導覽列中選擇註冊外部執行個體所在的區域。
3. 在導覽窗格中選擇 Clusters (叢集)，並選取託管執行個體的叢集。
4. 在 Cluster : ***name*** (叢集：名稱) 頁面上，選擇 Infrastructure (基礎基礎設施) 索引標籤。
5. 在 Container instances (容器執行個體) 中，選取要取消註冊的執行個體 ID。系統會將您重新導向至容器執行個體詳細資訊頁面。
6. 在 Container Instance : ***id*** (容器執行個體：ID) 頁面中選擇 Deregister (取消註冊)。
7. 在確認畫面上，選擇取消註冊。
8. 如果您完成容器執行個體，請終止基礎的 Amazon EC2 執行個體。如需詳細資訊，請參閱 Amazon EC2 使用者指南中的終止執行個體。

排空 Amazon ECS 容器實例

有時您可能需要從叢集中移除容器執行個體，例如執行系統更新或縮減叢集容量。Amazon ECS 可讓您將容器執行個體轉換為 DRAINING 狀態。這稱為容器執行個體耗盡。將容器執行個體設定為 DRAINING 時，Amazon ECS 會避免在容器執行個體中放置新的任務排程。

服務的耗盡行為

會立即停止屬於 PENDING 狀態之服務的任何任務。如果叢集中有可用的容器執行個體容量，服務排程器將會啟動取代任務。如果容器執行個體容量不足，則會傳送服務事件訊息，指出該問題。

在容器執行個體上處於 RUNNING 狀態的服務的任務會轉換為 STOPPED 狀態。服務排程器會嘗試根據服務的部署類型和部署組態參數 (minimumHealthyPercent 和 maximumPercent) 來取代任務。如需詳細資訊，請參閱 Amazon ECS 服務 及 Amazon ECS 服務定義參數。

- 如果 minimumHealthyPercent 低於 100%，則排程器在任務取代期間可以暫時忽略 desiredCount。例如，desiredCount 為四項任務，下限 50% 允許排程器先停止兩項現有的任務，再開始兩項新的任務。如果下限為 100%，則直到替代任務視為正常運作前，服務排程器都無法移除現有的任務。如果未使用負載平衡器的服務任務為 RUNNING 狀態，則視為運作良好。如果使用負載平衡器的服務任務為 RUNNING 狀態，且負載平衡器回報託管所在的容器執行個體運作良好，任務即視為運作良好。

> **Important**
>
> 如果您使用 Spot 執行個體且 `minimumHealthyPercent` 大於或等於 100%，則在 Spot 執行個體終止之前，服務將沒有足夠的時間取代任務。

- `maximumPercent` 參數代表任務取代期間的執行任務數量上限，這允許您定義替代批次大小。例如，如果 `desiredCount` 為四項任務，可先啟動四項新任務再停止四項要耗盡任務的上限為 200% (前提是有執行此作業所需的可用叢集資源)。如果上限為 100%，則在要耗盡的任務停止之前，皆無法啟動替代任務。

> **Important**
>
> 如果 `minimumHealthyPercent` 和 `maximumPercent` 兩者皆為 100%，則服務無法刪除現有任務，也無法啟動替換任務。這可以防止容器執行個體成功耗盡，並避免進行新部署。

獨立任務的耗盡行為

處於 `PENDING` 或 `RUNNING` 狀態的任何獨立任務不會受到影響；您必須等待其自行停止或手動停止。容器執行個體將維持 `DRAINING` 狀態。

當執行個體上執行的所有任務轉換為 `STOPPED` 狀態時，容器執行個體完成耗盡。容器執行個體會保持在 `DRAINING` 狀態，直到再次啟動或刪除為止。您可以使用具有`containerInstance`參數的ListTasks操作來驗證容器執行個體上的任務狀態，以取得執行個體上的任務清單，然後使用 Amazon Resource Name (ARN) 或每個任務的 ID 進行DescribeTasks操作以驗證任務狀態。

當您準備好讓容器執行個體重新開始託管任務時，您可以將容器執行個體的狀態從 `DRAINING` 變更為 `ACTIVE`。然後，Amazon ECS 服務排程器將再次考慮容器執行個體以進行任務放置。

程序

透過運用新的 AWS Management Console，可使用下列步驟將容器執行個體設定為耗盡。

您也可以使用 UpdateContainerInstancesStateAPI 動作或更新容器執行個體狀態命令，將容器執行個體的狀態變更為。`DRAINING`

AWS Management Console

1. 開啟主控台，網址為 https://console.aws.amazon.com/ecs/v2。

2. 在導覽窗格中，選擇叢集。
3. 在 Clusters (叢集) 頁面上，選擇託管您執行個體的叢集。
4. 在 Cluster : ***name*** (叢集：名稱) 頁面上，選擇 Infrastructure (基礎基礎設施) 索引標籤。然後，在 Container instances (容器執行個體)下，選取您要耗盡之每個容器執行個體的核取方塊。
5. 依序選擇動作、耗盡。

Amazon ECS Linux 容器代理程式

Amazon ECS 代理程式是在叢集註冊的每個容器執行個體上執行的程序。它有助於您的容器執行個體和 Amazon ECS 之間的通訊。

每個 Amazon ECS 容器代理程式版本都支援不同的功能集，並提供先前版本的錯誤修復。可能的話，我們建議您一律使用最新版本的 Amazon ECS 容器代理程式。若要將您的容器代理更新到最新版本，請參閱「更新 Amazon ECS 容器代理程式」。

若要查看每個代理版本中包含哪些功能和強化功能，請參閱 https://github.com/aws/amazon-ecs-agent/releases。

> **Important**
>
> 可靠指標的最低 Docker 版本是 Docker 版本 `v20.10.13` 及更新版本，該版本隨附於 Amazon ECS 最佳化 AMI `20220607` 及更新版本中。
> Amazon ECS 代理程式版本 `1.20.0` 和更新版本已不支援 `1.9.0` 之前的 Docker 版本。

生命週期

當 Amazon ECS 容器代理程式在您的叢集中註冊 Amazon EC2 執行個體時，Amazon EC2 執行個體會將其狀態報告為 `ACTIVE`，其代理程式連線狀態為 `TRUE`。這個容器執行個體可以接受執行任務請求。

如果您停止 (非終止) 容器執行個體，狀態會維持在 `ACTIVE`，但代理程式連線狀態會在幾分鐘內轉換成 `FALSE`。容器執行個體上執行的所有任務都會停止。如果您再次啟動容器執行個體，容器代理會與 Amazon ECS 服務重新連線，而您即可再次在執行個體上執行任務。

> **Important**
>
> 如果您停止和啟動容器執行個體，或重新啟動該執行個體，一些較舊版本的 Amazon ECS 容器代理程式會再次註冊執行個體，而未取消註冊原始容器執行個體 ID。在這種情況下，Amazon ECS 所列出的容器執行個體會比您實際在叢集中擁有的還多。(如果同一個 Amazon EC2 執行個體 ID 有重複的容器執行個體 ID，您可以安全地將列為 `ACTIVE` 但代理程式連線狀態為 `FALSE` 的重複項目取消註冊。) 最新版本的 Amazon ECS 容器代理程式已修正此問題。如需有關更新至目前版本的詳細資訊，請參閱「更新 Amazon ECS 容器代理程式」。

如果您將容器執行個體的狀態變更為 `DRAINING`，新的任務就不會放置在容器執行個體中。如果可能，容器執行個體上執行的所有服務任務都會移除，以便您可以執行系統更新。如需詳細資訊，請參閱排空 Amazon ECS 容器實例。

如果您取消註冊或終止容器執行個體，則容器執行個體狀態會立即變更為 `INACTIVE`，且在您列出容器執行個體時將不再回報該容器執行個體。不過，終止後的一小時內您仍然可以描述容器執行個體。一小時之後，即無法再進行執行個體描述。

> **Important**
>
> 您可以手動耗盡執行個體，或者建置 Auto Scaling 群組 lifecycle hook 以將執行個體狀態設定為 `DRAINING`。如需有關 Auto Scaling lifecycle hook 的詳細資訊，請參閱 Amazon EC2 Auto Scaling lifecycle hook。

Amazon ECS 最佳化 AMI

Amazon ECS 最佳化 AMI 的 Linux 變體使用 Amazon Linux 2 AMI 作為其基礎。透過查詢 Systems Manager 參數存放區 API，可以擷取每個變體的 Amazon Linux 2 來源 AMI 名稱。如需詳細資訊，請參閱 擷取 Amazon EC 最佳化的 Linux AMI 中繼資料。透過最新的 Amazon ECS 最佳化 Amazon Linux 2 AMI 啟動我們的容器執行個體時，您會收到最新的容器代理程式版本。若要使用最新的 Amazon ECS 最佳化 Amazon Linux 2 AMI 來啟動容器執行個體，請參閱啟動 Amazon ECS Linux 容器執行個體。

其他資訊

下列頁面提供變更的其他資訊：

- Amazon ECS 代理更新日誌 GitHub
- `ecs-init` 應用程式的原始程式碼，以及用來封裝代理程式的指令碼和組態，現在已成為代理程式儲存庫的一部分。對於舊版本`ecs-init`和包裝，請參閱 Amazon ECS-init 更改日誌 GitHub
- Amazon Linux 2 版本備註
- Docker 文件中的 Docker 引擎版本備註
- NVIDIA 文件中的 NVIDIA 驅動程式文件

Amazon ECS 容器代理程式組態

Amazon ECS 容器代理程式支援多個組態選項，其中大部分是透過環境變數設定的。

如果是透過 Linux 版的 Amazon ECS 最佳化 AMI 來啟動容器執行個體，您可以在 `/etc/ecs/ecs.config` 檔案中設定這些環境變數，然後重新啟動代理程式。您也可以在啟動時使用 Amazon EC2 使用者資料，將這些組態變數寫入容器執行個體。如需詳細資訊，請參閱 引導 Amazon ECS Linux 容器執行個體以傳遞資料。

如果您的容器執行個體是使用 Amazon ECS 最佳化 AMI 的 Windows 變體啟動，您可以使用 PowerShell SetEnvironmentVariable 命令設定這些環境變數，然後重新啟動代理程式。如需詳細資訊，請參閱 Amazon EC2 使用者指南中的啟動時在 Windows 執行個體上執行命令和the section called “引導容器實例”。

如果您是手動啟用 Amazon ECS 容器代理程式 (適用於非 Amazon ECS 最佳化 AMI)，則可以在用於啟用代理程式的 docker run 命令中使用這些環境變數。請搭配使用這些變數和語法 `--env=`*`VARIABLE_NAME`*`=`*`VARIABLE_VALUE`*。如需私有儲存庫的身分驗證登入資料這類敏感資訊，您應該將代理環境變數存放至一個檔案中，並使用 `--env-file` *`path_to_env_file`* 選項一次傳遞。您可使用下列命令來新增這些變數。

```
sudo systemctl stop ecs
sudo vi /etc/ecs/ecs.config
# And add the environment variables with VARIABLE_NAME=VARIABLE_VALUE format.
sudo systemctl start ecs
```

可用參數

如需可用 Amazon ECS 容器代理程式組態參數的相關資訊，請參閱上的 Amazon ECS 容器代理程式。 GitHub

在 Amazon S3 中存儲亞馬遜 ECS 容器實例配置

Amazon ECS 容器代理程式組態由環境變數控制。當容器代理程式啟動並隨之設定代理程式時，Linux 版的 Amazon ECS 最佳化 AMI 會在 /etc/ecs/ecs.config 中尋找這些變數。某些無害的環境變數 (例如 ECS_CLUSTER) 可以在啟動時透過 Amazon EC2 使用者資料傳遞給容器執行個體，並寫入此檔案而不會帶來什麼影響。不過，其他敏感資訊 (例如 AWS 憑證或ECS_ENGINE_AUTH_DATA變數) 絕對不應該傳遞至使用者資料中的執行個體，也不應該以允許這些資訊顯示/etc/ecs/ecs.config在.bash_history檔案中的方式寫入執行個體。

在 Amazon S3 中的私有儲存貯體內存放組態資訊，並對容器執行個體 IAM 角色授予唯讀存取權，這是一種允許容器執行個體在啟動時進行設定的安全且便利方法。您可以將 ecs.config 文件的複本儲存在私有儲存貯體中。然後，您可以使用 Amazon EC2 使用者資料來安裝組態資訊， AWS CLI 並將其複製到執行個體啟動/etc/ecs/ecs.config時。

若要在 Amazon S3 中存放 **ecs.config** 檔案

1. 您必須授與容器執行個體角色 (ecs InstanceRole) 許可，才能擁有 Amazon S3 的唯讀存取權。您可以通過將 AmazonS3 ReadOnly 訪問權限分配給該角色來完成此操作ecsInstanceRole。如需如何將原則附加至角色的詳細資訊，請參閱AWS Identity and Access Management 使用指南中的修改角色權限原則 (主控台)
2. 使用下列格式建立包含有效 Amazon ECS 代理程式組態變數的 ecs.config 檔案。本範例將設定私有登錄檔身分驗證。如需詳細資訊，請參閱 在 Amazon ECS 中使用非AWS 容器映像。

   ```
   ECS_ENGINE_AUTH_TYPE=dockercfg
   ECS_ENGINE_AUTH_DATA={"https://index.docker.io/v1/":
   {"auth":"zq212MzEXAMPLE7o6T25Dk0i","email":"email@example.com"}}
   ```

   > **Note**
   >
   > 如需可用 Amazon ECS 代理程式組態變數的完整清單，請參閱上的 Amazon ECS 容器代理程式。 GitHub

3. 若要存放您的組態檔案，請在 Amazon S3 中建立私有儲存貯體。如需詳細資訊，請參閱 Amazon Simple Storage Service 主控台使用者指南中的建立儲存貯體。
4. 將 ecs.config 檔案上傳至 S3 儲存貯體。如需詳細資訊，請參閱 Amazon Simple Storage Service 使用者指南中的新增物件至儲存貯體。

若要在啟動時從 Amazon S3 載入 **ecs.config** 檔案

1. 完成本節稍早的程序，以允許 Amazon S3 以唯讀方式存取您的容器執行個體，並將 ecs.config 檔案存放在私有 S3 儲存貯體。
2. 啟動新的容器執行個體，並在 EC2 使用者資料中使用下列範例指令碼。指令碼會安裝組態檔案 AWS CLI 並將其複製到/etc/ecs/ecs.config。如需詳細資訊，請參閱 啟動 Amazon ECS Linux 容器執行個體。

```
#!/bin/bash
yum install -y aws-cli
aws s3 cp s3://your_bucket_name/ecs.config /etc/ecs/ecs.config
```

安裝 Amazon ECS 容器代理程式

如果您想要在 Amazon ECS 叢集註冊 Amazon EC2 執行個體，而該執行個體並未使用以 Amazon ECS 最佳化 AMI 為基礎的 AMI，則可以使用下列程序手動安裝 Amazon ECS 容器代理程式。若要這麼做，您可以從其中一個區域 Amazon S3 儲存貯體或從 Amazon 彈性容器登錄公用下載代理程式。如果您從其中一個區域 Amazon S3 儲存貯體下載，您可以選擇性地使用 PGP 簽章驗證容器代理程式檔案的有效性。

> **Note**
>
> Amazon ECS 和 Docker 服務的 systemd 單位都有一個指令，要在啟動這兩個服務前等待 cloud-init 完成。在您的 Amazon EC2 使用者資料完成執行前，cloud-init 程序不會被視為完成。因此，透過 Amazon EC2 使用者資料啟動 Amazon ECS 或 Docker 可能會造成死鎖。若要使用 Amazon EC2 使用者資料來啟動容器代理程式，您可以使用 systemctl enable --now --no-block ecs.service。

在非 Amazon Linux EC2 執行個體上安裝 Amazon ECS 容器代理程式

若要在 Amazon EC2 執行個體上安裝 Amazon ECS 容器代理程式，您可以從其中一個區域 Amazon S3 儲存貯體下載代理程式並進行安裝。

> **Note**
>
> 使用非 Amazon Linux AMI 時，您的 Amazon EC2 執行個體需要 `cgroupfs` 支援 `cgroup` 驅動程式，以便 Amazon ECS 代理程式支援任務層級資源限制。如需詳細資訊，請參閱上 GitHub的 Amazon ECS 代理程式。

每個系統架構最新的 Amazon ECS 容器代理程式檔案 (依區域) 列出如下，以供參考。

區域	區域名稱	Amazon ECS init deb 檔案	Amazon ECS init rpm 檔案
us-east-2	美國東部 (俄亥俄)	Amazon ECS init amd64 (amd64) Amazon ECS init arm64 (arm64)	Amazon ECS init x86_64 (x86_64) Amazon ECS aarch64 (aarch64)
us-east-1	美國東部 (維吉尼亞北部)	Amazon ECS init amd64 (amd64) Amazon ECS init arm64 (arm64)	Amazon ECS init x86_64 (x86_64) Amazon ECS aarch64 (aarch64)
us-west-1	美國西部 (加利佛尼亞北部)	Amazon ECS init amd64 (amd64) Amazon ECS init arm64 (arm64)	Amazon ECS init x86_64 (x86_64) Amazon ECS aarch64 (aarch64)
us-west-2	美國西部 (奧勒岡)	Amazon ECS init amd64 (amd64) Amazon ECS init arm64 (arm64)	Amazon ECS init x86_64 (x86_64) Amazon ECS aarch64 (aarch64)
ap-east-1	亞太區域 (香港)	Amazon ECS init amd64 (amd64)	Amazon ECS init x86_64 (x86_64)

區域	區域名稱	Amazon ECS init deb 檔案	Amazon ECS init rpm 檔案
		Amazon ECS init arm64 (arm64)	Amazon ECS aarch64 (aarch64)
ap-northeast-1	亞太區域 (東京)	Amazon ECS init amd64 (amd64) Amazon ECS init arm64 (arm64)	Amazon ECS init x86_64 (x86_64) Amazon ECS aarch64 (aarch64)
ap-northeast-2	亞太區域 (首爾)	Amazon ECS init amd64 (amd64) Amazon ECS init arm64 (arm64)	Amazon ECS init x86_64 (x86_64) Amazon ECS aarch64 (aarch64)
ap-south-1	亞太區域 (孟買)	Amazon ECS init amd64 (amd64) Amazon ECS init arm64 (arm64)	Amazon ECS init x86_64 (x86_64) Amazon ECS aarch64 (aarch64)
ap-southeast-1	亞太區域 (新加坡)	Amazon ECS init amd64 (amd64) Amazon ECS init arm64 (arm64)	Amazon ECS init x86_64 (x86_64) Amazon ECS aarch64 (aarch64)
ap-southeast-2	亞太區域 (悉尼)	Amazon ECS init amd64 (amd64) Amazon ECS init arm64 (arm64)	Amazon ECS init x86_64 (x86_64) Amazon ECS aarch64 (aarch64)

區域	區域名稱	Amazon ECS init deb 檔案	Amazon ECS init rpm 檔案
ca-central-1	加拿大 (中部)	Amazon ECS init amd64 (amd64) Amazon ECS init arm64 (arm64)	Amazon ECS init x86_64 (x86_64) Amazon ECS aarch64 (aarch64)
eu-central-1	歐洲 (法蘭克福)	Amazon ECS init amd64 (amd64) Amazon ECS init arm64 (arm64)	Amazon ECS init x86_64 (x86_64) Amazon ECS aarch64 (aarch64)
eu-west-1	歐洲 (愛爾蘭)	Amazon ECS init amd64 (amd64) Amazon ECS init arm64 (arm64)	Amazon ECS init x86_64 (x86_64) Amazon ECS aarch64 (aarch64)
eu-west-2	歐洲 (倫敦)	Amazon ECS init amd64 (amd64) Amazon ECS init arm64 (arm64)	Amazon ECS init x86_64 (x86_64) Amazon ECS aarch64 (aarch64)
eu-west-3	Europe (Paris)	Amazon ECS init amd64 (amd64) Amazon ECS init arm64 (arm64)	Amazon ECS init x86_64 (x86_64) Amazon ECS aarch64 (aarch64)
sa-east-1	南美洲 (聖保羅)	Amazon ECS init amd64 (amd64) Amazon ECS init arm64 (arm64)	Amazon ECS init x86_64 Amazon ECS aarch64 (aarch64)

區域	區域名稱	Amazon ECS init deb 檔案	Amazon ECS init rpm 檔案
us-gov-east-1	AWS GovCloud (美國東部)	Amazon ECS init amd64 (amd64) Amazon ECS init arm64 (arm64)	Amazon ECS init x86_64 (x86_64) Amazon ECS aarch64 (aarch64)
us-gov-west-1	AWS GovCloud (美國西部)	Amazon ECS init amd64 (amd64) Amazon ECS init arm64 (arm64)	Amazon ECS init x86_64 (x86_64) Amazon ECS aarch64 (aarch64)

使用非 Amazon Linux AMI 在 Amazon EC2 執行個體上安裝 Amazon ECS 容器代理程式

1. 使用允許存取 Amazon ECS 的 IAM 角色來啟動 Amazon EC2 執行個體。如需詳細資訊，請參閱 Amazon ECS 容器執行個體 IAM 角色。
2. 連線到您的執行個體。
3. 在您的執行個體上安裝最新版本的 Docker。
4. 檢查您的 Docker 版本，驗證您的系統符合最低版本需求。

   > **Note**
   >
   > 可靠指標的最低 Docker 版本是 Docker 版本 `v20.10.13` 及更新版本，該版本隨附於 Amazon ECS 最佳化 AMI `20220607` 及更新版本中。
   > Amazon ECS 代理程式版本 `1.20.0` 和更新版本已不支援 `1.9.0` 之前的 Docker 版本。

   ```
   docker --version
   ```

5. 下載適用於您作業系統和系統架構的 Amazon ECS 代理程式檔案並加以安裝。

   對於 deb 架構：

```
ubuntu:~$ curl -O https://s3.us-west-2.amazonaws.com/amazon-ecs-agent-us-west-2/
amazon-ecs-init-latest.amd64.deb
ubuntu:~$ sudo dpkg -i amazon-ecs-init-latest.amd64.deb
```

對於 rpm 架構：

```
fedora:~$ curl -O https://s3.us-west-2.amazonaws.com/amazon-ecs-agent-us-west-2/
amazon-ecs-init-latest.x86_64.rpm
fedora:~$ sudo yum localinstall -y amazon-ecs-init-latest.x86_64.rpm
```

6. 編輯/lib/systemd/system/ecs.service檔案，並在[Unit]區段末尾新增下列行。

```
After=cloud-final.service
```

7. (選用) 若要將執行個體註冊到 default 叢集之外的叢集，請編輯 /etc/ecs/ecs.config 檔案並新增以下內容。以下範例會指定 MyCluster 叢集。

```
ECS_CLUSTER=MyCluster
```

如需這些和其他代理執行時間選項的詳細資訊，請參閱「Amazon ECS 容器代理程式組態」。

> **Note**
>
> 您可以選擇性地將您的代理程式環境變數存放在 Amazon S3 中 (可在啟動時使用 Amazon EC2 使用者資料將其下載到您的容器執行個體)。針對敏感性資訊 (例如私有存放庫的身分驗證登入資料)，此為建議選項。如需詳細資訊，請參閱在 Amazon S3 中存儲亞馬遜 ECS 容器實例配置及在 Amazon ECS 中使用非AWS 容器映像。

8. 啟動 ecs 服務。

```
ubuntu:~$ sudo systemctl start ecs
```

使用主機網路模式執行 Amazon ECS 代理程式

當執行 Amazon ECS 容器代理程式時，ecs-init 將使用 host 網路模式建立容器代理程式容器。對於容器代理程式容器，這是唯一支援的網路模式。

對於容器代理程式啟動的容器，這允許您封鎖對 Amazon EC2 執行個體中繼資料服務端點 (http://169.254.169.254) 的存取。這可確保容器不能從容器執行個體設定檔存取 IAM 角色憑證，並強制任務只使用 IAM 任務角色憑證。如需詳細資訊，請參閱 Amazon ECS 任務 IAM 角色。

這也可讓容器代理程式不會在 docker0 橋接上爭奪連線和網路流量。

Amazon ECS 容器代理程式日誌組態參數

Amazon ECS 容器代理程式會在您的容器執行個體儲存紀錄。

針對 1.36.0 版和更新版本的容器代理程式，根據預設，紀錄會位在 Linux 執行個體的 /var/log/ecs/ecs-agent.log，以及 Windows 執行個體的 C:\ProgramData\Amazon\ECS\log\ecs-agent.log。

針對 1.35.0 版和更舊版本的容器代理程式，根據預設，紀錄會位在 Linux 執行個體的 /var/log/ecs/ecs-agent.log.*timestamp*，以及 Windows 執行個體的 C:\ProgramData\Amazon\ECS\log\ecs-agent.log.*timestamp*。

根據預設，代理程式紀錄會每小時輪換一次，最多儲存 24 個日誌。

下列是容器代理程式組態變數，可用來變更預設的代理程式紀錄行為。如需詳細資訊，請參閱 Amazon ECS 容器代理程式組態。

ECS_LOGFILE

範例值：/ecs-agent.log

Linux 的預設值：Null

Windows 的預設值：Null

代理日誌應寫入其中的位置。如果您透過ecs-init執行代理程式 (使用 Amazon ECS 最佳化 AMI 時的預設方法)，則容器內路徑為/log，並將該路徑ecs-init掛接到主機/var/log/ecs/上。

ECS_LOGLEVEL

範例值：crit、error、warn、info、debug

Linux 的預設值：info

Windows 的預設值：info

要記錄的細節層次。

ECS_LOGLEVEL_ON_INSTANCE

範例值：none、crit、error、warn、info、debug

Linux 上的預設值：如果 ECS_LOG_DRIVER 明確設定為非空白值，則為 none；否則會與 ECS_LOGLEVEL 的值相同

Windows 上的預設值：如果 ECS_LOG_DRIVER 明確設定為非空白值，則為 none；否則會與 ECS_LOGLEVEL 的值相同

可用於覆寫 ECS_LOGLEVEL，並設定應該記錄在執行個體上日誌檔案中的詳細資訊層級，與記錄驅動程式中記錄的層級分開。如果明確設定記錄驅動程式，則預設會關閉執行個體記錄。您可以使用此變數重新開啟它們。

ECS_LOG_DRIVER

範例值：awslogs、fluentd、gelf、json-file、journald、logentries、syslog、splunk

Linux 的預設值：json-file

Windows 的預設值：不適用

判斷代理程式容器使用的記錄驅動程式。

ECS_LOG_ROLLOVER_TYPE

範例值：size、hourly

Linux 的預設值：hourly

Windows 的預設值：hourly

判斷容器代理程式記錄檔是每小時還是根據大小輪替。根據預設，代理日誌檔每小時輪換一次。

ECS_LOG_OUTPUT_FORMAT

範例值：logfmt、json

Linux 的預設值：logfmt

Windows 的預設值：logfmt

決定日誌輸出格式。使用json格式時，記錄中的每一行都是結構化的 JSON 對映。

ECS_LOG_MAX_FILE_SIZE_MB

範例值：10

Linux 的預設值：10

Windows 的預設值：10

當ECS_LOG_ROLLOVER_TYPE變數設定為時size，此變數會決定輪換之前記錄檔的大小上限 (以 MB 為單位)。如果累積類型設為 hourly，則此變數將遭忽略。

ECS_LOG_MAX_ROLL_COUNT

範例值：24

Linux 的預設值：24

Windows 的預設值：24

決定要保留的輪換日誌檔數目。達到此限制後，就會刪除較舊的日誌檔案。

針對 1.36.0 和更新版本的容器代理程式，可參考以下使用 logfmt 格式時的範例日誌檔。

```
level=info time=2019-12-12T23:43:29Z msg="Loading configuration" module=agent.go
level=info time=2019-12-12T23:43:29Z msg="Image excluded from cleanup: amazon/amazon-
ecs-agent:latest" module=parse.go
level=info time=2019-12-12T23:43:29Z msg="Image excluded from cleanup: amazon/amazon-
ecs-pause:0.1.0" module=parse.go
level=info time=2019-12-12T23:43:29Z msg="Amazon ECS agent Version: 1.36.0, Commit:
 ca640387" module=agent.go
level=info time=2019-12-12T23:43:29Z msg="Creating root ecs cgroup: /ecs"
 module=init_linux.go
level=info time=2019-12-12T23:43:29Z msg="Creating cgroup /ecs"
 module=cgroup_controller_linux.go
level=info time=2019-12-12T23:43:29Z msg="Loading state!" module=statemanager.go
level=info time=2019-12-12T23:43:29Z msg="Event stream ContainerChange start
 listening..." module=eventstream.go
level=info time=2019-12-12T23:43:29Z msg="Restored cluster 'auto-robc'" module=agent.go
level=info time=2019-12-12T23:43:29Z msg="Restored from checkpoint file. I
 am running as 'arn:aws:ecs:us-west-2:0123456789:container-instance/auto-
robc/3330a8a91d15464ea30662d5840164cd' in cluster 'auto-robc'" module=agent.go
```

以下是使用 JSON 格式時的範例日誌檔。

```
{"time": "2019-11-07T22:52:02Z", "level": "info", "msg": "Starting Amazon Elastic
 Container Service Agent", "module": "engine.go"}
```

針對 1.35.0 和更舊版本的容器代理程式，以下是日誌檔的範例。

```
2016-08-15T15:54:41Z [INFO] Starting Agent: Amazon ECS Agent - v1.12.0 (895f3c1)
2016-08-15T15:54:41Z [INFO] Loading configuration
2016-08-15T15:54:41Z [WARN] Invalid value for task cleanup duration, will be overridden
 to 3h0m0s, parsed value 0, minimum threshold 1m0s
2016-08-15T15:54:41Z [INFO] Checkpointing is enabled. Attempting to load state
2016-08-15T15:54:41Z [INFO] Loading state! module="statemanager"
2016-08-15T15:54:41Z [INFO] Detected Docker versions [1.17 1.18 1.19 1.20 1.21 1.22]
2016-08-15T15:54:41Z [INFO] Registering Instance with ECS
2016-08-15T15:54:41Z [INFO] Registered! module="api client"
```

為私有碼頭映像設定 Amazon ECS 容器執行個體

Amazon ECS 容器代理程式可利用基本身分驗證，使用私有登錄檔進行身分驗證。當您啟用私有登錄檔身分驗證時，您可以使用您任務定義中的私有 Docker 映像。只有使用 EC2 啟動類型的任務才支援此功能。

啟用私人登錄驗證的另一種方法，會用 AWS Secrets Manager 來安全地儲存您的私人登錄認證，然後在容器定義中參考它們。這可讓您的任務使用來自私有儲存庫的映像。此方法支援使用 EC2 或 Fargate 啟動類型的任務。如需詳細資訊，請參閱 在 Amazon ECS 中使用非AWS 容器映像。

Amazon ECS 容器代理程式會在啟動時尋找兩個環境變數：

- `ECS_ENGINE_AUTH_TYPE` 可指定要傳送之身分驗證資料的類型。
- `ECS_ENGINE_AUTH_DATA` 包含實際身分驗證的登入資料。

Linux 版 Amazon ECS 最佳化 AMI 會在容器執行個體啟動及每一次服務啟動時 (使用 sudo start ecs 命令)，針對這些變數掃描 `/etc/ecs/ecs.config` 檔案。不是 Amazon ECS 最佳化的 AMI 應將這些環境變數存放在一個檔案中，並使用 `--env-file` *`path_to_env_file`* 選項將它們傳遞到啟動容器代理程式的 docker run 命令。

> **Important**
>
> 我們不建議您在執行個體啟動時使用 Amazon EC2 使用者資料插入這些身分驗證環境變數，或使用 `--env` 選項將它們傳遞至 docker run 命令。這些方法不適合用於敏感性資料，例如身分

驗證登入資料。如需將安全地將身分驗證登入資料新增至容器執行個體的詳細資訊，請參閱 在 Amazon S3 中存儲亞馬遜 ECS 容器實例配置。

身分驗證格式

私有登錄檔身分驗證有兩種可用的格式，dockercfg 和 docker。

dockercfg 身分驗證格式

dockercfg 格式使用存放在執行 docker login 命令時建立之組態檔中的身分驗證資訊。您可以透過在您的本機系統上執行 docker login，輸入您的登錄使用者名稱、密碼及電子郵件地址來建立此檔案。您也可以登入容器執行個體，並在該處執行命令。根據您的 Docker 版本，這個檔案會做為 ~/.dockercfg 或 ~/.docker/config.json 儲存。

```
cat ~/.docker/config.json
```

輸出：

```
{
  "auths": {
    "https://index.docker.io/v1/": {
      "auth": "zq212MzEXAMPLE7o6T25Dk0i"
    }
  }
}
```

Important

較新版本的 Docker 會如上所示，使用外部 auths 物件建立組態檔。Amazon ECS 代理程式只支援以下格式的 dockercfg 身分驗證資料，而不包含 auths 物件。如果您已安裝 jq 公用程式，即可使用以下命令擷取此資料：cat ~/.docker/config.json | jq .auths

```
cat ~/.docker/config.json | jq .auths
```

輸出：

```
{
```

```
  "https://index.docker.io/v1/": {
    "auth": "zq212MzEXAMPLE7o6T25Dk0i",
    "email": "email@example.com"
  }
}
```

在上述範例中，以下環境變數應新增到 Amazon ECS 容器代理程式在執行時間載入的環境變數檔案 (適用於 Amazon ECS 最佳化 AMI 的 /etc/ecs/ecs.config)。如果您不使用 Amazon ECS 最佳化 AMI，且您使用 docker run 手動啟動代理程式，請在啟動代理程式時使用 --env-file *path_to_env_file* 選項指定環境變數檔案。

```
ECS_ENGINE_AUTH_TYPE=dockercfg
ECS_ENGINE_AUTH_DATA={"https://index.docker.io/v1/":
{"auth":"zq212MzEXAMPLE7o6T25Dk0i","email":"email@example.com"}}
```

您可以使用以下語法設定多個私有登錄檔：

```
ECS_ENGINE_AUTH_TYPE=dockercfg
ECS_ENGINE_AUTH_DATA={"repo.example-01.com":
{"auth":"zq212MzEXAMPLE7o6T25Dk0i","email":"email@example-01.com"},"repo.example-02.com":
{"auth":"fQ172MzEXAMPLEoF7225DU0j","email":"email@example-02.com"}}
```

docker 身分驗證格式

docker 格式會使用代理程式應進行身分驗證的登錄伺服器 JSON 表示法。它也包含了該登錄所需要的身分驗證參數 (例如該帳戶的使用者名稱、密碼和電子郵件地址)。針對 Docker Hub 帳戶，JSON 表示法如以下內容所示：

```
{
  "https://index.docker.io/v1/": {
    "username": "my_name",
    "password": "my_password",
    "email": "email@example.com"
  }
}
```

在此範例中，以下環境變數應新增到 Amazon ECS 容器代理程式在執行時間載入的環境變數檔案 (適用於 Amazon ECS 最佳化 AMI 的 /etc/ecs/ecs.config)。如果您不使用 Amazon ECS 最佳化 AMI，且您使用 docker run 手動啟動代理程式，請在啟動代理程式時使用 --env-file *path_to_env_file* 選項指定環境變數檔案。

```
ECS_ENGINE_AUTH_TYPE=docker
ECS_ENGINE_AUTH_DATA={"https://index.docker.io/v1/":
{"username":"my_name","password":"my_password","email":"email@example.com"}}
```

您可以使用以下語法設定多個私有登錄檔：

```
ECS_ENGINE_AUTH_TYPE=docker
ECS_ENGINE_AUTH_DATA={"repo.example-01.com":
{"username":"my_name","password":"my_password","email":"email@example-01.com"},"repo.example-02
{"username":"another_name","password":"another_password","email":"email@example-02.com"}}
```

程序

使用以下程序，開啟容器執行個體的私有登錄檔。

若要在 Amazon ECS 最佳化 AMI 中啟用私有登錄檔

1. 使用 SSH 登入您的容器執行個體。
2. 開啟 `/etc/ecs/ecs.config` 檔案，為您的登錄檔和帳戶新增 `ECS_ENGINE_AUTH_TYPE` 和 `ECS_ENGINE_AUTH_DATA` 值：

   ```
   sudo vi /etc/ecs/ecs.config
   ```

   此範例會驗證 Docker Hub 使用者帳戶：

   ```
   ECS_ENGINE_AUTH_TYPE=docker
   ECS_ENGINE_AUTH_DATA={"https://index.docker.io/v1/":
   {"username":"my_name","password":"my_password","email":"email@example.com"}}
   ```

3. 檢查您的代理程式是否是使用 `ECS_DATADIR` 環境變數儲存其狀態：

   ```
   docker inspect ecs-agent | grep ECS_DATADIR
   ```

   輸出：

   ```
   "ECS_DATADIR=/data",
   ```

> **Important**
>
> 若先前的命令並未傳回 ECS_DATADIR 環境變數，您必須停止任何在此容器執行個體上執行的任務，才能停止您的代理。較新的代理會使用 ECS_DATADIR 環境變數儲存其狀態，讓您可以在任務執行中時停止和啟動它們，而不會有任何問題。如需詳細資訊，請參閱 更新 Amazon ECS 容器代理程式。

4. 停止 ecs 服務：

```
sudo stop ecs
```

輸出：

```
ecs stop/waiting
```

5. 重新啟動 ecs 服務。

- 對於 Amazon ECS 最佳化 Amazon Linux 2 AMI：

```
sudo systemctl restart ecs
```

- 對於 Amazon ECS 最佳化 Amazon Linux AMI：

```
sudo stop ecs && sudo start ecs
```

6. (選用) 您可以驗證代理已在執行中，並透過查詢代理自我檢查 API 操作，查看您新的容器執行個體的一些資訊。如需詳細資訊，請參閱 the section called “容器內省”。

```
curl http://localhost:51678/v1/metadata
```

自動 Amazon ECS 任務和圖像清理

每一次任務放置在容器執行個體上時，Amazon ECS 容器代理便會檢查任務中參考的映像是否是存放庫中指定標籤的最近版本。如果沒有，預設行為可讓代理程式從個別的儲存庫提取映像。若您經常更新任務和服務中的映像，您的容器執行個體儲存體可能會很快的被不再使用且可能永遠不會再使用的 Docker 映像填滿。例如，您可以使用連續整合和連續部署 (CI/CD) 管道。

> **Note**
>
> 您可以使用 ECS_IMAGE_PULL_BEHAVIOR 參數自訂 Amazon ECS 代理程式映像提取行為。如需詳細資訊，請參閱 Amazon ECS 容器代理程式組態。

同樣的，屬於已停止任務的容器也會使用日誌資訊、資料磁碟區和其他成品取用容器執行個體儲存體。這些成品有助於偵錯未預期停止的容器，但其中大多數的儲存體都可在一段時間之後安全的釋放。

根據預設，Amazon ECS 容器代理會自動清除停止的任務和您容器執行個體上不再由任何任務使用的 Docker 映像。

> **Note**
>
> 自動化映像清除功能需要至少 1.13.0 版本的 Amazon ECS 容器代理程式。若要將您的代理更新到最新版本，請參閱「更新 Amazon ECS 容器代理程式」。

以下代理組態變數可用來調整您的自動化任務和映像清除體驗。如需如何在您的容器執行個體上設定這些變數的詳細資訊，請參閱「Amazon ECS 容器代理程式組態」。

`ECS_ENGINE_TASK_CLEANUP_WAIT_DURATION`

此變數可指定移除屬於已停止任務的任何容器前，應等待的時間。只要有容器仍在參考該映像，映像清除處理序便無法刪除映像。當映像不再由任何容器 (停止的容器或正在執行的容器) 參考時，映像便可進行清除。根據預設，此參數會設為 3 小時，但您可以將此期間減少至 1 秒鐘 (若您的應用程式需要的話)。如果您將值設定為小於 1 秒，則會忽略此參數。

`ECS_DISABLE_IMAGE_CLEANUP`

若您將此變數設為 `true`，則系統會關閉您的容器執行個體上的自動化映像清除，並且不會自動移除任何映像。

`ECS_IMAGE_CLEANUP_INTERVAL`

此變數指定自動化映像清除處理序檢查要刪除之映像的頻率。預設值是每 30 分鐘，但您最多可以將此期間減少至 10 分鐘，以更頻繁的移除映像。

`ECS_IMAGE_MINIMUM_CLEANUP_AGE`

此變數可指定提取映像後，到可清除該映像之間最小的時間長度。這可防止清除才剛提取的映像。預設值為 1 小時。

ECS_NUM_IMAGES_DELETE_PER_CYCLE

此變數會指定單一清除週期中可移除多少映像。預設值為 5，最小值為 1。

當 Amazon ECS 容器代理程式正在執行中，且自動化映像清除未關閉時，代理程式會檢查未被執行中或已停止的容器參考的 Docker 映像檔，檢查頻率由 ECS_IMAGE_CLEANUP_INTERVAL 變數決定。若有找到未使用的映像，且其時間比 ECS_IMAGE_MINIMUM_CLEANUP_AGE 變數指定的最小清除時間還舊，則代理最多會移除 ECS_NUM_IMAGES_DELETE_PER_CYCLE 變數所指定數量的映像。最近參考時間距離現在最久的映像會先遭到刪除。在移除映像後，代理會等待直到下一個間隔，並重複此程序。

在 Amazon ECS 上安排您的容器

Amazon Elastic Container Service (Amazon ECS) 是一種共用狀態、樂觀的並行系統，可為您的容器化工作負載提供彈性的排程功能。Amazon ECS 排程器會利用與 Amazon ECS API 相同的叢集狀態資訊，進行適當的置放決策。

Amazon ECS 提供為長時間執行的任務和應用程式服務排程器。它還提供了針對批次作業或單一執行工作執行獨立工作或排程工作的功能。您可以指定最符合您的需求的任何置放策略及執行任務的限制條件。例如，您可以指定任務跨多個可用區域或在單一可用區域內執行。另外，您還可以選擇性地將任務與您自己的自訂或第三方排程器整合。

選項	使用情況	其他資訊
服務	服務排程器適用於長時間執行的無狀態服務和應用程式。服務排程器也會選擇性地確保任務是針對 Elastic Load Balancing 負載平衡器所註冊。您可以更新由服務排程器維護的服務。這其中可能包括部署新的任務定義或變更正在執行的所需任務數。根據預設，服務排程器會將任務分散至多個可用區域。不過，您可以使用任務置放策略和限制條件，來自訂任務置放決策。	Amazon ECS 服務
獨立任務	獨立任務適用於諸如執行工作然後停止的批次工作之類的處理程序。例如，您可以在工作進入佇列時讓處理序呼叫 RunTask。任務會從佇列中提取工作、執行工作，然後結束。使用 RunTask，您可以允許預設任務置放策略在您的叢集上隨機分配任務。這可能最	Amazon ECS 獨立任務

選項	使用情況	其他資訊
	大限度降低單一執行個體取得不成比例之任務數的機率。	
排程任務	當您要在叢集中以設定的間隔執行工作時，排定的工作是適合的，您可以使用 EventBridge Scheduler 來建立排程。您可以為備份操作或日誌掃描執行任務。您建立的 EventBridge 排程器排程可以在指定的時間在叢集中執行一或多個工作。您的排程事件可以設定為特定的時間間隔 (每隔 *N* 分鐘、小時或天執行)。此外，對於更複雜的排程，您可以使用 `cron` 運算式。	使用 Amazon EventBridge 排程器來安排 Amazon ECS 任務

運算選項

使用 Amazon ECS，您可以指定執行任務或服務的基礎設施。您可以使用容量提供者策略或啟動類型。

對於 Fargate，容量提供者是 Fargate 和 Fargate 點。對於 EC2，容量提供者是具有已註冊容器執行個體的 Auto Scaling 群組。

容量提供者策略會將您的工作分配到與叢集相關聯的容量提供者。

只有已經與叢集關聯並具有 `ACTIVE` 或 `UPDATING` 狀態的容量提供者，才能在容量提供者策略中使用。您可以在建立叢集時將容量提供者關聯到叢集。

在容量提供者策略中，選用的基準值會指明指定容量提供者上至少會執行多少任務數量。容量提供者策略中只有一個容量提供者可以定義基礎。

權重值會決定使用指定容量提供者其已啟動任務總數的相對百分比。請考量下列範例。您的策略包含兩個容量提供者，且兩者的權重值都是 1。達到基準百分比時，任務會平均分配給兩個容量提供者。以此類推，假如您為 capacityProviderA 指定了權重值 1，為 capacityProviderB 指定了權重值 4。那麼，每有一個任務使用 capacityProviderA 執行，就會有四個任務使用 capacityProviderB 執行。

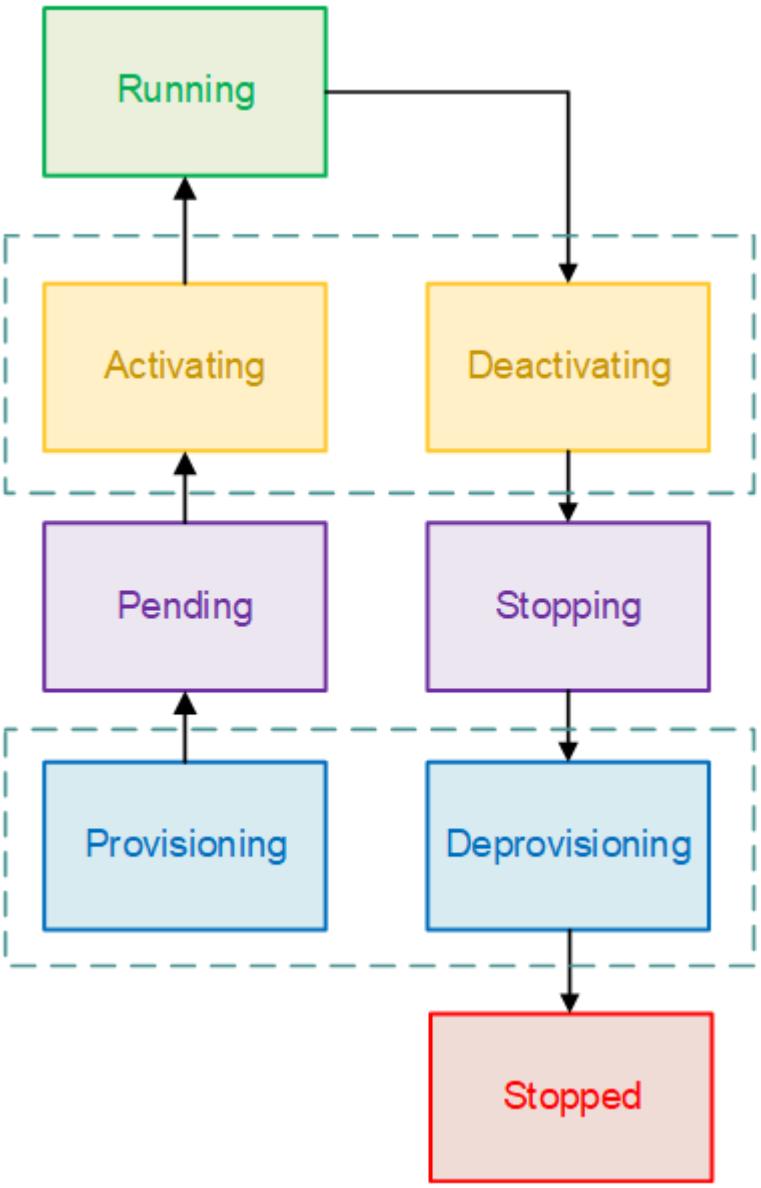
啟動類型會直接在 Fargate 或您手動註冊到叢集的 Amazon EC2 執行個體上啟動您的任務。

Amazon ECS 任務生命週期

無論是以手動方式或是做為服務的一部分，當任務啟動時，它可以在自動完成或手動停止之前通過數個狀態。有些任務是做為批次任務執行的，因此會很自然地從 PENDING 前進到 RUNNING，最後前進到 STOPPED。其他可做為服務一部分的任務會不限期的持續執行，或者可根據需要擴展或縮小。

當請求任務狀態變更時 (例如停止任務或更新需要擴展或縮小的服務計數時)，Amazon ECS 容器代理程式會以任務的最後已知狀態 (`lastStatus`)，以及任務的所需狀態 (`desiredStatus`) 來追蹤這些變更。最後已知狀態和所需狀態，都可在主控台或透過使用 API 或 AWS CLI說明任務來查看。

下面的流程圖顯示任務生命週期流程。



生命週期狀態

以下是每個任務生命週期狀態的說明。

佈建中

Amazon ECS 必須執行額外的步驟，然後再啟動任務。例如，對於使用 awsvpc 網路模式的任務，需要佈建彈性網路介面。

待定

這是一個轉換狀態，Amazon ECS 在容器代理程式上等待採取進一步的動作。任務會一直處於待處理狀態，直到任務有可用的資源為止。

啟動中

這是一種轉換狀態，其中 Amazon ECS 必須在啟動任務後但在任務可轉移為 RUNNING 狀態前執行額外的步驟。例如，對於已設定服務探索的任務，必須建立服務探索資源。任務若屬於設為要使用多個 Elastic Load Balancing 目標群組的服務，則目標群組註冊會發生於此狀態期間。

RUNNING (執行中)

任務成功執行中。

停用中

這是一種轉換狀態，其中 Amazon ECS 必須在任務停止之前執行其他步驟。例如，任務若屬於設為要使用多個 Elastic Load Balancing 目標群組的服務，則目標群組取消註冊會發生於此狀態期間。

停止中

這是一個轉換狀態，Amazon ECS 在容器代理程式上等待採取進一步的動作。

對於 Linux 容器，容器代理將發送SIGTERM信號以通知應用程序需要完成並關閉，然SIGKILL後在等待任務定義中設置的StopTimeout持續時間後發送一個。

取消佈建中

Amazon ECS 必須在任務停止後但在任務轉移為 STOPPED 狀態之前執行額外的步驟。例如，對於使用 awsvpc 網路模式的任務，需要分離並刪除彈性網路介面。

已停止

已成功停止任務。

如果您的工作因為錯誤而停止，請參閱查看 Amazon ECS 停止的任務錯誤。

DELETED

這是任務停止時的過渡狀態。此狀態不會在主控台中顯示，但是會顯示在 describe-tasks。

Amazon ECS 如何在容器執行個體上放置任務

您可以使用任務放置來設定 Amazon ECS，將任務放在符合特定條件的容器執行個體上，例如可用區域或執行個體類型。

以下是任務放置元件：

- 任務放置策略-用於為任務放置選擇容器實例或終止任務的算法。例如，Amazon ECS 可以隨機選取容器執行個體，也可以選取容器執行個體，以便將任務平均分配到一組執行個體。
- 任務群組-一組相關工作，例如資料庫工作。
- 任務放置限制-這些是將任務放置在容器實例上必須符合的規則。如果不滿足約束，則不會放置任務並保持在狀PENDING態。例如，您可以使用條件約束僅將工作放置在特定例證類型上。

Amazon ECS 對於啟動類型有不同的算法。

EC2 啟動類型

對於使用 EC2 啟動類型的任務，Amazon ECS 必須根據任務定義中指定的要求 (例如 CPU 和記憶體) 決定要放置任務的位置。同樣地，當您縮減任務計數時，Amazon ECS 必須判斷要終止的任務。您可以套用任務置放策略和限制條件，來自訂 Amazon ECS 如何放置和終止任務。

預設任務放置策略取決於您是手動執行工作 (獨立工作) 還是在服務中執行工作。如果任務是作為 Amazon ECS 服務的一部分來執行，任務置放策略會是使用 `attribute:ecs.availability-zone` 來 `spread`。服務中的任務無預設任務置放限制。如需詳細資訊，請參閱 在 Amazon ECS 上安排您的容器。

> **Note**
>
> 任務置放策略是一種最佳作法。即使最佳置放選項無法使用，Amazon ECS 仍然會嘗試放置任務。不過，任務置放限制條件具有約束性，且可能妨礙任務置放。

您可以同時使用任務置放策略和限制條件。例如，您可以使用任務放置策略和任務放置限制，根據每個可用區域內的記憶體，跨可用區域和分箱封裝任務來分配工作，但僅適用於 G2 執行個體。

Amazon ECS 放置任務時，會使用下列程序來選取容器執行個體：

1. 識別符合工作定義中 CPU、GPU、記憶體和連接埠需求的容器執行個體。
2. 識別符合任務放置限制的容器執行個體。
3. 識別符合任務放置策略的容器執行個體。
4. 選取要放置工作的容器執行個體。

Fargate 啟動類型

任務放置策略和限制條件不支援使用 Fargate 啟動類型的任務。Fargate 將盡力將任務分散至可存取的可用區域。如果容量提供者同時包含 Fargate 和 Fargate Spot，則每個容量提供者的分散行為均各自獨立。

使用策略定義 Amazon ECS 任務放置

對於使用 EC2 啟動類型的任務，Amazon ECS 必須根據任務定義中指定的要求 (例如 CPU 和記憶體) 決定要放置任務的位置。同樣地，當您縮減任務計數時，Amazon ECS 必須判斷要終止的任務。您可以套用任務置放策略和限制條件，來自訂 Amazon ECS 如何放置和終止任務。

預設任務放置策略取決於您是手動執行工作 (獨立工作) 還是在服務中執行工作。如果任務是作為 Amazon ECS 服務的一部分來執行，任務置放策略會是使用 `attribute:ecs.availability-zone` 來 `spread`。服務中的任務無預設任務置放限制。如需詳細資訊，請參閱 在 Amazon ECS 上安排您的容器。

> **Note**
>
> 任務置放策略是一種最佳作法。即使最佳置放選項無法使用，Amazon ECS 仍然會嘗試放置任務。不過，任務置放限制條件具有約束性，且可能妨礙任務置放。

您可以同時使用任務置放策略和限制條件。例如，您可以使用任務放置策略和任務放置限制，根據每個可用區域內的記憶體，跨可用區域和分箱封裝任務來分配工作，但僅適用於 G2 執行個體。

Amazon ECS 放置任務時，會使用下列程序來選取容器執行個體：

1. 識別符合工作定義中 CPU、GPU、記憶體和連接埠需求的容器執行個體。
2. 識別符合任務放置限制的容器執行個體。
3. 識別符合任務放置策略的容器執行個體。
4. 選取要放置工作的容器執行個體。

您可以在服務定義中指定任務放置策略，或使用`placementStrategy`參數指定任務定義。

```
"placementStrategy": [
    {
        "field": "The field to apply the placement strategy against",
        "type": "The placement strategy to use"
```

```
    }
]
```

您可以在執行工作 (RunTask)、建立新服務 () 或更新現有服務 (UpdateService) 時指定策略。CreateService

下表說明可用的類型和欄位。

type	有效的欄位值	
`binpack` 任務會放置於容器執行個體上，以便保留最少量的未使用 CPU 或記憶體。此策略會將使用中的容器執行個體數量減至最少。 當使用此策略並採取縮減動作時，Amazon ECS 會終止任務。此策略根據任務終止後保留在容器執行個體上的資源量來執行此操作。在任務終止後保留最多可用資源的容器執行個體，會終止該任務。	• cpu • memory	
`random` 隨機放置任務。	未使用	
`spread` 根據指定的值平均放置任務。服務任務會根據該服務的任務進行散佈。獨立的任務會以同一任務群組的任務為基礎分配。如需有關任務群組的詳細資訊，請參閱 群組相關的 Amazon ECS 任務 。	• `instanceId` (或`host`具有相同效果) • 套用至容器執行個體的任何平台或自訂屬性，例如 `attribute:ecs.availability-zone`	

type	有效的欄位值	
當使用 spread 策略並執行縮減動作時，Amazon ECS 會選擇個在可用區域之間保持平衡的任務進行終止。在一個可用區域內，任務會被隨機選取。		

任務放置策略也可以為現有服務更新。如需詳細資訊，請參閱 Amazon ECS 如何在容器執行個體上放置任務。

您可以透過依照您希望執行的順序建立策略陣列來建立使用多個策略的任務放置策略。例如，如果您想要跨可用區域分散任務，然後在每個可用區域內根據記憶體對任務進行分箱封裝，請指定可用區域策略，然後指定記憶體策略。如需策略範例，請參閱 Amazon ECS 任務放置策略示例。

Amazon ECS 任務放置策略示例

您可以使用下列動作來指定任務放置策略：CreateServiceUpdateService、和RunTask。

範例

- 將任務平均分佈至可用區域
- 將任務平均分佈至所有執行個體
- 根據記憶體對任務進行分箱封裝
- 隨機放置任務。
- 將任務平均分佈至各個可用區域，然後將任務平均分佈至每個可用區域內的執行個體
- 將任務平均分佈至各個可用區域，然後在每個可用區域內根據記憶體對任務進行分箱封裝
- 將任務平均分佈至執行個體，然後根據記憶體對任務進行分箱封裝

將任務平均分佈至可用區域

下列策略會將任務平均分散到各個可用區域。

```
"placementStrategy": [
    {
        "field": "attribute:ecs.availability-zone",
        "type": "spread"
    }
```

```
]
```

將任務平均分佈至所有執行個體

下列策略會將任務平均分散到所有執行個體。

```
"placementStrategy": [
    {
        "field": "instanceId",
        "type": "spread"
    }
]
```

根據記憶體對任務進行分箱封裝

下列策略會根據記憶體對任務進行分箱封裝。

```
"placementStrategy": [
    {
        "field": "memory",
        "type": "binpack"
    }
]
```

隨機放置任務。

下列策略會隨機放置任務。

```
"placementStrategy": [
    {
        "type": "random"
    }
]
```

將任務平均分佈至各個可用區域，然後將任務平均分佈至每個可用區域內的執行個體

下列策略會將任務平均分散到各個可用區域，然後將任務平均分散到每個可用區域內的執行個體。

```
"placementStrategy": [
    {
        "field": "attribute:ecs.availability-zone",
        "type": "spread"
```

```
    },
    {
        "field": "instanceId",
        "type": "spread"
    }
]
```

將任務平均分佈至各個可用區域，然後在每個可用區域內根據記憶體對任務進行分箱封裝

下列策略會將任務平均分散到各個可用區域，然後根據每個可用區域內的記憶體對任務進行分箱封裝。

```
"placementStrategy": [
    {
        "field": "attribute:ecs.availability-zone",
        "type": "spread"
    },
    {
        "field": "memory",
        "type": "binpack"
    }
]
```

將任務平均分佈至執行個體，然後根據記憶體對任務進行分箱封裝

下列策略會將任務平均分佈至所有執行個體，然後在每個執行個體內根據記憶體對任務進行分箱封裝。

```
"placementStrategy": [
    {
        "field": "instanceId",
        "type": "spread"
    },
    {
        "field": "memory",
        "type": "binpack"
    }
]
```

群組相關的 Amazon ECS 任務

您可以識別一組相關任務，並將其放置在任務群組中。在使用 spread 任務置放策略時，會將所有具有相同任務群組名稱的任務視為一組。例如，假設您在一個叢集執行不同的應用程式，例如資料庫和 Web 伺服器。為了確保您在各個可用區域中之資料庫的平衡，請將它們新增至名為 databases 的任

務群組，然後使用 spread 任務置放策略。如需詳細資訊，請參閱 使用策略定義 Amazon ECS 任務放置。

任務群組也可用作任務放置限制條件。當您在 memberOf 限制條件中指定任務群組時，任務只會傳送到在指定任務群組中執行的容器執行個體。如需範例，請參閱Amazon ECS 任務放置限制範例。

根據預設，若未指定自訂任務群組名稱，獨立任務會使用任務定義系列名稱 (如 family:my-task-definition) 做為任務群組名稱。若任務做為服務的一部分啟動，它將使用服務名稱做為任務群組名稱，而且該名稱無法被變更。

適用於任務群組的下列需求。

- 任務群組名稱必須在 255 個字元以下。
- 每個任務只能在一個群組中。
- 啟動任務之後，就無法修改其任務群組。

定義 Amazon ECS 用於執行任務的容器執行個體

任務放置限制是 Amazon ECS 用來判斷是否允許在執行個體上執行任務的容器執行個體的規則。至少有一個容器執行個體必須符合限制條件。如果沒有符合條件限制的執行個體，則任務將會保持 PENDING 狀態。當您建立新服務或更新現有服務時，您可以為服務的任務指定任務放置限制條件。

您可以使用placementConstraint參數在服務定義、任務定義或任務中指定任務位置限制。

```
"placementConstraint": [
    {
        "expression": "The expression that defines the task placement constraints",
        "type": "The placement constraint type to use"
    }
]
```

下表說明如何使用這些參數。

Constraint type (限制條件類型)	可以指定時
distinctInstance 將每個任務放置在不同的容器執行個體。	• 執行工作 RunTask • 創建一個新的服務 CreateService，

Constraint type (限制條件類型)	可以指定時	
Important 我們建議客戶希望對其任務進行強大隔離的客戶使用 Fargate。Fargate 會在硬體虛擬化環境中執行每項工作。這可確保這些容器化工作負載不會與其他工作共用網路介面、Fargate 暫時性儲存、CPU 或記憶體。如需詳細資訊，請參閱 AWS Fargate.		
`memberOf` 在滿足運算式的容器執行個體上放置任務。	• 執行工作 RunTask • 創建一個新的服務 CreateService， • 建立新的任務定義定 RegisterTask義 • 建立任務定RegisterTask義的新版本修訂 • 更新服務 UpdateService	

使用`memberOf`約束類型時，您可以使用叢集查詢語言建立運算式，該語言定義了 Amazon ECS 可在其中放置任務的容器執行個體。運算式是您依屬性分組容器執行個體的一種方式。運算式會進入的`expression` 參數`placementConstraint`。

Amazon ECS 容器實例屬性

您可以將自訂中繼資料新增至容器執行個體，稱為屬性。每個屬性都有名稱和選用字串值。您可以使用 Amazon ECS 所提供的內建屬性，或定義自訂屬性。

以下章節包含範例內建屬性、選擇性屬性和自訂屬性。

內建屬性

Amazon ECS 會將下列屬性自動套用至您的容器執行個體。

`ecs.ami-id`

用來啟動執行個體的 AMI ID。此屬性的範例值為 `ami-1234abcd`。

`ecs.availability-zone`

執行個體的可用區域。此屬性的範例值為 `us-east-1a`。

`ecs.instance-type`

執行個體的執行個體類型。此屬性的範例值為 `g2.2xlarge`。

`ecs.os-type`

執行個體的作業系統。此屬性的可能值為 `linux` 和 `windows`。

`ecs.os-family`

執行個體的作業系統版本。

若為 Linux 執行個體，有效值為 `LINUX`。若為 Windows 執行個體，ECS 會以 `WINDOWS_SERVER_<OS_Release>_<FULL or CORE>` 格式設定值。有效值為 `WINDOWS_SERVER_2022_FULL`、`WINDOWS_SERVER_2022_CORE`、`WINDOWS_SERVER_20H2_CORE`、`W` 與 `WINDOWS_SERVER_2016_FULL`。

這對於 Windows 容器很重要， Windows containers on AWS Fargate 因為每個 Windows 容器的作業系統版本都必須與主機的作業系統版本相符。如果容器映像的 Windows 版本與主機不同，則容器不會啟動。如需詳細資訊，請參閱 Microsoft 文件網站上的 Windows 容器版本相容性。

如果您的叢集執行多個 Windows 版本，您可以使用置放條件限制：`memberOf(attribute:ecs.os-family == WINDOWS_SERVER_<OS_Release>_<FULL or CORE>)`，確保將任務置放在執行相同版本的 EC2 執行個體上。如需詳細資訊，請參閱 the section called “擷取 Amazon EC 最佳化視窗 AMI 中繼資料”。

`ecs.cpu-architecture`

執行個體的 CPU 架構。此屬性的範例值為 `x86_64` 和 `arm64`。

`ecs.vpc-id`

執行個體啟動所在位置的 VPC。此屬性的範例值為 `vpc-1234abcd`。

`ecs.subnet-id`

執行個體使用的子網路。此屬性的範例值為 `subnet-1234abcd`。

選擇性屬性

Amazon ECS 可能會將以下屬性新增到您的容器執行個體。

`ecs.awsvpc-trunk-id`

如果此屬性存在，執行個體則具有一個幹線網路界面。如需詳細資訊，請參閱 增加 Amazon ECS Linux 容器實例網絡界面。

`ecs.outpost-arn`

如果此屬性存在，它會包含 Outpost 的 Amazon Resource Name (ARN)。如需詳細資訊，請參閱 the section called “Amazon 彈性容器服務 AWS Outposts”。

`ecs.capability.external`

若存在此屬性，執行個體將被識別為外部執行個體。如需詳細資訊，請參閱 適用於外部啟動類型的 Amazon ECS 叢集。

自訂屬性

您可以將自訂屬性套用至容器執行個體。例如，您可以定義名稱為 "stack" 且值為 "prod" 的屬性。

在指定自訂屬性時，必須考慮下列事項。

- `name` 必須包含 1 至 128 個字元，而名稱可能包含字母 (大小寫)、數字、連字號、底線、正斜線、反斜線或句號。
- `value` 必須包含 1 至 128 個字元，而且可能包含字母 (大小寫)、數字、連字號、底線、句號、@ 符號、正斜線、反斜線、冒號或空格。值不得包含任何前置或結尾空格。

建立運算式以定義 Amazon ECS 任務的容器執行個體

叢集查詢是可讓您將物件分組的運算式。例如，您可以依屬性 (例如可用區域、執行個體類型或自訂中繼資料) 將容器執行個體分組。如需詳細資訊，請參閱 Amazon ECS 容器實例屬性。

在您定義一組容器執行個體之後，即可自訂 Amazon ECS 以根據群組在容器執行個體上放置任務。如需詳細資訊，請參閱 以 Amazon ECS 任務的形式執行應用程式 和 使用主控台建立 Amazon ECS 服務。您也可以在列出容器執行個體時套用群組篩選條件。

運算式語法

運算式的語法如下：

```
subject operator [argument]
```

主旨

要評估的屬性或欄位。

`agentConnected`

依 Amazon ECS 容器代理程式連線狀態來選取容器執行個體。您可以使用此篩選條件來搜尋具有代理程式已中斷連線的執行個體容器。

有效運算子：equals (==)、not_equals (!=)、in、not_in (!in)、matches (=~)、not_matches (!~)

`agentVersion`

依 Amazon ECS 容器代理程式版本來選取容器執行個體。您可以使用此篩選條件，尋找正在執行過期版本 Amazon ECS 容器代理程式的執行個體。

有效運算子：equals (==)、not_equals (!=)、greater_than (>)、greater_than_equal (>=)、less_than (<)、less_than_equal (<=)

`attribute:`*`attribute-name`*

依屬性選取容器執行個體。如需詳細資訊，請參閱 Amazon ECS 容器實例屬性。

`ec2InstanceId`

依 Amazon EC2 執行個體 ID 選取容器執行個體。

有效運算子：equals (==)、not_equals (!=)、in、not_in (!in)、matches (=~)、not_matches (!~)

`registeredAt`

依容器執行個體註冊日期來選取容器執行個體。您可以使用此篩選條件，尋找新註冊執行個體或非常舊的執行個體。

有效運算子：equals (==)、not_equals (!=)、greater_than (>)、greater_than_equal (>=)、less_than (<)、less_than_equal (<=)

有效日期格式：2018-06-18T22:28:28+00:00, 2018-06-18T22:28:28Z, 2018-06-18T22:28:28, 2018-06-18

`runningTasksCount`

依執行中的任務數量來選取容器執行個體。您可以使用此篩選條件來尋找空白或接近空白 (在其上執行的任務很少) 的執行個體。

有效運算子：equals (==)、not_equals (!=)、greater_than (>)、greater_than_equal (>=)、less_than (<)、less_than_equal (<=)

`task:group`

依任務群組選取容器執行個體。如需詳細資訊，請參閱 群組相關的 Amazon ECS 任務 。

運算子

比較運算子。下列是支援的運算子。

運算子	描述
==、equals	字串相等
!=、not_equals	字串不相等
>、greater_than	大於
>=、greater_than_equal	大於或等於
<、less_than	小於
<=、less_than_equal	小於或等於
exists	主旨存在
!exists、not_exists	主旨不存在
in	引數清單中的值
!in、not_in	不在引數清單中的值
=~、matches	模式相符
!~、not_matches	模式不相符

> **Note**
>
> 單一運算式不能包含括號。不過，可以使用括號指定複合運算式中的優先順序。

引數

對於許多運算子而言，引數是一種常值。

in 和 not_in 運算子需要有參數清單做為引數。您可以指定引數清單，如下所示：

```
[argument1, argument2, ..., argumentN]
```

matches 和 not_matches 運算子需要有符合 Java 一般運算式語法的引數。如需詳細資訊，請參閱 java.util.regex.Pattern。

複合運算式

您可以使用下列布林值運算子來合併運算式：

- &&, 和
- ||, 或
- !, 非

您可以使用括號來指定優先順序：

```
(expression1 or expression2) and expression3
```

範例運算式

下列是範例運算式。

範例：字串相等

下列運算式會選取具有指定之執行個體類型的執行個體。

```
attribute:ecs.instance-type == t2.small
```

範例：引數清單

下列運算式會選取 us-east-1a 或 us-east-1b 可用區域中的執行個體。

```
attribute:ecs.availability-zone in [us-east-1a, us-east-1b]
```

範例：複合運算式

下列運算式會選取不在 us-east-1d 可用區域中的 G2 執行個體。

```
attribute:ecs.instance-type =~ g2.* and attribute:ecs.availability-zone != us-east-1d
```

範例：任務親和性

下列運算式會選取在 `service:production` 群組中託管任務的執行個體。

```
task:group == service:production
```

範例：任務反親和性

下列運算式會選取未在資料庫群組中託管任務的執行個體。

```
not(task:group == database)
```

範例：執行中任務計數

下列運算式會選取僅執行一項任務的執行個體。

```
runningTasksCount == 1
```

範例：Amazon ECS 容器代理程式版本

下列運算式會選取正在執行低於 1.14.5 版本容器代理程式的執行個體。

```
agentVersion < 1.14.5
```

範例：執行個體註冊時間

下列運算式會選取在 2018 年 2 月 13 日之前註冊的執行個體。

```
registeredAt < 2018-02-13
```

範例：Amazon EC2 執行個體 ID

下列運算式會選取具有下列 Amazon EC2 執行個體 ID 的執行個體。

```
ec2InstanceId in ['i-abcd1234', 'i-wxyx7890']
```

Amazon ECS 任務放置限制範例

下列是任務置放條件限制的範例。

此範例使用memberOf條件約束將工作放置在 t2 執行個體上。您可以使用下列動作來指定：CreateServiceUpdateService、、定RegisterTask義和RunTask。

```
"placementConstraints": [
    {
        "expression": "attribute:ecs.instance-type =~ t2.*",
        "type": "memberOf"
    }
]
```

範例使用 memberOf 限制條件以便將複本任務與常駐程式服務 daemon-service 任務群組中的任務放置到執行個體上，還要顧及其他被指定的任何任務置放策略。此限制可確保常駐程式服務任務在執行複本服務任務之前放置在 EC2 執行個體上。

以常駐程式服務的名稱取代 daemon-service。

```
"placementConstraints": [
    {
        "expression": "task:group == service:daemon-service",
        "type": "memberOf"
    }
]
```

範例使用 memberOf 限制條件以便將任務與 databases 任務群組中的其他任務放置到執行個體上，還要顧及其他被指定的任何任務置放策略。如需有關任務群組的詳細資訊，請參閱 群組相關的 Amazon ECS 任務 。您可以使用下列動作來指定：CreateServiceUpdateService、、定RegisterTask義和RunTask。

```
"placementConstraints": [
```

```
    {
        "expression": "task:group == databases",
        "type": "memberOf"
    }
]
```

distinctInstance 條件限制會在不同執行個體的群組中放置每個任務。您可以使用下列動作來指定：CreateServiceUpdateService、和 RunTask

```
"placementConstraints": [
    {
        "type": "distinctInstance"
    }
]
```

Amazon ECS 獨立任務

當您擁有執行某些工作的應用程式，然後停止 (例如批次處理程序) 時，您可以將應用程式當做工作執行。如果您想要執行一次工作，可以使用主控台 AWS CLI、API 或 SDK。

如果您需要以速率為基礎、cron 型或一次性排程執行應用程式，您可以使用「排程器」建立排程。EventBridge

工作流程

當您啟動 Amazon ECS 任務 (獨立任務或 Amazon ECS 服務) 時，會建立任務並最初移至該狀態PROVISIONING。當任務處於PROVISIONING狀態時，任務和容器都不存在，因為 Amazon ECS 需要尋找放置任務的運算容量。

Amazon ECS 會根據您的啟動類型或容量供應商組態，為您的任務選擇適當的運算容量。您可以將容量提供者和容量提供者策略與 Fargate 和 Amazon EC2 啟動類型一起使用。使用 Fargate，您不必考慮叢集容量的佈建、設定和擴展。Fargate 會為您的任務負責所有基礎設施管理。對於 EC2 啟動類型，您可以透過將 Amazon EC2 執行個體註冊到叢集來管理叢集容量，也可以使用叢集 auto 擴展來簡化運算容量管理。叢集 auto 動擴展會處理動態擴展叢集容量，以便您可以專注於執行工作。Amazon ECS 會根據您在任務定義中指定的需求 (例如 CPU 和記憶體)，以及放置限制和策略來決定放置任務的位置。如需詳細資訊，請參閱Amazon ECS 如何在容器執行個體上放置任務。

如果您使用已啟用受管擴展的容量提供者，則由於缺乏運算容量而無法啟動的工作會移至PROVISIONING狀態，而非立即失敗。找到放置任務的容量後，Amazon ECS 會佈建必要的附件

(例如，awsvpc模式下任務的彈性網路界面 (ENI))。它會使用 Amazon ECS 容器代理程式提取您的容器映像，然後啟動您的容器。佈建完成並啟動相關容器之後，Amazon ECS 會將任務移至RUNNING狀態。如需有關工作狀態的資訊，請參閱Amazon ECS 任務生命週期。

優化 Amazon ECS 任務啟動時間

為了加快任務啟動速度，請考慮以下建議。

- 快取容器映像檔和 binpack 執行個體

  如果您使用 EC2 啟動類型，則可以將 Amazon ECS 容器代理程式提取行為設定為ECS_IMAGE_PULL_BEHAVIOR：prefer-cached如果沒有快取的影像，則會從遠端拉取影像。否則，將使用執行個體上的快取映像，關閉容器的自動影像清理功能，以確保不會移除快取的影像。如此可減少後續啟動的影像擷取時間。當您的容器執行個體具有較高的工作密度時，快取的效果會更大，您可以使用binpack放置策略進行設定。快取容器映像對於通常具有大量 (數十 GB) 容器映像大小的 Windows 型工作負載來說尤其有用。使用binpack放置策略時，您還可以考慮使用彈性網絡接口（ENI）中繼在每個容器實例上使用awsvpc網絡模式放置更多任務。ENI 中繼功能會增加您可以在awsvpc模式下執行的工作數量。例如，一個可能只支援同時執行 2 個工作的 c5.large 執行個體，最多可以透過 ENI 中繼執行 10 個工作。

- 選擇最佳網路模式

  雖然有許多awsvpc網路模式最理想的執行個體，但此網路模式本質上可能會增加任務啟動延遲，因為對於awsvpc模式中的每個任務，Amazon ECS 工作流程都需要叫用 Amazon EC2 API 來佈建和連接 ENI，這會為您的任務啟動增加數秒的額外負荷。相比之下，使用awsvpc網路模式的主要優點是每個工作都有一個安全群組來允許或拒絕流量。這意味著您可以更靈活地控制任務和服務之間的通信，以更細微的級別。如果部署速度是您的優先考量，您可以考慮使用bridge模式加快工作啟動速度。如需詳細資訊，請參閱 the section called “AWSVPC 網路模式”。

- 追蹤您的任務啟動生命週期，以尋找最佳化

  通常很難知道應用程序啟動所需的時間量。在應用程式啟動期間啟動容器映像、執行啟動指令碼和其他設定可能需要大量的時間。您可以使用 Task 中繼資料端點張貼指標，以追蹤應用程式啟動時間，從ContainerStartTime應用程式準備好提供流量為止。有了這些資料，您就可以瞭解應用程式如何促成總啟動時間，並找出可以減少不必要的應用程式特定額外負荷並最佳化容器映像的區域。如需詳細資訊，請參閱 優化 Amazon ECS 容量和可用性。

- 選擇最佳執行個體類型 (適用於 EC2 啟動類型)

  根據您在工作上設定的資源保留區 (例如 CPU、記憶體) 來選擇正確的執行個體類型。因此，調整例證大小時，您可以計算單一例證上可以放置多少任務。在 m5.large 執行個體中託管 4 個需要 0.5

vCPU 和 2GB 記憶體保留的工作 (支援 2 個 vCPU 和 8 GB 記憶體) 的簡單範例。此任務定義的保留項目會充分利用執行個體的資源。

以 Amazon ECS 任務的形式執行應用程式

您可以使用建立一次性程序的工作 AWS Management Console。

若要建立獨立工作 (AWS Management Console)

1. 開啟主控台，網址為 https://console.aws.amazon.com/ecs/v2。
2. Amazon ECS 主控台可讓您從叢集詳細資訊頁面或任務定義修訂版清單建立獨立任務。根據您選擇的資源頁面，使用下列步驟建立獨立作業。

啟動服務的資源	步驟
叢集詳細資訊頁面...	a. 在叢集頁面，選取您要在哪個叢集中建立服務。 b. 在 Tasks (任務) 標籤中，選擇 Run new task (執行新任務)。
任務定義修訂頁面...	a. 在「工作定義」頁面上，選擇工作定義族群以顯示該族群的修訂版本。 b. 選取您要使用的版本修訂。 c. 從「部署」功能表中，選擇「執行工作」。

3. (選擇性) 您可以在「運算組態 (進階)」區段中選擇工作的分配方式。您可以使用容量提供者策略或 Launch 類型。若要使用容量提供者策略，您必須在叢集層級設定容量提供者。如果您尚未將叢集設定為使用容量提供者，請改用啟動類型。

分發方法	步驟
容量提供者策略	a. 在 Compute options (運算選項) 區段中，選取

分發方法	步驟	
	Capacity provider strategy (容量提供者策略)。 b. 選擇策略： • 若要使用叢集的預設容量提供者策略，選擇使用叢集預設。 • 若您的叢集沒有預設容量提供者策略，或要使用自訂策略，選擇 Use custom (使用自訂)、Add capacity provider strategy (新增容量提供者策略) 並透過指定 Base (基礎)、Capacity provider (容量提供者) 和 Weight (權重) 來定義您的自訂容量提供者策略。 Note 若要在策略中使用容量提供者，容量提供者必須與叢集相關聯。	

分發方法	步驟
啟動類型	a. 在運算選項區段中，選取啟動類型。 b. 針對 啟動類型，選擇啟動類型。 c. (選用) 當指定 Fargate 啟動類型時，針對平台版本，指定要使用的平台版本。如果未指定平台版本，將使用 LATEST 平台版本。

4. 針對 Application type (應用程式類型)，選擇 Task (任務)。
5. 在任務定義中，選擇任務定義系列和修訂。

> **Important**
>
> 主控台會驗證選擇，以確保所選任務定義系列和修訂與定義的運算組態相容。

6. 針對 Desired tasks (所需任務)，輸入要啟動的任務數目。
7. 如果任務定義使用 awsvpc 網路模式，請展開聯網。使用下列步驟以指定自訂組態。
   a. 針對 VPC，選擇要使用的 VPC。
   b. 針對子網路，在 VPC 中選擇一個或多個子網路，而任務排程器在放置任務時會考慮該 VPC。

   > **Important**
   >
   > awsvpc 網路模式只支援私有子網路。任務不會接收公有 IP 地址。因此必須使用 NAT 閘道進行對外網際網路存取，且入站網際網路流量會透過負載平衡器進行路由。

   c. 在安全群組中，您可以選擇現有的安全群組，或建立新的安全群組。若要使用現有的安全群組，選擇該安全群組並移至下一個步驟。若要建立新的安全群組，請選擇 建立新安全群組。您必須指定安全群組名稱、描述，然後為該安全群組新增一條或更多傳入規則。
   d. 針對 Public IP (公有 IP)，選擇是否為任務的彈性網路界面 (ENI) 自動指派公有 IP 地址。

AWS Fargate 任務可以在公共子網中運行時分配一個公共 IP 地址，以便他們有一個路由到互聯網。如需詳細資訊，請參閱 AWS Fargate的 Amazon Elastic Container Service 使用者指南中的 Fargate 任務聯網。

8. 如果您的工作使用與部署時設定相容的資料磁碟區，您可以透過擴充磁碟區來設定磁碟區。

   磁碟區名稱和磁碟區類型是在建立工作定義修訂版時設定的，而且在執行獨立工作時無法變更。若要更新磁碟區名稱與類型，您必須建立新的作業定義修訂版本，並使用新的修訂版本來執行作業。

設定此磁碟區類型	執行此作業
Amazon EBS	a. 針對 EBS 磁碟區類型，選擇要附加至工作的 EBS 磁碟區類型。 b. 在大小 (GiB) 中，輸入磁碟區大小的有效值 (以 GiB) 為單位。您可以指定最小 1 GiB 和最大 16,384 GiB 磁碟區大小。除非您提供快照 ID，否則此值是必要的。 c. 若為 IOPS，請輸入磁碟區應提供的最大輸入/輸出作業數 (IOPS)。此值僅適用於`io1`、`io2`、和`gp3`磁碟區類型。 d. 對於輸送量 (MIB/s)，輸入磁碟區應提供的輸送量，以 MB/秒為單位 (MiBps 或 MIB/s)。此值僅適用於`gp3`磁碟區類型。 e. 對於快照 ID，請選擇現有的 Amazon EBS 磁碟區快照，或者如果要從快照建立磁碟區，請輸入快照的 ARN。您也可以不選擇或

設定此磁碟區類型	執行此作業	
	輸入快照 ID 來建立新的空白磁碟區。 f. 針對「終止」原則，如果您希望在工作終止後保留為工作附件所設定的磁碟區，請取消選取核取方塊。依預設，工作終止時，會刪除附加至工作的 EBS 磁碟區。 g. 在 [檔案系統類型] 中，選擇磁碟區上資料儲存和擷取要使用的檔案系統類型。您可以選擇作業系統預設值或特定檔案系統類型。Linux 的預設值為XFS。對於從快照建立的磁碟區，您必須指定與建立快照時磁碟區所使用的相同檔案系統類型。如果文件系統類型不匹配，則任務將無法啟動。 h. 對於基礎設施角色，請選擇具有必要許可的 IAM 角色，以允許 Amazon ECS 管理任務的 Amazon EBS 磁碟區。您可以將AmazonECS Infrastru ctureRole PolicyForVolumes 受管理的原則附加至角色，也可以使用原則做為指南，建立並附加具有符合您特定需求的權限的	

設定此磁碟區類型	執行此作業	
	自己原則。如需必要權限的詳細資訊，請參閱〈〉Amazon ECS 基礎設施 IAM 角色。 i. 對於加密，如果您要使用 Amazon EBS 加密的預設設定，請選擇預設。如果您的帳戶預設設定了加密，則磁碟區將使用設定中指定的 AWS Key Management Service (AWS KMS) 金鑰加密。如果您選擇預設且 Amazon EBS 預設加密並未開啟，則磁碟區將不加密。 如果您選擇 [自訂]，您可以指定您選擇 AWS KMS key 的磁碟區加密。 如果您選擇 [無]，除非您預設已設定加密，或是從加密快照建立磁碟區，否則磁碟區將不加密。 j. 如果您已選擇「加密自訂」，則必須指 AWS KMS key 定要使用的項目。對於 KMS 金鑰，請選擇 AWS KMS key 或輸入金鑰 ARN。如果您選擇使用對稱的客戶管理金鑰來加密磁碟區，請確定您擁有在 AWS KMS key 政策中定義的正確權限。如需詳細資訊，請參閱	

設定此磁碟區類型	執行此作業	
	Amazon EBS 磁碟區的資料加密。 k. (選擇性) 在標籤下，您可以透過從任務定義傳播標籤或提供您自己的標籤，將標籤新增至 Amazon EBS 磁碟區。 如果您要從工作定義傳輸標籤，請選擇「工作定義」做為「傳輸標籤來源」。如果您選擇不傳播，或者如果您沒有選擇值，則不會傳播標籤。 如果您想要提供自己的標籤，請選擇 [新增標籤]，然後為您新增的每個標籤提供索引鍵和值。 如需標記 Amazon EBS 磁碟區的詳細資訊，請參閱標記 Amazon EBS 磁碟區。	

9. (選用) 若要使用預設以外的任務置放策略，請展開任務置放，然後從下列選項中選擇。

   如需詳細資訊，請參閱 Amazon ECS 如何在容器執行個體上放置任務。

   - AZ 平衡分攤 — 跨可用區域和可用區域中的容器執行個體分配任務。
   - AZ 平衡 BinPack — 在可用記憶體最少的可用性區域和容器執行個體之間分配工作。
   - BinPack— 根據最少可用的 CPU 或記憶體量來分配任務。
   - 每個主機一個工作 — 在每個容器執行個體上放置服務中最多一個工作。
   - 自訂 — 定義您自己的任務放置策略。

   如果選擇自訂，則請定義置放任務的演算法，以及在任務置放期間考慮的規則。

- 在策略下，針對類型和欄位，選擇演算法以及要用於演算法的實體。

  您最多可新增 5 項策略。

- 在 Constraint (限制條件)，針對 Type (類型) 和 Expression (運算式)，選擇要用於限制條件的規則與屬性。

  例如，若要設定限制條件，以將任務置放在 T2 執行個體上，針對運算式，請輸入 attribute:ecs.instance-type =~ t2.*。

  您最多可新增 10 個限制條件。

10. (選用) 若要覆寫任務 IAM 角色或任務定義中定義的任務執行角色，請展開 Task overrides (任務覆寫)，然後完成下列步驟：

    a. 在任務角色中，為此任務選擇 IAM 角色。如需詳細資訊，請參閱 Amazon ECS 任務 IAM 角色。

       僅顯示具有 `ecs-tasks.amazonaws.com` 信任關係的角色。如需為您的任務建立 IAM 角色的說明，請參閱 建立任務 IAM 角色。

    b. 在任務執行角色中，選擇任務執行角色。如需詳細資訊，請參閱 Amazon ECS 任務執行 IAM 角色。

11. (選用) 若要覆寫容器命令和環境變數，請展開 Container Overrides (容器覆寫)，然後展開容器。

    - 若要將命令傳送至任務定義命令以外的容器，針對 Command override (命令複寫)，請輸入 Docker 命令。

      如需 Docker 執行命令的詳細資訊，請參閱《Docker 參考手冊》中的 Docker 執行參考資料。

    - 若要新增環境變數，選擇 Add Environment Variable (新增環境變數)。針對 Key (索引鍵)，輸入您的環境變數的名稱。針對 Value (值)，為您的環境值輸入一個字串值 (未使用雙引號 (" ") 括住)。

      AWS 以雙引號 (」「) 圍住字串，並以下列格式將字串傳遞至容器：

      ```
      MY_ENV_VAR="This variable contains a string."
      ```

12. (選用) 為協助識別您的任務，請展開 Tags (標籤) 區段，然後設定標籤。

    若要讓 Amazon ECS 使用叢集名稱和任務定義標籤，自動標記所有新啟動的任務，請選取開啟 Amazon ECS 受管標籤，然後選取任務定義。

新增或移除標籤。

- [新增標籤] 選擇新增標籤，然後執行下列操作：
  - 在索引鍵中，輸入索引鍵名稱。
  - 在值中，進入索引鍵值。
- [移除標籤] 在標籤旁邊，選擇移除標籤。

13. 選擇建立。

使用 Amazon EventBridge 排程器來安排 Amazon ECS 任務

EventBridge Scheduler 是無伺服器排程器，可讓您從單一中央受管理的服務建立、執行及管理工作。它提供與事件匯流排和規則無關的一次性和週期性排程功能。 EventBridge Scheduler 具有高度可自訂性，並提供比 EventBridge 排程規則更高的延展性，並提供更廣泛的目標 API 作業和 AWS 服務。EventBridge 排程器提供下列排程，您可以在 [EventBridge 排程器] 主控台中為工作設定這些排程：

- 速率型
- Cron 型

  您可以在任何時區中設定 Cron 型排程。
- 一次性排程

  您可以在任何時區設定一次性排程。

您可以使用 Amazon 調度程序安 EventBridge 排您的 Amazon ECS。

雖然您可以在 Amazon ECS 主控台中建立排程任務，但 EventBridge 排程器主控台目前提供了更多功能。

在排程任務之前，請完成以下步驟：

1. 使用 VPC 主控台取得執行任務所在的子網路 ID 以及子網路的安全群組 ID。如需詳細資訊，請參閱《Amazon VPC 使用者指南》中的檢視子網路和檢視您的安全群組。
2. 設定 EventBridge 排程器執行角色。如需詳細資訊，請參閱 Amazon EventBridge 排程器使用者指南中的設定執行角色。

使用主控台建立新排程

1. 在 https://console.aws.amazon.com/scheduler/home 打開 Amazon EventBridge 調度程序控制台。
2. 在排程頁面上，選擇建立排程。
3. 在指定排程詳細資訊頁面的排程名稱和描述區段中，執行以下動作：

   a. 在排程名稱中，輸入排程的名稱，例如 **`MyTestSchedule`**。

   b. (選用) 在描述中，輸入對排程的描述，例如 **`TestSchedule`**。

   c. 在「排程」群組中，選擇排程群組。如果您沒有群組，請選擇預設值。若要建立排程群組，請選擇建立自己的排程。

      您可以使用排程群組，為不同群組的排程加上標籤。

4. 選擇排程選項。

頻率	執行此作業...
一次性排程 一次性排程只會在您指定的日期與時間調用目標一次。	針對日期和時間執行以下動作： • 依 YYYY/MM/DD 格式輸入有效日期。 • 依 hh:mm 格式輸入時間戳記 (24 小時)。 • 針對時區選擇時區。
週期性排程 週期性排程會依您指定的頻率，使用 cron 或 Rate 運算式調用目標。	a. 在排程模式中，執行下列其中一項動作： • 若要使用 Cron 運算式定義排程，請選擇 Cron 排程，然後輸入 Cron 運算式。 • 若要使用 Rate 表達式定義排程，請選擇 Rate 排程，然後輸入 Rate 表達式。

頻率	執行此作業...	
	如需 Cron 和費率運算式的詳細資訊，請參閱 Amazon 排程器使用者指南中的 EventBridge 排程器上的 EventBridge 排程類型。 b. 對於彈性時段，選擇關閉可關閉此選項，或者也能選擇其中一個預先定義的時間範圍。例如，如果您選擇 15 分鐘並設定週期性排程，每小時調用目標一次，則排程會在每小時一開始的 15 分鐘內執行。	

5. (選用) 如果您在上一個步驟中選擇週期性排程，請在時間範圍區段執行以下動作：
   a. 針對時區選擇時區。
   b. 對於開始日期和時間，依 YYYY/MM/DD 格式輸入有效日期，接著依 24 小時的 hh:mm 格式指定時間戳記。
   c. 對於結束日期和時間，依 YYYY/MM/DD 格式輸入有效日期，接著依 24 小時的 hh:mm 格式指定時間戳記。
6. 選擇下一步。
7. 在選取目標頁面上，執行下列動作：
   a. 選擇所有 API，然後在搜尋方塊中輸入 ECS。
   b. 選取 Amazon ECS。
   c. 在搜尋方塊中輸入 RunTask，然後選擇RunTask。
   d. 在 ECS 叢集中，選擇叢集。
   e. 在 ECS 任務中，選擇要用於任務的任務定義。
   f. 若要使用啟動類型，請展開運算選項，然後選取啟動類型。然後，選擇啟動類型。

      當指定 Fargate 啟動類型時，在平台版本中，輸入要使用的平台版本。若沒有指定平台，則會使用 LATEST 平台版本。

g. 在子網路中，輸入要在其中執行任務的子網路 ID。

h. 在安全群組中，輸入子網路的安全群組 ID。

i. (選用) 若要使用預設以外的任務置放策略，請展開置放限制條件，然後輸入限制條件。

如需詳細資訊，請參閱 Amazon ECS 如何在容器執行個體上放置任務。

j. (選用) 為協助識別您的任務，請在標籤下設定標籤。

若要讓 Amazon ECS 使用任務定義標籤，自動標記所有新啟動的任務，請選取啟用 Amazon ECS 受管標籤。

8. 選擇下一步。

9. 在設定頁面執行以下動作：

a. 若要開啟排程，請在排程狀態底下切換到啟用排程。

b. 若要設定排程的重試政策，請在重試政策和無效字母佇列 (DLQ) 底下執行以下動作：

- 切換到重試。
- 對於事件的保留時間上限，請輸入 EventBridge 排程器必須保留未處理事件的最大小時數和最小時數。
- 時間最長可設為 24 小時。
- 針對重試次數上限，輸入目標傳回錯誤時， EventBridge 排程器重試排程的次數上限。

最大值為重試 185 次。

使用重試原則時，如果排程無法呼叫其目標， EventBridge 排程器會重新執行排程。一旦設定此功能，您就必須設定排程的最長保留時間和重試次數。

c. 選擇 EventBridge 排程器儲存未傳遞事件的位置。

無效字母佇列 (DLQ) 選項	執行此作業...
不儲存	選擇無。
將活動儲存在建立排程的相同 AWS 帳戶 位置	a. 選擇選取我中的一個 Amazon SQS 佇列 AWS 帳戶 作為 DL Q。

無效字母佇列 (DLQ) 選項	執行此作業...	
	b. 選擇 Amazon SQS 佇列的 Amazon Resource Name (ARN)。	
將活動儲存在與建立排程不同 AWS 帳戶 的位置	a. 選擇 「AWS 帳戶 將其他地方的 Amazon SQS 佇列指定為 DL Q」。 b. 輸入 Amazon SQS 佇列的 Amazon Resource Name (ARN)。	

d. 若要使用由客戶管理的金鑰加密您的目標輸入，請在加密底下選擇自訂加密設定 (進階)。

   如果選擇此選項，請輸入現有的 KMS 金鑰 ARN，或選擇建立 AWS KMS key，以導覽至 AWS KMS 控制台。如需 EventBridge 排程器如何加密靜態資料的詳細資訊，請參閱 Amazon EventBridge 排程器使用者指南中的靜態加密。

e. 對於許可，請選擇使用現有角色，然後選取角色。

   若要讓 EventBridge 排程器為您建立新的執行角色，請選擇 [為此排程建立新角色]。接著輸入角色名稱。如果您選擇此選項， EventBridge Scheduler 會將範本化目標所需的必要權限附加至角色。

10. 選擇下一步。
11. 在檢閱和建立排程頁面上，檢閱排程的詳細資訊。在每個區段中選擇編輯，即可返回該步驟並編輯其詳細資訊。
12. 選擇建立排程。

    您可以在排程頁面檢視新建立和現有的排程。在狀態欄底下，確認您的新排程狀態為已啟用。

後續步驟

您可以使用 EventBridge 排程器主控台或管理排程。 AWS CLI 如需詳細資訊，請參閱 Amazon EventBridge 排程器使用者指南中的管理排程。

停止 Amazon ECS 任務

如果您不再需要讓獨立工作保持執行，您可以停止工作。Amazon ECS 主控台可讓您輕鬆停止一或多個任務。

如果您要停止服務，請參閱使用主控台刪除 Amazon ECS 服務。

若要停止獨立工作 (AWS Management Console)

1. 開啟主控台，網址為 https://console.aws.amazon.com/ecs/v2。
2. 在導覽窗格中，選擇叢集。
3. 在「叢集」頁面上，選擇要導覽至叢集詳細資訊頁面的叢集。
4. 在叢集詳細資訊頁面上，選擇 [工作] 索引標籤。
5. 您可以使用 [篩選器啟動類型] 清單，依啟動類型篩選工作。

要停止的工作	步驟
一個或多個	a. 選取工作，然後選擇停止，停止已選取。 b. 在 [停止工作確認] 頁面上，選擇 [停止]
全部	**Important** 如果您選擇使用主控台停止所有任務，Amazon ECS 會停止屬於服務一部分的所有獨立任務和任務。因此，我們建議您在使用此選項時小心。 a. 選擇停止 > 全部停止。

要停止的工作	步驟	
	b. 在 [停止工作確認] 頁面上，輸入 [停止所有工作]，然後選擇 [停止]。	

Amazon ECS 服務

您可以利用 Amazon ECS 服務在 Amazon ECS 叢集中同時執行並維持指定數目的任務定義執行個體。如果您有任務產生故障或停止，Amazon ECS 服務排程器就會根據您的任務定義啟動另一個執行個體來取而代之。這有助於維持服務中所需的任務數量。

您也可以在負載平衡器後方選擇性地執行您的服務。負載平衡器可將流量分散到與服務關聯的任務。

我們建議將服務排程器用於長時間執行的無狀態服務及應用程式。服務排程器可確保系統遵循您指定的排程策略，並在任務失敗時重新排程任務。例如，若基本的基礎設施發生故障，服務排程器會重新排程任務。您可以利用任務置放策略和限制條件來自訂排程器如何放置和終止任務。若服務中的任務停止，排程器會啟動新任務進行取代。此流程會根據您服務使用的排程策略持續進行，直至服務執行的任務達到您所需的數量為止。服務的排程策略也叫做 service type (服務類型)。

在容器運作狀態檢查或負載平衡器目標群組運作狀態檢查失敗後，服務排程器也會取代判定為狀況不佳的任務。此取代取決於 `maximumPercent` 和 `desiredCount` 服務定義參數。如果任務標記為運作狀態不佳，服務排程器會先啟動替代任務。然後，會發生以下情況。

- 如果取代工作的健全狀況狀態為HEALTHY，服務排程器會停止運作狀況不良的工作
- 如果替代任務的運作狀態為 UNHEALTHY，排程器會停止運作狀態不佳的替代任務或現有運作狀態不佳的任務，以使任務總計數等於 `desiredCount`。

如果 `maximumPercent` 參數限制排程器先啟動替代任務，排程器會隨機停止運作狀態不佳的任務，以釋放容量，然後啟動替代任務。開始和停止程序會繼續進行，直到所有運作狀態不佳的任務都會取代為運作狀態良好的 一旦已取代了所有運作狀態不佳的任務，而且只執行運作狀態良好的任務，如果任務總數超過 `desiredCount`，則運作良好的任務會隨機停止，直到任務總計數等於 `desiredCount` 為止。如需有關 `maximumPercent` 和 `desiredCount` 的詳細資訊，請參閱服務定義參數。

服務排程器包含的邏輯，可在任務重複啟動失敗的情況下，針對任務重新啟動的頻率進行調節。如果任務在未進入 RUNNING 的狀態下停止，服務排程器會減少啟動嘗試並傳送出服務事件訊息。此行為可以讓您在問題解決前，避免失敗的任務占用不必要的資源。在服務更新之後，服務排程器就會恢復正常的

排程行為。如需更多詳細資訊，請參閱 Amazon ECS 服務節流邏輯 和 檢視 Amazon ECS 服務事件訊息。

現已推出兩個服務排程器策略概念：

- REPLICA - 複本排程策略會置放並在整個叢集中維持所需的任務數量。根據預設，服務排程器會將任務分散至各個可用區域。您可以使用任務置放策略和限制條件，來自訂任務置放決策。如需詳細資訊，請參閱 複本策略。
- DAEMON - 常駐程式排程策略會在每個符合您於叢集中所指定所有任務置放限制條件的作用中容器執行個體上，準確地部署一個任務。使用這項策略時，不需指定所需的任務數量、任務置放策略或使用 Service Auto Scaling 政策。如需詳細資訊，請參閱 守護進程策。

> **Note**
>
> Fargate 任務不支援 DAEMON 排程策略。

守護進程策

常駐程式排程策略會在每個活動容器執行個體上準確部署一個任務，其可滿足叢集中所有指定的任務置放限制條件。服務排程器會評估執行中工作的工作放置限制，並停止不符合放置限制的工作。使用此策略時，您不需要指定所需的工作數目、工作放置策略或使用 Service Auto Scaling 原則。

Amazon ECS 會為常駐程式任務保留容器執行個體運算資源，包括 CPU、記憶體和網路介面。當您在具有其他複本服務的叢集上啟動常駐程式服務時，Amazon ECS 會將常駐程式任務設為最優先。這表示協助程式工作是第一個在執行個體上啟動的工作，也是停止所有複本工作之後停止的最後一個工作。此策略可確保資源不會被擱置的複本任務使用，且可供常駐程式任務使用。

常駐程式服務排程器不會將任何任務置放於具有 DRAINING 狀態的執行個體上。如果容器執行個體轉換為 DRAINING 狀態，常駐程式會停止運作。新的容器執行個體加入叢集時，服務排程器也會加以監控，並將常駐程式任務新增到這些執行個體。

當您指定部署規劃時，maximumPercent參數的值必須是 100 (以百分比指定)，如果未設定，則使用的預設值。minimumHealthyPercent參數的預設值為 0 (以百分比指定)。

當您變更常駐程式服務的置放限制條件時，您必須重新啟動服務。Amazon ECS 會為常駐程式任務動態更新合格執行個體上保留的資源。對於現有執行個體，排程器會嘗試將任務放置在執行個體上。

當任務定義中的任務大小或容器資源保留改變時，會啟動一個新的部署。Amazon ECS 會為常駐程式挑選更新的 CPU 和記憶體保留，然後為常駐程式任務封鎖容量。

若上述任一情境的資源不足，將發生下列情況：

- 任務置放失敗。
- 會產生一個 CloudWatch 事件。
- Amazon ECS 透過等待資源轉為可用以繼續嘗試與排程執行個體上的任務。
- Amazon ECS 會釋放任何不再符合置放限制標準的預留執行個體，並且停止對應的常駐程式任務。

常駐程式排程策略可被用於下列情況：

- 執行應用程式容器
- 執行支援容器以記錄、監控和追蹤任務

使用 Fargate 啟動類型或 CODE_DEPLOY 或 EXTERNAL 部署控制器類型的任務，不支援常駐程式排程策略。

當服務排程器停止執行中的任務時，會嘗試維護叢集中之可用區域間的平衡。排程器使用以下邏輯：

- 如已定義置放策略，請使用該策略選取要終止的任務。例如，如果服務已定義可用區域分佈策略，則會選取能讓剩餘任務完美分佈的任務。
- 如未定義任何置放策略，請使用下列邏輯維護叢集中之可用區域的平衡：
  - 排序有效的容器執行個體。優先考慮在此服務的個別可用區域中，擁有最多執行中任務數量的執行個體。例如，如果區域 A 有一項執行中的服務任務，而區域 B 和 C 各有兩項執行中的服務任務，則最適合終止的即為區域 B 或 C 中的容器執行個體。
  - 根據先前的步驟，停止最佳可用區域中容器執行個體上的任務。偏好此服務執行中任務數量最多的容器執行個體。

複本策略

複本排程策略會在叢集中放置並維持所需的任務數量。

對於在 Fargate 上執行任務的服務，當服務排程器啟動新任務或停止執行任務時，服務排程器會盡力嘗試在可用區域之間維持平衡。您無須指定任務置放策略或限制條件。

當您在 EC2 執行個體上建立執行任務的服務時，您可以選擇性地指定任務置放策略和限制條件，以自訂任務置放決策。如果未指定任務置放策略或限制，預設情況下，服務排程器會將任務分散到可用區域。服務排程器使用以下邏輯：

- 決定您叢集中的哪些容器執行個體可支援服務的任務定義 (例如，必要的 CPU、記憶體、連接埠和容器執行個體屬性)。
- 決定哪些容器執行個體可滿足針對服務所定義的任何置放限制。
- 當您有取決於常駐程式服務的複本服務時 (例如，必須先執行常駐程式日誌路由器任務，才能使用記錄)，請建立任務置放限制，以確保常駐程式服務任務在複本服務任務之前置放在 EC2 執行個體上。如需詳細資訊，請參閱 Amazon ECS 任務放置限制範例。
- 如已定義置放策略，請使用該策略從剩餘的待選項目中選取執行個體。
- 如未定義任何置放策略，請使用下列邏輯平衡您叢集中可用區域的任務：
  - 排序有效的容器執行個體。優先考慮在此服務的個別可用區域中，執行中任務數量最少的執行個體。例如，如果區域 A 有一項執行中的服務任務，而區域 B 和 C 都沒有，則最適合置放的即為區域 B 或 C 中的有效容器執行個體。
  - 根據先前的步驟，將新的服務任務置放在最佳可用區域的有效容器執行個體。偏好此服務執行中任務數量最少的容器執行個體。

Amazon ECS 服務參數的最佳實務

為了確保沒有應用程式停機時間，部署程序如下：

1. 啟動新的應用程式容器，同時保持現有容器的執行狀態。
2. 檢查新容器是否健康。
3. 停止舊的容器。

根據您的部署配置以及叢集中未保留的可用空間量，可能需要多輪次才能完成以新任務取代所有舊任務。

您可以使用兩個 ECS 服務組態選項來修改編號：

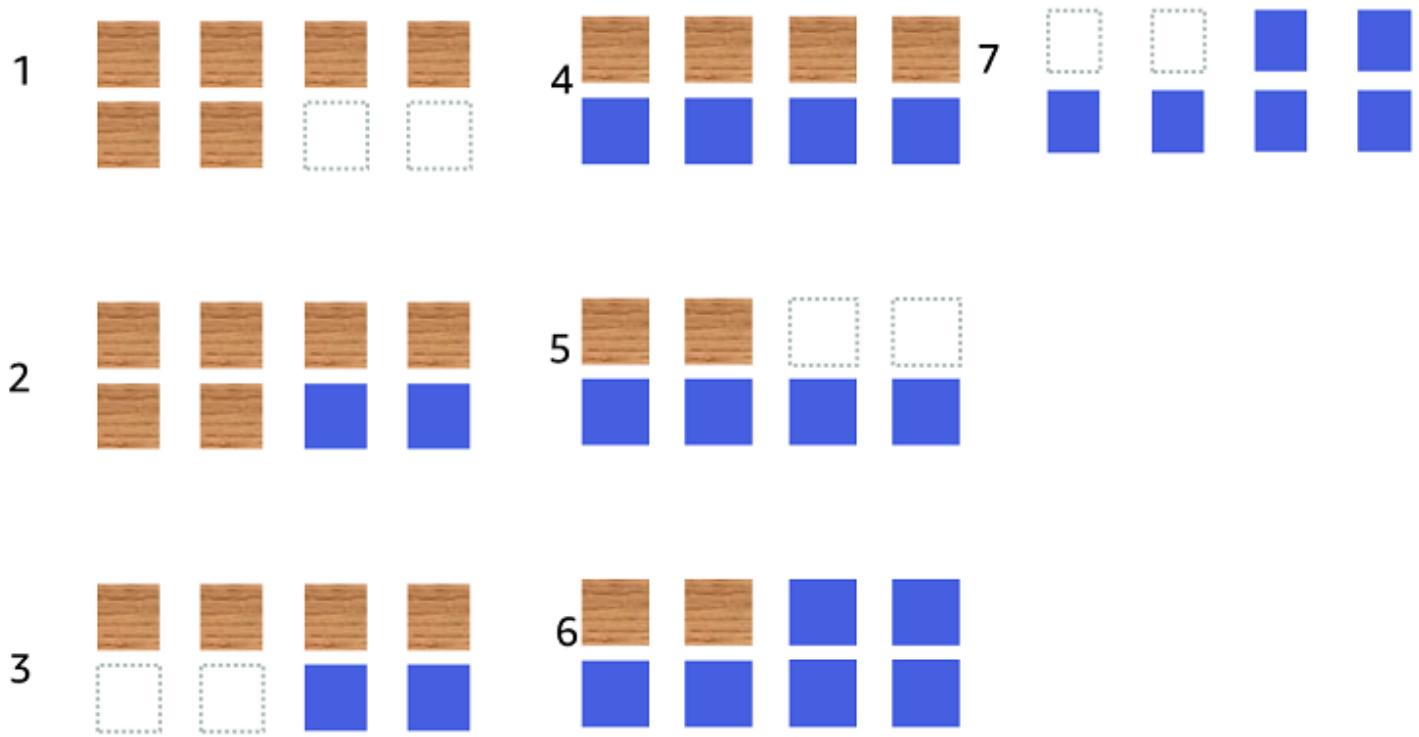
- `minimumHealthyPercent`：100% (預設值)

  部署期間必須保持在RUNNING狀態下的服務工作數目下限。這是desiredCount四捨五入到最接近整數的百分比。此參數可讓您在不使用額外叢集容量的情況下進行部署。

- `maximumPercent`：百分之二十 (預設值)

  部署期間，RUNNING或PENDING狀態允許的服務工作數目上限。這是desiredCount四捨五入到最接近的整數的百分比。

請考慮下列有六個 tan 工作的服務，部署在叢集中，總共有八個工作的空間。預設的 Amazon ECS 服務組態選項不允許部署低於六個所需任務的 100%。



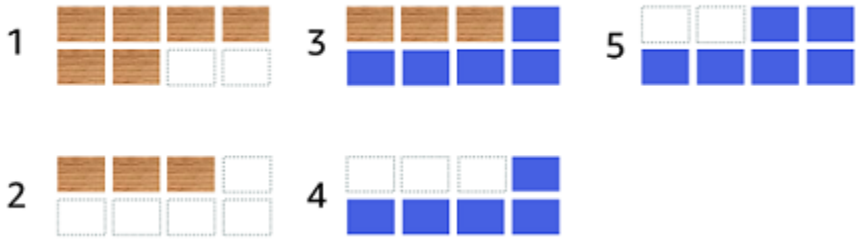
部署程序如下：

1. 目標是用藍色任務替換棕褐色任務。
2. 排程器會啟動兩個新的藍色工作，因為預設設定要求有六個執行中的工作。
3. 排程器會停止其中兩個 tan 工作，因為總共會有六個工作 (四個 tan 和兩個藍色)。
4. 排程器會啟動兩個額外的藍色工作。
5. 排程器會關閉兩個 tan 工作。
6. 排程器會啟動兩個額外的藍色工作。
7. 排程器會關閉最後兩個 tan 工作。

在上述範例中，如果您使用選項的預設值，則每個啟動的新工作都需要等待 2.5 分鐘。此外，負載平衡器可能需要等待 5 分鐘才能停止舊工作。

您可以將`minimumHealthyPercent`值設定為 50% 來加速部署。

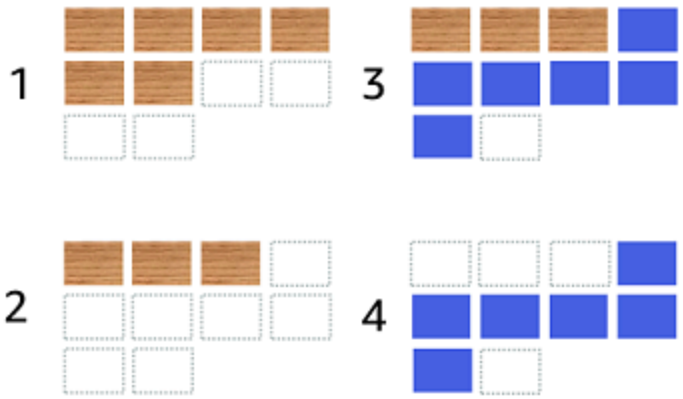
請考慮下列有六個 tan 工作的服務，部署在叢集中，總共有八個工作的空間。



部署程序如下：

1. 目標是用藍色任務替換棕褐色任務。
2. 排程器會停止三個 TAN 工作。仍然有三個棕褐色任務正在運行，這些任務符合該`minimumHealthyPercent`值。
3. 排程器會啟動五個藍色工作。
4. 排程器會停止剩餘的三個 TAN 工作。
5. 排程器會啟動最後的藍色工作。

您也可以新增額外的可用空間，以便執行其他工作。



部署程序如下：

1. 目標是用藍色任務替換棕褐色任務。
2. 排程器會停止三個棕褐色工作
3. 調度程序啟動六個藍色任務
4. 排程器會停止三個 TAN 工作。

當您的任務閒置一段時間且沒有高使用率時，請使用以下 Amazon ECS 服務組態選項的值。

- `minimumHealthyPercent`：50%
- `maximumPercent`: 二百分之二十

使用主控台建立 Amazon ECS 服務

您可以使用主控台建立服務。

使用主控台時應考慮以下項目：

- 分發任務有兩種運算選項。
  - capacity provider strategy (容量供應商策略)，使 Amazon ECS 將您的任務分發至一個或多個容量供應商。
  - launch type (啟動類型)，使 Amazon ECS 直接在 Fargate 或註冊到叢集的 Amazon EC2 執行個體上啟動我們的任務。
- 使用 `awsvpc` 網路模式的任務定義或被設定使用負載平衡器的服務必須有聯網組態。根據預設，主控台會在預設的 Amazon VPC 中選擇預設的 Amazon VPC，以及全部子網路和預設安全群組。
- 預設任務放置策略會將工作平均分配到可用區域。
- 當您使用 Launch Type (啟動類型) 進行服務部署時，服務預設會在叢集 VPC 的子網路中啟動。
- 針對 capacity provider strategy (容量提供者策略)，主控台預設會選取運算選項。以下說明主控台用來選擇預設值的順序：
  - 若您的叢集定義了預設容量提供者策略，則會選取該叢集。
  - 如果您的叢集未定義預設容量提供者策略，但您已將 Fargate 容量提供者新增至叢集，則會選取使用容量提供者的自訂`FARGATE`容量提供者策略。
  - 如果您的叢集未定義預設容量提供者策略，但您已將一或多個 Auto Scaling 群組容量提供者新增至叢集，則會選取 [使用自訂 (進階)] 選項，您需要手動定義策略。
  - 若您的叢集未定義預設容量提供者策略，也沒有為叢集新增容量提供者，則會選擇 Fargate 啟動類型。

- 預設部署失敗偵測預設選項是使用 Amazon ECS 部署斷路器選項搭配「故障時復原」選項。

  如需詳細資訊，請參閱 Amazon ECS 部署斷路器如何偵測故障。
- 如果您要使用藍/綠部署選項，請決定 CodeDeploy 移動應用程式的方式。以下是可用的選項：
  - CodeDeploy預設 .ECS AllAt 一次：一次將所有流量轉移到更新後的 Amazon ECS 容器
  - CodeDeploy預設值 10 PercentEvery 1 分鐘：每分鐘移動 10% 的流量，直到所有流量轉移為止。
  - CodeDeploy預設值 10 PercentEvery 3 分鐘：每 3 分鐘移動 10% 的流量，直到所有流量轉移為止。
  - CodeDeploy預設。百分之十 5 分鐘：在第一個增量中移動 10% 的流量。剩餘的 90% 會在五分鐘之後部署。
  - CodeDeploy預設。百分之十 15 分鐘：在第一個增量中移動 10% 的流量。剩餘的 90% 會在 15 分鐘之後部署。
- 如果您需要應用程式連線到在 Amazon ECS 中執行的其他應用程式，請確定適合您架構的選項。如需詳細資訊，請參閱 互連 Amazon ECS 服務。
- 您必須使用 AWS CloudFormation 或 AWS Command Line Interface 來部署使用下列任一參數的服務：
  - 使用自訂指標追蹤政策
  - 更新服務 — 您無法更新`awsvpc`網路組態和健全狀況檢查寬限期。

  若要取得有關如何使用建立服務的資訊 AWS CLI，請參閱〈AWS Command Line Interface 參考〉create-service中的〈〉。

  若要取得有關如何使用建立服務的資訊 AWS CloudFormation，請參閱《使AWS CloudFormation 用指南》AWS::ECS::Service中的〈〉。

快速建立服務

您可以使用主控台來快速建立並部署服務。服務具有下列組態：

- 在與叢集關聯的 VPC 和子網路中部署
- 部署一項任務
- 使用滾動部署
- 搭配預設容量供應商使用容量供應商策略
- 使用部署斷路器偵測故障，並將選項設定為在失敗時自動回復部署

若要使用預設參數部署服務，請遵循下列步驟。

建立服務 (Amazon ECS 主控台)

1. 開啟主控台，網址為 https://console.aws.amazon.com/ecs/v2。
2. 在導覽頁面中，選擇叢集)。
3. 在 [叢集] 頁面上，選擇要在其中建立服務的叢集。
4. 在 Services (服務) 索引標籤上，選擇 Create (建立)。
5. 在 Deployment configuration (部署組態)，指定應用程式的部署方式。
   a. 針對 Application type (應用程式類型)，選擇 Service (服務)。
   b. 針對 Task definition (任務定義)，選擇要使用的任務定義系列和修訂。
   c. 針對 Service name (服務名稱)，輸入服務的名稱。
   d. 針對 Desired tasks (所需任務)，輸入要在服務中啟動並維護的任務數。
6. (選用) 為協助識別您的服務和任務，請展開 Tags (標籤) 區段，然後設定標籤。

   若要讓 Amazon ECS 使用叢集名稱和任務定義標籤，自動標記所有新啟動的任務，請選取 Turn on Amazon ECS managed tags (開啟 Amazon ECS 受管標籤)，然後選取 Task definitions (任務定義)。

   若要讓 Amazon ECS 使用叢集名稱和服務標籤，自動標記所有新啟動的任務，請選取 Turn on Amazon ECS managed tags (開啟 Amazon ECS 受管標籤)，然後選取 Service (服務)。

   新增或移除標籤。

   - [新增標籤] 選擇新增標籤，然後執行下列操作：
     - 在索引鍵中，輸入索引鍵名稱。
     - 在值中，進入索引鍵值。
   - [移除標籤] 在標籤旁邊，選擇 移除標籤。

使用定義的參數建立服務

若要使用已定義的參數來建立服務，請依照下列步驟執行。

建立服務 (Amazon ECS 主控台)

1. 開啟主控台，網址為 https://console.aws.amazon.com/ecs/v2。

2. 決定您要從中啟動服務的資源。

啟動服務的資源	步驟
叢集	a. 在叢集頁面，選取您要在哪個叢集中建立服務。 b. 在 Services (服務) 索引標籤上，選擇 Create (建立)。
啟動類型	a. 在 [作業定義] 頁面上，選取作業定義旁邊的選項按鈕。 b. 在 [部署] 功能表上，選擇 [建立服務]。

3. (選用) 選擇任務在叢集基礎設施中的分佈方式。展開 Compute configuration (運算組態)，然後選擇您的選項。

分發方法	步驟
容量提供者策略	a. 在 Compute options (運算選項) 下，選擇 Capacity provider strategy (容量提供者策略)。 b. 選擇策略： • 若要使用叢集的預設容量提供者策略，選擇使用叢集預設。 • 若您的叢集沒有預設容量提供者策略，或要使用自訂策略，選擇 Use custom (使用自訂)、Add capacity provider strategy (新增容量提供者策略)，然

分發方法	步驟
	後透過指定 Base (基礎)、Capacity provider (容量提供者) 和 Weight (權重) 來定義您的自訂容量提供者策略。 **Note** 若要在策略中使用容量提供者，容量提供者必須與叢集相關聯。
啟動類型	a. 在運算選項區段中，選取啟動類型。 b. 針對 啟動類型，選擇啟動類型。 c. (選用) 當指定 Fargate 啟動類型時，針對平台版本，指定要使用的平台版本。如果未指定平台版本，將使用 LATEST 平台版本。

4. 若要指定服務的部署方式，請移至 [部署設定] 區段，然後選擇您的選項。

   a. 在應用程式類型中，沿用服務選項。

   b. 針對 Task definition (任務定義) 和 Revision (修訂)，選擇要使用的任務定義系列和修訂。

   c. 針對 Service name (服務名稱)，輸入服務的名稱。

   d. 針對 Service type (服務類型)，選擇服務排程策略。

      - 若要讓排程器在每個活動容器執行個體上準確部署一個任務，且滿足所有任務置放限制條件，請選擇 Daemon (常駐程式)。
      - 若要讓排程器在叢集中置放並維持所需的任務數量，請選擇 Replica (複寫)。

e. 如果您選擇 Replica (複寫)，針對 Desired tasks (所需任務)，輸入要在服務中啟動並維護的任務數。

f. 決定服務的部署類型。展開 [部署選項]，然後指定下列參數。

部署類型	步驟
滾動更新	a. 針對 Min running tasks (執行中任務下限)，輸入部署期間必須維持在 RUNNING 狀態的服務任務數量下限，它是所需任務數量的百分比 (無條件進位到最接近的整數)。如需詳細資訊，請參閱部署組態。 b. 針對 Max running tasks (執行中任務上限)，輸入部署期間允許的處於 RUNNING 或 PENDING 狀態的服務任務數目上限，它是所需任務數量的百分比 (無條件捨去到最接近的整數)。
藍/綠部署	a. 對於部署組態，請選擇在部署期間將生產流量 CodeDeploy 路由至取代工作集的方式。 b. 對於的服務角色 CodeDeploy，請選擇服務用來向授權發出 API 請求的 IAM 角色 AWS 服務。

g. 若要設定 Amazon ECS 如何偵測並處理部署失敗，請展開 Deployment failure detection (部署失敗偵測)，然後選擇您的選項。

i. 若要在任務無法啟動時停止部署，請選取 Use the Amazon ECS deployment circuit breaker (使用 Amazon ECS 部署斷路器)。

若要讓軟體在部署斷路器將部署設定為失敗狀態時，自動將部署復原至上次完成的部署狀態，請選取「失敗時復原」。

ii. 若要停止以應用程式測量結果為基礎的部署，請選取使用 CloudWatch 警示。然後，從CloudWatch 警報名稱中選擇鬧鐘。若要建立新鬧鐘，請前往主 CloudWatch 控台。

若要讓軟體在 CloudWatch 警示將部署設定為失敗狀態時，自動將部署回復至上次完成的部署狀態，請選取失敗時復原。

5. (選用) 若要使用 Service Connect，請選取 Turn on Service Connect (開啟 Service Connect)，然後指定下列項目：

a. 在 Service Connect configuration (Service Connect 組態) 下，指定用戶端模式。

- 如果您的服務執行的網路用戶端應用程式只需要連線到命名空間中的其他服務，請選擇僅用戶端。
- 如果服務執行的是網路或 Web 服務應用程式，且需要為此服務提供端點，並連線至命名空間中的其他服務，請選擇 Client and server (用戶端和伺服器)。

b. 若要使用非預設叢集命名空間的命名空間，請在 Namespace (命名空間) 欄位中選擇服務命名空間。

c. (選用) 選擇使用日誌收集選項來指定日誌組態。每個可用的日誌驅動程序式都有要指定的日誌驅動程序選項。預設選項會將容器記錄檔傳送至 CloudWatch 記錄檔。其他記錄驅動程式選項是使用設定的 AWS FireLens。如需詳細資訊，請參閱 將 Amazon ECS 日誌傳送到 AWS 服務或 AWS Partner。

下方更詳細地描述了每個容器日誌目的地。

- Amazon CloudWatch — 設定任務以將容器日誌傳送到 CloudWatch 日誌。系統會提供預設的記錄驅動程式選項，以代表您建立 CloudWatch 記錄群組。若要指定不同的日誌群組名稱，請變更驅動程式選項值。
- Amazon 資料防火管 — 設定任務以將容器日誌傳送到 Firehose。提供預設的記錄驅動程式選項，可將記錄檔傳送至 Firehose 傳送串流。若要指定不同的交付串流名稱，請變更驅動程式選項值。

- Amazon Kinesis Data Streams — 設定任務以將容器日誌傳送到 Kinesis Data Streams。提供預設的記錄驅動程式選項，可將記錄檔傳送至 Kinesis Data Streams。若要指定不同的串流名稱，請變更驅動程式選項值。
- Amazon OpenSearch 服務 — 設定將容器日誌傳送到 OpenSearch 服務網域的任務。務必提供日誌驅動程式選項。
- Amazon S3 — 設定任務以將容器日誌傳送到 Amazon S3 儲存貯體。提供預設日誌驅動程式選項，但您必須指定有效的 Amazon S3 儲存貯體名稱。

6. (選擇性) 若要使用服務探索，請選取使用服務探索，然後指定下列項目。

   a. 若要使用新的命名空間，請選擇 [設定命名空間] 底下的 [建立新命名空間]，然後提供命名空間名稱和說明。若要使用現有的命名空間，請選擇 [選取現有的命名空間]，然後選擇您要使用的命名空間。

   b. 提供服務探索服務資訊，例如服務的名稱和描述。

   c. 若要讓 Amazon ECS 執行定期容器層級運作狀態檢查，請選取啟用 Amazon ECS 任務運作狀態傳播。

   d. 針對 DNS record type (DNS 紀錄類型)，選擇您要為服務建立的 DNS 紀錄類型。Amazon ECS 服務探索僅支援 A 和 SRV 記錄，視您的任務定義指定的網路模式而定。如需有關此類記錄類型的詳細資訊，請參閱 Amazon Route 53 開發人員指南中的支援的 DNS 紀錄類型。

      - 如果服務任務指定的任務定義使用的是 `bridge` 或 `host` 網路模式，僅支援類型 SRV 記錄。選擇容器名稱和連接埠組合以與該紀錄建立關聯。
      - 如果服務任務指定的任務定義使用的是 `awsvpc` 網路模式，選取 A 或 SRV 紀錄類型。如果您選擇 A，請跳至下一個步驟。如果選取 SRV，指定可找到該服務的連接埠，或是容器名稱和連接埠組合，以與該記錄建立關聯。

      針對 TTL，輸入 DNS 解析器和網頁瀏覽器快取記錄集的時間 (以秒為單位)。

7. (選用) 若要設定服務的負載平衡器，請展開 Load balancing (負載平衡) 區段。

   選擇負載平衡器。

使用此負載平衡器	執行此作業
Application Load Balancer	a. 針對 Load balancer type (負載平衡器類型)，選

使用此負載平衡器	執行此作業	
	擇 Application Load Balancer。 b. 選擇 Create a new load balancer (建立新的負載平衡器) 以建立新的 Application Load Balancer，或者 Use an existing load balancer (使用現有負載平衡器) 以選擇現有的 Application Load Balancer。 c. 針對 Load balancer name (負載平衡器名稱)，輸入不重複的名稱。 d. 針對 Choose container to load balance (選擇進行負載平衡的容器)，選擇託管服務的容器。 e. 針對 Listener (接聽程式)，為 Application Load Balancer 輸入連接埠和協定，以偵聽連接請求。根據預設，負載平衡器將被設定為使用連接埠 80 和 HTTP。 f. 針對 Target group name (目標群組名稱)，為目標群組輸入名稱和協定，Application Load Balancer 會將請求路由至該群組。根據預設，目標群組會將請求路由至您的任務定義中定義的第一個容器。	

使用此負載平衡器	執行此作業	
	g. 對於逐位延遲，請輸入負載平衡器將目標狀態變更為的秒數。UNUSED預設為 300 秒。 h. 針對 Health check path (運作狀態檢查路徑)，輸入容器中的現有路徑，Application Load Balancer 會在該容器中定期傳送請求，以驗證 Application Load Balancer 和容器之間的連接狀態。預設使用根目錄 (/)。 i. 針對 Health check grace period (運作狀態檢查寬限期)，輸入服務排程器應忽略狀態不良的 Elastic Load Balancing 目標運作狀態檢查的時長 (秒)。	

使用此負載平衡器	執行此作業	
Network Load Balancer	a. 針對 Load balancer type (負載平衡器類型)，選取 Network Load Balancer。 b. 針對 Load Balancer (負載平衡器)，選擇現有的 Network Load Balancer。 c. 針對 Choose container to load balance (選擇進行負載平衡的容器)，選擇託管服務的容器。 d. 針對 Target group name (目標群組名稱)，為目標群組輸入名稱和協定，Network Load Balancer 會將請求路由至該群組。根據預設，目標群組會將請求路由至您的任務定義中定義的第一個容器。 e. 對於逐位延遲，請輸入負載平衡器將目標狀態變更為的秒數。UNUSED預設為 300 秒。 f. 針對 Health check path (運作狀態檢查路徑)，輸入容器中的現有路徑，Network Load Balancer 會在該容器中定期傳送請求，以驗證 Application Load Balancer 和容器之間的連接狀態。預設使用根目錄 (/)。 g. 針對 Health check grace period (運作狀態檢查寬限	

使用此負載平衡器	執行此作業	
	期)，輸入服務排程器應忽略狀態不良的 Elastic Load Balancing 目標運作狀態檢查的時長 (秒)。	

8. (選擇性) 若要設定服務 Auto Scaling，請展開服務自動擴展，然後指定下列參數。

   a. 若要使用服務自動擴展，請選取 Service auto scaling (服務自動擴展)。

   b. 對於工作數目下限，請輸入要使用的服務 auto 擴展的工作數目下限。所需的計數不會低於此計數。

   c. 針對工作數目上限，輸入要使用的服務 auto 擴展的工作數目上限。所需的計數不會高於此計數。

   d. 選擇政策類型。在「資源調整」政策類型下，選擇下列其中一個選項。

若要使用此政策，請參閱。	執行此作業...	
目標追蹤	a. 針對 Scaling policy type (擴展政策類型)，選擇 Target tracking (目標追蹤)。 b. 針對 Policy name (政策名稱)，輸入政策的名稱。 c. 針對 ECS service metric (ECS 服務指標)，選擇下列其中一項指標。 • ECS ServiceAverage CPU 使用率 — 服務的平均 CPU 使用率。 • ECS ServiceAverage MemoryUtilization — 服務的平均記憶體使用率。	

若要使用此政策，請參閱。	執行此作業...	
	• ALB RequestCount PerTarget —「Application Load Balancer」目標群組中每個目標完成的要求數目。 d. 針對 Target value (目標值)，輸入服務為選取的指標保持的值。 e. 對於向外延展冷卻時間，請輸入在其他向外延展活動開始之前必須經過的向外延展活動 (新增任務) 之後的時間量 (以秒為單位)。 f. 對於縮放冷卻時間，請輸入在其他縮放活動開始之前必須經過的縮放活動（刪除任務）之後的時間量（以秒為單位）。 g. 若要防止政策執行縮減活動，請選取 Turn off scale-in (關閉縮減)。 h. • (選用) 如果您希望擴展政策擴展以增加流量，但不需要在流量減少時擴展，請選取關閉擴展政策。	

若要使用此政策，請參閱。	執行此作業...	
步驟擴展	a. 針對 Scaling policy type (擴展政策類型)，選擇 Step scaling (步驟擴展)。 b. 在 Policy Name (政策名稱) 輸入政策的名稱。 c. 針對 Alarm name (警示名稱)，輸入警示的唯一名稱。 d. 針對 Amazon ECS service metric (Amazon ECS 服務指標)，選擇用於警示的指標。 e. 針對 Statistic (統計資料)，選擇警示統計資料。 f. 針對 Period (期間)，選擇警示的期間。 g. 針對 Alarm condition (警示條件)，選擇如何比較選取的指標與定義的閾值。 h. 針對 Threshold to compare metrics (比較閾值與指標) 和 Evaluation period to initiate alarm (啟動警示的評估期)，輸入用於警示的閾值以及評估閾值的時間長度。 i. 在 Scaling actions (擴展動作) 下，執行下列動作：	

若要使用此政策，請參閱。	執行此作業...	
	• 在動作中，選取是否要新增、移除或設定服務的特定所需計數。 • 如果您選擇新增或移除工作，在「值」中，輸入啟動縮放動作時要新增或移除的工作數目 (或現有工作的百分比)。如果您選擇設定所需的計數，請輸入任務數量。對於 Type (類型)，選取 Value (值) 是整數或是現有所需計數的百分比值。 • 針對 Lower bound (下限) 和 Upper bound (上限)，輸入步驟擴展調整的下界限和上界限。根據預設，新增政策的下限為警示閾值，而上限為無限大正數 (+) 值。根據預設，移除政策的上限為警示閾值，而下限為無限小負數 (-) 值。 • (選用) 新增其他擴展選項。選擇「新增縮放」動作，然後重複「縮放」動作步驟。 • 對於冷卻時間，請輸入等待先前縮放活動生效的時間（以秒為單位）。對於新增政策，這是擴展活動之後的時	

若要使用此政策，請參閱。	執行此作業...	
	間，擴展政策會封鎖擴展活動，並限制一次可向外延展的工作數目。對於移除原則，這是在縮放活動之後必須經過的時間，才能開始另一個縮放活動。	

9. (選用) 若要使用預設以外的任務置放策略，請展開任務置放，然後從下列選項中選擇。

   如需詳細資訊，請參閱 Amazon ECS 如何在容器執行個體上放置任務。

   - AZ 平衡分攤 — 跨可用區域和可用區域中的容器執行個體分配任務。
   - AZ 平衡 BinPack — 在可用記憶體最少的可用性區域和容器執行個體之間分配工作。
   - BinPack— 根據最少可用的 CPU 或記憶體量來分配任務。
   - 每個主機一個工作 — 在每個容器執行個體上放置服務中最多一個工作。
   - 自訂 — 定義您自己的任務放置策略。

   如果選擇自訂，則請定義置放任務的演算法，以及在任務置放期間考慮的規則。

   - 在策略下，針對類型和欄位，選擇演算法以及要用於演算法的實體。

     您最多可新增 5 項策略。
   - 在 Constraint (限制條件)，針對 Type (類型) 和 Expression (運算式)，選擇要用於限制條件的規則與屬性。

     例如，若要設定限制條件，以將任務置放在 T2 執行個體上，針對運算式，請輸入 attribute:ecs.instance-type =~ t2.*。

     您最多可新增 10 個限制條件。

10. 如果任務定義使用 `awsvpc` 網路模式，請展開 Networking (聯網)。使用下列步驟以指定自訂組態。

    a. 針對 VPC，選擇要使用的 VPC。

    b. 針對子網路，在 VPC 中選擇一個或多個子網路，而任務排程器在放置任務時會考慮該 VPC。

> **Important**
>
> awsvpc 網路模式只支援私有子網路。任務不會接收公有 IP 地址。因此必須使用 NAT 閘道進行對外網際網路存取，且入站網際網路流量會透過負載平衡器進行路由。

c. 針對 Security groups (安全群組)，您可以選取現有的安全群組，或建立新的安全群組。若要使用現有的安全群組，選擇該安全群組並移至下一個步驟。若要建立新的安全群組，請選擇建立新安全群組。您必須指定安全群組名稱、描述，然後為該安全群組新增一條或更多傳入規則。

11. 如果您的工作使用與部署時設定相容的資料磁碟區，您可以透過擴充磁碟區來設定磁碟區。

    磁碟區名稱和磁碟區類型是在您建立工作定義修訂時設定的，而且在建立服務時無法變更。若要更新磁碟區名稱與類型，您必須建立新的作業定義修訂版本，並使用新的修訂版本建立服務。

設定此磁碟區類型	執行此作業
Amazon EBS	a. 針對 EBS 磁碟區類型，選擇要附加至工作的 EBS 磁碟區類型。 b. 在大小 (GiB) 中，輸入磁碟區大小的有效值 (以 GiB) 為單位。您可以指定最小 1 GiB 和最大 16,384 GiB 磁碟區大小。除非您提供快照 ID，否則此值是必要的。 c. 若為 IOPS，請輸入磁碟區應提供的最大輸入/輸出作業數 (IOPS)。此值僅適用於io1io2、和gp3磁碟區類型。 d. 對於輸送量 (MIB/s)，輸入磁碟區應提供的輸送量，以 MB/秒為單位 (MiBps

設定此磁碟區類型	執行此作業	
	或 MIB/s)。此值僅適用於gp3磁碟區類型。 e. 對於快照 ID，請選擇現有的 Amazon EBS 磁碟區快照，或者如果要從快照建立磁碟區，請輸入快照的 ARN。您也可以不選擇或輸入快照 ID 來建立新的空白磁碟區。 f. 在 [檔案系統類型] 中，選擇磁碟區上資料儲存和擷取要使用的檔案系統類型。您可以選擇作業系統預設值或特定檔案系統類型。Linux 的預設值為XFS。對於從快照建立的磁碟區，您必須指定與建立快照時磁碟區所使用的相同檔案系統類型。如果文件系統類型不匹配，則任務將無法啟動。 g. 對於基礎設施角色，請選擇具有必要許可的 IAM 角色，以允許 Amazon ECS 管理任務的 Amazon EBS 磁碟區。您可以將AmazonECS Infrastru ctureRole PolicyForVolumes 受管理的原則附加至角色，也可以使用原則做為指南，建立並附加具有符合您特定需求的權限的自己	

設定此磁碟區類型	執行此作業	
	原則。如需有關必要許可的詳細資訊，請參閱 Amazon ECS 基礎設施 IAM 角色。 h. 對於加密，如果您要使用 Amazon EBS 加密的預設設定，請選擇預設。如果您的帳戶預設設定了加密，則磁碟區將使用設定中指定的 AWS Key Management Service (AWS KMS) 金鑰加密。如果您選擇預設且 Amazon EBS 預設加密並未開啟，則磁碟區將不加密。 如果您選擇 [自訂]，您可以指定您選擇 AWS KMS key 的磁碟區加密。 如果您選擇 [無]，除非您預設已設定加密，或是從加密快照建立磁碟區，否則磁碟區將不加密。 i. 如果您已選擇「加密自訂」，則必須指 AWS KMS key 定要使用的項目。對於 KMS 金鑰，請選擇 AWS KMS key 或輸入金鑰 ARN。如果您選擇使用對稱的客戶管理金鑰來加密磁碟區，請確定您擁有在 AWS KMS key 政策中定義的正確權限。如需詳細資訊，請參閱	

設定此磁碟區類型	執行此作業
	Amazon EBS 磁碟區的資料加密。 j. (選擇性) 在標籤下，您可以透過從任務定義或服務傳播標籤，或提供您自己的標籤，將標籤新增至 Amazon EBS 磁碟區。 如果您要從工作定義傳輸標籤，請選擇「工作定義」做為「傳輸標籤來源」。如果您要從服務傳輸標記，請選擇服務做為傳輸標記來源。如果您選擇不傳播，或者如果您沒有選擇值，則不會傳播標籤。 如果您想要提供自己的標籤，請選擇 [新增標籤]，然後為您新增的每個標籤提供索引鍵和值。 如需標記 Amazon EBS 磁碟區的詳細資訊，請參閱標記 Amazon EBS 磁碟區。

12. (選用) 為協助識別您的服務和任務，請展開 Tags (標籤) 區段，然後設定標籤。

    若要讓 Amazon ECS 使用叢集名稱和任務定義標籤，自動標記所有新啟動的任務，請選取 Turn on Amazon ECS managed tags (開啟 Amazon ECS 受管標籤)，然後針對 Propagate tags from (傳播標籤來源)，選取 Task definitions (任務定義)。

    若要讓 Amazon ECS 使用叢集名稱和服務標籤，自動標記所有新啟動的任務，請選取 Turn on Amazon ECS managed tags (開啟 Amazon ECS 受管標籤)，然後針對 Propagate tags from (傳播標籤來源)，選取 Service (服務)。

新增或移除標籤。

- [新增標籤] 選擇新增標籤，然後執行下列操作：
  - 在索引鍵中，輸入索引鍵名稱。
  - 在值中，進入索引鍵值。
- [移除標籤] 在標籤旁邊，選擇 移除標籤。

使用主控台更新 Amazon ECS 服務

您可以使用 Amazon ECS 主控台更新 Amazon ECS 服務。目前的服務組態會預先填入。您可以更新任務定義、所需任務計數、容量提供者策略、平台版本和部署組態；或這些的任意組合。

如需有關如何更新藍/綠部署組態的相關資訊，請參閱 使用主控台更新Amazon ECS 藍色/綠色部署。

使用主控台時應考慮以下項目：

如果您要暫時停止服務，請將 [所需的工作] 設定為 0。然後，當您準備好啟動服務時，請使用原始「想要的」任務計數更新服務。

使用主控台時應考慮以下項目：

- 您必須使用 AWS Command Line Interface 來更新使用下列任一參數的服務：
  - 藍/綠部署
  - 服務探索 — 您只能檢視您的服務探索組態。
  - 使用自訂指標追蹤政策
  - 更新服務 — 您無法更新`awsvpc`網路組態和健全狀況檢查寬限期。

  如需有關如何使用更新服務的資訊 AWS CLI，請參閱《AWS Command Line Interface 參考》update-service中的〈〉。
- 如果您要在任務定義中變更容器使用的連接埠，您可能需要更新容器執行個體的安全群組，才能使用更新的連接埠。
- Amazon ECS 不會自動更新與 Elastic Load Balancing 負載平衡器或 Amazon ECS 容器執行個體相關聯的安全群組。
- 如果您的服務使用負載平衡器，於建立時為您的服務定義的負載平衡器組態無法使用主控台變更。您可以改用 AWS CLI 或 SDK 來修改負載平衡器組態。如需如何修改組態的相關資訊，請參閱 Amazon 彈性容器服務 API 參考UpdateService中的。

- 如果您為服務更新任務定義，則在負載平衡器組態中指定的容器名稱和容器連接埠必須保留在任務定義中。

您可以更新現有的服務，以變更某些服務組態參數，例如服務維護的任務數或任務使用的任務定義。另外，如果您使用的是 Fargate 任務類型，您可以變更任務使用的平台版本。使用 Linux 平台版本的服務無法更新為使用 Windows 平台版本，反之亦然。如果您的應用程式需要更多容量，您可以擴展您的服務。如果您有未使用的容量要縮減，您可以減少服務所需的任務數並釋出資源。

若要將更新容器映像用於任務，您可以使用該映像建立新任務定義修訂版，然後使用主控台中的 force new deployment (強制執行新部署) 選項來將其部署至服務。

服務排程器 (在服務的部署組態中) 使用運作狀態百分比下限和百分比上限參數，決定部署策略。

如果服務使用滾動更新 (ECS) 部署類型，運作狀態百分比下限代表部署期間，服務中必須維持在 RUNNING 狀態的任務數量下限，以所需任務數量的百分比表示 (無條件進位到最接近的整數)。如果服務包含的任務使用 EC2 啟動類型，即使有任何容器執行個體處於 DRAINING 狀態，此參數也適用。使用此參數進行部署，無須使用額外的叢集容量。例如，如果您的服務所需任務數量為四項，且運作狀態百分比下限為 50%，則排程器可先停止兩項現有的任務以釋出叢集容量，再啟動兩項新的任務。如果未使用負載平衡器的服務任務為 RUNNING 狀態，則視為運作良好。如果服務的任務使用負載平衡器，這些任務只要是處於 RUNNING 狀態，且負載平衡器也回報為運作良好，即視為運作良好。運作狀態百分比下限的預設值為 100%。

如果服務使用滾動更新 (ECS) 部署類型，則百分比上限參數代表在部署期間，服務中允許處於 PENDING、RUNNING 或 STOPPING 狀態的任務數量上限，以所需任務數量的百分比表示 (無條件捨去小數，只保留整數)。如果服務包含的任務使用 EC2 啟動類型，即使有任何容器執行個體處於 DRAINING 狀態，此參數也適用。使用此參數來定義部署批次大小。例如，如果您的服務有所需的四個任務數目及百分比上限值 200%，則排程器可先啟動四項新任務，再停止四項舊任務。前提是有執行此操作所需的叢集資源。百分比上限的預設值為 200%。

當服務排程器於更新期間取代任務時，服務會先從負載平衡器 (如果使用) 移除任務，並等待連線耗盡。然後，對任務中執行的容器發出相當於 docker stop 的命令。這會造成 SIGTERM 信號和 30 秒逾時，並在之後傳送 SIGKILL，強制停止容器。如果容器在接收到 SIGTERM 信號後於 30 秒內從容處理完畢並結束，就不會傳送任何 SIGKILL 信號。服務排程器依據您定義的運作狀態百分比下限和百分比上限設定，啟動和停止任務。

在容器運作狀態檢查或負載平衡器目標群組運作狀態檢查失敗後，服務排程器也會取代判定為狀況不佳的任務。此取代取決於 maximumPercent 和 desiredCount 服務定義參數。如果任務標記為運作狀態不佳，服務排程器會先啟動替代任務。然後，會發生以下情況。

- 如果取代工作的健全狀況狀態為HEALTHY，服務排程器會停止運作狀況不良的工作
- 如果替代任務的運作狀態為 UNHEALTHY，排程器會停止運作狀態不佳的替代任務或現有運作狀態不佳的任務，以使任務總計數等於 desiredCount。

如果 maximumPercent 參數限制排程器先啟動替代任務，排程器會隨機停止運作狀態不佳的任務，以釋放容量，然後啟動替代任務。開始和停止程序會繼續進行，直到所有運作狀態不佳的任務都會取代為運作狀態良好的 一旦已取代了所有運作狀態不佳的任務，而且只執行運作狀態良好的任務，如果任務總數超過 desiredCount，則運作良好的任務會隨機停止，直到任務總計數等於 desiredCount 為止。如需有關 maximumPercent 和 desiredCount 的詳細資訊，請參閱服務定義參數。

> **Important**
>
> 如果您要在任務定義中變更容器使用的連接埠，您可能需要更新容器執行個體的安全群組，才能使用更新的連接埠。
> 如果您為服務更新任務定義，則在服務建立時指定的容器名稱和容器連接埠必須保留在任務定義中。
> Amazon ECS 不會自動更新與 Elastic Load Balancing 負載平衡器或 Amazon ECS 容器執行個體相關聯的安全群組。

更新服務 (Amazon ECS 主控台)

1. 開啟主控台，網址為 https://console.aws.amazon.com/ecs/v2。
2. 在叢集頁面上，選擇叢集。
3. 在叢集詳細資料頁面的 [服務] 區段中，選取服務旁的核取方塊，然後選擇 [更新]。
4. 若要讓服務啟動新部署，請選取 Force new deployment (強制執行新部署)。
5. 在任務定義中，選擇任務定義系列和修訂。

   > **Important**
   >
   > 主控台會驗證選取的工作定義系列和修訂版是否與定義的計算規劃相容。如果您收到警告，請確認您的工作定義相容性和您選取的計算組態。

6. 針對 [想要的工作]，輸入您要針對服務執行的工作數目。

7. 針對 Min running tasks (執行中任務下限)，輸入部署期間必須維持在 RUNNING 狀態的服務任務數量下限，它是所需任務數量的百分比 (無條件進位到最接近的整數)。如需詳細資訊，請參閱部署組態。
8. 針對 Max running tasks (執行中任務上限)，輸入部署期間允許的處於 RUNNING 或 PENDING 狀態的服務任務數目上限，它是所需任務數量的百分比 (無條件捨去到最接近的整數)。
9. 若要設定 Amazon ECS 如何偵測並處理部署失敗，請展開 Deployment failure detection (部署失敗偵測)，然後選擇您的選項。

    a. 若要在任務無法啟動時停止部署，請選取 Use the Amazon ECS deployment circuit breaker (使用 Amazon ECS 部署斷路器)。

       若要讓軟體在部署斷路器將部署設定為失敗狀態時，自動將部署復原至上次完成的部署狀態，請選取「失敗時復原」。

    b. 若要停止以應用程式測量結果為基礎的部署，請選取使用 CloudWatch 警示。然後，從CloudWatch 警報名稱中選擇鬧鐘。若要建立新鬧鐘，請前往主 CloudWatch 控台。

       若要讓軟體在 CloudWatch 警示將部署設定為失敗狀態時，自動將部署回復至上次完成的部署狀態，請選取失敗時復原。

10. 若要變更運算選項，請展開 [運算組態]，然後執行下列動作：

    a. 對於開啟的服務 AWS Fargate，對於平台版本，請選擇新版本。

    b. 對於使用容量提供者策略的服務，對於容量提供者策略，請執行下列動作：

       - 若要新增其他容量提供者，請選擇 Add more (新增更多)。然後，針對 Capacity provider (容量提供者)，選擇容量提供者。
       - 若要移除容量提供者，請選擇容量提供者右側的 Remove (移除)。

       使用 Auto Scaling 群組容量提供者的服務無法更新為使用 Fargate 容量提供者。使用 Fargate 容量提供者的服務無法更新為使用 Auto Scaling 群組容量提供者。

11. (選擇性) 若要設定服務 Auto Scaling，請展開服務自動擴展，然後指定下列參數。

    a. 若要使用服務自動擴展，請選取 Service auto scaling (服務自動擴展)。

    b. 對於工作數目下限，請輸入要使用的服務 auto 擴展的工作數目下限。所需的計數不會低於此計數。

    c. 針對工作數目上限，輸入要使用的服務 auto 擴展的工作數目上限。所需的計數不會高於此計數。

d. 選擇政策類型。在「資源調整」政策類型下，選擇下列其中一個選項。

若要使用此政策，請參閱。	執行此作業...	
目標追蹤	a. 針對 Scaling policy type (擴展政策類型)，選擇 Target tracking (目標追蹤)。 b. 針對 Policy name (政策名稱)，輸入政策的名稱。 c. 針對 ECS service metric (ECS 服務指標)，選擇下列其中一項指標。 • ECS ServiceAverage CPU 使用率 — 服務的平均 CPU 使用率。 • ECS ServiceAverage MemoryUtilization — 服務的平均記憶體使用率。 • ALB RequestCount PerTarget —「Application Load Balancer」目標群組中每個目標完成的要求數目。 d. 針對 Target value (目標值)，輸入服務為選取的指標保持的值。 e. 對於向外延展冷卻時間，請輸入在其他向外延展活動開始之前必須經過的向外延展活動 (新增任務) 之後的時間量 (以秒為單位)。	

若要使用此政策，請參閱。	執行此作業...	
	f. 對於縮放冷卻時間，請輸入在其他縮放活動開始之前必須經過的縮放活動（刪除任務）之後的時間量（以秒為單位）。 g. 若要防止政策執行縮減活動，請選取 Turn off scale-in (關閉縮減)。 h. • (選用) 如果您希望擴展政策擴展以增加流量，但不需要在流量減少時擴展，請選取關閉擴展政策。	

若要使用此政策，請參閱。	執行此作業...	
步驟擴展	a. 針對 Scaling policy type (擴展政策類型)，選擇 Step scaling (步驟擴展)。 b. 在 Policy Name (政策名稱) 輸入政策的名稱。 c. 針對 Alarm name (警示名稱)，輸入警示的唯一名稱。 d. 針對 Amazon ECS service metric (Amazon ECS 服務指標)，選擇用於警示的指標。 e. 針對 Statistic (統計資料)，選擇警示統計資料。 f. 針對 Period (期間)，選擇警示的期間。 g. 針對 Alarm condition (警示條件)，選擇如何比較選取的指標與定義的閾值。 h. 針對 Threshold to compare metrics (比較閾值與指標) 和 Evaluation period to initiate alarm (啟動警示的評估期)，輸入用於警示的閾值以及評估閾值的時間長度。 i. 在 Scaling actions (擴展動作) 下，執行下列動作：	

若要使用此政策，請參閱。	執行此作業...	
	• 在動作中，選取是否要新增、移除或設定服務的特定所需計數。 • 如果您選擇新增或移除工作，在「值」中，輸入啟動縮放動作時要新增或移除的工作數目 (或現有工作的百分比)。如果您選擇設定所需的計數，請輸入任務數量。對於 Type (類型)，選取 Value (值) 是整數或是現有所需計數的百分比值。 • 針對 Lower bound (下限) 和 Upper bound (上限)，輸入步驟擴展調整的下界限和上界限。根據預設，新增政策的下限為警示閾值，而上限為無限大正數 (+) 值。根據預設，移除政策的上限為警示閾值，而下限為無限小負數 (-) 值。 • (選用) 新增其他擴展選項。選擇「新增縮放」動作，然後重複「縮放」動作步驟。 • 對於冷卻時間，請輸入等待先前縮放活動生效的時間（以秒為單位）。對於新增政策，這是擴展活動之後的時	

若要使用此政策，請參閱。	執行此作業...	
	間，擴展政策會封鎖擴展活動，並限制一次可向外延展的工作數目。對於移除原則，這是在縮放活動之後必須經過的時間，才能開始另一個縮放活動。	

12. (選用) 若要使用 Service Connect，請選取 Turn on Service Connect (開啟 Service Connect)，然後指定下列項目：

    a. 在 Service Connect configuration (Service Connect 組態) 下，指定用戶端模式。

    - 如果服務執行的網路用戶端應用程式只需要連線到命名空間中的其他服務，請選擇 Client side only (僅用戶端)。
    - 如果服務執行的是網路或 Web 服務應用程式，且需要為此服務提供端點，並連線至命名空間中的其他服務，請選擇 Client and server (用戶端和伺服器)。

    b. 若要使用非預設叢集命名空間的命名空間，請在 Namespace (命名空間) 欄位中選擇服務命名空間。

13. 如果您的工作使用與部署時設定相容的資料磁碟區，您可以透過擴充磁碟區來設定磁碟區。

    磁碟區名稱和磁碟區類型是在您建立工作定義修訂時設定的，而且在您更新服務時無法變更。若要更新磁碟區名稱與類型，您必須建立新的作業定義修訂版本，並使用新的修訂版本來更新服務。

設定此磁碟區類型	執行此作業	
Amazon EBS	a. 針對 EBS 磁碟區類型，選擇要附加至工作的 EBS 磁碟區類型。 b. 在大小 (GiB) 中，輸入磁碟區大小的有效值 (以 GiB) 為單位。您可以指定最小 1 GiB 和最大 16,384 GiB 磁碟區大小。除非您	

設定此磁碟區類型	執行此作業	
	提供快照 ID，否則此值是必要的。 c. 若為 IOPS，請輸入磁碟區應提供的最大輸入/輸出作業數 (IOPS)。此值僅適用於io1io2、和gp3磁碟區類型。 d. 對於輸送量 (MIB/s)，輸入磁碟區應提供的輸送量，以 MB/秒為單位 (MiBps 或 MIB/s)。此值僅適用於gp3磁碟區類型。 e. 對於快照 ID，請選擇現有的 Amazon EBS 磁碟區快照，或者如果要從快照建立磁碟區，請輸入快照的 ARN。您也可以不選擇或輸入快照 ID 來建立新的空白磁碟區。 f. 在 [檔案系統類型] 中，選擇磁碟區上資料儲存和擷取要使用的檔案系統類型。您可以選擇作業系統預設值或特定檔案系統類型。Linux 的預設值為XFS。對於從快照建立的磁碟區，您必須指定與建立快照時磁碟區所使用的相同檔案系統類型。如果文件系統類型不匹配，則任務將無法啟動。 g. 對於基礎設施角色，請選擇具有必要許可的 IAM 角色，以允許	

設定此磁碟區類型	執行此作業	
	Amazon ECS 管理任務的 Amazon EBS 磁碟區。您可以將AmazonECSInfrastructureRolePolicyForVolumes 受管理的原則附加至角色，也可以使用原則做為指南，建立並附加具有符合您特定需求的權限的自己原則。如需有關必要許可的詳細資訊，請參閱 Amazon ECS 基礎設施 IAM 角色。 h. 對於加密，如果您要使用 Amazon EBS 加密的預設設定，請選擇預設。如果您的帳戶預設設定了加密，則磁碟區將使用設定中指定的 AWS Key Management Service (AWS KMS) 金鑰加密。如果您選擇預設且 Amazon EBS 預設加密並未開啟，則磁碟區將不加密。 如果您選擇 [自訂]，您可以指定您選擇 AWS KMS key 的磁碟區加密。 如果您選擇 [無]，除非您預設已設定加密，或是從加密快照建立磁碟區，否則磁碟區將不加密。	

設定此磁碟區類型	執行此作業	
	i. 如果您已選擇「加密自訂」，則必須指 AWS KMS key 定要使用的項目。對於 KMS 金鑰，請選擇 AWS KMS key 或輸入金鑰 ARN。如果您選擇使用對稱的客戶管理金鑰來加密磁碟區，請確定您擁有在 AWS KMS key 政策中定義的正確權限。如需詳細資訊，請參閱 Amazon EBS 磁碟區的資料加密。 j. (選擇性) 在標籤下，您可以透過從任務定義或服務傳播標籤，或提供您自己的標籤，將標籤新增至 Amazon EBS 磁碟區。 如果您要從工作定義傳輸標籤，請選擇「工作定義」做為「傳輸標籤來源」。如果您要從服務傳輸標記，請選擇服務做為傳輸標記來源。如果您選擇不傳播，或者如果您沒有選擇值，則不會傳播標籤。 如果您想要提供自己的標籤，請選擇 [新增標籤]，然後為您新增的每個標籤提供索引鍵和值。	

設定此磁碟區類型	執行此作業
	如需標記 Amazon EBS 磁碟區的詳細資訊，請參閱標記 Amazon EBS 磁碟區。

14. (選用) 為協助識別您的服務，請展開 Tags (標籤) 區段，然後設定標籤。
    - [新增標記] 選擇 [新增標籤]，然後執行下列動作：
      - 在索引鍵中，輸入索引鍵名稱。
      - 在值中，進入索引鍵值。
    - [移除標籤] 在標籤旁邊，選擇 移除標籤。
15. 選擇更新。

使用主控台更新Amazon ECS 藍色/綠色部署

您可以使用 Amazon ECS 主控台更新藍/綠部署組態 會預先填入目前的藍/綠部署組態。您可以更新下列藍/綠部署選項：

- 部署群組名稱-部 CodeDeploy 署設定
- 應用程式名稱-部 CodeDeploy 署群組
- 部署組態-如何在部署期間將生產流量 CodeDeploy 路由至取代工作集
- 在負載平衡器上測試接聽程式- CodeDeploy 使用測試接聽程式將測試流量路由至部署期間的取代工作集

您必須先設定新選項，才能更新組態。

若要更新藍/綠部署組態 (Amazon ECS 主控台)

1. 開啟主控台，網址為 https://console.aws.amazon.com/ecs/v2。
2. 在 Clusters (叢集) 頁面上，選取您的叢集。
3. 在 Cluster overview (叢集概觀) 頁面中，選取服務，然後選擇 Update (更新)。
4. 展開 [部署選項-技術支援] CodeDeploy，然後選擇要更新的選項：
   - 若要修改部 CodeDeploy 署群組，請在應用程式名稱中選擇部署群組。

- 若要修改部 CodeDeploy 署設定，請在部署群組名稱中選擇群組。
- 若要修改在部署期間將生產流量 CodeDeploy 路由至取代工作集的方式，對於部署組態，請選擇選項。

5. 選取要在服務部署的新修訂版中運行的部署生命週期事件勾點和相關聯的 Lambda 函數。可用的生命週期勾點如下：

   - BeforeInstall— 在建立替代任務集之前，使用此部署生命週期事件掛接來叫用 Lambda 函數。此生命週期事件上的 Lambda 函數結果不會起始回復。
   - AfterInstall— 建立取代任務集後，使用此部署生命週期事件掛接來叫用 Lambda 函數。此生命週期事件上的 Lambda 函數結果可以起始回復。
   - BeforeAllow流量 — 在生產流量重新路由至取代任務集之前，使用此部署生命週期事件掛接呼叫 Lambda 函數。此生命週期事件上的 Lambda 函數結果可以起始回復。
   - AfterAllow流量 — 在生產流量重新路由至取代任務集後，使用此部署生命週期事件掛接呼叫 Lambda 函數。此生命週期事件上的 Lambda 函數結果可以起始回復。

6. 若要修改測試監聽器，請展開負載平衡，然後針對測試接聽程式進行 CodeDeploy 建置，選擇測試監聽器。
7. 選擇更新。

使用主控台刪除 Amazon ECS 服務

您可以使用主控台刪除 Amazon ECS 服務。刪除之前，服務會自動將規模縮減至零。與該服務關聯的負載平衡器或服務探索資源，不會受到服務刪除的影響。若要刪除您的 Elastic Load Balancing 資源，請視您的負載平衡器類型，參閱下列任一主題：刪除 Application Load Balancer 或刪除 Network Load Balancer。

當您刪除服務時，如果仍有執行中的工作需要清除，服務狀態會從移ACTIVE至DRAINING，而且服務在主控台或 ListServices API 作業中不再可見。在所有任務轉換為 STOPPING 或 STOPPED 狀態後，服務狀態會從 DRAINING 變為 INACTIVE。仍可透過 DescribeServices API 作業檢視DRAINING或INACTIVE狀態的服務。

> **Important**
>
> 如果您嘗試建立與現有服務名稱相同的新服務，其中一個ACTIVE或DRAINING狀態，就會收到錯誤訊息。

程序

1. 開啟主控台，網址為 https://console.aws.amazon.com/ecs/v2。
2. 在 Clusters (叢集) 頁面，為服務選取叢集。
3. 在 Clusters (叢集) 頁面上，選擇叢集。
4. 在 Cluster : ***name*** (叢集：名稱) 頁面上，選擇 Services (服務) 索引標籤。
5. 選取服務，然後選擇 Delete (刪除)。
6. 若要刪除服務，即使該服務並未縮減規模至零任務，請選取 Force delete service (強制刪除服務)。
7. 出現確認提示時，輸入 delete，然後選擇 [刪除]。

透過取代任務部署 Amazon ECS 服務

當您建立使用滾動更新 (ECS) 部署類型的服務時，Amazon ECS 服務排程器會以新任務取代目前執行中的任務。Amazon ECS 在滾動更新期間從服務新增或移除的任務數量是由服務部署組態控制。部署組態由以下項目組成：

- `minimumHealthyPercent` 代表在部署期間或容器執行個體耗盡時，應為服務執行的任務數量下限 (佔服務所需任務數量的百分比)。此值會向上捨入。例如，如果最小運作狀態良好的百分比為 50，且所需的任務計數為 4，則排程器可以先停止兩項現有的任務，再開始兩項新的任務。同樣地，如果最小運作狀態良好的百分比為 75%，而所需的任務計數為 2，則排程器無法停止任何任務，因為產生的值也是 2。

  如果任務狀況不佳，Amazon ECS 服務排程器會先啟動替換任務並維持任務，直到替換`minimumHealthyPercent`任務健康為止。隨著替換任務的啟動並變得健康，不健康的任務將逐漸停止。
- `maximumPercent` 代表在部署期間或容器執行個體耗盡時，應為服務執行的任務數目上限 (佔服務所需任務數量的百分比)。此值會向下捨去。例如，如果最大百分比為 200，且所需的任務計數為四，則排程器可以先停止四項現有的任務，再開始四項新的任務。同樣地，如果最大百分比為 125，且所需的任務計數為 3，則排程器無法停止任何任務，因為產生的值也是 3。

> **Important**
>
> 設定最小運作狀態良好的百分比或最大百分比時，您應確定排程器在部署初始化時，可以停止或啟動至少一項任務。如果您的服務部署因無效的部署組態而停滯，將傳送服務事件簡訊。如

需詳細資訊，請參閱 服務 (*service-name*) 在部署期間因服務部署組態而無法停止或啟動任務。請更新 minimumHealthyPercent 或「最大百分比」值，然後再試一次。。

滾動式部署會使用部署斷路器來判斷任務是否達到穩定狀態。部署斷路器可以選擇性在失敗時復原部署。

故障檢測

在部署失敗時快速識別，然後選擇性將失敗復原至上次運作中部署的兩種方法。

- the section called “ 部署斷路器如何偵測故障”
- the section called “ CloudWatch 警示如何偵測部署失敗”

可以單獨使用或一起使用這些方法。當同時使用兩種方法時，只要符合任一失敗方法的失敗條件，就會將部署設為失敗。

請遵循下列準則來協助判斷要使用的方法：

- 斷路器 – 當您想要在任務無法啟動而停止部署時，請使用此方法。
- CloudWatch alarm-當您要停止根據應用程式測量結果的部署時，請使用此方法。

Amazon ECS 部署斷路器如何偵測故障

部署斷路器是可確定任務是否達到穩定狀態的滾動式更新機制。部署斷路器邏輯有一個選項，可將失敗的部署自動復原至處於 COMPLETED 狀態的部署。

當服務部署狀態變更時，Amazon ECS 會將服務部署狀態變更事件傳送到 EventBridge。這提供一種程式設計方式來監控您的服務部署的狀態。如需詳細資訊，請參閱 Amazon ECS 服務部署狀態變更事件。我們建議您使用eventName的建立和監視 EventBridge 規則，以SERVICE_DEPLOYMENT_FAILED便您可以採取手動動作來開始部署。如需詳細資訊，請參閱 Amazon EventBridge 使用者指南中的建立 EventBridge規則。

當部署斷路器判斷部署失敗時，會尋找處於 COMPLETED 狀態的最新部署。這是部署斷路器用來做為復原部署的部署。復原開始時，部署會從 COMPLETED 變更為 IN_PROGRESS。這表示在部署到達 COMPLETED 狀態之前，不符合另一個復原的資格。當部署斷路器找不到處於 COMPLETED 狀態的部署時，斷路器不會啟動新任務，且會阻礙部署。

當您建立服務時，排程器會追蹤分兩個階段啟動失敗的工作。

- 階段 1-排程器會監視工作，以查看它們是否已轉換為 RUNNING 狀態。
  - 成功-部署有機會轉換至「已完成」狀態，因為有一個以上的工作已轉換為 RUNNING 狀態。跳過失敗準則，斷路器將移至階段 2。
  - 失敗-有些工作未轉換為 RUNNING 狀態，且部署可能會轉換為「失敗」狀態。
- 階段 2-當至少有一個工作處於 RUNNING 狀態時，部署會進入此階段。斷路器會檢查目前部署中正在評估的工作狀態檢查。通過驗證的健康狀態檢查包括 Elastic Load Balancing、 AWS Cloud Map 服務健康狀態檢查和容器健康狀態檢查
  - 成功-在執行中狀態中至少有一項工作，且運作狀態檢查已通過。
  - 失敗-因健全狀況檢查失敗而取代的工作已達到失敗臨界值。

在服務上使用部署斷路器方法時，請考慮下列事項。 EventBridge 產生規則。

- `DescribeServices` 回應可提供部署狀態洞察，`rolloutState` 和 `rolloutStateReason`。當啟動新的部署時，推展狀態從 `IN_PROGRESS` 狀態開始。當服務到達穩定狀態時，推展狀態轉移到 `COMPLETED`。若服務無法達到穩定狀態，而斷路器已開啟，則部署將轉換為 `FAILED` 狀態。處於 `FAILED` 狀態的部署不會啟動任何新任務。
- 除 Amazon ECS 為已開始和已完成的部署傳送服務部署狀態變更事件以外，Amazon ECS 還會在開啟的斷路器部署失敗時傳送事件。這些事件提供有關部署失敗原因的詳細資訊，或者部署是否因為回復而開始。如需詳細資訊，請參閱 Amazon ECS 服務部署狀態變更事件。
- 如果新的部署因為先前的部署失敗並發生復原而開始，服務部署狀態變更事件的 `reason` 欄位將指示部署因為復原而開始。
- 部署斷路器僅支援使用滾動更新 (ECS) 部署控制器的 Amazon ECS 服務。
- 您必須使用 Amazon ECS 主控台，或者在使用部署斷路器 AWS CLI 時搭配選 CloudWatch 項。如需詳細資訊，請參閱《AWS Command Line Interface 參考》中的 the section called “使用定義的參數建立服務” 和 create-service。

下列`create-service` AWS CLI 範例顯示如何在將部署斷路器與復原選項搭配使用時建立 Linux 服務。

```
aws ecs create-service \
     --service-name MyService \
     --deployment-controller type=ECS \
     --desired-count 3 \
```

```
    --deployment-configuration "deploymentCircuitBreaker={enable=true,rollback=true}"
\
    --task-definition sample-fargate:1 \
    --launch-type FARGATE \
    --platform-family LINUX \
    --platform-version 1.4.0 \
    --network-configuration
"awsvpcConfiguration={subnets=[subnet-12344321],securityGroups=[sg-12344321],assignPublicIp=EN
```

範例：

部署 1 處於 COMPLETED 狀態。

部署 2 無法啟動，因此斷路器會復原至部署 1。部署 1 會轉換至 IN_PROGRESS 狀態。

部署 3 會啟動，且 COMPLETED 狀態中沒有部署，因此部署 3 無法復原或啟動任務。

Failure threshold

部署斷路器會計算閾值，然後使用該值判斷何時將部署移動到 FAILED 狀態。

部署斷路器的最小臨界值為 3，最大臨界值為 200。並使用以下公式中的值來判斷部署失敗。

```
Minimum threshold <= 0.5 * desired task count => maximum threshold
```

當計算結果大於最小值 3，但小於最大值 200 時，會將失敗臨界值設定為計算的閾值 (四捨五入)。

> **Note**
>
> 您不能更改任何一個閾值。

部署狀態檢查有兩個階段。

1. 部署斷路器監視部署過程中的任務，並檢查處於 RUNNING 狀態的任務。當目前部署中的任務處於 RUNNING 狀態時，排程器會忽略故障條件，並繼續進行到下一個階段。當任務無法到達 RUNNING 狀態時，部署斷路器會在故障計數上增加一。當故障計數等於閾值時，部署將標記為 FAILED。
2. 當RUNNING狀態中有一個或多個任務時，就會輸入此階段。部署斷路器對目前部署中的任務，針對以下資源執行運作狀態檢查：
   - Elastic Load Balancing 負載平衡器

- AWS Cloud Map 服務
- Amazon ECS 容器運作狀態檢查

當任務的運作狀態檢查失敗時，部署斷路器會在故障計數上增加一。當故障計數等於閾值時，部署將標記為 FAILED。

下表提供一些範例。

所需的任務計數	計算	Threshold
1	`3 <= 0.5 * 1 => 200`	3 (計算值小於最小值)
25	`3 <= 0.5 * 25 => 200`	13 (值會無條件進位)
400	`3 <= 0.5 * 400 => 200`	200
800	`3 <= 0.5 * 800 => 200`	200 (計算值大於最大值)

例如，當臨界值為 3 時，斷路器會以設定為 0 的故障計數開始。當工作無法達到RUNNING狀態時，部署斷路器會將失敗次數增加 1。當失敗計數等於 3 時，部署會標記為FAILED。

如需有關使用復原選項的其他範例，請參閱 Announcing Amazon ECS deployment circuit breaker (推出 Amazon ECS 部署斷路器)。

CloudWatch 警示如何偵測 Amazon ECS 部署失敗

您可以設定 Amazon ECS，在偵測到指定 CloudWatch 警示進入ALARM狀態時，將部署設定為失敗。

您可以選擇性地設定組態，將失敗的部署復原至上次完成的部署。

下列create-service AWS CLI 範例顯示如何在將部署警示與復原選項搭配使用時建立 Linux 服務。

```
aws ecs create-service \
     --service-name MyService \
```

```
    --deployment-controller type=ECS \
    --desired-count 3 \
    --deployment-configuration
"alarms={alarmNames=[alarm1Name,alarm2Name],enable=true,rollback=true}" \
    --task-definition sample-fargate:1 \
    --launch-type FARGATE \
    --platform-family LINUX \
    --platform-version 1.4.0 \
    --network-configuration
"awsvpcConfiguration={subnets=[subnet-12344321],securityGroups=[sg-12344321],assignPublicIp=EN
```

在服務上使用 Amazon CloudWatch 警示方法時，請考慮下列事項。

- 封裝時間是新服務版本橫向擴展且舊服務版本縮減之後的一段時間，在此期間，Amazon ECS 會繼續監控與部署相關聯的警示。Amazon ECS 會根據與部署相關聯的警示組態來計算此時段。
- `deploymentConfiguration` 請求參數現在包含 `alarms` 資料類型。您可以指定警示名稱、是否使用該方法，以及是否在警示指出部署失敗時初始化復原。如需詳細資訊，請參閱 Amazon 彈性容器服務 API 參考CreateService中的。
- `DescribeServices` 回應可提供部署狀態洞察，`rolloutState` 和 `rolloutStateReason`。當啟動新的部署時，推展狀態從 `IN_PROGRESS` 狀態開始。當服務達到穩定狀態且封裝時間完成時，推展狀態轉換為 `COMPLETED`。若服務無法達到穩定狀態，而警示已進入 `ALARM` 狀態，部署將轉換為 `FAILED` 狀態。處於 `FAILED` 狀態的部署不會啟動任何新任務。
- 除 Amazon ECS 為已開始和已完成的部署傳送服務部署狀態變更事件以外，Amazon ECS 還會在使用警示的部署失敗時傳送事件。這些事件提供有關部署失敗原因的詳細資訊，或者部署是否因為回復而開始。如需詳細資訊，請參閱 Amazon ECS 服務部署狀態變更事件。
- 如果新的部署因為先前的部署失敗並已開啟復原而開始，服務部署狀態變更事件的 `reason` 欄位將指示部署因為復原而開始。
- 如果您使用部署斷路器和 Amazon CloudWatch 警示來偵測故障，任何一種方法都可以在符合任一方法的條件後立即啟動部署失敗。當您對初始化部署失敗的方法使用復原選項時，就會發生復原。
- 只有使用滾動更新 (ECS) 部署控制器的 Amazon ECS 服務才支援 Amazon CloudWatch 警示。
- 您可以使用 Amazon ECS 主控台來設定此選項，或使用 AWS CLI. 如需詳細資訊，請參閱《AWS Command Line Interface 參考》中的 the section called “使用定義的參數建立服務” 和 create-service。
- 您可能會注意到，部署狀態會維持 `IN_PROGRESS` 較長的時間。其原因是 Amazon ECS 在刪除作用中的部署之前不會變更狀態，並且直到封裝時間過後才會發生這種情況。視警示組態而定，部署需要的時間可能比您不使用警示多幾分鐘 (即使新的主要任務集已縱向擴展，而舊的部署也縮減規模)。如

果您使用 CloudFormation 逾時，請考慮增加逾時。如需詳細資訊，請參閱《AWS CloudFormation 使用者指南》中的在範本中建立等待條件。

- Amazon ECS 呼叫 `DescribeAlarms` 以輪詢警示。要`DescribeAlarms`計入與您帳戶相關聯之 CloudWatch 服務配額的呼叫。如果您有其他呼叫的 AWS 服務`DescribeAlarms`，可能會影響 Amazon ECS 輪詢警示。例如，如果其他服務撥出足夠的`DescribeAlarms`呼叫以達到配額，則該服務會受到限制，Amazon ECS 也會受到限制且無法輪詢警示。如果在節流期間產生警示，Amazon ECS 可能會遺漏警示，且可能不會發生回復。針對部署沒有其他影響。如需 CloudWatch 服務配額的詳細資訊，請參閱CloudWatch 使用指南中的CloudWatch 服務配額。
- 如果警示在部署開始時處於 `ALARM` 狀態，Amazon ECS 將不會在部署期間監控警示 (Amazon ECS 會忽略警示組態)。此行為可解決您要啟動新部署以修正初始部署失敗的情況。

建議的警示

我們建議您使用下列警示指標：

- 如果您使用 Application Load Balancer，請使用 `HTTPCode_ELB_5XX_Count` 和 `HTTPCode_ELB_4XX_Count` Application Load Balancer 指標。這些指標會檢查 HTTP 峰值。如需有關「Application Load Balancer」度量的詳細資訊，請參閱應用程式負載平衡器使用者指南中的應用程式負載平衡器的指CloudWatch 標。
- 如果您有現有的應用程式，請使用 `CPUUtilization` 和 `MemoryUtilization` 指標。這些指標會檢查叢集或服務使用的 CPU 和記憶體的百分比。如需詳細資訊，請參閱 the section called “考量事項”。
- 如果您在任務中使用 Amazon Simple Queue Service 佇列，請使用 `ApproximateNumberOfMessagesNotVisible` Amazon SQS 指標。此指標會檢查佇列中延遲且無法立即讀取的訊息數量。如需 Amazon SQS 指標的詳細資訊，請參閱 Amazon 簡單佇列服務開發人員 CloudWatch 指南中的 Amazon SQS 可用指標。

在部署之前驗證 Amazon ECS 服務的狀態

藍/綠部署類型使用由控制的藍/綠部署模型。 CodeDeploy使用此部署類型在傳送生產流量到服務之前，先驗證服務的新部署。如需詳細資訊，請參閱AWS CodeDeploy 使用者指南 CodeDeploy中的內容。在部署之前驗證 Amazon ECS 服務的狀態。

在藍/綠部署期間，流量有三種轉移方式：

- 金絲雀 — 流量以兩種增量轉移。您可以從預先定義的 canary 選項中選擇，這會指定流量轉移至您於第一次增量時更新之任務集的百分比，以及在剩餘流量於第二次增量前轉移的間隔 (以分鐘計)。

- 線性 — 流量以相等的增量移動，每個增量之間的分鐘數相等。您可從預先指定的線性選項中指定每次增量的流量轉移百分比例，以及在每個增量之間的分鐘數。
- 一次性全部 — 所有流量都會一次從原始任務集轉移到更新的任務集。

以下是 Amazon ECS 在服務使用藍/綠部署類型時使用的 CodeDeploy 元件：

CodeDeploy 應用

CodeDeploy 資源的集合。這包含一或多個部署群組。

CodeDeploy 部署群組

部署設定。這包含下列各項：

- Amazon ECS 叢集和服務
- 負載平衡器目標群組和接聽器資訊
- 部署復原策略
- 流量重新路由設定
- 原始修訂終止設定
- 部署組態
- CloudWatch 可設定以停止部署的警示組態
- 用於通知的 SNS 或 CloudWatch 事件設定

如需詳細資訊，請參閱 AWS CodeDeploy 使用者指南中的使用部署群組。

CodeDeploy 部署組態

指定部署期間如何將生產流量 CodeDeploy 路由至取代工作集。以下預先定義的線性和金絲雀部署組態可供使用。您也可以建立自訂定義的線性和金絲雀部署。如需詳細資訊，請參閱 AWS CodeDeploy 使用者指南中的使用部署組態。

- CodeDeployDefault.ECS AllAt 一次：一次將所有流量轉移到更新後的 Amazon ECS 容器
- CodeDeployDefault.ECS 10 PercentEvery 1 分鐘：每分鐘移動 10% 的流量，直到所有流量轉移為止。
- CodeDeployDefault.ECS 10 PercentEvery 3 分鐘：每 3 分鐘移動 10% 的流量，直到所有流量轉移為止。
- CodeDeployDefault.ECS 百分之十 5 分鐘：在第一個增量中移動 10% 的流量。剩餘的 90% 會在五分鐘之後部署。

- CodeDeploy預設。百分之十 15 分鐘：在第一個增量中移動 10% 的流量。剩餘的 90% 會在 15 分鐘之後部署。

修訂

修訂版是 CodeDeploy 應用程式規格檔案 (AppSpec 檔案)。在 AppSpec 檔案中，您可以指定工作定義的完整 ARN，以及取代工作集的容器和連接埠，以便在建立新部署時路由流量。容器名稱必須是在您的任務定義中參考的其中一個容器名稱。如果服務定義中的網路組態或平台版本已更新，您也必須在 AppSpec檔案中指定這些詳細資料。您也可以指定要在部署生命週期事件期間執行的 Lambda 函數。Lambda 函數可讓您在部署期間執行測試和傳回指標。若要取得更多資訊，請參閱 AWS CodeDeploy 使用指南中的〈AppSpec 檔案參考〉。

考量事項

使用藍/綠部署類型時，請考慮下列各項：

- 一開始建立使用藍/綠部署類型的 Amazon ECS 服務時，會建立 Amazon ECS 任務集。
- 您必須設定服務，以使用 Application Load Balancer 或 Network Load Balancer。以下是負載平衡器需求：
  - 您必須將生產接聽器新增到負載平衡器 (用於路由傳送生產流量)。
  - 可以將選用的測試接聽器新增到負載平衡器，後者用於路由測試流量。如果您指定測試接聽程式，則會將測試流量 CodeDeploy 路由至部署期間的取代工作集。
  - 生產和測試接聽器必須屬於相同的負載平衡器。
  - 您必須為負載平衡器定義目標群組。目標群組會將流量透過生產接聽器路由到服務中的原始任務集。
  - 當使用 Network Load Balancer 時，僅支援 `CodeDeployDefault.ECSAllAtOnce` 部署組態。
- 對於被設定為使用服務自動擴展和藍/綠部署類型的服務，自動擴展不會在部署期間被封鎖，但部署可能在某些情況下失敗。以下詳細說明此行為。
  - 如果服務正在擴展並啟動部署，則 CodeDeploy 會建立綠色工作集，並等待最多一個小時，讓綠色工作集達到穩定狀態，而且在執行之前不會轉移任何流量。
  - 如果服務正在進行藍/綠部署而發生擴展事件，流量將持續轉移 5 分鐘。如果服務在 5 分鐘內未達到穩定狀態， CodeDeploy 將停止部署並將其標記為失敗。
  - 如果服務正在進行藍/綠部署且發生擴展事件，所需任務計數可能會設定為意外值。這是由於自動擴展將正在執行的任務計數視為當前容量，而其是所需任務計數計算中所使用的適當任務數的兩倍。
- 使用 Fargate 啟動類型或 `CODE_DEPLOY` 部署控制器類型的任務，不支援 `DAEMON` 排程策略。

- 當您一開始建立 CodeDeploy 應用程式和部署群組時，必須指定下列項目：
  - 您必須為負載平衡器定義兩個目標群組。一個目標群組必須是建立 Amazon ECS 服務時為負載平衡器定義的初始目標群組。第二個目標群組的唯一要求是，它無法與服務所使用的負載平衡器以外的其他負載平衡器相關聯。
- 當您為 Amazon ECS 服務建立 CodeDeploy 部署時， CodeDeploy 會在部署中建立替代任務集 (或綠色任務集)。如果您將測試接聽程式新增至負載平衡器，請將測試流量 CodeDeploy 路由至取代工作集。此時您可以執行任何驗證測試。然後，根據部署群組的流量 CodeDeploy重新路由傳送設定，將生產流量從原始工作集重新路由傳送至取代工作集。

所需的 IAM 許可

透過 Amazon ECS 和 CodeDeploy API 的組合，可以實現藍色/綠色部署。使用者必須擁有這些服務的適當許可，才能在 AWS Management Console 或 SDK 中使用 Amazon ECS 藍/綠部署。 AWS CLI

除了建立和更新服務的標準 IAM 許可之外，Amazon ECS 也需要下列許可。已將這些權限新增至 `AmazonECS_FullAccess` IAM 政策。如需詳細資訊，請參閱 亞馬遜 FullAccess。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "codedeploy:CreateApplication",
                "codedeploy:CreateDeployment",
                "codedeploy:CreateDeploymentGroup",
                "codedeploy:GetApplication",
                "codedeploy:GetDeployment",
                "codedeploy:GetDeploymentGroup",
                "codedeploy:ListApplications",
                "codedeploy:ListDeploymentGroups",
                "codedeploy:ListDeployments",
                "codedeploy:StopDeployment",
                "codedeploy:GetDeploymentTarget",
                "codedeploy:ListDeploymentTargets",
                "codedeploy:GetDeploymentConfig",
                "codedeploy:GetApplicationRevision",
                "codedeploy:RegisterApplicationRevision",
                "codedeploy:BatchGetApplicationRevisions",
                "codedeploy:BatchGetDeploymentGroups",
                "codedeploy:BatchGetDeployments",
```

```
                "codedeploy:BatchGetApplications",
                "codedeploy:ListApplicationRevisions",
                "codedeploy:ListDeploymentConfigs",
                "codedeploy:ContinueDeployment",
                "sns:ListTopics",
                "cloudwatch:DescribeAlarms",
                "lambda:ListFunctions"
            ],
            "Resource": ["*"]
        }
    ]
}
```

> **Note**
>
> 除了執行任務和服務所需的標準 Amazon ECS 許可之外，使用者還需要 `iam:PassRole` 許可，才能使用任務的 IAM 角色。

CodeDeploy 需要許可才能呼叫 Amazon ECS API、修改 Elastic Load Balancing、叫用 Lambda 函數和描述 CloudWatch 警示，以及代表您修改服務所需計數的許可。建立使用藍/綠部署類型的 Amazon ECS 服務之前，您必須建立一個 IAM 角色 (`ecsCodeDeployRole`)。如需詳細資訊，請參閱 Amazon ECS CodeDeploy IAM 角色。

建立 Amazon ECS 服務範例 和 更新 Amazon ECS 服務示例 IAM 政策範例顯示使用者在 AWS Management Console上使用 Amazon ECS 藍/綠部署時所需的許可。

使用藍色/綠色部署部署Amazon ECS 服務

了解如何建立包含使用藍/綠部署類型的 Fargate 任務的 Amazon ECS 服務。 AWS CLI

> **Note**
>
> 已為 AWS CloudFormation新增對於執行藍/綠部署的支援。如需詳細資訊，請參閱使用AWS CloudFormation 者指南 AWS CloudFormation中的 CodeDeploy 使用執行 Amazon ECS 藍/綠部署。

必要條件

本教學課程假設已完成下列先決條件：

- 已安裝並設定最新版本的。 AWS CLI 如需有關安裝或升級的詳細資訊 AWS CLI，請參閱安裝 AWS Command Line Interface.
- 已完成「設定以使用 Amazon ECS。」中的步驟。
- 您的 AWS 使用者具有 亞馬遜 FullAccess IAM 政策範例中指定的必要許可。
- 您已建立 VPC 和安全群組。如需詳細資訊，請參閱 the section called “建立 Virtual Private Cloud”。
- 隨即建立 Amazon ECS CodeDeploy IAM 角色。如需詳細資訊，請參閱 Amazon ECS CodeDeploy IAM 角色。

步驟 1：建立 Application Load Balancer

使用藍/綠部署類型的 Amazon ECS 服務需要使用 Application Load Balancer 或 Network Load Balancer。本教學課程會使用 Application Load Balancer。

建立 Application Load Balancer

1. 使用 create-load-balancer 命令來建立 Application Load Balancer。指定並非來自與安全群組相同可用區域的兩個子網路。

```
aws elbv2 create-load-balancer \
     --name bluegreen-alb \
     --subnets subnet-abcd1234 subnet-abcd5678 \
     --security-groups sg-abcd1234 \
     --region us-east-1
```

   其輸出將包含負載平衡器的 Amazon Resource Name (ARN)，格式如下：

```
arn:aws:elasticloadbalancing:region:aws_account_id:loadbalancer/app/bluegreen-alb/
e5ba62739c16e642
```

2. 使用 create-target-group 命令建立目標群組。此目標群組會將流量路由到服務中的原始任務集。

```
aws elbv2 create-target-group \
     --name bluegreentarget1 \
     --protocol HTTP \
     --port 80 \
     --target-type ip \
     --vpc-id vpc-abcd1234 \
     --region us-east-1
```

其輸出將包含目標群組的 ARN，格式如下：

```
arn:aws:elasticloadbalancing:region:aws_account_id:targetgroup/
bluegreentarget1/209a844cd01825a4
```

3. 使用 create-listener 命令建立具有預設規則以將請求轉送至目標群組的負載平衡接聽程式。

```
aws elbv2 create-listener \
     --load-balancer-arn
 arn:aws:elasticloadbalancing:region:aws_account_id:loadbalancer/app/bluegreen-alb/
e5ba62739c16e642 \
     --protocol HTTP \
     --port 80 \
     --default-actions
 Type=forward,TargetGroupArn=arn:aws:elasticloadbalancing:region:aws_account_id:targetgroup
bluegreentarget1/209a844cd01825a4 \
     --region us-east-1
```

其輸出將包含接聽程式的 ARN，格式如下：

```
arn:aws:elasticloadbalancing:region:aws_account_id:listener/app/bluegreen-alb/
e5ba62739c16e642/665750bec1b03bd4
```

步驟 2：建立 Amazon ECS 叢集。

使用 create-cluster 命令來建立要使用之名為 `tutorial-bluegreen-cluster` 的叢集。

```
aws ecs create-cluster \
     --cluster-name tutorial-bluegreen-cluster \
     --region us-east-1
```

其輸出將包含叢集的 ARN，格式如下：

```
arn:aws:ecs:region:aws_account_id:cluster/tutorial-bluegreen-cluster
```

步驟 3：註冊任務定義

使用 register-task-definition 命令來註冊與 Fargate 相容的任務定義。這需要使用 `awsvpc` 網路模式。以下是用於此教學課程的任務定義範例。

首先，建立名為 fargate-task.json 且具有下列內容的檔案。確保您使用的是任務執行角色的 ARN。如需詳細資訊，請參閱 Amazon ECS 任務執行 IAM 角色。

```
{
    "family": "tutorial-task-def",
        "networkMode": "awsvpc",
        "containerDefinitions": [
            {
                "name": "sample-app",
                "image": "httpd:2.4",
                "portMappings": [
                    {
                        "containerPort": 80,
                        "hostPort": 80,
                        "protocol": "tcp"
                    }
                ],
                "essential": true,
                "entryPoint": [
                    "sh",
                    "-c"
                ],
                "command": [
                    "/bin/sh -c \"echo '<html> <head> <title>Amazon ECS Sample
 App</title> <style>body {margin-top: 40px; background-color: #00FFFF;} </style> </
head><body> <div style=color:white;text-align:center> <h1>Amazon ECS Sample App</h1>
 <h2>Congratulations!</h2> <p>Your application is now running on a container in Amazon
 ECS.</p> </div></body></html>' >  /usr/local/apache2/htdocs/index.html && httpd-
foreground\""
                ]
            }
        ],
        "requiresCompatibilities": [
            "FARGATE"
        ],
        "cpu": "256",
        "memory": "512",
        "executionRoleArn": "arn:aws:iam::aws_account_id:role/ecsTaskExecutionRole"
}
```

接下來，使用您建立的 fargate-task.json 檔案註冊任務定義。

```
aws ecs register-task-definition \
```

```
    --cli-input-json file://fargate-task.json \
    --region us-east-1
```

步驟 4：建立 Amazon ECS 服務

使用 create-service 命令來建立服務。

首先，建立名為 service-bluegreen.json 且具有下列內容的檔案。

```
{
    "cluster": "tutorial-bluegreen-cluster",
    "serviceName": "service-bluegreen",
    "taskDefinition": "tutorial-task-def",
    "loadBalancers": [
        {
            "targetGroupArn":
 "arn:aws:elasticloadbalancing:region:aws_account_id:targetgroup/
bluegreentarget1/209a844cd01825a4",
            "containerName": "sample-app",
            "containerPort": 80
        }
    ],
    "launchType": "FARGATE",
    "schedulingStrategy": "REPLICA",
    "deploymentController": {
        "type": "CODE_DEPLOY"
    },
    "platformVersion": "LATEST",
    "networkConfiguration": {
       "awsvpcConfiguration": {
          "assignPublicIp": "ENABLED",
          "securityGroups": [ "sg-abcd1234" ],
          "subnets": [ "subnet-abcd1234", "subnet-abcd5678" ]
       }
    },
    "desiredCount": 1
}
```

接下來，使用您建立的 service-bluegreen.json 檔案建立服務。

```
aws ecs create-service \
    --cli-input-json file://service-bluegreen.json \
    --region us-east-1
```

其輸出將包含服務的 ARN，格式如下：

```
arn:aws:ecs:region:aws_account_id:service/service-bluegreen
```

使用下列命令取得負載平衡器的 DNS 名稱。

```
aws elbv2 describe-load-balancers --name bluegreen-alb  --query
'LoadBalancers[*].DNSName'
```

在 Web 瀏覽器中輸入 DNS 名稱，您應該會看到一個網頁，其中顯示具有藍色背景的範例應用程式。

步驟 5：建立 AWS CodeDeploy 資源

使用下列步驟建立 CodeDeploy 應用程式、 CodeDeploy 部署群組的「Application Load Balancer」目標群組，以及部 CodeDeploy 署群組。

若要建立 CodeDeploy 資源

1. 使用「建立應用程式」指令建立應用程式。 CodeDeploy 指定 ECS 運算平台。

   ```
   aws deploy create-application \
        --application-name tutorial-bluegreen-app \
        --compute-platform ECS \
        --region us-east-1
   ```

   其輸出將包含應用程式 ID，格式如下：

   ```
   {
       "applicationId": "b8e9c1ef-3048-424e-9174-885d7dc9dc11"
   }
   ```

2. 使用「建立-目標群組」命令建立第二個「Application Load Balancer」目標群組，此群組將在建立部署群組時使用。 CodeDeploy

   ```
   aws elbv2 create-target-group \
        --name bluegreentarget2 \
        --protocol HTTP \
        --port 80 \
        --target-type ip \
        --vpc-id "vpc-0b6dd82c67d8012a1" \
        --region us-east-1
   ```

其輸出將包含目標群組的 ARN，格式如下：

```
arn:aws:elasticloadbalancing:region:aws_account_id:targetgroup/
bluegreentarget2/708d384187a3cfdc
```

3. 使用「建立-部署群組」命令建立部署群組。 CodeDeploy

首先，建立名為 tutorial-deployment-group.json 且具有下列內容的檔案。此範例會使用您建立的資源。對於serviceRoleArn，請指定您的 Amazon ECS CodeDeploy IAM 角色的 ARN。如需詳細資訊，請參閱 Amazon ECS CodeDeploy IAM 角色。

```
{
   "applicationName": "tutorial-bluegreen-app",
   "autoRollbackConfiguration": {
      "enabled": true,
      "events": [ "DEPLOYMENT_FAILURE" ]
   },
   "blueGreenDeploymentConfiguration": {
      "deploymentReadyOption": {
         "actionOnTimeout": "CONTINUE_DEPLOYMENT",
         "waitTimeInMinutes": 0
      },
      "terminateBlueInstancesOnDeploymentSuccess": {
         "action": "TERMINATE",
         "terminationWaitTimeInMinutes": 5
      }
   },
   "deploymentGroupName": "tutorial-bluegreen-dg",
   "deploymentStyle": {
      "deploymentOption": "WITH_TRAFFIC_CONTROL",
      "deploymentType": "BLUE_GREEN"
   },
   "loadBalancerInfo": {
      "targetGroupPairInfoList": [
        {
          "targetGroups": [
             {
                 "name": "bluegreentarget1"
             },
             {
                 "name": "bluegreentarget2"
             }
```

```
            ],
            "prodTrafficRoute": {
                "listenerArns": [
                    "arn:aws:elasticloadbalancing:region:aws_account_id:listener/
app/bluegreen-alb/e5ba62739c16e642/665750bec1b03bd4"
                ]
            }
        }
      ]
   },
   "serviceRoleArn": "arn:aws:iam::aws_account_id:role/ecsCodeDeployRole",
   "ecsServices": [
       {
           "serviceName": "service-bluegreen",
           "clusterName": "tutorial-bluegreen-cluster"
       }
   ]
}
```

然後建立部 CodeDeploy 署群組。

```
aws deploy create-deployment-group \
     --cli-input-json file://tutorial-deployment-group.json \
     --region us-east-1
```

其輸出將包含部署群組 ID，格式如下：

```
{
    "deploymentGroupId": "6fd9bdc6-dc51-4af5-ba5a-0a4a72431c88"
}
```

步驟 6：建立和監視 CodeDeploy部署

在建立 CodeDeploy 部署之前，請command在中更新工作定義，fargate-task.json如下所示，將範例應用程式背景顏色變更為綠色。

```
{
    ...
        "containerDefinitions": [
            {
                ...
```

```
                "command": [
                    "/bin/sh -c \"echo '<html> <head> <title>Amazon ECS Sample
 App</title> <style>body {margin-top: 40px; background-color: #097969;} </style> </
head><body> <div style=color:white;text-align:center> <h1>Amazon ECS Sample App</h1>
 <h2>Congratulations!</h2> <p>Your application is now running on a container in Amazon
 ECS.</p> </div></body></html>' >  /usr/local/apache2/htdocs/index.html && httpd-
foreground\""
                ]
            }
        ],
        ...
}
```

使用以下命令註冊更新的任務定義。

```
aws ecs register-task-definition \
     --cli-input-json file://fargate-task.json \
     --region us-east-1
```

現在，請使用下列步驟來建立並上傳應用程式規格AppSpec 檔案 (檔案) 和 CodeDeploy 部署。

若要建立和監視 CodeDeploy 部署

1. 使用以下步驟創建並上傳 AppSpec 文件。

   a. 建立`appspec.yaml`以 CodeDeploy 部署群組內容命名的檔案。此範例使用更新的任務定義。

   ```
   version: 0.0
   Resources:
     - TargetService:
         Type: AWS::ECS::Service
         Properties:
           TaskDefinition: "arn:aws:ecs:region:aws_account_id:task-
   definition/tutorial-task-def:2"
           LoadBalancerInfo:
             ContainerName: "sample-app"
             ContainerPort: 80
           PlatformVersion: "LATEST"
   ```

   b. 使用 s3 mb 命令為 AppSpec 檔案建立 Amazon S3 儲存貯體。

```
aws s3 mb s3://tutorial-bluegreen-bucket
```

c. 使用 s3 cp 命令將 AppSpec 檔案上傳到 Amazon S3 儲存貯體。

```
aws s3 cp ./appspec.yaml s3://tutorial-bluegreen-bucket/appspec.yaml
```

2. 使用下列步驟建立 CodeDeploy 部署。

a. 建立`create-deployment.json`以 CodeDeploy 部署內容命名的檔案。此範例會使用您先前在教學課程中建立的資源。

```
{
    "applicationName": "tutorial-bluegreen-app",
    "deploymentGroupName": "tutorial-bluegreen-dg",
    "revision": {
        "revisionType": "S3",
        "s3Location": {
            "bucket": "tutorial-bluegreen-bucket",
            "key": "appspec.yaml",
            "bundleType": "YAML"
        }
    }
}
```

b. 使用 create-deployment 命令來建立部署。

```
aws deploy create-deployment \
     --cli-input-json file://create-deployment.json \
     --region us-east-1
```

其輸出將包含部署 ID，格式如下：

```
{
    "deploymentId": "d-RPCR1U3TW"
}
```

3. 使用 get-deployment-target 命令，並從先前的輸出中指定 `deploymentId` 來取得部署的詳細資訊。

```
aws deploy get-deployment-target \
--deployment-id "d-IMJU3A8TW" \
```

```
--target-id tutorial-bluegreen-cluster:service-bluegreen \
--region us-east-1
```

最初，部署狀態為 InProgress。流量會導向至原始任務集，該任務集的 taskSetLabel 為 BLUE，狀態為 PRIMARY，且 trafficWeight 為 100.0。取代任務集的 taskSetLabel 為 GREEN，狀態為 ACTIVE，且 trafficWeight 為 0.0。您輸入 DNS 名稱的 Web 瀏覽器仍會以藍色背景顯示範例應用程式。

```
{
"deploymentTarget": {
"deploymentTargetType": "ECSTarget",
"ecsTarget": {
    "deploymentId": "d-RPCR1U3TW",
    "targetId": "tutorial-bluegreen-cluster:service-bluegreen",
    "targetArn": "arn:aws:ecs:region:aws_account_id:service/service-bluegreen",
    "lastUpdatedAt": "2023-08-10T12:07:24.797000-05:00",
    "lifecycleEvents": [
        {
            "lifecycleEventName": "BeforeInstall",
            "startTime": "2023-08-10T12:06:22.493000-05:00",
            "endTime": "2023-08-10T12:06:22.790000-05:00",
            "status": "Succeeded"
        },
        {
            "lifecycleEventName": "Install",
            "startTime": "2023-08-10T12:06:22.936000-05:00",
            "status": "InProgress"
        },
        {
            "lifecycleEventName": "AfterInstall",
            "status": "Pending"
        },
        {
            "lifecycleEventName": "BeforeAllowTraffic",
            "status": "Pending"
        },
        {
            "lifecycleEventName": "AllowTraffic",
            "status": "Pending"
        },
        {
            "lifecycleEventName": "AfterAllowTraffic",
```

```
            "status": "Pending"
        }
    ],
    "status": "InProgress",
    "taskSetsInfo": [
        {
            "identifer": "ecs-svc/9223370493423413672",
            "desiredCount": 1,
            "pendingCount": 0,
            "runningCount": 1,
            "status": "ACTIVE",
            "trafficWeight": 0.0,
            "targetGroup": {
                "name": "bluegreentarget2"
            },
            "taskSetLabel": "Green"
        },
        {
            "identifer": "ecs-svc/9223370493425779968",
            "desiredCount": 1,
            "pendingCount": 0,
            "runningCount": 1,
            "status": "PRIMARY",
            "trafficWeight": 100.0,
            "targetGroup": {
                "name": "bluegreentarget1"
            },
            "taskSetLabel": "Blue"
        }
    ]
}
}
}
```

繼續使用命令擷取部署詳細資訊，直到部署狀態為 Succeeded，如下列輸出所示。流量現在會重新導向至取代任務集，該任務集現在的狀態為 PRIMARY 且 trafficWeight 為 100.0。重新整理您輸入負載平衡器 DNS 名稱的 Web 瀏覽器，現在您應該會看到背景為綠色的範例應用程式。

```
{
"deploymentTarget": {
"deploymentTargetType": "ECSTarget",
"ecsTarget": {
    "deploymentId": "d-RPCR1U3TW",
```

```
    "targetId": "tutorial-bluegreen-cluster:service-bluegreen",
    "targetArn": "arn:aws:ecs:region:aws_account_id:service/service-bluegreen",
    "lastUpdatedAt": "2023-08-10T12:07:24.797000-05:00",
    "lifecycleEvents": [
        {
            "lifecycleEventName": "BeforeInstall",
            "startTime": "2023-08-10T12:06:22.493000-05:00",
            "endTime": "2023-08-10T12:06:22.790000-05:00",
            "status": "Succeeded"
        },
        {
            "lifecycleEventName": "Install",
            "startTime": "2023-08-10T12:06:22.936000-05:00",
            "endTime": "2023-08-10T12:08:25.939000-05:00",
            "status": "Succeeded"
        },
        {
            "lifecycleEventName": "AfterInstall",
            "startTime": "2023-08-10T12:08:26.089000-05:00",
            "endTime":  "2023-08-10T12:08:26.403000-05:00",
            "status": "Succeeded"
        },
        {
            "lifecycleEventName": "BeforeAllowTraffic",
            "startTime": "2023-08-10T12:08:26.926000-05:00",
            "endTime":  "2023-08-10T12:08:27.256000-05:00",
            "status": "Succeeded"
        },
        {
            "lifecycleEventName": "AllowTraffic",
            "startTime": "2023-08-10T12:08:27.416000-05:00",
            "endTime": "2023-08-10T12:08:28.195000-05:00",
            "status": "Succeeded"
        },
        {
            "lifecycleEventName": "AfterAllowTraffic",
            "startTime": "2023-08-10T12:08:28.715000-05:00",
            "endTime":  "2023-08-10T12:08:28.994000-05:00",
            "status": "Succeeded"
        }
    ],
    "status": "Succeeded",
    "taskSetsInfo": [
        {
```

```
            "identifer": "ecs-svc/9223370493425779968",
            "desiredCount": 1,
            "pendingCount": 0,
            "runningCount": 1,
            "status": "ACTIVE",
            "trafficWeight": 0.0,
            "targetGroup": {
                "name": "bluegreentarget1"
            },
            "taskSetLabel": "Blue"
        },
        {
            "identifer": "ecs-svc/9223370493423413672",
            "desiredCount": 1,
            "pendingCount": 0,
            "runningCount": 1,
            "status": "PRIMARY",
            "trafficWeight": 100.0,
            "targetGroup": {
                "name": "bluegreentarget2"
            },
            "taskSetLabel": "Green"
        }
    ]
}
}
}
```

步驟 7：清除

完成此教學課程時，清除與其相關的資源，以免未使用的資源產生費用。

清除教學課程資源

1. 使用 [刪除部署群組] 命令刪除部署群組。 CodeDeploy

```
aws deploy delete-deployment-group \
     --application-name tutorial-bluegreen-app \
     --deployment-group-name tutorial-bluegreen-dg \
     --region us-east-1
```

2. 使用刪除應用程式命令刪除應用程式。 CodeDeploy

```
aws deploy delete-application \
     --application-name tutorial-bluegreen-app \
     --region us-east-1
```

3. 使用 delete-service 命令來刪除 Amazon ECS 服務。使用 `--force` 旗標可讓您即使在服務未縮減為零個任務時仍能將其刪除。

```
aws ecs delete-service \
     --service arn:aws:ecs:region:aws_account_id:service/service-bluegreen \
     --force \
     --region us-east-1
```

4. 使用 delete-cluster 命令來刪除 Amazon ECS 叢集。

```
aws ecs delete-cluster \
     --cluster tutorial-bluegreen-cluster \
     --region us-east-1
```

5. 使用 s3 rm 命令從 Amazon S3 儲存貯體刪除 AppSpec 檔案。

```
aws s3 rm s3://tutorial-bluegreen-bucket/appspec.yaml
```

6. 使用 s3 rb 命令來刪除 Amazon S3 儲存貯體。

```
aws s3 rb s3://tutorial-bluegreen-bucket
```

7. 使用 delete-load-balancer 命令刪除 Application Load Balancer。

```
aws elbv2 delete-load-balancer \
     --load-balancer-arn
 arn:aws:elasticloadbalancing:region:aws_account_id:loadbalancer/app/bluegreen-alb/
e5ba62739c16e642 \
     --region us-east-1
```

8. 使用 delete-target-group 命令，刪除兩個 Application Load Balancer 目標群組。

```
aws elbv2 delete-target-group \
     --target-group-arn
 arn:aws:elasticloadbalancing:region:aws_account_id:targetgroup/
bluegreentarget1/209a844cd01825a4 \
     --region us-east-1
```

```
aws elbv2 delete-target-group \
     --target-group-arn
 arn:aws:elasticloadbalancing:region:aws_account_id:targetgroup/
bluegreentarget2/708d384187a3cfdc \
     --region us-east-1
```

使用第三方控制器部署 Amazon ECS 服務

外部部署類型可讓您使用任何第三方部署控制器，以完全控制 Amazon ECS 服務的部署程序。服務的詳細資訊是由服務管理 API 動作 (CreateService、UpdateService 和 DeleteService) 或任務設定管理 API 動作 (CreateTaskSet、UpdateTaskSet、UpdateServicePrimaryTaskSet 和 DeleteTaskSet) 所管理。每個 API 動作會管理服務定義參數的子集。

UpdateService API 動作更新服務的所需計數和運作狀態檢查寬限期參數。如果需要更新啟動類型、平台版本、負載平衡器詳細資訊、網路組態或任務定義，您必須建立新的任務集。

UpdateTaskSet API 動作只更新任務集的擴展參數。

UpdateServicePrimaryTaskSet API 動作修改服務中哪個任務集是主要任務集。當您呼叫 DescribeServices API 動作時，它會傳回主要任務集指定的所有欄位。如果更新服務的主要任務集，當定義新的主要任務時，存在於新主要任務集的任何任務集參數值 (不同於服務中的舊主要任務集) 將會更新為新的值。如果沒有為服務定義主要任務集，則描述服務時，任務集欄位是 null。

外部部署考量

使用外部部署類型時，請考慮下列各項：

- 支援的負載平衡器類型可為 Application Load Balancer 或 Network Load Balancer。
- Fargate 啟動類型或 EXTERNAL 部署控制器類型不支援 DAEMON 排程策略。

外部部署工作流程

以下是在 Amazon ECS 上管理外部部署的基本工作流程。

使用外部部署控制器管理 Amazon ECS 服務

1. 建立 Amazon ECS 服務。唯一的必要參數是服務名稱。使用外部部署控制器建立服務時，您可以指定以下參數。在服務內建立任務集時，指定所有其他服務參數。

`serviceName`

類型：字串

必要：是

您的服務名稱。可以包含最多可達 255 個字元 (大小寫)、數字、連字號和底線。叢集中不得有相同的服務名稱，但一個區域內或多個區域間的多個叢集中可以有類似的服務名稱。

`desiredCount`

指定的任務集任務定義的執行個體化數量，將放在服務內保持執行。

`deploymentConfiguration`

選用的部署參數，可控制在部署期間執行多少任務以及停止和啟動任務的順序。

`tags`

類型： 物件陣列

必要：否

您套用到服務以協助您分類和組織的中繼資料。每個標籤皆包含由您定義的一個金鑰與一個選用值。刪除服務時，也會一併刪除標籤。服務最多可套用 50 個標籤。如需詳細資訊，請參閱 標記 Amazon ECS 資源。

`key`

類型：字串

長度限制：長度下限為 1。長度上限為 128。

必要：否

組成標籤的鍵值組的一部分。索引鍵是一般標籤，作用就像更特定標籤值的類別。

`value`

類型：字串

長度限制：長度下限為 0。長度上限為 256。

必要：否

組成標籤的鍵值組的選用部分。值就像標籤類別 (索引鍵) 內的描述項。

`enableECSManagedTags`

指定是否對服務內的使用 Amazon ECS 受管標籤。如需詳細資訊，請參閱 使用標籤進行帳單。

`propagateTags`

類型：字串

有效值：`TASK_DEFINITION` | `SERVICE`

必要：否

指定是否將標籤從任務定義或服務複製到服務中的任務。如果沒有指定值，則不會複製標籤。標籤只能在建立服務期間複製到服務內的任務。若要在建立服務或任務後對任務新增標籤，請使用 `TagResource` API 動作。

`schedulingStrategy`

使用的排程策略。使用外部部署控制器的服務僅支援 `REPLICA` 排程策略。

`placementConstraints`

您服務的任務要使用的置放限制物件陣列。每項任務您最多可以指定 10 項限制 (此限制包含任務定義中的限制以及執行時間指定的限制)。若您使用的是 Fargate 啟動類型，則不支援任務置放限制條件。

`placementStrategy`

您服務的任務要使用的置放策略物件。每項服務您最多可以指定四項策略規則。

以下是使用外部部署控制器建立服務的服務定義範例。

```
{
    "cluster": "",
    "serviceName": "",
    "desiredCount": 0,
    "role": "",
    "deploymentConfiguration": {
        "maximumPercent": 0,
        "minimumHealthyPercent": 0
    },
```

```
    "placementConstraints": [
        {
            "type": "distinctInstance",
            "expression": ""
        }
    ],
    "placementStrategy": [
        {
            "type": "binpack",
            "field": ""
        }
    ],
    "schedulingStrategy": "REPLICA",
    "deploymentController": {
        "type": "EXTERNAL"
    },
    "tags": [
        {
            "key": "",
            "value": ""
        }
    ],
    "enableECSManagedTags": true,
    "propagateTags": "TASK_DEFINITION"
}
```

2. 建立初始任務集。任務集包含以下關於服務的詳細資訊：

   `taskDefinition`

   供任務集的任務使用的任務定義。

   `launchType`

   類型：字串

   有效值：EC2 | FARGATE | EXTERNAL

   必要：否

   您服務執行所在的啟動類型。如果未指定啟動類型，預設會使用預設的 `capacityProviderStrategy`。如需詳細資訊，請參閱 Amazon ECS 啟動類型。

   若有指定 `launchType`，則必須省略 `capacityProviderStrategy` 參數。

`platformVersion`

類型：字串

必要：否

您服務中任務正在其上執行的平台版本。平台版本僅會為使用 Fargate 啟動類型的任務指定。如果尚未指定，預設會使用最新版本 (LATEST)。

AWS Fargate 平台版本用於指 Fargate 任務基礎結構的特定運行時環境。在您執行任務或建立服務時指定 LATEST 平台版本，即可取得任務所能使用的最新平台版本。當您擴展服務規模時，這些任務會收到於服務的目前部署上所指定的平台版本。如需詳細資訊，請參閱 適用於 Amazon ECS 的 Fargate Linux 平台版本。

> **Note**
>
> 並未針對使用 EC2 啟動類型的任務指定平台版本。

`loadBalancers`

用來代表您的服務所使用之負載平衡器的負載平衡器物件。當使用外部部署控制器時，僅支援 Application Load Balancer 和 Network Load Balancer。如果您使用 Application Load Balancer，每個任務集僅允許一個 Application Load Balancer 目標群組。

以下程式碼片段顯示要使用的 `loadBalancer` 物件範例。

```
"loadBalancers": [
        {
            "targetGroupArn": "",
            "containerName": "",
            "containerPort": 0
        }
]
```

> **Note**
>
> 指定 `loadBalancer` 物件時，您必須指定 `targetGroupArn` 並省略 `loadBalancerName` 參數。

`networkConfiguration`

服務的網路組態。使用 `awsvpc` 網路模式的任務定義需要此參數，才能接收其自己的彈性網路介面，其他網路模式不支援此參數。如需有關 Fargate 啟動類型之網路的詳細資訊，請參閱適用於 Fargate 啟動類型的 Amazon ECS 任務聯網選項。

`serviceRegistries`

要指派給此服務的服務探索登錄檔詳細資訊。如需詳細資訊，請參閱 使用服務探索將 Amazon ECS 服務與 DNS 名稱連接。

`scale`

需要放在任務集來保持執行的任務數的浮點百分比。值是以服務的 `desiredCount` 的百分比總計指定。接受的值為 0 到 100 之間的數字。

以下是為外部部署控制器建立任務集的 JSON 範例。

```
{
    "service": "",
    "cluster": "",
    "externalId": "",
    "taskDefinition": "",
    "networkConfiguration": {
        "awsvpcConfiguration": {
            "subnets": [
                ""
            ],
            "securityGroups": [
                ""
            ],
            "assignPublicIp": "DISABLED"
        }
    },
    "loadBalancers": [
        {
            "targetGroupArn": "",
            "containerName": "",
            "containerPort": 0
        }
    ],
```

```
    "serviceRegistries": [
        {
            "registryArn": "",
            "port": 0,
            "containerName": "",
            "containerPort": 0
        }
    ],
    "launchType": "EC2",
    "capacityProviderStrategy": [
        {
            "capacityProvider": "",
            "weight": 0,
            "base": 0
        }
    ],
    "platformVersion": "",
    "scale": {
        "value": null,
        "unit": "PERCENT"
    },
    "clientToken": ""
}
```

3. 需要服務變更時，請根據您更新哪些參數，使用 UpdateService、UpdateTaskSet 或 CreateTaskSet API 動作。如果您建立任務集，請對服務中的每個任務集使用 scale 參數，以決定在服務中要保持執行多少個任務。例如，如果您有一個服務包含 tasksetA 並建立 tasksetB，您可能想要先測試 tasksetB 的有效性，然後才將生產流量轉移給它。您可以將這兩個任務集的 scale 都設為 100，而當您準備好將所有生產流量轉移至 tasksetB 時，您可以將 tasksetA 的 scale 更新為 0 以縮減它。

使用負載平衡來分配 Amazon ECS 服務流量

您可以選擇性地將服務設定為使用「Elastic Load Balancing」，將流量平均分配到服務中的任務。

> **Note**
>
> 使用任務集時，集中的所有任務必須全部設定為使用 Elastic Load Balancing 或不使用 Elastic Load Balancing。

託管在上的 Amazon ECS 服務 AWS Fargate 支援應用程式負載平衡器、網路負載平衡器和閘道負載平衡器。使用下表瞭解要使用的負載平衡器類型。

Load Balancer 類型	在這些情況下使用
Application Load Balancer	路由 HTTP/HTTPS (或第 7 層) 流量。 Application Load Balancer 提供數種功能，非常適合與 Amazon ECS 服務搭配使用： • 每項服務都可以透過指定多個目標群組，為來自多個負載平衡器的流量提供服務，以及公開多個負載平衡的連接埠。 • 它們受在 Fargate 和 EC2 執行個體上託管的任務的支援。 • Application Load Balancer 允許容器使用動態主機連接埠映射 (允許每個容器執行個體之相同服務中的多個任務)。 • Application Load Balancer 支援路徑類型路由和優先順序規則 (因此多個服務可以在單一 Application Load Balancer 上使用相同的接聽程式連接埠)。
Network Load Balancer	路由 TCP 或 UDP (或第 4 層) 流量。
Gateway Load Balancer	路由 TCP 或 UDP (或第 4 層) 流量。

Load Balancer 類型	在這些情況下使用	
	使用虛擬設備，例如防火牆，入侵檢測和防禦系統以及深度數據包檢測系統。	

建議您將應用程式負載平衡器用於 Amazon ECS 服務，以便您可以利用這些最新功能，除非您的服務需要僅適用於網路負載平衡器或閘道負載平衡器的功能。如需有關 Elastic Load Balancing 和負載平衡器類型區別的詳細資訊，請參閱 Elastic Load Balancing 使用者指南。

使用負載平衡器時，您只需按實際用量付費。如需詳細資訊，請參閱 Elastic Load Balancing 定價。

針對 Amazon ECS 優化負載平衡器運作狀態檢查參數

負載平衡器只會將要求路由到負載平衡器的可用區域中運作良好的目標。每個目標都註冊到一個目標群組。負載平衡器會使用目標群組健全狀況檢查設定來檢查每個目標的健全狀況。註冊目標後，它必須通過一次健康檢查才能被視為健康。Amazon ECS 監控負載平衡器。負載平衡器會定期傳送運作狀態檢查至 Amazon ECS 容器。Amazon ECS 代理程式會監控並等待負載平衡器報告容器健康狀態。在它認為容器處於健康狀態之前，它會執行此操作。

兩個 Elastic Load Balancing 健康狀態檢查參數會影響部署速度

- Health 狀態檢查間隔：決定個別容器健康狀態檢查之間的大約時間長度 (以秒為單位)。根據預設，負載平衡器會每隔 30 秒檢查一次。

  此參數的名稱為：

  - `HealthCheckIntervalSeconds`在 Elastic Load Balancing API 中
  - Amazon EC2 主控台上的間隔
- 健康臨界值計數：決定將健康狀況不良的容器視為健康狀況良好之前，所需的連續運作狀況檢查成功次 根據預設，負載平衡器需要通過五次健康狀態檢查，才能報告目標容器狀況良好。

  此參數的名稱為：

  - `HealthyThresholdCount`在 Elastic Load Balancing API 中
  - Amazon EC2 控制台上的健康閾值

使用預設設定時，判斷容器健全狀況的總時間為兩分 30 秒 (`30 seconds * 5 = 150 seconds`)。

如果您的服務在 10 秒內啟動並穩定，則可以加快運行狀態檢查過程。若要加快處理速度，請減少健康狀態檢查的次數和檢查之間的間隔。

- `HealthCheckIntervalSeconds`(Elastic Load Balancing API 名稱) 或間隔 (Amazon EC2 主控台名稱)：5
- `HealthyThresholdCount`(Elastic Load Balancing API 名稱) 或狀態良好閾值 (Amazon EC2 主控台名稱)：2

使用此設定時，健康狀態檢查程序需要 10 秒，而預設值為 2 分 30 秒。

如需有關 Elastic Load Balancing Health 全狀況檢查參數的詳細資訊，請參閱 E lastic Load Balancing 使用者指南中的目標群組健全狀況檢查。

優化 Amazon ECS 的負載平衡器連接排除參數

若要允許最佳化，用戶端會維護與容器服務的「保持作用中」連線。這允許來自該客戶端的後續請求重複使用現有的連接。當您想要停止容器的流量時，請通知負載平衡器。負載平衡器會定期檢查用戶端是否已關閉「保持作用中」連線。Amazon ECS 代理程式監控負載平衡器，並等待負載平衡器報告保持作用連線已關閉 (目標處於UNUSED狀態)。

負載平衡器等待將目標移至狀UNUSED態的時間量為取消註冊延遲。您可以設定下列負載平衡器參數來加速部署。

- `deregistration_delay.timeout_seconds`：300 (預設值)

當您的服務的回應時間不到一秒鐘時，請將參數設定為下列值，讓負載平衡器在中斷用戶端與後端服務之間的連線之前，僅等待 5 秒鐘：

- `deregistration_delay.timeout_seconds`：5

> **Note**
>
> 如果服務具有長期要求 (例如檔案上傳速度慢或串流連線)，請勿將值設定為 5 秒。

回應速度

Amazon ECS 會先傳送 SIGTERM 訊號至任務，以通知應用程式需要完成和關閉。然後，Amazon ECS 發送 SIGKILL 消息。當應用程式忽略 SIGTERM 時，Amazon ECS 服務必須等待傳送 SIGKILL 訊號才能終止程序。

Amazon ECS 等待發送 SIGKILL 消息的時間長度由以下 Amazon ECS 代理程式選項決定：

- `ECS_CONTAINER_STOP_TIMEOUT`：30 (預設值)

  如需容器代理程式參數的詳細資訊，請參閱上 GitHub的 Amazon ECS 容器代理程式。

若要加快等待期間，請將 Amazon ECS 代理程式參數設定為下列值：

> **Note**
>
> 如果您的應用程式耗時超過 1 秒，請將該值乘以 2，並使用該數字做為值。

- `ECS_CONTAINER_STOP_TIMEOUT`: 2

在這種情況下，Amazon ECS 會等待容器關閉 2 秒鐘，然後當應用程式沒有停止時，Amazon ECS 會傳送 SIGKILL 訊息。

您也可以修改應用程式程式碼，以捕獲 SIGTERM 訊號並對其做出反應。以下是中的範例 JavaScript：

```
process.on('SIGTERM', function() {
  server.close();
})
```

此程式碼會導致 HTTP 伺服器停止接聽任何新要求、完成回應任何執行中要求，然後 Node.js 處理序終止。這是因為它的事件循環沒有什麼可做的。鑑於此，如果進程僅需 500 毫秒即可完成其運行中請求，則它會提前終止，而不必等待停止超時並發送 SIGKILL。

使用 Amazon ECS 的 Application Load Balancer

Application Load Balancer 在應用程式層 (HTTP/HTTPS) 進行路由決策、支援以路徑為基礎的路由，並可將請求路由到您叢集中每個容器執行個體的一或多個連接埠。Application Load Balancer 支援動

態主機連接埠映射。例如，如果您的任務的容器定義指定 NGINX 容器連接埠使用連接埠 80，主機連接埠使用連接埠 0，則會從容器執行個體的暫時性連接埠範圍 (例如最新的 Amazon ECS 最佳化 AMI 為 32768 至 61000) 動態選擇主機連接埠。當工作啟動時，NGINX 容器會在應 Application Load Balancer 器中註冊為執行個體 ID 和連接埠組合，而流量會分配到與該容器對應的執行個體 ID 和連接埠。這個動態映射可讓您在同一個容器執行個體中的單一服務擁有多項任務。如需詳細資訊，請參閱 Application Load Balancer 使用者指南。

如需設定參數以加速部署的最佳作法的相關資訊，請參閱：

- 針對 Amazon ECS 優化負載平衡器運作狀態檢查參數
- 優化 Amazon ECS 的負載平衡器連接排除參數

將應用程式負載平衡器與 Amazon ECS 搭配使用時，請考慮下列事項：

- Amazon ECS 需要服務連結 IAM 角色，該角色會在任務建立和停止時提供向您的負載平衡器註冊與取消註冊目標所需的許可。如需詳細資訊，請參閱 使用 Amazon ECS 的服務連結角色。
- 目標群組的 IP 位址類型必須設定為 IPv4。
- 對於任務使用 `awsvpc` 網路模式的服務，當您為服務建立目標群組時，必須選擇 `ip` 做為目標類型，而非 `instance`。這是由於採用 `awsvpc` 網路模式的任務是與彈性網路介面相關聯，而非與 Amazon EC2 執行個體相關聯。
- 如果您的服務需要存取多個負載平衡連接埠，例如 HTTP/HTTPS 服務的連接埠 80 和連接埠 443，您可以設定兩個接聽程式。一個接聽程式負責將請求轉送至服務的 HTTPS，另一個接聽程式負責將 HTTP 請求重新導向至適當的 HTTPS 連接埠。如需詳細資訊，請參閱 Application Load Balancer 使用者指南中的為您的 Application Load Balancer 建立接聽程式。
- 負載平衡器子網路組態必須包含您容器執行個體所在的所有可用區域。
- 建立服務之後，負載平衡器組態無法從 AWS Management Console變更。您可以使用 AWS Copilot AWS CloudFormation、 AWS CLI 或 SDK 來修改ECS滾動式部署控制器的負載平衡器組態，而不能修改 AWS CodeDeploy 藍/綠或外部的負載平衡器組態。當您新增、更新或移除負載平衡器組態時，Amazon ECS 會使用更新後的 Elastic Load Balancing 組態啟動新的部署。這會導致任務向負載平衡器註冊和取消註冊。我們建議您在更新 Elastic Load Balancing 組態之前，先在測試環境中驗證。如需如何修改組態的相關資訊，請參閱 Amazon 彈性容器服務 API 參考UpdateService中的。
- 如果服務工作未通過負載平衡器健全狀況檢查準則，則工作會停止並重新啟動。此程序會持續到您的服務達到所需的執行任務數量為止。
- 如果您遇上啟用負載平衡器功能之服務的問題，請參閱疑難排解 Amazon ECS 中的服務負載平衡器。

- 您的任務和負載平衡器必須位於相同的 VPC 中。
- 針對每個服務使用唯一的目標群組。

  針對多個服務使用相同的目標群組，可能會導致服務部署期間發生問題。

如需如何建立 Application Load Balancer 的相關資訊，請參閱 Ap plication Load Balancer 中建立應用程式負載平衡器

使用 Amazon ECS 的 Network Load Balancer

Network Load Balancer 在傳輸層進行路由決策 (TCP/SSL)。每秒可以處理數百萬個請求。在負載平衡器收到連線後，會使用流程雜湊路由演算法從目標群組選取預設規則的目標。負載平衡器會嘗試開啟接聽程式設定中指定之連接埠上所選目標的 TCP 連線。負載平衡器會轉送此請求，且不會修改標頭。Network Load Balancer 支援動態主機連接埠映射。例如，如果您的任務的容器定義指定 NGINX 容器連接埠使用連接埠 80，主機連接埠使用連接埠 0，則會從容器執行個體的暫時性連接埠範圍 (例如最新的 Amazon ECS 最佳化 AMI 為 32768 至 61000) 動態選擇主機連接埠。啟動此任務時，NGINX 容器會以執行個體 ID 和連接埠組合在 Network Load Balancer 註冊，而流量將分散到與該容器對應的執行個體 ID 和連接埠。這個動態映射可讓您在同一個容器執行個體中的單一服務擁有多項任務。如需詳細資訊，請參閱 Network Load Balancer 使用者指南。

如需設定參數以加速部署的最佳作法的相關資訊，請參閱：

- 針對 Amazon ECS 優化負載平衡器運作狀態檢查參數
- 優化 Amazon ECS 的負載平衡器連接排除參數

將網路負載平衡器與 Amazon ECS 搭配使用時，請考慮下列事項：

- Amazon ECS 需要服務連結 IAM 角色，該角色會在任務建立和停止時提供向您的負載平衡器註冊與取消註冊目標所需的許可。如需詳細資訊，請參閱 使用 Amazon ECS 的服務連結角色。
- 您無法將五個以上的目標群組附加至服務。
- 對於任務使用 `awsvpc` 網路模式的服務，當您為服務建立目標群組時，必須選擇 `ip` 做為目標類型，而非 `instance`。這是由於採用 `awsvpc` 網路模式的任務是與彈性網絡介面相關聯，而非與 Amazon EC2 執行個體相關聯。
- 負載平衡器子網路組態必須包含您容器執行個體所在的所有可用區域。
- 建立服務之後，負載平衡器組態無法從 AWS Management Console變更。您可以使用 AWS Copilot AWS CloudFormation、 AWS CLI 或 SDK 來修改ECS滾動式部署控制器的負載平衡器組態，而不

能修改 AWS CodeDeploy 藍/綠或外部的負載平衡器組態。當您新增、更新或移除負載平衡器組態時，Amazon ECS 會使用更新後的 Elastic Load Balancing 組態啟動新的部署。這會導致任務向負載平衡器註冊和取消註冊。我們建議您在更新 Elastic Load Balancing 組態之前，先在測試環境中驗證。如需如何修改組態的相關資訊，請參閱 Amazon 彈性容器服務 API 參考UpdateService中的。

- 如果服務工作未通過負載平衡器健全狀況檢查準則，則工作會停止並重新啟動。此程序會持續到您的服務達到所需的執行任務數量為止。
- 當您使用設定 IP 位址做為目標且關閉用戶端 IP 保留的閘道 Load Balancer 時，會將要求視為來自閘道負載平衡器私人 IP 位址。這表示只要您允許目標安全性群組中的傳入要求和健康狀態檢查，閘道 Load Balancer 背後的服務就會有效地向全世界開放。
- 對於 Fargate 任務，您必須使用平台版本`1.4.0`（Linux）或`1.0.0`（視窗）。
- 如果您遇上啟用負載平衡器功能之服務的問題，請參閱疑難排解 Amazon ECS 中的服務負載平衡器。
- 您的任務和負載平衡器必須位於相同的 VPC 中。
- Network Load Balancer 用戶端 IP 位址保留與 Fargate 目標相容。
- 針對每個服務使用唯一的目標群組。

  針對多個服務使用相同的目標群組，可能會導致服務部署期間發生問題。

如需如何建立 Network Load Balancer 的相關資訊，請參閱在 Network Load Balancer 中建立網路負載平衡器

> **Important**
>
> 若服務的任務定義是採用 Fargate 啟動類型所需的 `awsvpc` 網路模式，則您必須選擇 `ip` 做為目標類型，而不是選擇 `instance`。因為採用 `awsvpc` 網路模式的任務是與彈性網路界面相關聯，而非與 Amazon EC2 執行個體相關聯。
> 如果執行個體類型如下：
> C1、CC1、CC2、CG1、CG2、CR1、G1、G2、HI1、HS1、M1、M2、M3 和 T1，就不能以執行個體 ID 來登錄執行個體。您可以使用 IP 地址來登錄這些執行個體。

使用 Amazon ECS 的閘道 Load Balancer

Gateway Load Balancer 在開放系統互相連線 (OSI) 模型的第三層，網路層運作。它會接聽所有連接埠的全部 IP 封包，並轉送流量至接聽程式規則中指定的目標群組。它使用 5 元組 (針對 TCP/UDP 流)

或 3 元組 (針對非 TCP/UDP 流) 維護特定目標應用裝置的流量黏性。例如，如果您的任務的容器定義指定 NGINX 容器連接埠使用連接埠 80，主機連接埠使用連接埠 0，則會從容器執行個體的暫時性連接埠範圍 (例如最新的 Amazon ECS 最佳化 AMI 為 32768 至 61000) 動態選擇主機連接埠。啟動工作時，NGINX 容器會在閘道 Load Balancer 中註冊為執行個體 ID 和連接埠組合，而流量會分配至與該容器對應的執行個體 ID 和連接埠。這個動態映射可讓您在同一個容器執行個體中的單一服務擁有多項任務。如需詳細資訊，請參閱閘道 Load Balancer 中的閘道負載平衡器是什麼。

如需設定參數以加速部署的最佳作法的相關資訊，請參閱：

- 針對 Amazon ECS 優化負載平衡器運作狀態檢查參數
- 優化 Amazon ECS 的負載平衡器連接排除參數

將閘道負載平衡器與 Amazon ECS 搭配使用時，請考慮下列事項：

- Amazon ECS 需要服務連結 IAM 角色，該角色會在任務建立和停止時提供向您的負載平衡器註冊與取消註冊目標所需的許可。如需詳細資訊，請參閱 使用 Amazon ECS 的服務連結角色。
- 對於任務使用 `awsvpc` 網路模式的服務，當您為服務建立目標群組時，必須選擇 `ip` 做為目標類型，而非 `instance`。這是由於採用 `awsvpc` 網路模式的任務是與彈性網絡介面相關聯，而非與 Amazon EC2 執行個體相關聯。
- 負載平衡器子網路組態必須包含您容器執行個體所在的所有可用區域。
- 建立服務之後，負載平衡器組態無法從 AWS Management Console變更。您可以使用 AWS Copilot AWS CloudFormation、 AWS CLI 或 SDK 來修改ECS滾動式部署控制器的負載平衡器組態，而不能修改 AWS CodeDeploy 藍/綠或外部的負載平衡器組態。當您新增、更新或移除負載平衡器組態時，Amazon ECS 會使用更新後的 Elastic Load Balancing 組態啟動新的部署。這會導致任務向負載平衡器註冊和取消註冊。我們建議您在更新 Elastic Load Balancing 組態之前，先在測試環境中驗證。如需如何修改組態的相關資訊，請參閱 Amazon 彈性容器服務 API 參考UpdateService中的。
- 如果服務工作未通過負載平衡器健全狀況檢查準則，則工作會停止並重新啟動。此程序會持續到您的服務達到所需的執行任務數量為止。
- 當您使用設定 IP 位址做為目標的閘道 Load Balancer 時，會將要求視為來自閘道負載平衡器私有 IP 位址。這表示只要您允許目標安全性群組中的傳入要求和健康狀態檢查，閘道 Load Balancer 背後的服務就會有效地向全世界開放。
- 對於 Fargate 任務，您必須使用平台版本`1.4.0`（Linux）或`1.0.0`（視窗）。
- 如果您遇上啟用負載平衡器功能之服務的問題，請參閱疑難排解 Amazon ECS 中的服務負載平衡器。
- 您的任務和負載平衡器必須位於相同的 VPC 中。

- 針對每個服務使用唯一的目標群組。

  針對多個服務使用相同的目標群組，可能會導致服務部署期間發生問題。

如需如何建立閘道 Load Balancer 的相關資訊，請參閱在閘道 Load Balancer 中建立閘道負載平衡器

> **Important**
>
> 若服務的任務定義是採用 Fargate 啟動類型所需的 awsvpc 網路模式，則您必須選擇 ip 做為目標類型，而不是選擇 instance。因為採用 awsvpc 網路模式的任務是與彈性網路界面相關聯，而非與 Amazon EC2 執行個體相關聯。
> 如果執行個體類型如下：
> C1、CC1、CC2、CG1、CG2、CR1、G1、G2、HI1、HS1、M1、M2、M3 和 T1，就不能以執行個體 ID 來登錄執行個體。您可以使用 IP 地址來登錄這些執行個體。

使用 Amazon ECS 服務註冊多個目標群組

當您在服務定義中指定多個目標群組時，您的 Amazon ECS 服務可以為來自多個負載平衡器的流量提供服務，以及公開多個負載平衡的連接埠。

若要建立指定多個目標群組的服務，您必須使用 Amazon ECS API AWS CLI、開發套件或 AWS CloudFormation 範本建立服務。建立服務之後，您可以使用 AWS Management Console檢視該服務以及向其註冊的目標群組。您必須使用 UpdateService 來修改現有服務的負載平衡器組態。

您可以使用以下格式在服務定義中指定多個目標群組。如需服務定義的完整語法，請參閱服務定義範本。

```
"loadBalancers":[
   {

 "targetGroupArn":"arn:aws:elasticloadbalancing:region:123456789012:targetgroup/
target_group_name_1/1234567890123456",
      "containerName":"container_name",
      "containerPort":container_port
   },
   {

 "targetGroupArn":"arn:aws:elasticloadbalancing:region:123456789012:targetgroup/
target_group_name_2/6543210987654321",
```

```
      "containerName":"container_name",
      "containerPort":container_port
   }
]
```

考量事項

當您在服務定義中指定多個目標群組時，應該考慮下列事項。

- 對於使用 Application Load Balancer 或 Network Load Balancer 的服務，您無法連接超過五個目標群組到一個服務。
- 只有在下列情況下支援在服務定義中指定多個目標群組：
  - 服務必須使用 Application Load Balancer 或 Network Load Balancer。
  - 服務必須使用滾動更新 (ECS) 部署控制器類型。
- 若服務同時包含使用 Fargate 和 EC2 啟動類型的任務，則該服務支援指定多個目標群組。
- 當建立指定多個目標群組的服務時，必須建立 Amazon ECS 服務連結角色。透過省略 API 請求中的 `role` 參數，或 AWS CloudFormation中的 `Role` 屬性來建立角色。如需詳細資訊，請參閱 使用 Amazon ECS 的服務連結角色。

範例服務定義

以下是在服務定義中指定多個目標群組的一些使用案例範例。如需服務定義的完整語法，請參閱服務定義範本。

針對內部和外部流量具有單獨的負載平衡器

在以下的使用案例中，服務會對相同的容器和連接埠使用兩個不同的負載平衡器，一個用於內部流量，另一個用於面向網際網路的流量。

```
"loadBalancers":[
   //Internal ELB
   {

 "targetGroupArn":"arn:aws:elasticloadbalancing:region:123456789012:targetgroup/
target_group_name_1/1234567890123456",
      "containerName":"nginx",
      "containerPort":8080
   },
   //Internet-facing ELB
```

```
   {

 "targetGroupArn":"arn:aws:elasticloadbalancing:region:123456789012:targetgroup/
target_group_name_2/6543210987654321",
      "containerName":"nginx",
      "containerPort":8080
   }
]
```

公開來自同一個容器的多個連接埠

在以下的使用案例中，服務會使用一個負載平衡器，但公開相同容器中的多個連接埠。例如，Jenkins 容器可公開適用於 Jenkins Web 界面的連接埠 8080，以及公開適用於 API 的連接埠 50000。

```
"loadBalancers":[
   {

 "targetGroupArn":"arn:aws:elasticloadbalancing:region:123456789012:targetgroup/
target_group_name_1/1234567890123456",
      "containerName":"jenkins",
      "containerPort":8080
   },
   {

 "targetGroupArn":"arn:aws:elasticloadbalancing:region:123456789012:targetgroup/
target_group_name_2/6543210987654321",
      "containerName":"jenkins",
      "containerPort":50000
   }
]
```

公開來自多個容器的連接埠

在以下的使用案例，服務會使用一個負載平衡器和兩個目標群組，以公開不同容器中的連接埠。

```
"loadBalancers":[
   {

 "targetGroupArn":"arn:aws:elasticloadbalancing:region:123456789012:targetgroup/
target_group_name_1/1234567890123456",
      "containerName":"webserver",
      "containerPort":80
```

```
   },
   {

 "targetGroupArn":"arn:aws:elasticloadbalancing:region:123456789012:targetgroup/
target_group_name_2/6543210987654321",
      "containerName":"database",
      "containerPort":3306
   }
]
```

自動擴展您的 Amazon ECS 服務

自動擴展是自動增加或減少 Amazon ECS 服務中所需任務數的功能。Amazon ECS 利用 Application Auto Scaling 服務來提供此功能。如需詳細資訊，請參閱 Application Auto Scaling 使用者指南。

Amazon ECS 會根 CloudWatch 據您服務的平均 CPU 和記憶體使用量發佈指標。如需詳細資訊，請參閱 Amazon ECS 服務使用率指標。您可以使用這些 CloudWatch 指標和其他指標向外擴充服務 (新增更多工作)，以在尖峰時段處理高需求，並在服務中擴充 (執行較少的工作)，以在低使用率期間降低成本。

Amazon ECS Service Auto Scaling 支援下列自動擴展類型：

- 使用目標指標值擴展您的 Amazon ECS 服務–根據特定指標的目標值，增加或減少您服務執行的任務數目。這與您運用電熱器維持家中溫度的方式很類似。您只要選取溫度，電熱器會自行執行其餘操作。
- 根據警示 CloudWatch 使用預先定義的增量來擴展 Amazon ECS 服務–根據一組依警示違規大小而變動的擴展調整 (稱為步驟調整)，增加或減少您服務執行的任務數目。
- 使用排程擴展您的 Amazon ECS 服務根據日期和時間，增加或減少服務執行的工作數。

考量事項

使用擴展策略時，考慮下列事項：

- Amazon ECS 會以 1 分鐘的間隔將指標傳送到 CloudWatch。指標必須等到叢集和服務傳送指標 CloudWatch，而且您無法為不存在的指標建立 CloudWatch 警示。
- 擴展政策支援冷卻時間。等待先前擴展活動生效的秒數。
  - 對於向外擴展事件，其目的是連續的規模擴展 (但並非過度)。在 Service Auto Scaling 使用擴展政策成功擴展之後，就會開始計算冷卻時間。除非初始化較大的橫向擴展或冷卻時間結束，否則擴展

政策不會再次增加所需的容量。在向外擴展冷卻期間有效時，由起始冷卻的向外擴展活動所增加的容量，將計入下次向外擴展活動所需的容量。

- 對於向內縮減事件，其用意在於保守縮減以保護應用程式的可用性，所以冷卻時間過後才會開放縮減活動。不過，若在向內縮減冷卻時間另有警示起始了橫向擴展活動，Service Auto Scaling 則會立即橫向擴展目標。在這種情況下，向內擴展冷卻期間隨即停止，且不會完成。

- 服務排程器隨時會使用所需的計數，但只要您具有服務的有效擴展政策和警示，Service Auto Scaling 就可以變更您手動設定的所需計數。
- 如果服務的所需計數設定為低於其最小容量值，且警示起始向外延展活動，Service Auto Scaling 會根據與警示相關聯的擴展政策，將所需計數擴展到最小容量值，然後依需要繼續向外擴充。不過，向內擴展活動不會調整所需的計數，因為它已低於最小容量值。
- 如果服務的所需計數設定為高於其最大容量值，且警示啟動了活動中的縮放比例，Service Auto Scaling 會根據與警示相關聯的擴展政策，將所需的計數擴展到最大容量值，然後依需要繼續擴充。不過，向外擴展活動不會調整所需的計數，因為它已高於最大容量值。
- 在擴展活動期間，服務中實際執行的任務計數是 Service Auto Scaling 用作起始點的值，而不是所需的計數。這應為處理容量。這可防止無法滿足的過度且失控的擴展，例如，有可能因容器執行個體資源不足而無法放置其他任務。如果容器執行個體容量稍後可用，則擱置的擴展活動可能會成功，接著在冷卻時間之後進行後續的擴展活動。
- 若希望您的任務計數在沒有待完成工作時縮減到零，將最小容量設定為 0。使用目標追蹤擴展政策，當實際容量為 0 而指標顯示有工作負載需求時，Service Auto Scaling 會在擴展前等待一個待傳送的資料點。在此情況下，它會向外擴展可能的數量下限做為起始點，然後以實際執行的任務計數為基礎繼續擴展。
- Application Auto Scaling 會在 Amazon ECS 部署正在進行時關閉縮減程序。不過，除非在部署期間暫停，否則向外擴展程序會繼續發生。如需詳細資訊，請參閱 服務自動擴展和部署。
- 您有多個適用於 Amazon ECS 任務的 Application Auto Scaling 選項。目標追蹤是最容易使用的模式。這樣一來，您僅需要為指標設定目標值，例如 CPU 平均使用率。然後，自動縮放器會自動管理獲得該值所需的任務數量。使用步驟擴展，您可以更快對需求變化做出反應，因為您定義了擴展指標的特定閾值，以及超越閾值時要新增或刪除的任務數量。而且，更重要的是，您可以透過最大限度減少閾值警報違反的時間來對需求變化做出非常快速的反應。

優化 Amazon ECS 服務 auto 擴展

Amazon ECS 服務是受管理的任務集合。每個服務都有關聯的任務定義、所需的任務計數以及可選的放置策略。Amazon ECS 服務 auto 擴展是透過 Application Auto Scaling 服務實作的。「Application Auto Scaling」會使用 CloudWatch 指標做為擴展指標的來源。它還使用 CloudWatch 警報來設置閾值，以確定何時擴展服務擴展或擴展。您可以設定測量結果目標 (稱為目標追蹤調整)，或指定臨界值

(稱為步驟調整)，來提供調整的臨界值。設定「Application Auto Scaling」之後，它會持續計算服務的適當所需工作計數。它還會在所需的任務計數發生變化時通知 Amazon ECS，方法是向外擴展或擴展。

若要有效使用服務 auto 調整規模，您必須選擇適當的擴展指標。

如果預測需求大於目前的容量，應該向外擴充應用程式。相反地，當資源超出需求時，應用程式也可以進行擴充，以節省成本。

識別度量

為了有效擴展，識別指示使用率或飽和度的指標至關重要。此量度必須顯示下列屬性，才能用於縮放比例。

- 量度必須與需求相關。當資源保持穩定但需求變更時，量度值也必須變更。當需求增加或減少時，量度應增加或減少。
- 度量值必須按照容量成比例調整。當需求保持不變時，新增更多資源必須導致標準值的比例變更。因此，將任務數量加倍應導致指標減少 50%。

識別使用率指標的最佳方法是在生產前環境 (例如測試環境) 中進行負載測試。商業和開源負載測試解決方案已廣泛使用。這些解決方案通常可以產生合成負載或模擬真實的使用者流量。

若要開始負載測試程序，請為應用程式的使用率指標建立儀表板。這些測量結果包括 CPU 使用率、記憶體使用率、I/O 作業、I/O 佇列深度和網路輸送量。您可以透過服務 (例如容器深入解析) 收集這些指標。如需詳細資訊，請參閱 使用容器洞察來監控 Amazon ECS 容器。在此過程中，請確定您收集並繪製應用程式回應時間或工作完成率的指標。

從一個小的請求或工作插入率開始。將此速率保持穩定數分鐘，以便您的應用程序進行預熱。然後，慢慢增加速度並保持穩定幾分鐘。重複此週期，每次都會提高速率，直到應用程式的回應或完成時間太慢，無法達到您的服務層級目標 (SLO) 為止。

在負載測試時，檢查每個使用率指標。隨負載而增加的指標是作為最佳使用率指標的主要候選人。

接下來，確定達到飽和度的資源。同時，還要檢查使用率指標，以查看哪一個先在高級別扁平化，或者達到峰值，然後首先崩潰您的應用程序。例如，如果在增加負載時 CPU 使用率從 0% 增加到 70-80%，那麼在添加更多負載後保持在該級別，那麼可以肯定地說 CPU 已飽和。根據 CPU 架構的不同，它可能永遠不會達到 100%。例如，假設記憶體使用率隨著您增加負載而增加，當應用程式達到任務或 Amazon EC2 執行個體記憶體限制時突然當機。在這種情況下，很可能是內存已完全消耗的情況。您的應用程式可能會耗用多個資源。因此，請選擇代表首先耗盡資源的指標。

最後，在任務或 Amazon EC2 執行個體數量增加一倍後，再次嘗試負載測試。假設關鍵量度以前一半的速率增加或減少。如果是這種情況，則度量與容量成正比。這是 auto 調整規模的良好使用率指標。

現在考慮這種假設的情況。假設您負載測試應用程序，並發現 CPU 使用率最終達到 80%，每秒 100 個請求。當添加更多負載時，它不會使 CPU 利用率再提高。但是，它確實使您的應用程序響應速度更慢。然後，您再次運行負載測試，將任務數量加倍，但將速率保持在其先前的峰值。如果您發現平均 CPU 使用率降至約 40%，則平均 CPU 使用率是擴展指標的最佳選擇。另一方面，如果 CPU 使用率在增加任務數量後仍保持在 80%，則平均 CPU 使用率不是一個很好的擴展指標。在這種情況下，需要更多的研究來找到合適的指標。

一般應用程式模型和縮放屬性

各種軟件都在上運行 AWS。許多工作負載都是本土開發的，而其他工作負載則以熱門的開放原始碼 無論它們的起源如何，我們都觀察到了一些常見的服務設計模式。如何有效縮放在很大程度上取決於陣列。

高效率的 CPU 繫結伺服器

高效率的 CPU 綁定服務器幾乎不使用 CPU 和網絡吞吐量以外的任何資源。每個請求都可以由應用程序單獨處理。請求不依賴於其他服務，例如數據庫。該應用程序可以處理數十萬個並發請求，並且可以有效地利用多個 CPU 來執行此操作。每個請求都是由具有低內存開銷的專用線程提供服務，或者有一個非同步事件循環在服務請求的每個 CPU 上運行。應用程式的每個複本同樣能夠處理要求。在 CPU 之前可能耗盡的唯一資源是網絡帶寬。在 CPU 邊界服務中，即使在尖峰輸送量下，記憶體使用率也只是可用資源的一小部分。

這種類型的應用程序適用於基於 CPU 的 auto 縮放。該應用程序在擴展方面享有最大的靈活性。它可以通過向其提供更大的 Amazon EC2 實例或 Fargate vCPUs 來垂直擴展。而且，它也可以通過添加更多複本水平縮放。新增更多複本或將執行個體大小增加一倍，相對於容量的平均 CPU 使用率減少一半。

如果您為此應用程式使用 Amazon EC2 容量，請考慮將其放置在運算優化執行個體上，例如`c5`或`c6g`系列。

高效率的記憶體繫結伺服器

高效率的記憶體繫結伺服器會為每個要求配置大量記憶體。在最大並行性 (但不一定是輸送量) 下，CPU 資源耗盡之前，記憶體就會耗盡。要求結束時，會釋放與要求相關聯的記憶體。只要有可用的記憶體，就可以接受其他要求。

這種類型的應用程序適用於基於內存的 auto 縮放。該應用程序在擴展方面享有最大的靈活性。它可以通過向其提供更大的 Amazon EC2 或 Fargate 內存資源來垂直擴展。而且，它也可以通過添加更多複

本水平縮放。新增更多複本或將執行個體大小增加一倍，可以將相對於容量的平均記憶體使用率減少一半。

如果您對此應用程式使用 Amazon EC2 容量，請考慮將其放在記憶體最佳化執行個體上，例如`r5`或`r6g`系列。

有些記憶體繫結應用程式在結束時不會釋放與要求相關聯的記憶體，因此減少並行運算不會導致使用的記憶體減少。為此，我們不建議您使用以記憶體為基礎的縮放。

以工作者為基礎的伺服器

以 Worker 為基礎的伺服器會依次處理每個個別 Worker 執行緒的一個要求。工作線程可以是輕量級線程，例如 POSIX 線程。它們也可以是重量較重的線程，例如 UNIX 進程。無論它們是哪個線程，應用程序總是可以支持的最大並發性。通常，並行限制會按比例設定為可用的記憶體資源。如果達到並行限制，則會將其他要求放入待處理佇列中。如果積壓佇列溢位，則會立即拒絕其他內送要求。適合這種模式的常見應用程序包括 Apache 的 Web 服務器和 Gunicorn。

請求並行通常是擴展此應用程序的最佳指標。因為每個複本都有並行限制，因此在達到平均限制之前向外擴充是很重要的。

取得要求並行量度的最佳方式是讓您的應用程式將其報告給 CloudWatch。應用程式的每個複本都可以高頻發佈並行要求數量，做為自訂量度。我們建議將頻率設定為每分鐘至少一次。收集數個報告之後，您可以使用平均並行作為縮放量度量。此量度的計算方式是將並行總數除以複本數目。例如，如果並行總數為 1000，而複本數為 10，則平均並行為 100。

如果您的應用程式位於應用程式負載平衡器之後，您也可以使用負載平衡器的`ActiveConnectionCount`指標作為擴展指標中的一個因素。`ActiveConnectionCount`測量結果必須除以複本數目，才能取得平均值。平均值必須用於縮放，而不是原始計數值。

為了使此設計發揮最佳效果，在低請求率下，響應延遲的標準差應該很小。我們建議在需求不足的期間，大部分的要求都會在短時間內得到回應，而且沒有太多要求的時間比平均回應時間長得多。平均回應時間應接近第 95 個百分位數的回應時間。否則，可能會導致佇列溢位。這會導致錯誤。我們建議您在必要時提供其他複本，以減輕溢位的風險。

等待伺服器

等待伺服器會針對每個要求執行一些處理，但高度依賴於一個或多個下游服務才能運作。容器應用程式通常會大量使用下游服務，例如資料庫和其他 API 服務。這些服務可能需要一些時間才能回應，特別是在高容量或高並行情況下。這是因為這些應用程序傾向於使用很少的 CPU 資源，並在可用內存方面利用其最大並發性。

等待服務適用於記憶體繫結伺服器模式或工作者型伺服器模式，視應用程式的設計方式而定。如果應用程式的並行僅受記憶體限制，則應使用平均記憶體使用率作為擴展指標。如果應用程式的並行是以 Worker 限制為基礎，則應使用平均並行作為縮放量度量。

基於 Java 的服務器

如果您的基於 Java 的服務器是 CPU 綁定的，並且按比例擴展到 CPU 資源，那麼它可能適合高效率的 CPU 綁定服務器模式。在這種情況下，平均 CPU 使用率可能適合作為擴展指標。但是，許多 Java 應用程序不受 CPU 限制，因此它們難以擴展。

為了獲得最佳效能，我們建議您將盡可能多的記憶體配置給 Java 虛擬機器 (JVM) 堆積。最新版本的 JVM（包括 Java 8 更新 191 或更新版本）會自動設置盡可能大的堆棧大小以適合容器。這表示在 Java 中，記憶體使用率很少與應用程式使用率成正比。隨著請求率和並發性的增加，內存使用率保持不變。因此，我們不建議根據記憶體使用率來擴展 Java 型伺服器。相反，我們通常建議根據 CPU 使用率進行擴展。

在某些情況下，基於 Java 的服務器在耗盡 CPU 之前會遇到堆積耗盡。如果您的應用程式容易在高並行時出現堆積耗盡，則平均連線是最佳的擴展指標。如果您的應用程式容易在高輸送量下出現堆積耗盡，則平均要求率是最佳的擴展指標。

使用其他垃圾收集執行階段的伺服器

許多伺服器應用程式都是以執行記憶體回收 (例如 .NET 和 Ruby) 的執行階段為基礎。這些伺服器應用程式可能適合先前描述的其中一種模式。但是，與 Java 一樣，我們不建議根據記憶體擴展這些應用程式，因為它們觀察到的平均記憶體使用率通常與輸送量或並行無關。

對於這些應用程式，如果應用程式是 CPU 繫結，建議您調整 CPU 使用率。否則，我們建議您根據負載測試結果，根據平均輸送量或平均並行擴展。

Job 處理器

許多工作負載涉及非同步工作處理 它們包括不會即時接收要求，而是訂閱工作佇列以接收工作的應用程式。對於這些類型的應用程式，適當的擴展指標幾乎總是佇列深度。佇列成長表示待處理工作超過處理能力，而空白佇列則表示有比工作更多的容量。

AWS 簡訊服務 (例如 Amazon SQS 和 Amazon Kinesis Data Streams) 提供可用於擴展的 CloudWatch 指標。對於 Amazon SQS 來說，`ApproximateNumberOfMessagesVisible`是最好的指標。對於 Kinesis Data Streams，請考慮使用 Kinesis 用戶端程式庫 (KCL) 所發佈的`MillisBehindLatest`量度。在使用此指標進行擴展之前，應先對所有消費者進行平均。

服務自動擴展和部署

Application Auto Scaling 會在 Amazon ECS 部署正在進行時關閉縮減程序。不過，除非在部署期間暫停，否則向外擴展程序會繼續發生。如果您想要在部署正在進行時暫停向外擴展程序，請執行下列步驟。

1. 呼叫 describe-scalable-targets 命令，指定在 Application Auto Scaling 中與可擴展目標關聯的服務資源 ID (範例：`service/default/sample-webapp`)。記錄輸出。您將在呼叫下一個命令時用到它。
2. 呼叫 register-scalable-target 命令，指定資源 ID、命名空間和可擴展維度。同時為 `DynamicScalingInSuspended` 和 `DynamicScalingOutSuspended` 指定 `true`。
3. 部署完成後，您可以呼叫 register-scalable-target 命令以繼續擴展。

如需詳細資訊，請參閱暫停和繼續擴展 Application Auto Scaling。

使用目標指標值擴展您的 Amazon ECS 服務

使用目標追蹤擴展政策，您可以選取指標及設定目標值。Amazon ECS 服務 Auto Scaling 會建立和管理控制擴展政策的 CloudWatch 警示，並根據指標和目標值計算擴展調整。擴展政策會視需要新增或移除服務任務，以維持等於或接近指定目標值的指標。除了維持接近目標值的指標之外，目標追蹤擴展政策也會因波動負載模式調整為指標中的波動，以及將服務中執行之數個任務中的快速波動降到最低。

考量事項

使用目標追蹤政策時，請考慮下列事項：

- 目標追蹤擴展政策假設在指定的指標超過目標值時，應執行向外擴展。您無法使用目標追蹤擴展政策在指定的指標低於目標值時執行向外擴展。
- 所指定指標的資料不足時，目標追蹤擴展政策不會執行擴展。政策不會執行向內擴展，因為向內擴展不會將資料不足解釋為低使用率。
- 您可能會看到目標值與實際指標資料點之間有些差距。原因是 Service Auto Scaling 在決定新增或移除多少容量時，一律以四捨五入來保守處理。這樣可防止新增不足的容量，或移除過多的容量。
- 為了確保應用程式可用性，服務可以按比例快速地向外擴展到指標，但需漸漸地逐步向內擴展。
- Application Auto Scaling 會在 Amazon ECS 部署正在進行時關閉縮減程序。不過，除非在部署期間暫停，否則向外擴展程序會繼續發生。如需詳細資訊，請參閱 服務自動擴展和部署。
- 您可以擁有 Amazon ECS 服務的多個目標追蹤擴展政策，但前提是每個政策都使用不同的指標。Service Auto Scaling 的用意一律以可用性為優先，因此其行為視目標追蹤政策是準備水平擴展

或縮減而有所不同。如果任何目標追蹤政策已準備好橫向擴展，其就會將服務橫向擴展，但只有在所有目標追蹤政策 (已開啟縮減部分) 都已準備好要縮減時才會縮減。

- 請勿編輯或刪除 Service Auto Scaling 針對目標追蹤擴展政策所管理的 CloudWatch 警示。當您刪除擴展政策時，Service Auto Scaling 會自動刪除警示。
- 藍/綠部署類型不支援目標追蹤擴展政策的 `ALBRequestCountPerTarget` 指標。

如需有關目標追蹤擴展政策的詳細資訊，請參閱《Application Auto Scaling 使用者指南》中的目標追蹤擴展政策。

使用 Amazon ECS 主控台為您的 Amazon ECS 服務設定目標擴展政策

1. 除了用於建立和更新服務的標準 IAM 許可之外，您還需要其他許可。如需詳細資訊，請參閱 Amazon ECS 服務 auto 擴展所需的 IAM 許可。
2. 您可以在建立或更新服務時設定擴展政策。如需詳細資訊，請參閱下列其中一個項目：
   - 使用定義的參數建立服務— 創建一個新的服務
   - 使用主控台更新 Amazon ECS 服務— 更新現有服務

若要設定 Amazon ECS 服務的目標擴展政策，請使用 AWS CLI

1. 除了用於建立和更新服務的標準 IAM 許可之外，您還需要其他許可。如需詳細資訊，請參閱 Amazon ECS 服務 auto 擴展所需的 IAM 許可。
2. 使用 register-scalable-target 命令，將 Amazon ECS 服務註冊為可擴展性目標。
3. 使用 put-scaling-policy 命令，來建立擴展政策。

根據警示 CloudWatch 使用預先定義的增量來擴展 Amazon ECS 服務

使用步驟擴展政策，您可以指定啟動擴展程序的 CloudWatch 警示。例如，如果您想要在 CPU 使用率達到特定層級時向外延展，請使用提供的`CPUUtilization`指標建立警示。建立步進規模調整政策時，您必須指定下列規模調整類型的其中一種：

- 新增 — 依指定的容量單位數目或目前容量的指定百分比增加工作數目。
- 移除 — 依指定的容量單位數目或目前容量的指定百分比減少工作數目。
- 設定為-將工作數目設定為指定的容量單位數。

例如，假設目標容量為和已實現的容量為 10，而且規模調整政策會增加 1。違反警示時，自動擴展程序會增加 1 到 10，以獲得 11，因此 Amazon ECS 會針對該服務啟動 1 個任務。

強烈建議您使用目標追蹤擴展政策來擴展指標，例如平均 CPU 使用率或每個目標的平均要求計數。在容量減少時容量增加和增加的量度可用於按比例擴展或使用目標追蹤的工作數量。這有助於確保服務 Auto Scaling 緊密遵循應用程式的需求曲線。

如需步驟調整政策及其運作方式的概觀，請參閱《應用程式 Auto Scaling 使用者指南》中的步驟調整政策。閱讀本簡介後，請參閱以下各節，了解如何使用主控台和 AWS Command Line Interface為 Amazon ECS 設定步驟擴展。

使用 Amazon ECS 主控台為您的 Amazon ECS 服務設定步驟擴展政策

1. 除了用於建立和更新服務的標準 IAM 許可之外，您還需要其他許可。如需詳細資訊，請參閱 Amazon ECS 服務 auto 擴展所需的 IAM 許可。
2. 您可以在建立或更新服務時設定擴展政策。如需詳細資訊，請參閱下列其中一個項目：
   - 使用定義的參數建立服務— 創建一個新的服務
   - 使用主控台更新 Amazon ECS 服務— 更新現有服務

若要設定 Amazon ECS 服務的步驟擴展政策，請使用 AWS CLI

1. 除了用於建立和更新服務的標準 IAM 許可之外，您還需要其他許可。如需詳細資訊，請參閱 Amazon ECS 服務 auto 擴展所需的 IAM 許可。
2. 使用 register-scalable-target 命令，將 Amazon ECS 服務註冊為可擴展性目標。
3. 使用 put-scaling-policy 命令，來建立擴展政策。
4. 使用放置量度警示指令建立啟動資源調整政策的警示。

使用排程擴展您的 Amazon ECS 服務

透過排程擴展，您可以建立排程動作，根據所預測的負載變動，在特定時間增加或減少容量，自動調整應用程式規模。如此一來，您便能主動調整應用程式規模，以符合負載變動預測。

藉由這些排定的擴展動作，您可將費用和效能調整到最佳狀態。您的應用程式可以有足夠的容量處理週間的流量高峰，但在其他時段不必過度佈建不需要的容量。

您可以搭配利用排程擴展和擴展政策，以主動和被動的方式處理規模擴展作業，同時享有這兩種方法的好處。執行排定的擴展動作後，擴展政策可以繼續決定是否進一步擴展容量。這有助於確保您有足夠的

容量來處理應用程式的負載。當應用程式擴展以滿足需求時，目前的容量必須落在您排定的動作所設定的容量上下限之內。

您可以使用設定排程調整 AWS CLI。如需有關排程調整的詳細資訊，請參閱應用程式 Auto Scaling 使用者指南中的排程調整

互連 Amazon ECS 服務

在 Amazon ECS 任務中執行的應用程式通常需要接收來自網際網路的連線，或通常需要連線到在 Amazon ECS 服務中執行的其他應用程式。如果需要來自網際網路的外部連線，建議您使用 Elastic Load Balancing。如需有關整合負載平衡的詳細資訊，請參閱the section called “使用負載平衡來分配服務流量”。

如果您需要應用程式連線到在 Amazon ECS 服務中執行的其他應用程式，Amazon ECS 提供以下無需負載平衡器的方法：

- Amazon ECS Service Connect

  我們建議使用服務 Connect，它提供 Amazon ECS 組態以進行服務探索、連線和流量監控。透過 Service Connect，您的應用程式可以使用簡短名稱和標準連接埠來連接到相同叢集中的 Amazon ECS 服務，以及其他叢集 (包括相同的 VPC)。 AWS 區域

  當您使用 Service Connect 時，Amazon ECS 會管理服務探索的所有部分：建立可探索的名稱、在任務開始和停止時動態管理每個任務的項目，以及在設定為探索名稱的每個任務中執行代理程式。您的應用程式可以使用 DNS 名稱的標準功能並建立連線來查詢名稱。如果您的應用程式已執行這項操作，則不需要修改應用程式即可使用 Service Connect。

  您可以在每個服務和任務定義中提供完整的配置。Amazon ECS 會在每個服務部署中管理此組態的變更，以確保部署中的所有任務都以相同的方式運作。例如，DNS 即服務探索的常見問題是控制移轉。如果您將 DNS 名稱變更為指向新的取代 IP 位址，則可能需要所有用戶端開始使用新服務之前的 TTL 時間上限。使用服務 Connect 時，用戶端部署會透過取代用戶端工作來更新組態。您可以設定部署斷路器和其他部署規劃，以與任何其他部署相同的方式影響 Service Connect 變更。

  如需詳細資訊，請參閱 使用服務 Connect 以簡短名稱連接 Amazon ECS 服務。

- Amazon ECS 服務探索

  另一種 service-to-service 溝通方式是使用服務探索進行直接通訊。在這種方法中，您可以使用與 Amazon ECS 的 AWS Cloud Map 服務探索整合。Amazon ECS 會使用服務探索將啟動的任務清單同步到 AWS Cloud Map，該清單會維護 DNS 主機名稱，該主機名稱可解析為該特定服務中一或多

個任務的內部 IP 位址。Amazon VPC 中的其他服務可以使用此 DNS 主機名稱，使用其內部 IP 位址將流量直接傳送到另一個容器。

這種 service-to-service 通信方法提供了低延遲。容器之間沒有額外的組件。流量直接從一個容器傳輸到另一個容器。

使用awsvpc網絡模式，其中每個任務都有自己唯一的 IP 地址時，這種方法是合適的。大多數軟件僅支持直接解析為 IP 地址的 DNS A 記錄的使用。使用awsvpc網絡模式時，每個任務的 IP 地址都是一A條記錄。但是，如果您使用的是bridge網絡模式，則多個容器可能共享相同的 IP 地址。此外，動態連接埠對應會導致容器在該單一 IP 位址上隨機指派連接埠號碼。在這一點上，A記錄不再足以發現服務。您還必須使用SRV記錄。這種類型的記錄可以追蹤 IP 位址和連接埠號碼，但需要您適當地設定應用程式。您使用的某些預先建置的應用程式可能不支援SRV記錄。

awsvpc網路模式的另一個優點是每個服務都有唯一的安全性群組。您可以將此安全性群組設定為僅允許來自需要與該服務通訊之特定上游服務的連入連線。

使用服務發現直接 service-to-service 通信的主要缺點是，您必須實現額外的邏輯以重試並處理連接故障。DNS 記錄有一個 time-to-live (TTL) 期間，可控制快取的時間長度。DNS 記錄更新和快取過期需要一些時間，這樣您的應用程式才能取得最新版本的 DNS 記錄。因此，您的應用程序最終可能會解析 DNS 記錄以指向不再存在的另一個容器。您的應用程序需要處理重試並具有忽略錯誤後端的邏輯。

如需更多資訊，請參閱使用服務探索將 Amazon ECS 服務與 DNS 名稱連接

網路模式相容性資料表

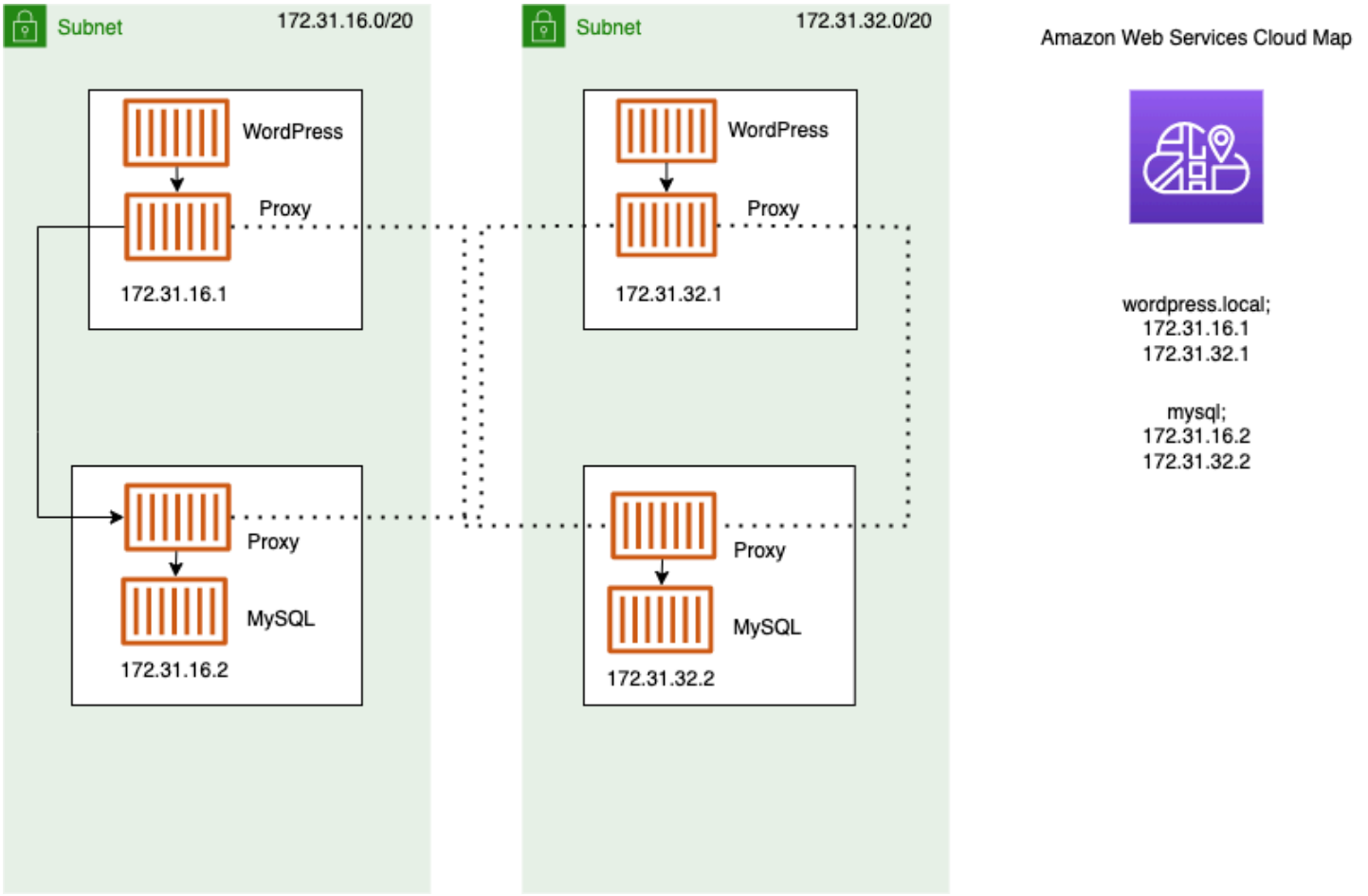
下表說明這些選項與任務網路模式之間的相容性。在資料表中，「用戶端」是指從 Amazon ECS 任務內部建立連線的應用程式。

互連線選項	橋接	**awsvpc**	主機
服務探索	是的，但用戶端需要知道在 DNS 中不含 hostPort 的 SRV 記錄。	是	是的，但用戶端需要知道在 DNS 中不含 hostPort 的 SRV 記錄。
Service Connect	是	是	否

使用服務 Connect 以簡短名稱連接 Amazon ECS 服務

Amazon ECS 服務 Connect 提供 Amazon ECS 配置的 service-to-service 通信管理。它會在 Amazon ECS 中建置服務探索和服務網格。這可在您依服務部署管理的每項服務內部提供完整的組態、在不依賴於 VPC DNS 組態的命名空間內參照服務的統一方式，以及用來監視所有應用程式的標準化指標和記錄。服務 Connect 只會互連服務。

下圖顯示 VPC 和 2 個服務中具有 2 個子網路的 Service Connect 網路範例。在每個子網路中 WordPress 以 1 個工作執行的用戶端服務。一種伺服器服務，在每個子網路中執行具有 1 項任務的 MySQL。這兩種服務都具有高度可用性，並且對任務和可用區域問題具有彈性，因為每個服務都會執行分散在 2 個子網路上的多個任務。實心箭頭顯示從 WordPress MySQL 的連接。例如，從具有 IP 位址的工作中的 WordPress 容器內執行的 `mysql --host=mysql` CLI 命令`172.31.16.1`。該命令在 MySQL 預設連接埠上使用簡短名稱 `mysql`。此名稱和連接埠會在相同任務中連線至 Service Connect Proxy。 WordPress 工作中的 Proxy 會在離群值偵測中使用循環配置資源負載平衡，以及任何先前的失敗資訊來挑選要連線到的 MySQL 工作。如圖中的實心箭頭所示，Proxy 連線至具 IP 地址 `172.31.16.2` MySQL 任務中的第二個 Proxy。在同一任務中，第二個 Proxy 連線至本機 MySQL 伺服器。這兩個代理伺服器都會報告 Amazon ECS 和 Amazon CloudWatch 主控台中圖形中可見的連線效能，以便您以相同的方式從各種應用程式取得效能指標。



以下術語會與 Service Connect 搭配使用。

連接埠名稱

將名稱指派給特定連接埠映射的 Amazon ECS 任務定義組態。此組態僅供 Amazon ECS Service Connect 使用。

用戶端別名

Amazon ECS 服務組態，用於指派端點中使用的連接埠號碼。此外，用戶端別名可以指派端點的 DNS 名稱，以覆寫探索名稱。如果 Amazon ECS 服務中未提供探索名稱，則用戶端別名會覆寫連接埠名稱作為端點名稱。如需端點範例，請參閱端點的定義。您可以將多個用戶端別名指派給 Amazon ECS 服務。此組態僅供 Amazon ECS Service Connect 使用。

探索名稱

此為選用的中繼名稱，您可以從任務定義為指定的連接埠建立此名稱。此名稱是用來建立 AWS Cloud Map 服務。如果未提供此名稱，則會使用任務定義中的連接埠名稱。您可以將多個探索名稱指派給 Amazon ECS 服務的特定連接埠。此組態僅供 Amazon ECS Service Connect 使用。

AWS Cloud Map 服務名稱在命名空間中必須是唯一的。由於此限制，對於每個命名空間中的特定任務定義，如果未提供探索名稱，您只能有一個 Service Connect 組態。

端點

連線到 API 或網站的 URL。URL 包含協定、DNS 名稱和連接埠。如需一般端點的更多資訊，請參閱 Amazon Web Services 一般參考內AWS 詞彙表中的端點。

Service Connect 可建立與 Amazon ECS 服務連線的端點，並將 Amazon ECS 服務中的任務設定為連線到端點。URL 包含協定、DNS 名稱和連接埠。您可以在任務定義中選取協定和連接埠名稱，因為連接埠必須比對容器映像內的應用程式。在服務中，您可以依名稱選取每個連接埠，並且可以指派 DNS 名稱。如果您未在 Amazon ECS 服務組態中指定 DNS 名稱，則會依預設使用任務定義中的連接埠名稱。例如，Service Connect 端點可以是 `http://blog:80`、`grpc://checkout:8080` 或 `http://_db.production.internal:99`。

Service Connect 服務

Amazon ECS 服務中單一端點的組態。這是 Service Connect 組態的一部分，包含主控台中 Service Connect and discovery name configuration (Service Connect 和探索名稱組態) 中的單一列，或是 Amazon ECS 服務 JSON 組態內 `services` 清單中的某個物件。此組態僅供 Amazon ECS Service Connect 使用。

如需詳細資訊，請參閱 ServiceConnect Amazon 彈性容器服務 API 參考中的服務。

命名空間

與服務 Connect 搭配使用的 AWS Cloud Map 命名空間的簡短名稱或完整 Amazon 資源名稱 (ARN)。命名空間必須與 Amazon ECS 服務和叢集位於相同 AWS 區域 的名稱。中的命名空間類型 AWS Cloud Map 不會影響服務 Connect。

服務 Connect 使用 AWS Cloud Map 命名空間做為彼此交談之 Amazon ECS 任務的邏輯分組。每個 Amazon ECS 服務只能屬於一個命名空間。命名空間內的服務可以分散 AWS 區域 在同一個群集內的不同 Amazon ECS 叢集。 AWS 帳戶您可以根據任何標準自由組織服務。

用戶端服務

執行網路用戶端應用程式的服務。此服務必須設定命名空間。服務中的每個任務都可以透過 Service Connect Proxy 容器探索，並連線到命名空間中的所有端點。

如果任務中任何容器需要從命名空間中的服務連線到端點，請選擇用戶端服務。如果前端、反向 Proxy 或負載平衡器應用程式透過其他方法 (例如 Elastic Load Balancing) 接收外部流量，則可使用此類型的 Service Connect 組態。

用戶端-伺服器服務

執行網路或 Web 服務應用程式的 Amazon ECS 服務。此服務必須具有命名空間，並且至少已設定一個端點。您可以使用端點連上服務中的每項任務。Service Connect Proxy 容器會接聽端點名稱和連接埠，以將流量引導至任務中的應用程式容器。

如果有任何容器在連接埠上公開並接聽網路流量，請選擇用戶端-伺服器服務。這些應用程式不需要連線到相同命名空間中的其他用戶端-伺服器服務，但需要用戶端設定。後端、中介軟體、業務層或大多數微服務都可以使用這種類型的 Service Connect 組態。如果您希望前端、反向 Proxy 或負載平衡器應用程式接收來自相同命名空間中使用 Service Connect 設定的其他服務流量，則這些服務應使用此類型的 Service Connect 組態。

Service Connect 功能會建立相關服務的虛擬網路。您可以跨多個不同的命名空間使用相同的服務組態，以執行獨立但相同的應用程式集合。Service Connect 會定義 Amazon ECS 服務中的 Proxy 容器。如此一來，您可以使用相同的任務定義，在具有不同 Service Connect 組態的不同命名空間中執行相同的應用程式。服務所執行的每項工作都會在工作中執行 Proxy 容器。

Service Connect 適用於相同命名空間內 Amazon ECS 服務之間的連線。對於下列應用程式，您需要使用額外的互連線方法，才能連線到使用 Service Connect 設定的 Amazon ECS 服務：

- 在其他命名空間中設定的工作
- 未針對服務 Connect 設定的工作
- Amazon ECS 以外的其他應用

這些應用程式可以透過 Service Connect Proxy 連線，但無法解析 Service Connect 端點名稱。

為了讓這些應用程式解析 Amazon ECS 任務的 IP 位址，您需要使用其他互連方法。

定價

Amazon ECS 服務 Connect 定價取決於您使用的是 Amazon EC2 基礎設施還是使用 AWS Fargate Amazon EC2 基礎設施託管容器化工作負載。開啟使用 Amazon ECS 時 AWS Outposts，定價方式與您直接使用 Amazon EC2 時所使用的模型相同。如需詳細資訊，請參閱 Amazon ECS 定價。

AWS Cloud Map 當服務 Connect 使用它時，使用是完全免費的。

Amazon ECS 服務 Connect 組件

使用 Amazon ECS 服務 Connect 時，您可以將每個 Amazon ECS 服務設定為執行接收網路請求 (用戶端伺服器服務) 的伺服器應用程式，或執行發出請求的用戶端應用程式 (用戶端服務)。

當您準備開始使用 Service Connect 時，請先從用戶端-伺服器服務開始。您可以將 Service Connect 組態新增至新服務或現有服務。Amazon ECS 會在命名空間中建立服務 Connect 端點。此外，Amazon ECS 會在服務中建立新部署來取代目前正在執行的任務。

現有任務和其他應用程式可以繼續連線到現有端點和外部應用程式。如果用戶端伺服器服務透過向外延展來新增工作，則來自用戶端的新連線將會在所有工作之間取得平衡。如果用戶端-伺服器服務已更新，來自用戶端的新連線將會在新版本的工作之間取得平衡。

現有任務無法解析並連線到新端點。只有在相同命名空間中具有 Service Connect 組態且在此部署後開始執行的新工作才能解析並連線至此端點。

這意味著用戶端應用程式的運算子會判斷其應用程式的組態何時變更，即使伺服器應用程式的運算子可以隨時變更其組態也是如此。每次部署命名空間中的任何服務時，命名空間中的端點清單都可以變更。現有工作和取代工作的行為會繼續與最近一次部署之後的行為相同。

請考慮下列範例。

首先，假設您要在單一 AWS CloudFormation 範本和單一 AWS CloudFormation 堆疊中建立可供公用網際網路使用的應用程式。公共發現和可達性應該由最後創建 AWS CloudFormation，包括前端客戶端服務。需要按照此順序建立該服務，以防止在一段時間內，前端用戶端服務正在執行並可公開使用，但後端用戶端服務不是。如此可避免在該期間向大眾傳送錯誤訊息。在中 AWS CloudFormation，您必須使`dependsOn`用指示無法 parallel 或同時建立多個 Amazon ECS 服務。 AWS CloudFormation 您應該為用戶端任務連線的每個後端用戶端-伺服器服務，將 `dependsOn` 新增至前端用戶端服務。

其次，假設前端服務存在，但沒有 Service Connect 組態。任務正在連線到現有的後端服務。請先使用 DNS 中或前端使用的 `clientAlias` 相同名稱，將用戶端-伺服器 Service Connect 組態新增至後端服務。這會建立新的部署，因此所有部署回復偵測 AWS Management Console，或者 AWS CLI， AWS SDK 和其他方法來回復後端服務，並將後端服務還原至先前的部署和設定。如果您對後端服務的效能和行為感到滿意，請將用戶端或用戶端-伺服器 Service Connect 組態新增至前端服務。只有新部署中的任務會使用在這些新任務中新增的 Service Connect Proxy。如果您在使用此組態時遇到問題，您可以使用部署復原偵測或 AWS Management Console、 AWS SDK 和其他方法來復原後端服務 AWS CLI，並將後端服務還原至先前的部署和組態，以還原至先前的組態。如果您使用另一個以 DNS 為基礎的服務探索系統，而非 Service Connect，則任何前端或用戶端應用程式會在本機 DNS 快取到期後開始使用新端點和變更的端點組態，該過程通常需要數小時。

聯網

依預設，服務連線 Proxy 會接聽`containerPort`來自作業定義 Connect 埠對應的。您的安全群組規則必須允許從用戶端執行的子網路傳入 (輸入) 流量至此連接埠。

即使您在 Service Connect 服務組態中設定連接埠號碼，也不會變更 Service Connect Proxy 所接聽用戶端-伺服器服務的連接埠。當您設定此連接埠號碼時，Amazon ECS 會在這些任務內的 Service Connect Proxy 上變更用戶端服務連線到其中的端點連接埠。用戶端服務中的 Proxy 會使用 `containerPort` 連線至用戶端-伺服器服務中的 Proxy。

如果您想要變更 Service Connect Proxy 接聽的連接埠，請變更用戶端-伺服器服務 Service Connect 組態中的 `ingressPortOverride`。如果您變更此連接埠號碼，則必須允許此連接埠上由此服務的流量所使用的輸入流量。

應用程式傳送到針對 Service Connect 設定的 Amazon ECS 服務的流量，會要求 Amazon VPC 和子網路具有路由表規則和網路 ACL 規則，以允許您正在使用的 `containerPort` 和 `ingressPortOverride` 連接埠號碼。

您可以使用服務 Connect 在 VPC 之間傳送流量。路由表規則、網路 ACL 和安全性群組的相同需求適用於這兩個 VPC。

例如，兩個叢集會在不同的 VPC 中建立任務。每個叢集中的服務會設定為使用相同的命名空間。這兩個服務中的應用程式無需任何 VPC DNS 組態，即可解析命名空間中的每個端點。但是，除非 VPC 對等互連、VPC 或子網路路由表以及 VPC 網路 ACL 允許和連接埠號碼上的流量，否則 Proxy 無法連線。`containerPort` `ingressPortOverride`

對於使用bridge網路模式的工作，您必須建立具有輸入規則的安全性群組，該規則允許較高動態連接埠範圍上的流量。然後，將安全群組指派給服務 Connect 叢集中的所有 EC2 執行個體。

Service Connect Proxy

如果您使用服務 Connect 組態建立或更新服務，Amazon ECS 會在每個新任務啟動時新增一個新的容器。這種使用單獨容器的模式稱為 `sidecar`。此容器在任務定義中不存在，您無法對其進行設定。Amazon ECS 會管理服務中的容器組態。這可讓您在不使用 Service Connect 的情況下，在多個服務、命名空間和工作之間重複使用相同的工作定義。

Proxy 資源

- 對於工作定義，您必須設定 CPU 和記憶體參數。

  我們建議在 Service Connect Proxy 容器的任務 CPU 和記憶體中，新增 256 個 CPU 單元和至少 64 MiB 記憶體。在 AWS Fargate 上，您可以設定的最低記憶體容量是 512 MiB 的記憶體。在 Amazon EC2 上，需要任務定義內存。
- 對於服務，您可以在「服務 Connect」配置中設置日誌配置。

- 如果您希望此服務中的任務在尖峰負載時每秒接收的請求數超過 500 個，建議在此 Service Connect Proxy 容器的任務定義中，將 512 個 CPU 單元新增至任務 CPU。
- 如果您希望在命名空間中建立超過 100 個 Service Connect 服務，或在命名空間內所有 Amazon ECS 服務中建立總共 2000 個任務，建議在 Service Connect Proxy 容器的任務記憶體中新增 128 MiB 記憶體。您應該在命名空間中所有 Amazon ECS 服務使用的每個任務定義中執行此操作。

代理組態

您的應用程式會以與應用程式所在的相同任務連線至附屬容器中的 Proxy。Amazon ECS 會設定任務和容器，以便應用程式僅在應用程式連線至相同命名空間中的端點名稱時才連線到 Proxy。所有其他流量都不會使用 Proxy。其他流量包括相同 VPC 中的 IP 位址、 AWS 服務端點和外部流量。

負載平衡

服務連線會將 Proxy 設定為使用循環配置策略，在 Service Connect 端點中的任務之間進行負載平衡。位於連線來源任務中的本機 Proxy，會挑選提供端點之用戶端-伺服器服務中的其中一項任務。

例如，假設在名為 local 的命名空間 WordPress 中設定為用戶端服務的服務中執行的工作。有另一個包含具有執行 MySQL 資料庫 2 個任務的服務。此服務設定為在相同命名空間中透過 Service Connect 提供名為 `mysql` 的端點。在 WordPress 工作中，應用 WordPress 程式會使用端點名稱連線到資料庫。與此名稱的連線會移至在相同工作的附屬容器中執行的 Proxy。然後，Proxy 可以使用循環配置策略連線至任一 MySQL 任務。

負載平衡策略：循環配置

異常值偵測

此功能會使用 Proxy 先前連線失敗的相關資料，以避免傳送新的連線至連線失敗的主機。Service Connect 會設定 Proxy 的異常值偵測功能，以提供被動運作狀態檢查。

使用前面的例子，代理可以連接到任一 MySQL 任務。如果 Proxy 與特定 MySQL 任務進行了多個連線，並且在過去的 30 秒內有 5 次以上的連線失敗，那麼 Proxy 會在 30 到 300 秒內避免該 MySQL 任務。

重試

Service Connect 會將 Proxy 設定為重試透過 Proxy 傳遞且失敗的連線，而第二次嘗試則避免使用先前連線的主機。這可確保透過 Service Connect 的每個連線都不會因為一次性原因而失敗。

重試次數：2

逾時

Service Connect 會設定 Proxy 等待用戶端-伺服器應用程式回應的最長時間。預設逾時值為 15 秒，但可以更新。

可選參數：

閒置逾時-閒置時連線保持作用中狀態的時間 (以秒為單位)。0停用的值idleTimeout。

HTTP/HTTP2/的idleTimeout預設值GRPC為 5 分鐘。

idleTimeout預設值TCP為 1 小時。

per RequestTimeout-等待上游回應每個請求完整回應的時間量。的值0會關閉perRequestTimeout。這只能在appProtocol應用程序容器是HTTP/HTTP2/時設置GRPC。預設值為 15 秒。

> **Note**
>
> 如果設定idleTimeout為小於的時間perRequestTimeout，連線將會在到達時關閉，而不idleTimeout是perRequestTimeout。

考量事項

使用服務 Connect 時，請考量下列事項：

- 在 Fargate 中運行的任務必須使用 Fargate Linux 平台1.4.0或更高版本才能使用服務 Connect。
- 容器執行個體上的 Amazon ECS 代理程式版本必須等於1.67.2或更高版本。
- 容器執行個體必須執行 Amazon ECS 最佳化 Amazon Linux 2023 AMI 版本 20230428 或更高版本，或 Amazon ECS 最佳化 Amazon Linux 2 AMI 版本 2.0.20221115，才能使用 Service Connect。除 Amazon ECS 容器代理程式外，這些版本也具有 Service Connect 代理程式。如需有關服務 Connect 代理程式的詳細資訊，請參閱上 GitHub的 Amazon ECS 服務 Connect 代理程式。
- 容器執行個體必須具有資源 arn:aws:ecs:*region*:*0123456789012*:task-set/*cluster*/* 的 ecs:Poll 許可。如果您使用的是 ecsInstanceRole，則不需要新增其他許可。AmazonEC2ContainerServiceforEC2Role 受管政策具有必要的許可。如需詳細資訊，請參閱 Amazon ECS 容器執行個體 IAM 角色。
- Service Connect 只支援使用滾動部署的服務。

- 使用`bridge`網路模式並使用服務 Connect 的工作不支援`hostname`容器定義參數。
- 任務定義必須設定任務記憶體限制，才能使用 Service Connect。如需詳細資訊，請參閱 Service Connect Proxy。
- 不支援設定容器記憶體限制的工作定義。

  您可以在容器上設定容器記憶體限制，但是您必須將任務記憶體限制設為大於容器記憶體限制總和的數字。任務限制中未在容器限制中配置的其他 CPU 和記憶體，會由 Service Connect Proxy 容器和其他未設定容器限制的容器使用。如需詳細資訊，請參閱 Service Connect Proxy。
- 您可以將服務連線設定為使用相同區域中相同區域中的任何 AWS Cloud Map 命名空間 AWS 帳戶。
- 每個服務只能屬於一個命名空間。
- 僅支援服務建立的工作。
- 所有端點在命名空間內必須不重複。
- 所有探索名稱在命名空間內必須不重複。
- 您必須先重新部署現有服務，應用程式才能解析新的端點。不會將最近部署後新增至命名空間的新端點新增至任務組態。如需詳細資訊，請參閱 the section called “服務 Connect 元件”。
- 刪除叢集時，服務 Connect 不會刪除命名空間。您必須在中刪除命名空間 AWS Cloud Map。
- Application Load Balancer 流量預設為透過網路模式下的 Service Connect 代理程式`awsvpc`路由。如果您想要非服務流量略過服務連線代理程式，請使用 Service Connect 服務組態中的`ingressPortOverride`參數。

服務 Connect 不支援下列項目：

- Windows 容器
- HTTP
- 獨立工作
- 使用藍色/綠色和外部部署類型的服務
- Service Connect 不支援 Amazon ECS Anywhere 的 `External` 容器執行個體。
- PPV2

提供 Service Connect 的地區

Amazon ECS 服務 Connect 在以下 AWS 區域提供：

區域名稱	區域
美國東部 (俄亥俄)	us-east-2
美國東部 (維吉尼亞北部)	us-east-1
美國西部 (加利佛尼亞北部)	us-west-1
美國西部 (奧勒岡)	us-west-2
非洲 (開普敦)	af-south-1
亞太區域 (香港)	ap-east-1
亞太區域 (雅加達)	ap-southeast-3
亞太區域 (孟買)	ap-south-1
亞太區域 (海德拉巴)	ap-south-2
亞太區域 (大阪)	ap-northeast-3
亞太區域 (首爾)	ap-northeast-2
亞太區域 (新加坡)	ap-southeast-1
亞太區域 (雪梨)	ap-southeast-2
亞太區域 (墨爾本)	ap-southeast-4
亞太區域 (東京)	ap-northeast-1
加拿大 (中部)	ca-central-1
加拿大西部 (卡加利)	ca-west-1
中國 (北京)	cn-north-1 (注意：服務 Connect 的 TLS 不適用於此區域。)
中國 (寧夏)	cn-西北 -1 (注意：服務 Connect 的 TLS 在此地區不提供。)

區域名稱	區域
歐洲 (法蘭克福)	eu-central-1
歐洲 (愛爾蘭)	eu-west-1
歐洲 (倫敦)	eu-west-2
歐洲 (巴黎)	eu-west-3
歐洲 (米蘭)	eu-south-1
歐洲 (西班牙)	eu-south-2
歐洲 (斯德哥爾摩)	eu-north-1
歐洲 (蘇黎世)	eu-central-2
以色列 (特拉維夫)	il-central-1
中東 (巴林)	me-south-1
中東 (阿拉伯聯合大公國)	me-central-1
南美洲 (聖保羅)	sa-east-1

Amazon ECS 服務 Connect 配置概觀

當您使用服務 Connect 時，您需要在資源中設定一些參數。

需要為服務 Connect 設定的 Amazon ECS 資源

參數位置	應用程式類型	描述	必要
任務定義	用戶端	用戶端任務定義中的 Service Connect 沒有可用的變更。	N/A
任務定義	用戶端-伺服器	伺服器必須將 name 欄位新增至容器 `portMappings` 中的連接埠。如需更多資訊，請參閱`portMappings`	是

參數位置	應用程式類型	描述	必要
任務定義	用戶端-伺服器	伺服器可以選擇性地提供應用程式協定 (例如 HTTP)，以接收其伺服器應用程式 (例如 HTTP 5xx) 的協定特定指標。	否
服務定義	用戶端	用戶端服務必須新增 serviceConnectConfiguration，才能設定要加入的命名空間。此命名空間必須包含此服務需要探索的所有伺服器服務。如需詳細資訊，請參閱「serviceConnectConfiguration」。	是
服務定義	用戶端-伺服器	伺服器服務必須新增 serviceConnectConfiguration，才能設定可從中使用服務的 DNS 名稱、連接埠號碼和命名空間。如需詳細資訊，請參閱「serviceConnectConfiguration」。	是
叢集	用戶端	叢集可以新增預設的 Service Connect 命名空間。在服務中設定 Service Connect 時，叢集中的新服務會繼承命名空間。	否
叢集	用戶端-伺服器	在套用至伺服器服務的叢集中，Service Connect 沒有可用的變更。伺服器任務定義和服務必須設定相應的組態。	N/A

Service Connect 的設定步驟概觀

下列步驟提供如何設定服務 Connect 的概觀。

Important

- 服務 Connect 會在您的帳戶中建立 AWS Cloud Map 服務。透過手動註冊/取消註冊執行個體、變更執行個體屬性或刪除服務來修改這些 AWS Cloud Map 資源，可能會導致應用程式流量或後續部署產生非預期的行為。
- 服務 Connect 不支援工作定義中的連結。

1. 將連接埠名稱新增至任務定義中的連接埠對映。此外，您可以識別應用程式的第 7 層協定，以取得其他指標。
2. 建立具有 AWS Cloud Map 命名空間的叢集或個別建立命名空間。對於簡單的組織，請使用您想要的命名空間名稱建立叢集，並為命名空間指定相同的名稱。在這種情況下，Amazon ECS 會使用必要的組態建立新的 HTTP 命名空間。服務 Connect 不會在 Amazon 路線 53 中使用或創建 DNS 託管區域。
3. 設定服務以在命名空間內建立 Service Connect 端點。
4. 部署服務以建立端點。Amazon ECS 會將 Service Connect Proxy 容器新增至每項任務，並在 AWS Cloud Map中建立 Service Connect 端點。此容器未在任務定義中設定，而且任務定義可以重複使用而不需要修改，以在相同的命名空間或多個命名空間中建立多個服務。
5. 將用戶端應用程式部署為服務，以連線到端點。Amazon ECS 會在每項任務中透過 Service Connect Proxy，將這些應用程式連線至 Service Connect 端點。

   應用程式僅使用 Proxy 來連線至 Service Connect 端點。沒有其他設定可使用 Proxy。Proxy 會執行循環配置負載平衡、異常值偵測和重試。如需 Proxy 的詳細資訊，請參閱 Service Connect Proxy。
6. 通過 Amazon 中的服務 Connect 代理監控流量 CloudWatch。

叢集組態

您可以在建立或更新叢集時，為服務 Connect 設定預設命名空間。如果您指定的命名空間名稱在相同的 AWS 區域 和帳戶中不存在，則會建立新的 HTTP 命名空間。

如果您建立叢集並指定預設的 Service Connect 命名空間，則叢集會在 Amazon ECS 建立命名空間時在 `PROVISIONING` 狀態中等待。您可以看到 `attachment` 所處的叢集狀態顯示命名空間的狀態。依預設，附件不會顯示在中 AWS CLI，您必須新增才能看`--include ATTACHMENTS`到它們。

服務配置

Service Connect 針對最低組態要求而設計。您必須為要在任務定義中與 Service Connect 搭配使用的每個連接埠映射設定名稱。在服務中，您需要開啟 Service Connect，然後選取命名空間來建立用戶端服務。若要建立用戶端-伺服器服務，您需要新增符合其中一個連接埠映射名稱的單一 Service Connect 服務組態。Amazon ECS 會重複使用任務定義中的連接埠號碼和連接埠名稱，以定義 Service Connect 服務和端點。若要覆寫這些值，您可以在主控台中使用 Discovery (探索)、DNS 和 Port (連接埠) 等其他參數，或分別在 Amazon ECS API 中使用 `discoveryName` 和 `clientAliases`。

下列範例示範在相同的 Amazon ECS 服務中一起使用的各種 Service Connect 組態。雖然會提供 Shell 註解，但請注意，用於 Amazon ECS 服務的 JSON 組態不支援註解。

```
{
    ...
    serviceConnectConfiguration: {
        enabled: true,
        namespace: "internal",
        #config for client services can end here, only these two parameters are
 required.
        services: [{
                portName: "http"
            }, #minimal client - server service config can end here.portName must match
 the "name"
            parameter of a port mapping in the task definition. {
                discoveryName: "http-second"
                #name the discoveryName to avoid a Task def port name collision with
 the minimal config in the same Cloud Map namespace
                portName: "http"
            },
            {
                clientAliases: [{
                        dnsName: "db",
                        port: 81
                    }] #use when the port in Task def is not the port that client apps
 use.Client apps can use http: //db:81 to connect
                    discoveryName: "http-three"
                portName: "http"
            },
            {
                clientAliases: [{
                    dnsName: "db.app",
                    port: 81
```

```
                }] #use when the port in Task def is not the port that client apps
 use.duplicates are fine as long as the discoveryName is different.
                discoveryName: "http-four"
                portName: "http",
                ingressPortOverride: 99 #If App should also accept traffic directly on
 Task def port.
            }
        ]
    }
}
```

加密 Amazon ECS 服務 Connect 流量

Amazon ECS 服務 Connect 支援使用 Amazon ECS 服務的傳輸層安全性 (TLS) 憑證的自動流量加密。當您將 Amazon ECS 服務指向 AWS Private Certificate Authority (AWS Private CA) 時，Amazon ECS 會自動佈建 TLS 憑證，以加密 Amazon ECS 服務 Connect 服務之間的流量。Amazon ECS 會產生、輪換和分發用於流量加密的 TLS 憑證。

Service Connect 的自動流量加密使用領先業界的加密功能來保護您的服務間通訊，協助您符合安全性需求。它支持帶有256-bit ECDSA和2048-bit RSA加密的 AWS Private Certificate Authority TLS 證書。根據預設，支援 TLS 1.3，但不支援 TLS 1.0-1.2。您也可以完全控制私有憑證和簽署金鑰，以協助您符合合規性需求。

> **Note**
>
> 若要使用 TLS 1.3，您必須在目標的接聽程式上啟用它。
> 只有透過 Amazon ECS 代理程式傳遞的入站和輸出流量才會加密。

AWS Private Certificate Authority 憑證和服務 Connect

發行憑證需要額外的 IAM 許可。Amazon ECS 提供受管的資源信任政策，概述了許可集。如需有關此政策的詳細資訊，請參閱 AmazonEC InfrastructureRole PolicyFor ServiceConnect TransportLayer s 安全性。

AWS Private Certificate Authority 服務 Connect 模式

AWS Private Certificate Authority 可以在兩種模式下運行：通用和短命。

- 一般用途-可以設定任何到期日的憑證。

- 短期憑證，有效期上限為七天。

雖然 Amazon ECS 支援這兩種模式，但我們建議您使用短期憑證。根據預設，憑證每五天輪換一次，而在短期模式下執行可大幅節省一般用途的成本。

Service Connect 不支援憑證撤銷，而是利用具有頻繁憑證輪換的短期憑證。您有權使用 Secrets Manager 中的受管理輪換來修改輪換頻率、停用或刪除密碼，但這樣做可能會產生下列可能的後果。

- 更短的循環頻率-較短的旋轉頻率會產生更高的成本 AWS Private CA，因為 AWS KMS 和 Secrets Manager 和 Auto Scaling 經歷了輪換工作負載增加。
- 更長的旋轉頻率-如果旋轉頻率超過七天，則應用程式的通訊會失敗。
- 刪除機密-刪除機密會導致輪替失敗，並影響客戶應用程式通訊。

如果您的秘密輪替失敗，則會在中發布一個`RotationFailed`事件AWS CloudTrail。您也可以為的設定CloudWatch 鬧鐘`RotationFailed`。

> **Important**
>
> 不要將複本區域添加到秘密中。這樣做可以防止 Amazon ECS 刪除密碼，因為 Amazon ECS 沒有從複寫中移除區域的權限。如果您已新增複寫，請執行下列命令。
>
> ```
> aws secretsmanager remove-regions-from-replication \
>  --secret-id SecretId \
>  --remove-replica-regions region-name
> ```

下屬憑證授權單位

您可以將任何 AWS Private CA根目錄或從屬端帶到服務 Connect TLS，以發行服務的最終實體憑證。所提供的發行人在任何地方都被視為簽署者和信任根。您可以從不同的下屬 CA 為應用程式的不同部分發行最終實體憑證。使用時 AWS CLI，請提供 CA 的 Amazon 資源名稱 (ARN) 以建立信任鏈。

內部部署憑證授

若要使用內部部署 CA，請在 AWS Private Certificate Authority中建立並設定從屬 CA。這可確保針對 Amazon ECS 工作負載發行的所有 TLS 憑證與您在內部部署執行的工作負載共用信任鏈，並且能夠安全地連線。

> **Important**
>
> **`AmazonECSManaged : true`**在您的 AWS Private CA.

基礎設施即程式碼

將服務 Connect TLS 與基礎結構即程式碼 (IaC) 工具搭配使用時，請務必正確設定相依性，以避免問題，例如服務卡在耗盡中。您的 AWS KMS 金鑰 (如果提供的話) IAM 角色和 AWS Private CA 相依性應在 Amazon ECS 服務之後刪除。

服務 Connect 和 AWS Key Management Service

您可以使用AWS Key Management Service來加密和解密您的服務 Connect 資源。 AWS KMS 是一項管理的服務，您 AWS 可以在其中製作和管理保護資料的加密金鑰。

AWS KMS 搭配 Service Connect 使用時，您可以選擇使用為您 AWS 管理的 AWS 擁有金鑰，也可以選擇現有的 AWS KMS 金鑰。您也可以建立要使用的新 AWS KMS 金鑰。

提供您自己的加密金鑰

您可以提供自己的金鑰材料，也可以透過將自己的金鑰 AWS Key Management Service 匯入來使用外部金鑰存放區 AWS KMS，然後在 Amazon ECS 服務 Connect 中指定該金鑰的 Amazon 資源名稱 (ARN)。

以下是一個示例 AWS KMS 策略。將*#####*值取代為您自己的值。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "id",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::111122223333:role/role-name"
      },
      "Action": [
        "kms:Encrypt",
        "kms:Decrypt",
        "kms:GenerateDataKey",
        "kms:GenerateDataKeyPair"
```

```
      ],
      "Resource": "*"
    }
  ]
}
```

如需關鍵原則的詳細資訊，請參閱AWS Key Management Service 開發人員指南中的建立金鑰政策。

> **Note**
>
> 服務 Connect 僅支援對稱加密 AWS KMS 金鑰。您無法使用任何其他類型的 AWS KMS 金鑰來加密您的服務 Connect 資源。如需判斷 AWS KMS 金鑰是否為對稱加密金鑰的說明，請參閱識別對稱和非對稱 AWS KMS金鑰。

如需 AWS Key Management Service 對稱加密金鑰的詳細資訊，請參閱AWS Key Management Service 開發人員指南中的對稱加密 AWS KMS 金鑰。

為 Amazon ECS 服務 Connect 啟用 TLS

您可以在建立或更新服務 Connect 服務時啟用流量加密。

為現有命名空間中的服務啟用流量加密 AWS Management Console

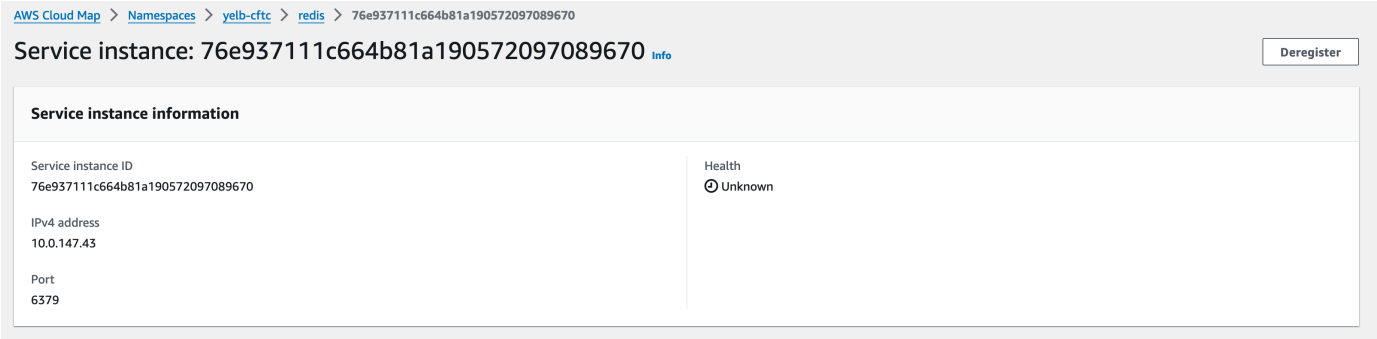
1. 開啟主控台，網址為 https://console.aws.amazon.com/ecs/v2。
2. 在導覽窗格中，選擇 Namespaces (命名空間)。
3. 選擇您要啟用流量加密的服務的命名空間。
4. 選擇您要啟用流量加密的服務。
5. 選擇右上角的「更新服務」，然後向下滾動到「服務 Connect」部分。
6. 選擇服務資訊下的 [開啟流量加密] 以啟用 TLS。
7. 對於服務 Connect TLS 角色，請選擇現有角色或建立新角色。
8. 針對簽署者憑證授權單位，選擇現有的憑證授權單位或建立新的憑證授權單位。
9. 在 [選擇] 中 AWS KMS key，選擇 AWS 擁有和受管理的金鑰，或者您可以選擇不同的金鑰。您也可以選擇建立一個新的。

如需使用為您的服務設定 TLS 的範例，設定 Amazon ECS 服務與 AWS CLI. AWS CLI

確認已為 Amazon ECS 服務 Connect 啟用 TLS

服務 Connect 會在服務 Connect 代理程式中起始 TLS，並在目的地代理程式終止它。因此，應用程式程式碼永遠不會看到 TLS 互動。請使用下列步驟來確認 TLS 已啟用。

1. 確定您的應用程式映像具有 `openssl` CLI。
2. 在您的服務上啟用 ECS Exec，以透過 SSM 連線到您的工作。或者，您可以在與服務相同的 Amazon VPC 中啟動 Amazon EC2 執行個體。
3. 從您要驗證的服務擷取工作的 IP 和連接埠。例如，如果您的`redis`服務已開啟 TLS，您可以透過導覽至、尋找服務 AWS Cloud Map，以及查看某個執行個體的 IP 和連接埠來擷取其任務 IP。



4. 使用類似以下示`execute-command`例中的任何任務登錄。或者，登入步驟 2 中建立的 Amazon EC2 執行個體。

```
$ aws ecs execute-command --cluster cluster-name \
    --task < TASK_ID>  \
    --container app \
    --interactive \
    --command "/bin/sh"
```

> **Note**
>
> 直接呼叫 DNS 名稱並不會顯示憑證。

5. 在連線的命令介面中，使用 `openssl` CLI 驗證並檢視連接至工作的憑證。

   範例：

```
openssl s_client -connect 10.0.147.43:6379 < /dev/null 2> /dev/null \
| openssl x509 -noout -text
```

回應範例：

```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            <serial-number>
        Signature Algorithm: ecdsa-with-SHA256
        Issuer: <issuer>
        Validity
            Not Before: Jan 23 21:38:12 2024 GMT
            Not After : Jan 30 22:38:12 2024 GMT
        Subject: <subject>
        Subject Public Key Info:
            Public Key Algorithm: id-ecPublicKey
                Public-Key: (256 bit)
                pub:
                    <pub>
                ASN1 OID: prime256v1
                NIST CURVE: P-256
        X509v3 extensions:
            X509v3 Subject Alternative Name:
                DNS:redis.yelb-cftc
            X509v3 Basic Constraints:
                CA:FALSE
            X509v3 Authority Key Identifier:
                keyid:<key-id>

            X509v3 Subject Key Identifier:
                1D:<id>
            X509v3 Key Usage: critical
                Digital Signature, Key Encipherment
            X509v3 Extended Key Usage:
                TLS Web Server Authentication, TLS Web Client Authentication
    Signature Algorithm: ecdsa-with-SHA256
        <hash>
```

設定 Amazon ECS 服務與 AWS CLI

您可以為使用服務 Connect 的 Fargate 任務建立 Amazon ECS 服務。 AWS CLI

必要條件

以下是「服務 Connect」的先決條件：

- 確認「區域」支援「服務 Connect」。如需詳細資訊，請參閱 Regions with Service Connect。
- 確認已安裝並設定最新版本的。 AWS CLI 如需詳細資訊，請參閱安裝 AWS Command Line Interface。
- 您的 AWS 使用者具有 亞馬遜 FullAccess IAM 政策範例中指定的必要許可。
- 您已建立要使用的 VPC、子網路、路由表和安全群組。如需詳細資訊，請參閱 the section called “建立 Virtual Private Cloud”。
- 您具有名為 `ecsTaskExecutionRole` 的任務執行角色，且 `AmazonECSTaskExecutionRolePolicy` 受管政策已附加至該角色。此角色允許 Fargate 將 NGINX 應用程式日誌和服務 Connect 代理日誌寫入 Amazon 日誌。 CloudWatch 如需詳細資訊，請參閱 建立任務執行 角色。

步驟 1：建立叢集

使用下列步驟建立 Amazon ECS 叢集和命名空間。

若要建立 Amazon ECS 叢集和 AWS Cloud Map 命名空間

1. 建立要使用的名為 `tutorial` 的 Amazon ECS 叢集。參數 `--service-connect-defaults` 會設定叢集的預設命名空間。在範例輸出中，此帳戶中`service-connect`不存在名稱的 AWS Cloud Map 命名空間 AWS 區域，因此命名空間是由 Amazon ECS 建立的。該命名空間是在帳戶的 AWS Cloud Map 中建立，並且對所有其他命名空間都可見，因此請使用可指示此用途的名稱。

   ```
   aws ecs create-cluster --cluster-name tutorial --service-connect-defaults
    namespace=service-connect
   ```

   輸出：

   ```
   {
       "cluster": {
           "clusterArn": "arn:aws:ecs:us-west-2:123456789012:cluster/tutorial",
           "clusterName": "tutorial",
           "serviceConnectDefaults": {
               "namespace": "arn:aws:servicediscovery:us-
   west-2:123456789012:namespace/ns-EXAMPLE"
           },
   ```

```
        "status": "PROVISIONING",
        "registeredContainerInstancesCount": 0,
        "runningTasksCount": 0,
        "pendingTasksCount": 0,
        "activeServicesCount": 0,
        "statistics": [],
        "tags": [],
        "settings": [
            {
                "name": "containerInsights",
                "value": "disabled"
            }
        ],
        "capacityProviders": [],
        "defaultCapacityProviderStrategy": [],
        "attachments": [
            {
                "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
                "type": "sc",
                "status": "ATTACHING",
                "details": []
            }
        ],
        "attachmentsStatus": "UPDATE_IN_PROGRESS"
    }
}
}
```

2. 確認已建立叢集：

```
aws ecs describe-clusters --clusters tutorial
```

輸出：

```
{
    "clusters": [
        {
            "clusterArn": "arn:aws:ecs:us-west-2:123456789012:cluster/tutorial",
            "clusterName": "tutorial",
            "serviceConnectDefaults": {
                "namespace": "arn:aws:servicediscovery:us-
west-2:123456789012:namespace/ns-EXAMPLE"
            },
```

```
            "status": "ACTIVE",
            "registeredContainerInstancesCount": 0,
            "runningTasksCount": 0,
            "pendingTasksCount": 0,
            "activeServicesCount": 0,
            "statistics": [],
            "tags": [],
            "settings": [],
            "capacityProviders": [],
            "defaultCapacityProviderStrategy": []
        }
    ],
    "failures": []
}
```

3. (選擇性) 確認已在中建立命名空間 AWS Cloud Map。您可以使用 AWS Management Console 或一般 AWS CLI 模型組態，因為它是在中建立的 AWS Cloud Map。

   例如，使用 AWS CLI：

   ```
   aws servicediscovery --region us-west-2 get-namespace --id ns-EXAMPLE
   ```

   輸出：

   ```
   {
       "Namespace": {
           "Id": "ns-EXAMPLE",
           "Arn": "arn:aws:servicediscovery:us-west-2:123456789012:namespace/ns-
   EXAMPLE",
           "Name": "service-connect",
           "Type": "HTTP",
           "Properties": {
               "DnsProperties": {
                   "SOA": {}
               },
               "HttpProperties": {
                   "HttpName": "service-connect"
               }
           },
           "CreateDate": 1661749852.422,
           "CreatorRequestId": "service-connect"
       }
   ```

```
}
```

步驟 2：為伺服器建立服務

Service Connect 功能適用於在 Amazon ECS 上進行多個應用程式的互連線。這些應用程式中至少有一個需要提供要連線的 Web 服務。在此步驟中，您要建立如下項目：

- 會使用未修改的官方 NGINX 容器映像且包含 Service Connect 組態的任務定義。
- Amazon ECS 服務定義可設定服務 Connect 以針對此服務的流量提供服務探索和服務網狀代理。此組態會重複使用叢集組態中的預設命名空間，以減少您為每個服務所做的服務組態數量。
- Amazon ECS 服務。該服務會使用任務定義執行一項任務，並為 Service Connect Proxy 插入額外的容器。Proxy 會在任務定義中容器連接埠映射的連接埠上接聽。在 Amazon ECS 中執行的用戶端應用程式中，用戶端任務中的 Proxy 會接聽任務定義連接埠名稱、服務探索名稱或服務用戶端別名名稱的輸出連線，以及來自用戶端別名的連接埠號碼。

使用服務 Connect 建立 Web 服務

1. 註冊與 Fargate 相容的任務定義，並使用 `awsvpc` 網路模式。請遵循下列步驟：

   a. 使用以下任務定義的內容，建立名為 `service-connect-nginx.json` 的檔案：

      此任務定義透過將 `name` 和 `appProtocol` 參數新增至連接埠映射來設定 Service Connect。使用多個連接埠時，連接埠名稱有助於您識別服務組態中的此連接埠。依預設，連接埠名稱也會用作可探索的名稱，以供命名空間中的其他應用程式使用。

      任務定義包含任務 IAM 角色，因為服務已啟用 ECS Exec。

      > **Important**
      >
      > 此任務定義使用 a `logConfiguration` 將 nginx 輸出從 `stdout` Amazon 日誌發送`stderr`到亞馬遜 CloudWatch 日誌。此工作執行角色沒有建立 CloudWatch 記錄檔群組所需的額外權限。使用或在 CloudWatch 記錄檔中建立記錄群 AWS Management Console 組 AWS CLI。如果您不想將 nginx 日誌發送到 CloudWatch 日誌，則可以刪除. `logConfiguration`
      > 將任務執行角色中的 AWS 帳戶 id 替換為您的 AWS 帳戶 id。

```
{
    "family": "service-connect-nginx",
    "executionRoleArn": "arn:aws:iam::123456789012:role/ecsTaskExecutionRole",
    "taskRoleArn": "arn:aws:iam::123456789012:role/ecsTaskRole",
    "networkMode": "awsvpc",
    "containerDefinitions": [
        {
        "name": "webserver",
        "image": "public.ecr.aws/docker/library/nginx:latest",
        "cpu": 100,
        "portMappings": [
            {
                "name": "nginx",
                "containerPort": 80,
                "protocol": "tcp",
                "appProtocol": "http"
            }
        ],
        "essential": true,
        "logConfiguration": {
            "logDriver": "awslogs",
            "options": {
                "awslogs-group": "/ecs/service-connect-nginx",
                "awslogs-region": "region",
                "awslogs-stream-prefix": "nginx"
            }
        }
        }
    ],
    "cpu": "256",
    "memory": "512"
}
```

b. 使用 service-connect-nginx.json 檔案註冊任務定義：

```
aws ecs register-task-definition --cli-input-json file://service-connect-nginx.json
```

2. 建立服務：

a. 使用您要建立的 Amazon ECS 服務的內容，建立名為 service-connect-nginx-service.json 的檔案。此範例使用上個步驟中建立的任務定義。需要 awsvpcConfiguration，因為任務定義範例使用 awsvpc 網路模式。

   當您建立 ECS 服務時，指定 Fargate 啟動類型以及支援 Service Connect 的 LATEST 平台版本。securityGroups 和 subnets 所屬的 VPC 必須具有使用 Amazon ECS 的需求。您可以從 Amazon VPC 主控台取得安全群組和子網路識別碼。

   此服務透過新增 serviceConnectConfiguration 參數來設定 Service Connect。不需要命名空間，因為叢集已設定預設命名空間。在命名空間的 ECS 中執行的用戶端應用程式使用 portName 和 clientAliases 中的連接埠連線至此服務。例如，由於 nginx 在根位置 / 提供歡迎頁面，因此可以使用 http://nginx:80/ 連上此服務。未在 Amazon ECS 中執行或不在相同命名空間中的外部應用程式，可以使用任務的 IP 地址和任務定義中的連接埠號碼，透過 Service Connect Proxy 連上此應用程式。對於您的tls組態，arn請為您awsPcaAuthorityArnkmsKey、您roleArn的和 IAM 角色新增憑證。

   此服務使用logConfiguration將服務連接代理輸出從 Amazon 日誌stdout和傳送stderr到 Amazon CloudWatch 日誌。此工作執行角色沒有建立 CloudWatch 記錄檔群組所需的額外權限。使用或在 CloudWatch 記錄檔中建立記錄群 AWS Management Console 組 AWS CLI。建議您建立此記錄群組，並將 Proxy 記錄儲存在記錄 CloudWatch 檔中。如果您不想將代理日誌發送到日 CloudWatch 誌，則可以刪除logConfiguration.

```
{
    "cluster": "tutorial",
    "deploymentConfiguration": {
        "maximumPercent": 200,
        "minimumHealthyPercent": 0
    },
    "deploymentController": {
        "type": "ECS"
    },
    "desiredCount": 1,
    "enableECSManagedTags": true,
    "enableExecuteCommand": true,
    "launchType": "FARGATE",
    "networkConfiguration": {
        "awsvpcConfiguration": {
            "assignPublicIp": "ENABLED",
            "securityGroups": [
```

```
                "sg-EXAMPLE"
            ],
            "subnets": [
                "subnet-EXAMPLE",
                "subnet-EXAMPLE",
                "subnet-EXAMPLE"
            ]
           }
    },
    "platformVersion": "LATEST",
    "propagateTags": "SERVICE",
    "serviceName": "service-connect-nginx-service",
    "serviceConnectConfiguration": {
        "enabled": true,
        "services": [
            {
                "portName": "nginx",
                "clientAliases": [
                    {
                        "port": 80
                    }
                ],
                "tls": {
                   "issuerCertificateAuthority": {
                      "awsPcaAuthorityArn": "certificateArn"
                   },
                   "kmsKey": "kmsKey",
                   "roleArn": "iamRoleArn"
                }
            }
        ],
        "logConfiguration": {
            "logDriver": "awslogs",
            "options": {
                "awslogs-group": "/ecs/service-connect-proxy",
                "awslogs-region": "region",
                "awslogs-stream-prefix": "service-connect-proxy"
            }
        }
    },
    "taskDefinition": "service-connect-nginx"
}
```

b. 使用以下service-connect-nginx-service.json檔案建立服務：

```
aws ecs create-service --cluster tutorial --cli-input-json file://service-
connect-nginx-service.json
```

輸出：

```
{
    "service": {
        "serviceArn": "arn:aws:ecs:us-west-2:123456789012:service/tutorial/
service-connect-nginx-service",
        "serviceName": "service-connect-nginx-service",
        "clusterArn": "arn:aws:ecs:us-west-2:123456789012:cluster/tutorial",
        "loadBalancers": [],
        "serviceRegistries": [],
        "status": "ACTIVE",
        "desiredCount": 1,
        "runningCount": 0,
        "pendingCount": 0,
        "launchType": "FARGATE",
        "platformVersion": "LATEST",
        "platformFamily": "Linux",
        "taskDefinition": "arn:aws:ecs:us-west-2:123456789012:task-definition/
service-connect-nginx:1",
        "deploymentConfiguration": {
            "deploymentCircuitBreaker": {
                "enable": false,
                "rollback": false
            },
            "maximumPercent": 200,
            "minimumHealthyPercent": 0
        },
        "deployments": [
            {
                "id": "ecs-svc/3763308422771520962",
                "status": "PRIMARY",
                "taskDefinition": "arn:aws:ecs:us-west-2:123456789012:task-
definition/service-connect-nginx:1",
                "desiredCount": 1,
                "pendingCount": 0,
                "runningCount": 0,
```

```
                "failedTasks": 0,
                "createdAt": 1661210032.602,
                "updatedAt": 1661210032.602,
                "launchType": "FARGATE",
                "platformVersion": "1.4.0",
                "platformFamily": "Linux",
                "networkConfiguration": {
                    "awsvpcConfiguration": {
                        "assignPublicIp": "ENABLED",
                        "securityGroups": [
                            "sg-EXAMPLE"
                        ],
                        "subnets": [
                            "subnet-EXAMPLEf",
                            "subnet-EXAMPLE",
                            "subnet-EXAMPLE"
                        ]
                    }
                },
                "rolloutState": "IN_PROGRESS",
                "rolloutStateReason": "ECS deployment ecs-
svc/3763308422771520962 in progress.",
                "failedLaunchTaskCount": 0,
                "replacedTaskCount": 0,
                "serviceConnectConfiguration": {
                    "enabled": true,
                    "namespace": "service-connect",
                    "services": [
                        {
                            "portName": "nginx",
                            "clientAliases": [
                                {
                                    "port": 80
                                }
                            ]
                        }
                    ],
                    "logConfiguration": {
                        "logDriver": "awslogs",
                        "options": {
                            "awslogs-group": "/ecs/service-connect-proxy",
                            "awslogs-region": "us-west-2",
                            "awslogs-stream-prefix": "service-connect-proxy"
                        },
```

```
                        "secretOptions": []
                    }
                },
                "serviceConnectResources": [
                    {
                        "discoveryName": "nginx",
                        "discoveryArn": "arn:aws:servicediscovery:us-
west-2:123456789012:service/srv-EXAMPLE"
                    }
                ]
            }
        ],
        "roleArn": "arn:aws:iam::123456789012:role/aws-service-role/
ecs.amazonaws.com/AWSServiceRoleForECS",
        "version": 0,
        "events": [],
        "createdAt": 1661210032.602,
        "placementConstraints": [],
        "placementStrategy": [],
        "networkConfiguration": {
            "awsvpcConfiguration": {
                "assignPublicIp": "ENABLED",
                "securityGroups": [
                    "sg-EXAMPLE"
                ],
                "subnets": [
                    "subnet-EXAMPLE",
                    "subnet-EXAMPLE",
                    "subnet-EXAMPLE"
                ]
            }
        },
        "schedulingStrategy": "REPLICA",
        "enableECSManagedTags": true,
        "propagateTags": "SERVICE",
        "enableExecuteCommand": true
    }
}
```

您提供的 `serviceConnectConfiguration` 會出現在輸出的第一個部署中。當您以需要對任務進行變更的方式對 ECS 服務進行變更時，Amazon ECS 會建立新的部署。

步驟 3：確認您是否可以連線

若要確認 Service Connect 已設定且正常運作，請依照下列步驟從外部應用程式連線至 Web 服務。然後，請參閱服務 Connect Proxy 建立 CloudWatch 的其他度量。

從外部應用程式連線至 Web 服務

- 使用任務 IP 地址連線至任務 IP 地址和容器連接埠

  使用 AWS CLI 取得作業 ID，使用`aws ecs list-tasks --cluster tutorial`。

  如果子網路和安全群組允許任務定義的連接埠上來自公用網際網路的流量，您可以從電腦連線到公用 IP。然而，公用 IP 無法從「描述任務」中獲取，因此步驟涉及到 Amazon EC2 AWS Management Console 或獲取 elastic network interface AWS CLI 的詳細信息。

  在此範例中，相同 VPC 中的 Amazon EC2 執行個體使用的是任務的私有 IP。該應用程式是 nginx，但 `server: envoy` 標題顯示使用的是 Service Connect Proxy。Service Connect Proxy 正在任務定義中容器連接埠上接聽。

```
$ curl -v 10.0.19.50:80/
*   Trying 10.0.19.50:80...
* Connected to 10.0.19.50 (10.0.19.50) port 80 (#0)
> GET / HTTP/1.1
> Host: 10.0.19.50
> User-Agent: curl/7.79.1
> Accept: */*
>
* Mark bundle as not supporting multiuse
< HTTP/1.1 200 OK
< server: envoy
< date: Tue, 23 Aug 2022 03:53:06 GMT
< content-type: text/html
< content-length: 612
< last-modified: Tue, 16 Apr 2019 13:08:19 GMT
< etag: "5cb5d3c3-264"
< accept-ranges: bytes
< x-envoy-upstream-service-time: 0
<
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
```

```
<style>
    body {
        width: 35em;
        margin: 0 auto;
        font-family: Tahoma, Verdana, Arial, sans-serif;
    }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
```

檢視 Service Connect 指標

服務 Connect 線代理伺服器會在度量中建立應用程式 (HTTP、HTTP2、gRPC 或 TCP 連線) 度 CloudWatch 量。使用 CloudWatch 主控台時，請查看 Amazon ECS 命名空間下的DiscoveryNameDiscoveryName ServiceName、 (、 、 ClusterName)、TargetDiscovery名稱和 (TargetDiscovery名稱、 ServiceName、 ClusterName) 的其他度量維度。如需有關這些指標和維度的詳細資訊，請參閱 Amazon CloudWatch Logs 使用者指南中的檢視可用指標。

使用服務探索將 Amazon ECS 服務與 DNS 名稱連接

您可以選擇性地設定 Amazon ECS 服務，以使用 Amazon ECS 服務探索。服務探索使用 AWS Cloud Map API 動作來管理 Amazon ECS 服務的 HTTP 和 DNS 命名空間。如需詳細資訊，請參閱AWS Cloud Map 開發人員指南 AWS Cloud Map中的內容。

下列 AWS 區域提供服務探索：

區域名稱	區域
美國東部 (維吉尼亞北部)	us-east-1
美國東部 (俄亥俄)	us-east-2
美國西部 (加州北部)	us-west-1
美國西部 (奧勒岡)	us-west-2
非洲 (開普敦)	af-south-1
亞太區域 (香港)	ap-east-1
亞太區域 (孟買)	ap-south-1
亞太區域 (海德拉巴)	ap-south-2
亞太區域 (東京)	ap-northeast-1
亞太區域 (首爾)	ap-northeast-2
亞太區域 (大阪)	ap-northeast-3
亞太區域 (新加坡)	ap-southeast-1
亞太區域 (雪梨)	ap-southeast-2
亞太區域 (雅加達)	ap-southeast-3
亞太區域 (墨爾本)	ap-southeast-4
加拿大 (中部)	ca-central-1
加拿大西部 (卡加利)	ca-west-1
中國 (北京)	cn-north-1
中國 (寧夏)	cn-northwest-1
歐洲 (法蘭克福)	eu-central-1

區域名稱	區域
歐洲 (蘇黎世)	eu-central-2
歐洲 (愛爾蘭)	eu-west-1
歐洲 (倫敦)	eu-west-2
歐洲 (巴黎)	eu-west-3
歐洲 (米蘭)	eu-south-1
歐洲 (斯德哥爾摩)	eu-north-1
以色列 (特拉維夫)	il-central-1
歐洲 (西班牙)	eu-south-2
中東 (阿拉伯聯合大公國)	me-central-1
中東 (巴林)	me-south-1
南美洲 (聖保羅)	sa-east-1
AWS GovCloud (美國東部)	us-gov-east-1
AWS GovCloud (美國西部)	us-gov-west-1

服務探索概念

服務探索包含下列元件：

- 服務探索命名空間：擁有相同網域名稱 (例如 `example.com`) 的服務探索服務的邏輯群組。此為您要向其路由流量的網域名稱。您可以透過呼叫`aws servicediscovery create-private-dns-namespace`命令或在 Amazon ECS 主控台中建立命名空間。您可以使用 `aws servicediscovery list-namespaces` 命令來檢視由目前帳戶所建立的命名空間摘要資訊。如需有關服務探索指令的詳細資訊`create-private-dns-namespace`，請參閱 AWS Cloud Map (服務探索) AWS CLI 參考指南*`list-namespaces`*中的和。
- 服務探索服務：存在於服務探索命名空間中，由命名空間的服務名稱和 DNS 設定組成。它提供以下核心元件：

  - 服務登錄：可讓您透過 DNS 或 AWS Cloud Map API 動作查詢服務，並取回一或多個可用來連線至服務的端點。
- 服務探索執行個體：存在於服務探索服務內，由服務目錄中每個 Amazon ECS 服務相關聯的屬性組成。
  - 執行個體屬性：針對每個設定為使用的服務探索的 Amazon ECS 服務，以下中繼資料會新增為自訂屬性：
    - **AWS_INSTANCE_IPV4**— 就A記錄而言，Route 53 的 IPv4 位址會傳回以回應 DNS 查詢，並在探查執行個體詳細資料時 AWS Cloud Map 傳回，例如，`192.0.2.44`。
    - **AWS_INSTANCE_PORT**：與服務探索服務相關聯的連接埠值。
    - **AVAILABILITY_ZONE**：在其中啟動任務的可用區域。對於使用 EC2 啟動類型的任務，這是容器執行個體所在的可用區域。對於使用 Fargate 啟動類型的任務，這是彈性網路介面所在的可用區域。
    - **REGION**：任務所在的區域。
    - **ECS_SERVICE_NAME**：任務所屬的 Amazon ECS 服務名稱。
    - **ECS_CLUSTER_NAME**：任務所屬的 Amazon ECS 叢集名稱。
    - **EC2_INSTANCE_ID**：放置任務的容器執行個體的 ID。如果任務使用 Fargate 啟動類型，則不會新增這個自訂屬性。
    - **ECS_TASK_DEFINITION_FAMILY**：任務使用的任務定義系列。
    - **ECS_TASK_SET_EXTERNAL_ID**：如果任務集合是為外部部署建立，並與服務探索登錄關聯，則 `ECS_TASK_SET_EXTERNAL_ID` 屬性將包含任務集合的外部 ID。
- Amazon ECS 運作狀態檢查：Amazon ECS 會定期執行容器層級的運作狀態檢查。端點若未通過運作狀態檢查，則會從 DNS 路由中移除並標記為運作狀態不佳。

服務探索考量

使用服務探索時應考慮以下事項：

- 服務探索支援在 Fargate 上使用 1.1.0 或更新平台版本的任務。如需詳細資訊，請參閱 適用於 Amazon ECS 的 Fargate Linux 平台版本。
- 被設定為使用服務探索的服務有每個服務 1,000 項任務的限制。這是由於 Route 53 服務配額所致。
- Amazon ECS 主控台的建立服務工作流程僅支援將服務註冊到私有 DNS 命名空間。建立 AWS Cloud Map 私有 DNS 命名空間時，會自動建立 Route 53 私有託管區域。
- 必須設定 VPC DNS 屬性，以成功進行 DNS 解析。如需有關如何設定屬性的詳細資訊，請參閱 Amazon VPC 使用者指南中的 VPC 中的 DNS 支援。

- 為服務探索服務建立的 DNS 記錄總是會使用任務的私有 IP 地址註冊，而非公有 IP 地址 (即使使用公有命名空間)。
- 服務探索要求任務指定 `awsvpc`、`bridge` 或 `host` 網路模式 (不支援 none)。
- 如果您的服務任務定義使用 `awsvpc` 網路模式，您可以為每個服務任務建立任意組合的 `A` 或 SRV 紀錄。如果您使用 SRV 記錄，必須具備連接埠。
- 如果服務任務定義使用 `bridge` 或 `host` 網路模式，則 SRV 記錄是唯一受支援的 DNS 記錄類型。為每個服務任務建立 SRV 記錄。SRV 紀錄必須從任務定義中指定容器名稱和容器連接埠組合。
- 您可以在 VPC 內查詢服務探索服務的 DNS 記錄。其採用的格式為：`<service discovery service name>.<service discovery namespace>`。
- 針對服務名稱執行 DNS 查詢時，`A` 記錄會傳回一組對應到您的任務的 IP 地址。SRV 記錄會針對每個任務傳回一組 IP 地址和連接埠。
- 如果您有八個以下正常運作的記錄，Route 53 會使用所有正常記錄回應所有 DNS 查詢。
- 當所有記錄都狀況不良，Route 53 會使用最多八個狀況不良的記錄來回應 DNS 查詢。
- 您可以為負載平衡器後方的服務設定服務探索，但會將服務探索流量一律路由到任務，而不是負載平衡器。
- 服務探索不支援使用 Classic Load Balancer。
- 針對您的服務探索服務，我們建議您使用由 Amazon ECS 管理的容器層級運作狀態檢查。
  - HealthCheckCustomConfig— Amazon ECS 代表您管理運作狀態檢查。Amazon ECS 利用來自容器和運作狀態檢查的資訊和您的任務狀態，透過 AWS Cloud Map更新運作狀態。當您建立服務探索服務時，會使用 `--health-check-custom-config` 參數來指定。如需詳細資訊，請參閱 AWS Cloud Map API 參考HealthCheckCustomConfig中的。
- 使用服務探索時建立的 AWS Cloud Map 資源必須手動清除。
- 工作和執行個體會註冊為`UNHEALTHY`直到容器健全狀況檢查傳回值為止。如果健全狀況檢查通過，狀態會更新為`HEALTHY`。如果容器健全狀況檢查失敗，則會取消註冊服務探索執行個體。

服務探索定價

使用 Amazon ECS 服務探索的客戶需要支付 Route 53 資源和 AWS Cloud Map 探索 API 操作的費用。其中包括建立 Route 53 託管區域和查詢服務登錄檔的費用。如需詳細資訊，請參閱 AWS Cloud Map 開發人員指南中的 AWS Cloud Map 定價。

Amazon ECS 會執行容器層級運作狀態檢查，並將其公開給 AWS Cloud Map 自訂運作狀態檢查 API 作業。這目前提供給客戶免費使用。如果您針對公開的任務設定其他網路運作狀態檢查，則會向您收取其費用。

建立使用服務探索的 Amazon ECS 服務

了解如何建立包含 Fargate 任務的服務，該任務使用服務探索與. AWS CLI

如需支援服務探索 AWS 區域 的清單，請參閱使用服務探索將 Amazon ECS 服務與 DNS 名稱連接。

如需支援 Fargate 的區域的資訊，請參閱the section called “AWS Fargate 地區”。

必要條件

在您開始教學課程之前，請務必先達成以下先決條件：

- 已安裝並設定最新版本的。 AWS CLI 如需詳細資訊，請參閱安裝 AWS Command Line Interface。
- 設定以使用 Amazon ECS。 中所述的步驟已完成。
- 您的 AWS 使用者具有 亞馬遜 FullAccess IAM 政策範例中指定的必要許可。
- 您已建立至少一個 VPC 和一個安全群組。如需詳細資訊，請參閱 the section called “建立 Virtual Private Cloud”。

步驟 1：在中建立服務探索資源 AWS Cloud Map

請依照下列步驟建立服務探索命名空間和服務探索服務。

1. 建立私有雲端資源服務探索命名空間。此範例會建立名為 `tutorial` 的命名空間。將 *vpc-abcd1234* 替換為您現有其中一個 VPC 的識別碼。

```
aws servicediscovery create-private-dns-namespace \
      --name tutorial \
      --vpc vpc-abcd1234
```

   以下是此命令的輸出：

```
{
    "OperationId": "h2qe3s6dxftvvt7riu6lfy2f6c3jlhf4-je6chs2e"
}
```

2. 使用上一步輸出的 `OperationId`，驗證私有命名空間是否已成功建立。請記下命名空間 ID，因為您在後續命令中使用它。

```
aws servicediscovery get-operation \
```

```
      --operation-id h2qe3s6dxftvvt7riu6lfy2f6c3jlhf4-je6chs2e
```

輸出如下。

```
{
    "Operation": {
        "Id": "h2qe3s6dxftvvt7riu6lfy2f6c3jlhf4-je6chs2e",
        "Type": "CREATE_NAMESPACE",
        "Status": "SUCCESS",
        "CreateDate": 1519777852.502,
        "UpdateDate": 1519777856.086,
        "Targets": {
           "NAMESPACE": "ns-uejictsjen2i4eeg"
        }
    }
}
```

3. 使用上一步輸出中的 NAMESPACE ID，建立服務探索服務。此範例會建立名為 myapplication 的服務。請記下服務 ID 和 ARN，因為您在後續命令中使用它們。

```
aws servicediscovery create-service \
      --name myapplication \
      --dns-config "NamespaceId="ns-
uejictsjen2i4eeg",DnsRecords=[{Type="A",TTL="300"}]" \
      --health-check-custom-config FailureThreshold=1
```

輸出如下。

```
{
    "Service": {
       "Id": "srv-utcrh6wavdkggqtk",
        "Arn": "arn:aws:servicediscovery:region:aws_account_id:service/srv-
utcrh6wavdkggqtk",
        "Name": "myapplication",
        "DnsConfig": {
            "NamespaceId": "ns-uejictsjen2i4eeg",
            "DnsRecords": [
                {
                    "Type": "A",
                    "TTL": 300
                }
            ]
```

```
        },
        "HealthCheckCustomConfig": {
            "FailureThreshold": 1
        },
        "CreatorRequestId": "e49a8797-b735-481b-a657-b74d1d6734eb"
    }
}
```

步驟 2：建立 Amazon ECS 資源

請依照下列步驟建立您的 Amazon ECS 叢集、任務定義和服務。

1. 建立 Amazon ECS 叢集 此範例會建立名稱為 `tutorial` 的叢集。

```
aws ecs create-cluster \
      --cluster-name tutorial
```

2. 註冊與 Fargate 相容的任務定義，並使用 `awsvpc` 網路模式。請遵循下列步驟：

   a. 使用以下任務定義的內容，建立名為 `fargate-task.json` 的檔案：

```
{
    "family": "tutorial-task-def",
        "networkMode": "awsvpc",
        "containerDefinitions": [
            {
                "name": "sample-app",
                "image": "httpd:2.4",
                "portMappings": [
                    {
                        "containerPort": 80,
                        "hostPort": 80,
                        "protocol": "tcp"
                    }
                ],
                "essential": true,
                "entryPoint": [
                    "sh",
                    "-c"
                ],
                "command": [
```

```
                    "/bin/sh -c \"echo '<html> <head> <title>Amazon ECS Sample
 App</title> <style>body {margin-top: 40px; background-color: #333;} </style>
 </head><body> <div style=color:white;text-align:center> <h1>Amazon ECS Sample
 App</h1> <h2>Congratulations!</h2> <p>Your application is now running on a
 container in Amazon ECS.</p> </div></body></html>' >  /usr/local/apache2/
htdocs/index.html && httpd-foreground\""
                ]
            }
        ],
        "requiresCompatibilities": [
            "FARGATE"
        ],
        "cpu": "256",
        "memory": "512"
}
```

b. 使用 fargate-task.json 註冊任務定義。

```
aws ecs register-task-definition \
      --cli-input-json file://fargate-task.json
```

3. 依照下列步驟建立 ECS 服務：

a. 使用您要建立的 ECS 服務的內容，建立名為 ecs-service-discovery.json 的檔案。此範例使用上個步驟中建立的任務定義。需要 awsvpcConfiguration，因為任務定義範例使用 awsvpc 網路模式。

當您建立 ECS 服務時，指定 Fargate 啟動類型以及 LATEST 支援服務探索的平台版本。在 AWS Cloud Map 中建立服務探索服務時，registryArn 是傳回的 ARN。securityGroups 和 subnets 必須屬於用來建立 Cloud Map 命名空間的 VPC。您可以從 Amazon VPC 主控台取得安全群組和子網路識別碼。

```
{
    "cluster": "tutorial",
    "serviceName": "ecs-service-discovery",
    "taskDefinition": "tutorial-task-def",
    "serviceRegistries": [
       {
          "registryArn":
 "arn:aws:servicediscovery:region:aws_account_id:service/srv-utcrh6wavdkggqtk"
       }
    ],
```

```
    "launchType": "FARGATE",
    "platformVersion": "LATEST",
    "networkConfiguration": {
       "awsvpcConfiguration": {
          "assignPublicIp": "ENABLED",
          "securityGroups": [ "sg-abcd1234" ],
          "subnets": [ "subnet-abcd1234" ]
       }
    },
    "desiredCount": 1
}
```

b. 使用 ecs-service-discovery.json 建立您的 ECS 服務。

```
aws ecs create-service \
      --cli-input-json file://ecs-service-discovery.json
```

步驟 3：驗證服務探索 AWS Cloud Map

您可以查詢服務探索資訊，以確認一切都已正確建立。設定服務探索之後，您可以使用 AWS Cloud Map API 作業，也可以dig從 VPC 中的執行個體呼叫。請遵循下列步驟：

1. 使用服務探索的服務 ID，來列出服務探索執行個體。請記下執行個體 ID (以粗體標示) 以進行資源清理。

```
aws servicediscovery list-instances \
      --service-id srv-utcrh6wavdkggqtk
```

輸出如下。

```
{
    "Instances": [
        {
            "Id": "16becc26-8558-4af1-9fbd-f81be062a266",
            "Attributes": {
                "AWS_INSTANCE_IPV4": "172.31.87.2"
                "AWS_INSTANCE_PORT": "80",
                "AVAILABILITY_ZONE": "us-east-1a",
                "REGION": "us-east-1",
                "ECS_SERVICE_NAME": "ecs-service-discovery",
                "ECS_CLUSTER_NAME": "tutorial",
```

```
                "ECS_TASK_DEFINITION_FAMILY": "tutorial-task-def"
            }
        }
    ]
}
```

2. 使用服務探索命名空間、服務和其他參數如 ECS 叢集名稱，來查詢服務探索執行個體的詳細資訊。

```
aws servicediscovery discover-instances \
      --namespace-name tutorial \
      --service-name myapplication \
      --query-parameters ECS_CLUSTER_NAME=tutorial
```

3. 在 Route 53 託管區域為服務探索服務建立的 DNS 記錄，可使用以下 AWS CLI 命令查詢：

   a. 使用命名空間 ID 取得命名空間的相關資訊，其中包括 Route 53 託管區域 ID。

```
aws servicediscovery \
      get-namespace --id ns-uejictsjen2i4eeg
```

   輸出如下。

```
{
    "Namespace": {
        "Id": "ns-uejictsjen2i4eeg",
        "Arn": "arn:aws:servicediscovery:region:aws_account_id:namespace/ns-
uejictsjen2i4eeg",
        "Name": "tutorial",
        "Type": "DNS_PRIVATE",
        "Properties": {
             "DnsProperties": {
                "HostedZoneId": "Z35JQ4ZFDRYPLV"
            }
        },
        "CreateDate": 1519777852.502,
        "CreatorRequestId": "9049a1d5-25e4-4115-8625-96dbda9a6093"
    }
}
```

   b. 使用上一步中的 Route 53 託管區域 ID (請參閱粗體文字)，取得託管區域的資源紀錄集。

```
aws route53 list-resource-record-sets \
      --hosted-zone-id Z35JQ4ZFDRYPLV
```

4. 您還可以使用 dig 從您的 VPC 內的執行個體查詢 DNS。

```
dig +short myapplication.tutorial
```

步驟 4：清理

當您完成此教學課程時，清除相關聯的資源以避免未使用的資源產生費用。請遵循下列步驟：

1. 使用您之前提到的服務 ID 和執行個體 ID，取消註冊服務探索服務執行個體。

```
aws servicediscovery deregister-instance \
      --service-id srv-utcrh6wavdkggqtk \
      --instance-id 16becc26-8558-4af1-9fbd-f81be062a266
```

輸出如下。

```
{
    "OperationId": "xhu73bsertlyffhm3faqi7kumsmx274n-jh0zimzv"
}
```

2. 使用來自上一步輸出的 OperationId，請驗證已成功取消註冊服務探索服務。

```
aws servicediscovery get-operation \
      --operation-id xhu73bsertlyffhm3faqi7kumsmx274n-jh0zimzv
```

```
{
  "Operation": {
        "Id": "xhu73bsertlyffhm3faqi7kumsmx274n-jh0zimzv",
        "Type": "DEREGISTER_INSTANCE",
        "Status": "SUCCESS",
        "CreateDate": 1525984073.707,
        "UpdateDate": 1525984076.426,
        "Targets": {
            "INSTANCE": "16becc26-8558-4af1-9fbd-f81be062a266",
            "ROUTE_53_CHANGE_ID": "C5NSRG1J4I1FH",
            "SERVICE": "srv-utcrh6wavdkggqtk"
```

```
        }
    }
}
```

3. 使用服務 ID 刪除服務探索服務。

```
aws servicediscovery delete-service \
      --id srv-utcrh6wavdkggqtk
```

4. 使用命名空間 ID 刪除服務探索命名空間 。

```
aws servicediscovery delete-namespace \
      --id ns-uejictsjen2i4eeg
```

輸出如下。

```
{
    "OperationId": "c3ncqglftesw4ibgj5baz6ktaoh6cg4t-jh0ztysj"
}
```

5. 使用上一步輸出的 OperationId，確認服務探索命名空間已成功刪除。

```
aws servicediscovery get-operation \
      --operation-id c3ncqglftesw4ibgj5baz6ktaoh6cg4t-jh0ztysj
```

輸出如下。

```
{
    "Operation": {
        "Id": "c3ncqglftesw4ibgj5baz6ktaoh6cg4t-jh0ztysj",
        "Type": "DELETE_NAMESPACE",
        "Status": "SUCCESS",
        "CreateDate": 1525984602.211,
        "UpdateDate": 1525984602.558,
        "Targets": {
            "NAMESPACE": "ns-rymlehshst7hhukh",
            "ROUTE_53_CHANGE_ID": "CJP2A2M86XW3O"
        }
    }
}
```

6. 將 Amazon ECS 服務所需的計數更新為 0。您必須執行此操作，才能在後續步驟中刪除服務。

```
aws ecs update-service \
      --cluster tutorial \
      --service ecs-service-discovery \
      --desired-count 0
```

7. 刪除 Amazon ECS 服務：

```
aws ecs delete-service \
      --cluster tutorial \
      --service ecs-service-discovery
```

8. 刪除 Amazon ECS 叢集：

```
aws ecs delete-cluster \
      --cluster tutorial
```

保護您的 Amazon ECS 任務不會因擴充事件而終止

您可以使用 Amazon ECS 任務擴展保護，保護您的任務不會因服務 auto 擴展或部署的擴充事件而終止。

某些應用程式需要一種機制來保護關鍵任務，避免任務在使用率較低時或服務部署期間因縮減事件終止。例如：

- 您擁有佇列處理非同步應用程式，例如影片轉碼工作，即使累積服務使用率很低，其中部分任務仍需要執行數小時。
- 您的遊戲應用程式會以 Amazon ECS 任務的形式執行遊戲伺服器，即使所有使用者都已登出，也需要繼續執行，以減少伺服器重新開機的啟動延遲。
- 部署新的程式碼版本時，您需要繼續執行任務，因為重新處理的費用很昂貴。

若要保護屬於服務的任務不會在縮減事件中終止，請將 `protectionEnabled` 屬性設定為 `true`。依預設，任務受保護的時間為 2 小時。您可以使用 `expiresInMinutes` 屬性自訂保護期間。任務的保護時間最短可至 1 分鐘，最長可達 2880 分鐘 (48 小時)。

任務完成必要的工作之後，您可以將 `protectionEnabled` 屬性設定為 `false`，以便隨後的縮減事件終止任務。

任務縮減保護機制

您可以使用 Amazon ECS 容器代理程式端點或 Amazon ECS API 來設定和取得任務縮減保護。

- Amazon ECS 容器代理程式端點

  我們建議將 Amazon ECS 容器代理程式端點，用於可自行判斷是否需要保護的任務。針對以佇列為基礎或工作處理的工作負載使用此方法。

  當容器開始處理工作時 (例如使用 SQS 訊息)，您可以從容器內透過任務縮減保護端點路徑 `$ECS_AGENT_URI/task-protection/v1/state` 來設定 `ProtectionEnabled` 屬性。Amazon ECS 不會在縮減事件期間終止此任務。工作完成其工作之後，您可以使用相同的端點清除`ProtectionEnabled`屬性，使工作有資格在後續擴充事件期間終止。

  如需 Amazon ECS 容器代理程式端點的詳細資訊，請參閱Amazon ECS 任務擴展保護端點。

- Amazon ECS API

  如果您的應用程式具有追蹤使用中任務狀態的元件，則可以使用 Amazon ECS API 來設定和擷取任務擴充保護。使用 `UpdateTaskProtection` 將一或多個任務標記為受保護。用`GetTaskProtection`於擷取防護狀態。

  如果應用程式以 Amazon ECS 任務的形式託管遊戲伺服器工作階段，則可使用此方法。當使用者登入伺服器 (任務) 上的工作階段時，您可以將任務標示為受保護。使用者登出後，您可以清除專門針對此任務的保護設定，或根據您希望伺服器保持閒置的需求，定期清除不再具有作用中工作階段的類似任務的保護設定。

  如需詳細資訊，請參閱 Amazon 彈性容器服務 API 參考中的GetTask保UpdateTask護和保護。

您可以將這兩種方法結合起來使用。例如，使用 Amazon ECS 代理程式端點，從容器內設定任務保護，並使用 Amazon ECS API，從外部控制器服務移除對任務保護的設定。

考量事項

使用任務縮減保護之前，請考量下列幾點：

- 建議使用 Amazon ECS 容器代理程式端點，因為 Amazon ECS 代理程式具有內建的重試機制和更簡單的介面。
- 您可以為已開啟保護的任務呼叫 `UpdateTaskProtection`，以重設任務縮減保護過期期間。

- 判斷任務需要多久才能完成其必要的工作，並相應地設定 `expiresInMinutes` 屬性。如果您設定的保護過期期間超過必要的時間，則會產生費用，並會在部署新任務時面臨延遲。
- Amazon ECS 容器代理程式 `1.65.0` 或更新版本支援任務縮減保護。

  使用較舊版本的 Amazon ECS 容器代理程式可在 Amazon EC2 執行個體上新增此功能的支援，方法是將代理程式更新為最新版本。如需詳細資訊，請參閱 更新 Amazon ECS 容器代理程式。
- 部署考量事項：
  - 如果服務使用滾動更新，則會建立新任務，但在 `protectionEnabled` 設定清除或過期之前執行舊版本的任務都不會終止。您可以將部署組態中的 `maximumPercentage` 參數調整為可允許在舊任務受保護時建立新任務的值。
  - 如果套用了藍/綠更新，若任務有 `protectionEnabled`，則不會移除具有受保護任務的藍色部署。流量將被轉移到出現的新任務，並且只有在清除或過期時`protectionEnabled`才會刪除較舊的任務。視 CodeDeploy 或 CloudFormation 更新的逾時而定，部署可能會逾時，而較舊的 Blue 工作可能仍然存在。
  - 如果您使用 CloudFormation，則更新堆疊會有 3 小時逾時。因此，如果您將工作保護設定為 3 小時以上，則 CloudFormation 部署可能會導致失敗和復原。

    在您的舊任務受到保護期間， CloudFormation 堆棧顯示`UPDATE_IN_PROGRESS`。如果任務縮減保護遭移除或在 3 小時視窗內過期，部署將會成功並變為 `UPDATE_COMPLETE` 狀態。如果部署停滯在 `UPDATE_IN_PROGRESS` 的時間超過 3 小時，其將失敗並顯示 `UPDATE_FAILED` 狀態，然後會復原至舊任務集。
  - 當受保護的任務導致部署 (滾動或藍/綠) 無法進入穩定狀態，Amazon ECS 會傳送服務事件，以便您採取補救措施。嘗試更新任務的受保護狀態時，如果您收到 `DEPLOYMENT_BLOCKED` 錯誤訊息，則表示該服務的受保護任務比服務所需的任務計數更多。若要解決此錯誤，請執行以下其中一個動作：
    - 等待目前的任務保護過期。然後設定任務保護。
    - 判斷可以停止哪些任務。然後使用 `UpdateTaskProtection`，並將 `protectionEnabled` 選項設為 `false` 來完成這些任務。
    - 將服務的所需任務計數提高到超過受保護任務的數目。

任務縮減保護所需的 IAM 許可

任務必須具有具有下列許可的 Amazon ECS 任務角色：

- `ecs:GetTaskProtection`：允許 Amazon ECS 容器代理程式呼叫 `GetTaskProtection`。

- ecs:UpdateTaskProtection：允許 Amazon ECS 容器代理程式呼叫 UpdateTaskProtection。

Amazon ECS 任務擴展保護端點

Amazon ECS 容器代理程式會自動將 ECS_AGENT_URI 環境變體注入 Amazon ECS 任務的容器中，以提供與容器代理程式 API 端點互動的方法。

我們建議將 Amazon ECS 容器代理程式端點，用於可自行判斷是否需要保護的任務。

當容器開始處理工作時，您可以使用容器內的工作縮放保護端點路徑$ECS_AGENT_URI/task-protection/v1/state來設定protectionEnabled屬性。

從容器內使用此 URI 的 PUT 要求來設定工作縮放保護。對此 URI 的 GET 要求會傳回工作目前的保護狀態。

工作擴充保護要求參數

您可以使用具有下列請求參數的 ${ECS_AGENT_URI}/task-protection/v1/state 端點來設定任務縮減保護。

ProtectionEnabled

指定true此選項可將工作標示為要保護。指定false移除保護，並使工作有資格終止。

類型：布林值

必要：是

ExpiresInMinutes

工作受到保護的分鐘數。您可以指定最短 1 分鐘到最長 2,880 分鐘 (48 小時)。在此期間，您的任務不會因 Service Auto Scaling 或部署中的縮減事件而終止。在此時段過後，將 protectionEnabled 參數設為 false。

如果您未指定時間，則會自動保護任務 120 分鐘 (2 小時)。

類型：整數

必要：否

下列範例示範如何設定具有不同時段的任務保護。

如何使用預設時段保護工作的範例

此範例示範如何保護預設時段為 2 小時的任務。

```
curl --request PUT --header 'Content-Type: application/json' ${ECS_AGENT_URI}/task-
protection/v1/state --data '{"ProtectionEnabled":true}'
```

如何保護工作 60 分鐘的範例

此範例示範如何使用 expiresInMinutes 參數保護任務 60 分鐘。

```
curl --request PUT --header 'Content-Type: application/json' ${ECS_AGENT_URI}/task-
protection/v1/state --data '{"ProtectionEnabled":true,"ExpiresInMinutes":60}'
```

如何保護工作 24 小時的範例

此範例示範如何使用 expiresInMinutes 參數保護任務 24 小時。

```
curl --request PUT --header 'Content-Type: application/json' ${ECS_AGENT_URI}/task-
protection/v1/state --data '{"ProtectionEnabled":true,"ExpiresInMinutes":1440}'
```

PUT 請求返回以下響應。

```
{
  "protection": {
    "ExpirationDate": "2023-12-20T21:57:44.837Z",
    "ProtectionEnabled": true,
    "TaskArn": "arn:aws:ecs:us-west-2:111122223333:task/1234567890abcdef0"
  }
}
```

工作縮放保護回應參數

JSON 回應中的任務縮減保護端點 ${ECS_AGENT_URI}/task-protection/v1/state 會傳回下列資訊。

ExpirationDate

任務保護將到期的時間 (以 Epoch 時間表示)。如果工作未受保護，則此值為 null。

ProtectionEnabled

任務的保護狀態。如果已針對任務啟用縮減保護，則值為 true。否則為 false。

`TaskArn`

容器所屬任務的完整 Amazon Resource Name (ARN)。

下列範例顯示了為受保護任務傳回的詳細資訊。

```
curl --request GET ${ECS_AGENT_URI}/task-protection/v1/state
```

```
{
    "protection":{
        "ExpirationDate":"2023-12-20T21:57:44Z",
        "ProtectionEnabled":true,
        "TaskArn":"arn:aws:ecs:us-west-2:111122223333:task/1234567890abcdef0"
    }
}
```

發生失敗時會傳回下列資訊。

`Arn`

任務的完整 Amazon Resource Name (ARN)。

`Detail`

與失敗相關的詳細資訊。

`Reason`

失敗的原因。

下列範例顯示了為未保護任務傳回的詳細資訊。

```
{
    "failure":{
        "Arn":"arn:aws:ecs:us-west-2:111122223333:task/1234567890abcdef0",
        "Detail":null,
        "Reason":"TASK_NOT_VALID"
    }
}
```

發生例外狀況時會傳回下列資訊。

`requestID`

會導致例外狀況的 Amazon ECS API 呼叫的 AWS 請求識別碼。

`Arn`

任務或服務的完整 Amazon Resource Name (ARN)。

`Code`

錯誤代碼。

`Message`

錯誤訊息。

> **Note**
>
> 如果出現 `RequestError` 或 `RequestTimeout` 錯誤，則可能是聯網問題。嘗試對 Amazon ECS 使用 VPC 端點。

以下範例顯示了發生錯誤時傳回的詳細資訊。

```
{
    "requestID":"12345-abc-6789-0123-abc",
    "error":{
        "Arn":"arn:aws:ecs:us-west-2:555555555555:task/my-cluster-
name/1234567890abcdef0",
        "Code":"AccessDeniedException",
        "Message":"User: arn:aws:sts::444455556666:assumed-role/my-ecs-task-
role/1234567890abcdef0 is not authorized to perform: ecs:GetTaskProtection on resource:
 arn:aws:ecs:us-west-2:555555555555:task/test/1234567890abcdef0 because no identity-
based policy allows the ecs:GetTaskProtection action"
    }
}
```

如果 Amazon ECS 代理程式因為網路問題或 Amazon ECS 控制平面關閉等原因而無法從 Amazon ECS 端點獲得回應，則會出現下列錯誤。

```
{
  "error": {
    "Arn": "arn:aws:ecs:us-west-2:555555555555:task/my-cluster-name/1234567890abcdef0",
    "Code": "RequestCanceled",
```

```
    "Message": "Timed out calling Amazon ECS Task Protection API"
  }
}
```

當 Amazon ECS 代理程式從 Amazon ECS 中取得限流例外狀況時，會出現下列錯誤。

```
{
  "requestID": "12345-abc-6789-0123-abc",
  "error": {
    "Arn": "arn:aws:ecs:us-west-2:555555555555:task/my-cluster-name/1234567890abcdef0",
    "Code": "ThrottlingException",
    "Message": "Rate exceeded"
  }
}
```

Amazon ECS 服務節流邏輯

Amazon ECS 服務排程器包含的邏輯，可在任務重複啟動失敗的情況下，針對服務任務啟動的頻率進行調節。

如果服務的工作重複無法進入狀RUNNING態 (從STOPPED狀態直接進行)，則後續重新啟動嘗試之間的時間會逐漸增加，最多可達 27 分鐘。PENDING此最長期間將來可能會有所變更。此行為能降低失敗的任務對 Amazon ECS 叢集資源或 Fargate 基礎設施成本所造成的影響。如果您的服務啟動了限流邏輯，您會收到以下的服務事件訊息：

```
(service service-name) is unable to consistently start tasks successfully.
```

Amazon ECS 永遠不會阻止失敗的服務重試。除了增加重新啟動的時間間隔，它也不會嘗試對其進行任何修改。服務限流邏輯沒有任何可供使用者調校的參數。

如果您更新服務使用新的任務定義，您的服務就會立即回復到正常、非調節的狀態。如需詳細資訊，請參閱 使用主控台更新 Amazon ECS 服務。

以下是啟動此邏輯的一些常見原因。我們建議您採取手動動作來解決此問題：

- 您的叢集中缺乏裝載任務的資源，例如連接埠、記憶體或 CPU 單位。在這種情況下，您也會看到「資源不足服務事件訊息」。
- Amazon ECS 容器代理程式無法提取您的任務 Docker 映像檔。這可能是因為錯誤的容器映像名稱、映像、或標籤，或缺乏私有登錄檔身分驗證或許可。在這種情況下，您也會在「CannotPullContainerError停止的任務錯誤」中看到。

- 您的容器執行個體磁碟空間不足以建立容器。在這種情況下，您也會在「CannotCreateContainerError停止的任務錯誤」中看到。如需詳細資訊，請參閱 Amazon ECS 中的泊塢視API error (500): devmapper窗疑難排解。

> **Important**
>
> 到達 RUNNING 狀態之後停止的任務不會啟動限流邏輯或相關聯的服務事件訊息。例如，假設服務的 Elastic Load Balancing 運作狀態檢查失敗導致任務標示為運作狀態不佳，而 Amazon ECS 取消註冊並停止了任務。這樣並不會觸發限流。即使任務的容器指令立即以非零結束代碼結束，任務仍已轉換為 RUNNING 狀態。因為指令錯誤而立即失敗的任務不會觸發限流或服務事件訊息。

Amazon ECS 服務定義參數

服務定義會定義您的 Amazon ECS 服務的執行方式。下列參數可以在服務定義中指定。

啟動類型

`launchType`

類型：字串

有效值：EC2 | FARGATE | EXTERNAL

必要：否

您服務執行所在的啟動類型，預設會使用預設的 `capacityProviderStrategy`。如需詳細資訊，請參閱 Amazon ECS 啟動類型。

若有指定 `launchType`，則必須省略 `capacityProviderStrategy` 參數。

容量提供者策略

`capacityProviderStrategy`

類型： 物件陣列

必要：否

要用於服務的容量提供者策略。

容量提供者策略包含一個或多個容量提供者，以及指派給這些容量提供者的 base 與 weight。容量提供者必須與要在容量提供者策略中使用的叢集相關聯。 PutClusterCapacityProviders API 用於將容量提供者與叢集建立關聯。只能使用具有 ACTIVE 或 UPDATING 狀態的容量提供者。

若有指定 capacityProviderStrategy，則必須省略 launchType 參數。如果沒有指定 capacityProviderStrategy 或 launchType，則會使用叢集的 defaultCapacityProviderStrategy。

如果您想要指定使用 Auto Scaling 群組的容量提供者，則必須已經建立該容量提供者。您可以使用 CreateCapacityProvider API 作業建立新的容量提供者。

若要使用 AWS Fargate 容量提供者，請指定FARGATE或FARGATE_SPOT容量提供者。 AWS Fargate 容量提供者適用於所有帳戶，只需要與叢集建立關聯即可使用。

建立叢集之後， PutClusterCapacityProviders API 作業可用來更新叢集的可用容量提供者清單。

capacityProvider

類型：字串

必要：是

容量提供者的簡短名稱或完整 Amazon Resource Name (ARN)。

weight

類型：整數

有效範圍：介於 0 到 1,000 之間的整數。

必要：否

加權值會指定應使用指定容量提供者其已啟動任務總數的相對百分比。

例如，假設您的策略包含兩個容量提供者，且兩者的加權值都是一。當基本條件符合時，工作會平均分配給兩個容量提供者。以此類推，假設您為 capacityProviderA 指定了加權值 1；並為 capacityProviderB 指定了加權值 4。那麼，每有一個任務使用 capacityProviderA 執行，就會有四個任務是使用 capacityProviderB 執行。

base

類型：整數

有效範圍：介於 0 到 100,000 之間的整數。

必要：否

基礎值指定至少要在指定容量提供者上執行多少任務數量。容量提供者策略中只有一個容量提供者可以定義基礎。

任務定義

`taskDefinition`

類型：字串

必要：否

`family` 和 `revision` (`family:revision`) 或在您服務中執行的任務定義完整 Amazon Resource Name (ARN)。如果 `revision` 未指定，則會使用指定系列的最新 `ACTIVE` 修訂版。

在使用循環更新 (ECS) 部署控制器時，必須指定任務定義。

平台作業系統

`platformFamily`

類型：字串

必要：有條件

預設：Linux

在 Fargate 上託管的 Amazon ECS 服務需要此參數。

Amazon Amazon EC2 上託管的亞馬遜 ECS 服務會忽略此參數。

執行服務的容器上的作業系統。有效值為 `LINUX`、`WINDOWS_SERVER_2019_FULL`、`WINDOWS_SERVER_2019_CORE`、`WINDOWS_SERVER_2022_` 與 `WINDOWS_SERVER_2022_CORE`。

為服務指定的每個任務的 `platformFamily` 值必須與服務 `platformFamily` 值相符。例如，如果您將 `platformFamily` 設定為 `WINDOWS_SERVER_2019_FULL`，則所有任務的 `platformFamily` 值都必須為 `WINDOWS_SERVER_2019_FULL`。

平台版本

`platformVersion`

類型：字串

必要：否

您服務中任務正在其上執行的平台版本。平台版本僅會為使用 Fargate 啟動類型的任務指定。如果尚未指定，預設會使用最新版本 (LATEST)。

AWS Fargate 平台版本用於指 Fargate 任務基礎結構的特定運行時環境。在您執行任務或建立服務時指定 LATEST 平台版本，即可取得任務所能使用的最新平台版本。當您擴展服務規模時，這些任務會收到於服務的目前部署上所指定的平台版本。如需詳細資訊，請參閱 適用於 Amazon ECS 的 Fargate Linux 平台版本。

> **Note**
>
> 並未針對使用 EC2 啟動類型的任務指定平台版本。

叢集

`cluster`

類型：字串

必要：否

您服務執行所在之叢集的 Amazon Resource Name (ARN) 簡稱或全稱。若您未指定叢集，即假設您會使用 `default` 叢集。

服務名稱

`serviceName`

類型：字串

必要：是

您的服務名稱。可以包含最多可達 255 個字元 (大小寫)、數字、連字號和底線。叢集中不得有相同的服務名稱，但一個區域內或多個區域間的多個叢集中可以有類似的服務名稱。

排程策略

`schedulingStrategy`

類型：字串

有效值：`REPLICA | DAEMON`

必要：否

使用的排程策略。如果未指定排程策略，則會使用 `REPLICA` 策略。如需詳細資訊，請參閱 Amazon ECS 服務。

現已推出兩個服務排程器策略概念：

- `REPLICA` - 複本排程策略會置放並在整個叢集中維持所需的任務數量。根據預設，服務排程器會將任務分散至各個可用區域。您可以使用任務置放策略和限制條件，來自訂任務置放決策。如需詳細資訊，請參閱 複本策略。
- `DAEMON` - 常駐程式排程策略會在每個符合您於叢集中所指定所有任務置放限制條件的作用中容器執行個體上，準確地部署一個任務。使用這項策略時，不需指定所需的任務數量、任務置放策略或使用 Service Auto Scaling 政策。如需詳細資訊，請參閱 守護進程策。

> **Note**
>
> Fargate 任務不支援 `DAEMON` 排程策略。

所需計數

`desiredCount`

類型：整數

必要：否

於指定的任務定義中，要放置在您服務上並保持執行的實例數量。

如果使用 `REPLICA` 排程策略，則需要此參數。如果該服務使用 `DAEMON` 排程策略，則此參數是選擇性的。

部署組態

deploymentConfiguration

類型：物件

必要：否

選用的部署參數，可控制在部署期間執行多少任務以及停止和啟動任務的順序。

maximumPercent

類型：整數

必要：否

如果服務使用滾動更新 (ECS) 部署類型，則 maximumPercent 參數代表在部署期間，服務中允許處於 RUNNING、STOPPING 或 PENDING 狀態的任務數量上限。以百分比表示 desiredCount 即無條件捨去至最接近的整數。您可以使用此參數來定義部署批次大小。例如，如果您的服務使用 REPLICA 服務排程器、有四個任務的 desiredCount，且 maximumPercent 值為 200%，則排程器可能先啟動四項新任務，再停止四項舊任務。前提是有執行此操作所需的叢集資源。使用 REPLICA 服務排程器的服務，其預設 maximumPercent 值為 200%。

如果您的服務使用 DAEMON 服務排程器類型，maximumPercent 應該維持 100%。這是預設值。

部署期間的任務最大數量等於 desiredCount 乘以 maximumPercent/100，無條件捨去到最接近的整數值。

如果服務使用的是藍/綠 (CODE_DEPLOY) 或 EXTERNAL 部署類型以及使用 EC2 啟動類型的任務，則百分比上限值設為預設值，並用來定義當容器執行個體為 DRAINING 狀態時，保持在 RUNNING 狀態之服務的任務數目上限。如果服務中的任務使用 Fargate 啟動類型，則不使用百分比上限值，雖然在描述服務時會傳回此值。

minimumHealthyPercent

類型：整數

必要：否

如果服務使用滾動更新 (ECS) 部署類型，則 minimumHealthyPercent 表示部署期間必須保持在 RUNNING 狀態的服務任務數量下限。這表示為無條件進位到最接近整數的

desiredCount 的百分比。您可以使用此參數，而無須使用額外的叢集容量進行部署。例如，如果您的服務有四項任務的 desiredCount，且 minimumHealthyPercent 為 50%，則排程器可能先停止兩項現有的任務以釋出叢集容量，再啟動兩項新的任務。

對於不使用負載平衡器的服務，請考慮下列事項：

- 如果服務中任務內的所有必要容器都通過了運作狀態檢查，則服務就會被視為狀態良好。
- 如果工作沒有已定義運作狀態檢查的必要容器，則服務排程器會在任務達到 RUNNING 狀態時等待 40 秒，然後再將任務計入運作狀態最低百分比總計。
- 如果任務有一或多個已定義運作狀態檢查的必要容器，服務排程器會等待任務達到正常狀態，然後將其計算為運作狀態最低百分比總計。當任務內的所有必要容器都通過其運作狀態檢查時，工作就會被視為狀態良好。服務排程器可以等待的時間，是由容器運作狀態檢查設定所決定。如需詳細資訊，請參閱 運作狀態檢查。

對於會使用負載平衡器的服務，請考慮下列事項：

- 如果任務沒有定義運作狀態檢查的必要容器，則服務排程器會等待負載平衡器目標群組運作狀態檢查回傳狀況良好狀態，然後將任務計算為良好狀態最低百分比總計。
- 如果工作具有已定義運作狀態檢查的必要容器，服務排程器會等待工作達到正常狀態，以及負載平衡器目標群組運作狀態檢查回傳狀況良好狀態，然後再將工作計算為最低良好狀態百分比總計。

minimumHealthyPercent 的複本服務預設值是 100%。使用服務排程之DAEMON服務的預設minimumHealthyPercent值為 0% (AWS SDK 和 API)，而. AWS CLI AWS Management Console

部署期間運作良好任務的最低數量等於 desiredCount 乘以 minimumHealthyPercent/100，無條件進位到最接近的整數值。

如果服務使用的是藍/綠 (CODE_DEPLOY) 或 EXTERNAL 部署類型，並且執行使用 EC2 啟動類型的任務，則運作狀態百分比下限值設為預設值，並用來定義當容器執行個體為 DRAINING 狀態時，保持在 RUNNING 狀態之服務的任務數量下限。若服務使用的是藍/綠 (CODE_DEPLOY) 或 EXTERNAL 部署類型，並且執行使用 Fargate 啟動類型的任務，則不使用運作良好百分比下限值，雖然描述服務時會傳回此值。

部署控制器

deploymentController

類型：物件

必要：否

用於服務的部署控制器。如果沒有指定部署控制器，則使用 ECS 控制器。如需詳細資訊，請參閱 Amazon ECS 服務。

type

類型：字串

有效值：ECS | CODE_DEPLOY | EXTERNAL

必要：是

要使用的部署控制器類型。部署控制器有三種類型：

ECS

輪流更新 (ECS) 部署類型涉及以最新版本替換目前執行的容器版本。調整在服務部署期間允許的運作良好任務的最小和最大數量，可以控制 Amazon ECS 在滾動式更新期間新增或移除的容器數量，如同 deploymentConfiguration 中所指定。

CODE_DEPLOY

藍/綠 (CODE_DEPLOY) 部署類型使用支援的藍/綠部署模型 CodeDeploy，可讓您在將生產流量傳送至服務之前，先驗證服務的新部署。

EXTERNAL

當您想要使用任何第三方部署控制器，以完全控制 Amazon ECS 服務的部署程序時，請使用外部部署類型。

任務置放

placementConstraints

類型： 物件陣列

必要：否

您服務的任務要使用的置放限制物件陣列。每項任務您最多可以指定 10 項限制。此限制包含任務定義中的限制以及執行時間指定的限制。若您使用的是 Fargate 啟動類型，則不支援任務置放限制條件。

type

類型：字串

必要：否

限制類型。使用 distinctInstance 以確保特定群組中的每個任務執行於不同的容器執行個體。使用 memberOf，以將選擇限制於特定群組或有效的待選項目。任務定義並不支援 distinctInstance 值。

expression

類型：字串

必要：否

限制所套用的叢集查詢語言運算式。如果限制條件類型為 distinctInstance，則無法指定運算式。如需詳細資訊，請參閱 建立運算式以定義 Amazon ECS 任務的容器執行個體。

placementStrategy

類型： 物件陣列

必要：否

您服務的任務要使用的置放策略物件。每項服務您最多可以指定四項策略規則。

type

類型：字串

有效值：random | spread | binpack

必要：否

置放策略類型。random 置放策略會將任務隨機置放於各個可用的待選項目。spread 置放策略根據 field 參數，以平均分散的方式置放於各個可用的待選項目。binpack 策略會根據以 field 參數指定的結果，將任務置放於資源最少的可用待選項目。例如，如果您在記憶體上進行 binpack，任務會放置在剩餘記憶體最少，但仍足以執行任務的執行個體上。

field

類型：字串

必要：否

套用置放策略的欄位。若是 spread 置放策略，有效的值為 instanceId (或擁有相同效果的 host)，或任何平台以及套用至 attribute:ecs.availability-zone 等容器執行個體的自訂屬性。若是 binpack 置放策略，有效值為 cpu 與 memory。若是 random 置放策略，此欄位並不使用。

標籤

`tags`

類型： 物件陣列

必要：否

您套用到服務以協助您分類和組織的中繼資料。每個標籤皆包含由您定義的一個金鑰與一個選用值。刪除服務時，也會一併刪除標籤。服務最多可套用 50 個標籤。如需詳細資訊，請參閱 標記 Amazon ECS 資源。

`key`

類型：字串

長度限制：長度下限為 1。長度上限為 128。

必要：否

組成標籤的鍵值組的一部分。索引鍵是一般標籤，作用就像更特定標籤值的類別。

`value`

類型：字串

長度限制：長度下限為 0。長度上限為 256。

必要：否

組成標籤的鍵值組的選用部分。值就像標籤類別 (索引鍵) 內的描述項。

`enableECSManagedTags`

類型：布林值

有效值：`true` | `false`

必要：否

指定是否對服務內的任務使用 Amazon ECS 受管標籤。如未指定任何值，則預設值為 `false`。如需詳細資訊，請參閱 使用標籤進行帳單。

`propagateTags`

類型：字串

有效值：TASK_DEFINITION | SERVICE

必要：否

指定是否將標籤從任務定義或服務複製到服務中的任務。如果沒有指定值，則不會複製標籤。標籤只能在建立服務期間複製到服務內的任務。若要在建立服務或任務後對任務新增標籤，請使用 TagResource API 動作。

網路組態

networkConfiguration

類型：物件

必要：否

服務的網路組態。使用 awsvpc 網路模式的任務定義需要此參數，才能接收其自己的彈性網路介面，其他網路模式不支援此參數。如果使用 Fargate 啟動類型，則需要 awsvpc 網路模式。如需 Amazon EC2 啟動類型的聯網相關資訊，請參閱適用於 EC2 啟動類型的 Amazon ECS 任務聯網選項。如需有關 Fargate 啟動類型之網路的詳細資訊，請參閱 Fargate 工作網路。

awsvpcConfiguration

類型：物件

必要：否

代表任務或服務之子網路和安全群組的物件。

subnets

類型：字串陣列

必要：是

與任務或服務相關聯的子網路。根據 awsvpcConfiguration 限制僅能指定 16 個子網路。

securityGroups

類型：字串陣列

必要：否

與任務或服務相關聯的安全群組。如果您未指定安全群組，則會使用預設的 VPC 安全群組。根據 awsvpcConfiguration 限制僅能指定五個安全群組。

assignPublicIP

類型：字串

有效值：ENABLED | DISABLED

必要：否

無論任務的彈性網路介面是否收到公有 IP 地址。如果未指定任何值，則會使用 DISABLED 的預設值。

healthCheckGracePeriodSeconds

類型：整數

必要：否

Amazon ECS 服務排程器在任務進入 RUNNING 狀態後，應忽略狀況不良的 Elastic Load Balancing 目標運作狀態檢查、容器運作狀態檢查和 Route 53 運作狀態檢查的期間 (秒)。這僅在您的服務已設定為使用負載平衡器時才會有效。如果您的服務已定義負載平衡器，且您未指定運作狀態檢查寬限期值，這時將會使用 0 的預設值。

若您的服務任務啟動及回應運作狀態檢查需要一些時間，您可以指定運作狀態檢查之寬限期間最高達 2,147,483,647 秒。在此期間，ECS 服務排程器會忽略運作狀態檢查狀態。此寬限期可防止 ECS 服務排程器將任務標示為狀況不良並在其來得及標示前加以停止。

如果您不使用 Elastic Load Balancing，建議您在任務定義運作狀態檢查參數中使用 startPeriod。如需詳細資訊，請參閱使用容器運作狀態檢查判斷 Amazon ECS 任務運作狀態。

loadBalancers

類型： 物件陣列

必要：否

用來代表您的服務所使用之負載平衡器的負載平衡器物件。針對使用 Application Load Balancer 或 Network Load Balancer 的任務，您最多僅能連接五個目標群組到一個服務。

建立服務之後，負載平衡器組態無法從 AWS Management Console變更。您可以使用 AWS Copilot AWS CloudFormation、 AWS CLI 或 SDK 來修改ECS滾動式部署控制器的負載平衡器組態，而不能修改 AWS CodeDeploy 藍/綠或外部的負載平衡器組態。當您新增、更新或移除負載平衡器組態時，Amazon ECS 會使用更新後的 Elastic Load Balancing 組態啟動新的部署。

這會導致任務向負載平衡器註冊和取消註冊。我們建議您在更新 Elastic Load Balancing 組態之前，先在測試環境中驗證。如需如何修改組態的相關資訊，請參閱 Amazon 彈性容器服務 API 參考UpdateService中的。

針對 Application Load Balancer 和 Network Load Balancer，此物件必須包含負載平衡器目標群組 ARN、容器名稱 (這會出現在容器定義中) 和容器連接埠，以從負載平衡器存取。當此服務的任務置於容器執行個體，該容器執行個體和連接埠組合就會註冊為指定之目標群組中的目標。

`targetGroupArn`

類型：字串

必要：否

與服務相關聯的 Elastic Load Balancing 目標群組的完整 Amazon Resource Name (ARN)。

僅在使用 Application Load Balancer 或Network Load Balancer時，才指定目標群組 ARN。

`loadBalancerName`

類型：字串

必要：否

要與服務建立關聯的負載平衡器名稱。

如果您使用 Application Load Balancer 或 Network Load Balancer，請略去負載平衡器名稱參數。

`containerName`

類型：字串

必要：否

要與負載平衡器建立關聯的容器 (這會出現在容器定義中) 名稱。

`containerPort`

類型：整數

必要：否

要與負載平衡器建立關聯之容器的連接埠。此連接埠必須對應至服務中任務所使用之任務定義中的 `containerPort`。針對使用 EC2 啟動類型的任務，容器執行個體必須允許傳入連接埠對應的 `hostPort` 的流量。

`role`

類型：字串

必要：否

IAM 角色的簡稱或完整 ARN，此角色允許 Amazon ECS 代您呼叫您的負載平衡器。只有在您對服務使用具有單一目標群組的負載平衡器，且您的任務定義沒有使用 `awsvpc` 網路模式時，才允許此參數。如果您指定 `role` 參數，您還必須以 `loadBalancers` 參數指定負載平衡器物件。

如果您指定的角色有 `/` 以外的路徑，則您必須指定完整的角色 ARN (建議項目) 或在角色名稱前加上路徑。例如，如果名稱為 `bar` 的角色有路徑 `/foo/`，則您可指定 `/foo/bar` 為角色名稱。如需詳細資訊，請參閱 IAM 使用者指南中的易記名稱和路徑。

> **Important**
>
> 若您的帳戶已建立 Amazon ECS 服務連結角色，則根據預設會為您的服務使用該角色，除非您在此處另行指定。若您的任務定義使用 awsvpc 網路模式，則服務連結角色為必要項目，此時您便不應在此處指定角色。如需詳細資訊，請參閱 使用 Amazon ECS 的服務連結角色。

`serviceConnectConfiguration`

類型：物件

必要：否

可讓此服務探索及連線至服務，並可供命名空間內其他服務探索及連線的組態。

如需詳細資訊，請參閱 使用服務 Connect 以簡短名稱連接 Amazon ECS 服務。

`enabled`

類型：布林值

必要：是

指定是否要搭配此服務使用 Service Connect。

`namespace`

類型：字串

必要：否

與服務 Connect 搭配使用的 AWS Cloud Map 命名空間的簡短名稱或完整 Amazon 資源名稱 (ARN)。命名空間必須位於與 Amazon ECS 服務和叢集相同的 AWS 區域 。命名空間的類型不會影響 Service Connect。如需有關的詳細資訊 AWS Cloud Map，請參閱開AWS Cloud Map 發人員指南中的使用服務。

`services`

類型： 物件陣列

必要：否

Service Connect 服務物件的陣列。這些是其他 Amazon ECS 服務用來連線到此服務的名稱和別名 (又稱為端點)。

對於屬於命名空間成員且僅連線至命名空間內其他服務的「用戶端」Amazon ECS 服務來說，此欄位不是必要欄位。以接受傳入請求的前端應用程式為例，這些請求由連接到服務的負載平衡器傳入或透過其他方式傳入。

物件會從工作定義中選取連接埠、指派 AWS Cloud Map 服務名稱，以及用戶端應用程式參照此服務的別名陣列 (也稱為端點) 和連接埠。

`portName`

類型：字串

必要：是

`portName` 必須符合此 Amazon ECS 服務任務定義中來自所有容器的其中一個 `portMappings` 的 `name`。

`discoveryName`

類型：字串

必要：否

這`discoveryName`是 Amazon ECS 為此 Amazon ECS AWS Cloud Map 服務創建的新服務的名稱。此名稱在 AWS Cloud Map 命名空間中必須不重複。

若沒有指定此欄位，則使用 `portName`。

`clientAliases`

類型： 物件陣列

必要：否

此 Service Connect 服務的用戶端別名清單。您可以使用這些別名，指派用戶端應用程式可使用的名稱。您可以在此清單中擁有的用戶端別名數量上限為 1。

每個別名 (「端點」) 都是其他 Amazon ECS 服務 (「用戶端」) 可用來連線至此服務的 DNS 名稱和連接埠號碼。

每個名稱和連接埠組合在命名空間內必須不重複。

這些名稱在用戶端服務的每個任務中設定，而不是在 AWS Cloud Map中設定。解析這些名稱的 DNS 請求不會離開任務，也不會計入每個彈性網路介面每秒對 DNS 請求的配額。

`port`

類型：整數

必要：是

Service Connect Proxy 的接聽連接埠號碼。此連接埠可在相同命名空間內的所有任務中使用。

若要避免在用戶端 Amazon ECS 服務中變更應用程式，請將此設定為用戶端應用程式預設使用的相同連接埠。

`dnsName`

類型：字串

必要：否

dnsName 是您在用戶端任務應用程式中用來連線到此服務的名稱。名稱必須是有效的 DNS 標籤。

如果未指定欄位，則預設值為 `discoveryName.namespace`。如果未指定 `discoveryName`，則會使用來自任務定義的 `portName`。

若要避免在用戶端 Amazon ECS 服務中變更應用程式，請將此設定為用戶端應用程式預設使用的相同名稱。例如，一些常見名稱為 `database`、`db`，或資料庫的小寫名稱，例如 `mysql` 或 `redis`。

`ingressPortOverride`

類型：整數

必要：否

(選用) Service Connect Proxy 接聽所用的連接埠號碼。

使用此欄位的值以略過此應用程式任務定義中指定之 `portMapping` 所指定的連接埠號碼上流量的 Proxy，然後在 Amazon VPC 安全群組中使用此值，以允許流量進入此 Amazon ECS 服務的 Proxy。

在 `awsvpc` 模式中，預設值是在此應用程式任務定義中指定之 `portMapping` 所指定的容器連接埠號碼。在 `bridge` 模式中，預設值是 Service Connect Proxy 的動態暫時性連接埠。

`logConfiguration`

類型：LogConfiguration物件

必要：否

這會定義 Service Connect Proxy 日誌的發佈位置。在非預期事件期間使用日誌進行偵錯。此組態會在此 Amazon ECS 服務的每個任務中，設定 Service Connect Proxy 容器中的 `logConfiguration` 參數。未在任務定義中指定 Proxy 容器。

我們建議您使用與此 Amazon ECS 服務任務定義的應用程式容器相同的日誌組態。對於 FireLens，這是應用程式容器的記錄組態。它不是使用`fluent-bit`或容器映像的 FireLens 日誌路由`fluentd` 器容器。

`serviceRegistries`

類型： 物件陣列

必要：否

您的服務探索組態詳細資訊。如需詳細資訊，請參閱 使用服務探索將 Amazon ECS 服務與 DNS 名稱連接。

`registryArn`

類型：字串

必要：否

服務登錄檔的 Amazon Resource Name (ARN)。目前支援的服務登錄為 AWS Cloud Map。如需詳細資訊，請參閱 AWS Cloud Map 開發人員指南中的使用服務。

`port`

類型：整數

必要：否

如果您的服務探索服務指定了 SRV 記錄，則使用該連接埠值。如果已使用 awsvpc 網路模式和 SRV 記錄，則此欄位為必要項目。

containerName

類型：字串

必要：否

容器名稱值用於您的「服務探索」服務。該值已於任務定義中指定。如果服務任務指定的任務定義是使用 bridge 或 host 網路模式，您必須透過任務定義指定 containerName 與 containerPort 組合。如果服務任務指定的任務定義使用 awsvpc網路模式，並已使用類型 SRV DNS 記錄，您必須指定 containerName 和 containerPort 組合或 port 值，但不得同時指定兩者。

containerPort

類型：整數

必要：否

用於您的「服務探索」服務的連接埠值。該值已於任務定義中指定。如果服務任務指定的任務定義是使用 bridge 或 host 網路模式，您必須透過任務定義指定 containerName 與 containerPort 組合。如果服務任務指定的任務定義使用 awsvpc 網路模式，並已使用類型 SRV DNS 記錄，您必須指定 containerName 和 containerPort 組合或 port 值，但不得同時指定兩者。

用戶端字符

clientToken

類型：字串

必要：否

由您提供的唯一且區分大小寫的識別符，以確保請求的等冪性。長度上限為 32 個 ASCII 字元。

磁碟區組態

volumeConfigurations

類型：物件

必要：否

將用於為服務管理的工作建立磁碟區的組態。會針對服務中的每個工作建立一個磁碟區。只有在任務定義configuredAtLaunch中標記為的磁碟區才能使用此物件來配置。若要將 Amazon EBS 資料磁碟區附加到由服務管理的任務，需要此物件。如需詳細資訊，請參閱 Amazon EBS 磁碟區。

name

類型：字串

必要：是

建立或更新服務時所設定的磁碟區名稱。允許最多 255 個字母（大寫和小寫），數字，底線（_）和連字符（-）。此值必須符合工作定義中指定的磁碟區名稱。

managedEBSVolume

類型：物件

必要：否

Amazon EBS 磁碟區的磁碟區組態，這些磁碟區會連接到建立或更新服務時由服務管理的任務。

encrypted

類型：布林值

必要：否

有效值：true|false

指定是否將連接至服務管理的任務的 Amazon EBS 磁碟區加密。如果您已預設為帳戶開啟 Amazon EBS 加密，則會覆寫此設定，且磁碟區將加密。如需有關預設 EBS 加密的詳細資訊，請參閱 Amazon EC2 使用者指南中的預設加密。

kmsKeyId

類型：字串

必要：否

用於 Amazon EBS 加密的 AWS Key Management Service (AWS KMS) 金鑰識別碼。如果未指定此參數，則會使 AWS KMS key 用您的 Amazon EBS 專用。如果指定 KmsKeyId，加密狀態必須是 true。

您可以使用下列任何一項指定 KMS 金鑰：

- 金鑰 ID — 例如，1234abcd-12ab-34cd-56ef-1234567890ab。
- 金鑰別名 — 例如，alias/ExampleAlias。
- 關鍵 ARN — 例如，arn:aws:kms:us-east-1:012345678910:key/1234abcd-12ab-34cd-56ef-1234567890ab.
- 別名 ARN — 例如，arn:aws:kms:us-east-1:012345678910:alias/ExampleAlias。

> **Important**
>
> AWS 以非同步方式驗證 KMS 金鑰。因此，如果您指定無效的 ID、別名或 ARN，動作可能會看起來成功，但最終會失敗。如需詳細資訊，請參閱疑難排解 Amazon EBS 磁碟區附件問題。

`volumeType`

類型：字串

必要：否

有效值：gp2gp3| io1 | io2 | sc1 | st1 | standard

EBS 磁碟區類型。如需有關磁碟區類型的詳細資訊，請參閱 Amazon EC2 使用者指南中的 Amazon EBS 磁碟區類型。預設磁碟區類型為 gp3。

> **Note**
>
> 針對standard為附加至 Fargate 任務而設定的 Amazon EBS 磁碟區不支援磁碟區類型。

`sizeInGiB`

類型：整數

必要：否

有效範圍：介於 1 和 16,384 之間的整數

EBS 磁碟區的大小 (以 GiB 為單位)。如果您未提供快照 ID 來設定附件的磁碟區，則必須提供大小值。如果您使用快照設定附件的磁碟區，預設值為快照大小。然後，您可以指定大於或等於快照大小的大小。

對於gp2和gp3磁碟區類型，有效範圍是 1-16,384。

對於io1和io2磁碟區類型，有效範圍是 4-16,384。

對於st1和sc1磁碟區類型，有效範圍是 125-16,384。

對於standard磁碟區類型，有效範圍是 1-1,024。

snapshotId

類型：字串

必要：否

現有 EBS 磁碟區的快照識別碼，用來建立附加至 ECS 工作的新磁碟區。

iops

類型：整數

必要：否

每秒 I/O 操作的次數 (IOPS)。對 gp3、io1 和 io2 磁碟區而言，這代表針對磁碟區佈建的 IOPS 數量。對於gp2磁碟區，此值代表磁碟區的基準效能，以及磁碟區累積 I/O 積分以進行成組分解的速率。這是 io1 和 io2 磁碟區的必要參數。gp2、st1、或standard磁碟區不支援此參數。sc1

對於gp3磁碟區，有效的值範圍為 3,000 到 16,000。

對於io1磁碟區，有效的值範圍是 100 到 64,000。

對於io2磁碟區，有效的值範圍是 100 到 64,000。

throughput

類型：整數

必要：否

為附加至服務所管理之工作的磁碟區佈建輸送量。

> **Important**
>
> 此參數僅支援gp3磁碟區。

`roleArn`

類型：字串

必要：是

基礎設施 (IAM) 角色的 Amazon 資源 ARN AWS Identity and Access Management (ARN)，可為您的任務提供 Amazon ECS 許可來管理 Amazon EBS 資源。如需詳細資訊，請參閱 Amazon ECS 基礎設施 IAM 角色。

`tagSpecifications`

類型：物件

必要：否

要套用至服務受管 Amazon EBS 磁碟區的標籤規格。

`resourceType`

類型：字串

必要：是

有效值：`volume`

建立時要標記的資源類型。

`tags`

類型： 物件陣列

必要：否

您套用至磁碟區的中繼資料，以協助您對其進行分類和整理。每個標籤都包含一個鍵和一個可選值，這兩個值都是您定義的。 `AmazonECSCreated`並且`AmazonECSManaged`是 Amazon ECS 代表您新增的保留標籤，因此您最多可以指定 48 個自己的標籤。刪除磁碟區時，也會刪除標籤。如需詳細資訊，請參閱 標記 Amazon ECS 資源。

key

類型：字串

長度限制：長度下限為 1。長度上限為 128。

必要：否

組成標籤的鍵值對的一部分。索引鍵是一般標籤，作用就像更特定標籤值的類別。

value

類型：字串

長度限制：長度下限為 0。長度上限為 256。

必要：否

組成標籤的鍵值對的可選部分。值就像標籤類別 (索引鍵) 內的描述項。

propagateTags

類型：字串

有效值：TASK_DEFINITION | SERVICE | NONE

必要：否

指定要將標籤從工作定義還是將服務複製到磁碟區。如NONE果指定或未指定值，則不會複製標籤。

fileSystemType

類型：字串

必要：否

有效值：xfs|ext3|ext4

磁碟區上檔案系統的類型。磁碟區的檔案系統類型決定資料在磁碟區中儲存和擷取的方式。對於從快照建立的磁碟區，您必須指定與建立快照時磁碟區所使用的相同檔案系統類型。如果文件系統類型不匹配，則任務將無法啟動。附加至 Linux 工作的磁碟區的預設值為XFS。

服務定義範本

以下顯示 Amazon ECS 服務定義的 JSON 表示法。

Amazon EC2 啟動類型

```
{
    "cluster": "",
    "serviceName": "",
    "taskDefinition": "",
    "loadBalancers": [
        {
            "targetGroupArn": "",
            "loadBalancerName": "",
            "containerName": "",
            "containerPort": 0
        }
    ],
    "serviceRegistries": [
        {
            "registryArn": "",
            "port": 0,
            "containerName": "",
            "containerPort": 0
        }
    ],
    "desiredCount": 0,
    "clientToken": "",
    "launchType": "EC2",
    "capacityProviderStrategy": [
        {
            "capacityProvider": "",
            "weight": 0,
            "base": 0
        }
    ],
    "platformVersion": "",
    "role": "",
    "deploymentConfiguration": {
        "deploymentCircuitBreaker": {
            "enable": true,
            "rollback": true
        },
        "maximumPercent": 0,
        "minimumHealthyPercent": 0,
        "alarms": {
            "alarmNames": [
                ""
```

```
            ],
            "enable": true,
            "rollback": true
        }
    },
    "placementConstraints": [
        {
            "type": "distinctInstance",
            "expression": ""
        }
    ],
    "placementStrategy": [
        {
            "type": "binpack",
            "field": ""
        }
    ],
    "networkConfiguration": {
        "awsvpcConfiguration": {
            "subnets": [
                ""
            ],
            "securityGroups": [
                ""
            ],
            "assignPublicIp": "DISABLED"
        }
    },
    "healthCheckGracePeriodSeconds": 0,
    "schedulingStrategy": "REPLICA",
    "deploymentController": {
        "type": "EXTERNAL"
    },
    "tags": [
        {
            "key": "",
            "value": ""
        }
    ],
    "enableECSManagedTags": true,
    "propagateTags": "TASK_DEFINITION",
    "enableExecuteCommand": true,
    "serviceConnectConfiguration": {
        "enabled": true,
```

```
        "namespace": "",
        "services": [
            {
                "portName": "",
                "discoveryName": "",
                "clientAliases": [
                    {
                        "port": 0,
                        "dnsName": ""
                    }
                ],
                "ingressPortOverride": 0
            }
        ],
        "logConfiguration": {
            "logDriver": "journald",
            "options": {
                "KeyName": ""
            },
            "secretOptions": [
                {
                    "name": "",
                    "valueFrom": ""
                }
            ]
        }
    },
    "volumeConfigurations": [
        {
            "name": "",
            "managedEBSVolume": {
                "encrypted": true,
                "kmsKeyId": "",
                "volumeType": "",
                "sizeInGiB": 0,
                "snapshotId": "",
                "iops": 0,
                "throughput": 0,
                "tagSpecifications": [
                    {
                        "resourceType": "volume",
                        "tags": [
                            {
                                "key": "",
```

```
                                "value": ""
                            }
                        ],
                        "propagateTags": "NONE"
                    }
                ],
                "roleArn": "",
                "filesystemType": ""
            }
        }
    ]
}
```

Fargate 啟動類型

```
{
    "cluster": "",
    "serviceName": "",
    "taskDefinition": "",
    "loadBalancers": [
        {
            "targetGroupArn": "",
            "loadBalancerName": "",
            "containerName": "",
            "containerPort": 0
        }
    ],
    "serviceRegistries": [
        {
            "registryArn": "",
            "port": 0,
            "containerName": "",
            "containerPort": 0
        }
    ],
    "desiredCount": 0,
    "clientToken": "",
    "launchType": "FARGATE",
    "capacityProviderStrategy": [
        {
            "capacityProvider": "",
            "weight": 0,
            "base": 0
```

```
        }
    ],
    "platformVersion": "",
    "platformFamily": "",
    "role": "",
    "deploymentConfiguration": {
        "deploymentCircuitBreaker": {
            "enable": true,
            "rollback": true
        },
        "maximumPercent": 0,
        "minimumHealthyPercent": 0,
        "alarms": {
            "alarmNames": [
                ""
            ],
            "enable": true,
            "rollback": true
        }
    },
    "placementStrategy": [
        {
            "type": "binpack",
            "field": ""
        }
    ],
    "networkConfiguration": {
        "awsvpcConfiguration": {
            "subnets": [
                ""
            ],
            "securityGroups": [
                ""
            ],
            "assignPublicIp": "DISABLED"
        }
    },
    "healthCheckGracePeriodSeconds": 0,
    "schedulingStrategy": "REPLICA",
    "deploymentController": {
        "type": "EXTERNAL"
    },
    "tags": [
        {
```

```
            "key": "",
            "value": ""
        }
    ],
    "enableECSManagedTags": true,
    "propagateTags": "TASK_DEFINITION",
    "enableExecuteCommand": true,
    "serviceConnectConfiguration": {
        "enabled": true,
        "namespace": "",
        "services": [
            {
                "portName": "",
                "discoveryName": "",
                "clientAliases": [
                    {
                        "port": 0,
                        "dnsName": ""
                    }
                ],
                "ingressPortOverride": 0
            }
        ],
        "logConfiguration": {
            "logDriver": "journald",
            "options": {
                "KeyName": ""
            },
            "secretOptions": [
                {
                    "name": "",
                    "valueFrom": ""
                }
            ]
        }
    },
    "volumeConfigurations": [
        {
            "name": "",
            "managedEBSVolume": {
                "encrypted": true,
                "kmsKeyId": "",
                "volumeType": "",
                "sizeInGiB": 0,
```

```
                "snapshotId": "",
                "iops": 0,
                "throughput": 0,
                "tagSpecifications": [
                    {
                        "resourceType": "volume",
                        "tags": [
                            {
                                "key": "",
                                "value": ""
                            }
                        ],
                        "propagateTags": "NONE"
                    }
                ],
                "roleArn": "",
                "filesystemType": ""
            }
        }
    ]
}
```

您可以使用下列 AWS CLI 命令建立此服務定義範本。

```
aws ecs create-service --generate-cli-skeleton
```

標記 Amazon ECS 資源

為協助您管理 Amazon ECS 資源，您可以使用標籤選擇性地將您自己的中繼資料指派給每個資源。每個標籤皆包含索引鍵與選用值。

您可以使用標籤以不同方式將 Amazon ECS 資源分類，例如依用途、擁有者或環境。這在您擁有許多相同類型的資源時很有用。您可以根據您指派給資源的標籤快速識別特定資源。例如，您可以為帳戶的 Amazon ECS 容器執行個體定義一組標籤，這可以協助您追蹤每個執行個體的擁有者和堆疊層級。

您可以將標籤用於成本和用量報告。您可以使用這些報告分析 Amazon ECS 資源的成本和用量。如需詳細資訊，請參閱 the section called “用量報告”。

> **Warning**
>
> 有許多 API 會傳回標籤鍵及其值。拒絕存取 `DescribeTags` 不會自動拒絕存取其他 API 傳回的標籤。根據最佳實務，建議您不要在標籤中包含敏感資料。

我們建議您為每種資源類型建立符合您需求的標籤金鑰。您可以使用一致的標籤索引鍵組，讓您更輕鬆的管理您的資源。您可以根據您新增的標籤搜尋和篩選資源。

標籤對 Amazon ECS 來說不具有任何語意意義，並會嚴格解譯為字元字串。您可以編輯標籤金鑰和值，並且可以隨時從資源移除標籤。您可以將標籤的值設為空白字串，但您無法將標籤的值設為 Null。若您將與現有標籤具有相同鍵的標籤新增到該資源，則新值會覆寫早前的值。刪除資源時，也會刪除資源的所有標籤。

如果您使用 AWS Identity and Access Management (IAM)，您可以控制 AWS 帳戶中哪些使用者有權管理標籤。

如何標記資源

Amazon ECS 任務、服務、任務定義和叢集有多種標記方式：

- 使用者使用 AWS Management Console、Amazon ECS API 或 AWS 開發套件手動標記資源。AWS CLI
- 使用者需要建立服務或執行獨立任務，然後選取 Amazon ECS 受管標籤選項。

  Amazon ECS 會自動標記所有新啟動的任務。如需詳細資訊，請參閱 the section called “Amazon ECS 受管標籤”。

- 使用者可使用主控台建立資源。主控台會自動標記資源。

  這些標籤會在和 AWS SDK 回應中傳回 AWS CLI，並顯示在主控台中。您無法修改或刪除這些標籤。

  如需有關新增標籤的資訊，請參閱 Amazon ECS 資源的標記支援資料表中的主控台自動新增的標籤欄。

如果您在建立資源時指定標籤，但卻無法套用標籤，Amazon ECS 會復原建立程序。這可確保資源不是具有標籤建立，就是不會建立，因此無論何時都不會有不具有標籤的資源。藉由在建立時為資源建立標籤，您可以消除在資源建立後執行自訂標籤指令碼的必要。

下表說明支援標記的 Amazon ECS 資源。

Amazon ECS 資源的標記支援

資源	支援標籤	支援標籤傳播	主控台自動新增的標籤
Amazon ECS 任務	是	是，從任務定義。	索引鍵：`aws:ecs:clusterName` Value (值)：`cluster-name`
Amazon ECS 服務	是	是，從任務定義或服務到服務中的任務。	索引鍵：`ecs:service:stackId` 值`arn:aws:cloudformation: arn`
Amazon ECS 任務集	是	否	N/A
Amazon ECS 任務定義	是	否	索引鍵：`ecs:taskDefinition:createdFrom`

資源	支援標籤	支援標籤傳播	主控台自動新增的標籤
			Value (值)：ecs-console-v2
Amazon ECS 叢集	是	否	索引鍵：aws:cloudformation:logical-id Value (值)：ECSCluster 索引鍵：aws:cloudformation:stack-id Value (值)：arn:aws:cloudformation: *arn* 索引鍵：aws:cloudformation:stack-name Value (值)：ECS-Console-V2-Cluster- *EXAMPLE*
Amazon ECS 容器執行個體。	是	是，來自 Amazon EC2 執行個體。如需詳細資訊，請參閱 將標籤新增至 Amazon ECS 容器執行個體。	N/A
Amazon ECS 外部執行個體	是	否	N/A

資源	支援標籤	支援標籤傳播	主控台自動新增的標籤
Amazon ECS 容量提供者	是。 您無法標記預先定義的 FARGATE 和 FARGATE_SPOT 容量提供者。	否	N/A

在建立期間標記資源

下列資源支援使用 Amazon ECS API 或 AWS SDK 在建立時進行標記： AWS CLI

- Amazon ECS 任務
- Amazon ECS 服務
- Amazon ECS 任務定義
- Amazon ECS 任務集
- Amazon ECS 叢集
- Amazon ECS 容器執行個體。
- Amazon ECS 容量提供者

Amazon ECS 可選擇使用標記授權來建立資源。設定 AWS 帳戶 為標記授權時，使用者必須具有建立資源之動作的權限，例如`ecsCreateCluster`。如果您在資源建立動作中指定標籤，請 AWS 執行其他授權以確認使用者或角色是否具有建立標籤的權限。因此，您必須授予使用 `ecs:TagResource` 動作的明確許可。如需詳細資訊，請參閱 the section called “在建立期間標記資源”。如需有關如何設定選項的資訊，請參閱the section called “標記授權”。

限制

以下限制適用於標籤：

- 資源最多可與 50 個標籤建立關聯。
- 單一資源的標籤索引鍵不能重複。每個標籤索引鍵都必須是唯一的，而且只能有一個值。

- 索引鍵的長度上限是 128 個 UTF-8 字元。
- 索引鍵的長度上限是 256 個 UTF-8 字元。
- 如果多個 AWS 服務 和資源使用您的標記結構描述，請限制您使用的字元類型。某些服務可能對允許的字元設有限制。通常允許的字元為：字母、數字和空格，以及下列字元：**+ - = . _ : / @**。
- 標籤鍵與值皆區分大小寫。
- 您無法使用 `aws:`、`AWS:` 或任何大小寫組合作為索引鍵或值的字首。這些僅保留 AWS 用途。您不可編輯或刪除具此字首的標籤金鑰或值。具有此前綴的標籤不會計入您的 tags-per-resource 限制。

Amazon ECS 受管標籤

當您使用 Amazon ECS 管理的標籤時，Amazon ECS 會使用叢集資訊和使用者新增的任務定義標籤或服務標籤，自動標記所有新啟動的任務和任何附加到任務的 Amazon EBS 磁碟區。以下說明新增的標籤：

- 獨立任務 – 索引鍵為 `aws:ecs:clusterName` 且值設定為叢集名稱的標籤。使用者新增的所有任務定義標籤。連接到獨立任務的 Amazon EBS 磁碟區將會收到標籤，其中包含金鑰為`aws:ecs:clusterName`和叢集名稱設定值的標籤。如需 Amazon EBS 磁碟區標記的詳細資訊，請參閱標記 Amazon EBS 磁碟區。
- 屬於服務一部分的任務 – 索引鍵為 `aws:ecs:clusterName` 且值設定為叢集名稱的標籤。索引鍵為 `aws:ecs:serviceName` 且值設定為服務名稱的標籤。來自下列其中一項資源的標籤：
  - 任務定義 – 使用者新增的所有任務定義標籤。
  - 服務 – 使用者新增的所有服務標籤。

  連接到屬於服務一部分之任務的 Amazon EBS 磁碟區將會收到一個標籤，其中包含「金鑰為」`aws:ecs:clusterName` 和「值」設定為叢集名稱的標籤，以及將「金鑰為」`aws:ecs:serviceName` 和「值」設定為服務名稱的標籤。如需 Amazon EBS 磁碟區標記的詳細資訊，請參閱標記 Amazon EBS 磁碟區。

對此功能而言，下列選項為必要：

- 您必須選擇加入新的 Amazon Resource Name (ARN) 和資源識別符 (ID) 格式。如需詳細資訊，請參閱 Amazon Resource Name (ARN) 和 ID。
- 當您使用 API 建立服務或執行工作時，您必須為 `run-task` 和 `create-service` 將 `enableECSManagedTags` 設定至 `true`。如需詳細資訊，請參閱 AWS Command Line Interface API 參考中的 create-service 和 run-task。

- Amazon ECS 使用受管標籤來判斷何時啟用某些功能，例如叢集 Auto Scaling。我們建議您不要手動修改標籤，以便 Amazon ECS 能夠有效地管理這些功能。

使用標籤進行帳單

AWS 提供名為「Cost Explorer」的報告工具，可用來分析 Amazon ECS 資源的成本和用量。

您可以使用 Cost Explorer 來檢視用量和成本的圖表。您可以檢視過去 13 個月以來的資料，並預測未來三個月的可能花費。您可以使用 Cost Explorer 來查看在一段時間內的 AWS 資源支出模式。例如，您可以用它來找出需進一步調查的領域，以及查看您可用來了解成本的趨勢。您也可以指定資料的時間範圍，以及根據天或月檢視時間資料。

您可以將 Amazon ECS 受管標籤或使用者新增的標籤用於成本和用量報告。如需詳細資訊，請參閱 Amazon ECS 用量報告。

若想要查看合併資源的成本，您可根據具有相同標籤金鑰值的資源來整理您的帳單資訊。例如，您可以使用特定應用程式名稱來標記數個資源，然後整理帳單資訊以查看該應用程式跨數項服務的總成本。如需有關使用標籤設定成本分配報告的詳細資訊，請參閱《AWS Billing 使用者指南》中的每月成本分配報告。

此外，您可以開啟拆分成本分配資料，在成本和用量報告中取得任務層級 CPU 和記憶體用量資料。如需詳細資訊，請參閱 任務層級成本和用量報告。

> **Note**
>
> 若您開啟報告，目前月份的資料會在 24 小時之後提供檢視。

向 Amazon ECS 資源添加標籤

您可以標記新的或現有的工作、服務、工作定義或叢集。如需標記容器執行個體的相關資訊，請參閱將標籤新增至 Amazon ECS 容器執行個體。

> **Warning**
>
> 請勿在標籤中加入個人身分識別資訊 (PII) 或其他機密或敏感資訊。許多 AWS 服務均可存取標籤，包括帳單。標籤不適用於私人或敏感資料。

建立資源時，您可以使用以下資源來指定標籤。

任務	主控台	AWS CLI	API 動作
執行一或多個任務。	以 Amazon ECS 任務的形式執行應用程式	run-task	RunTask
建立服務。	使用主控台建立 Amazon ECS 服務	create-service	CreateService
建立任務集。	使用第三方控制器部署 Amazon ECS 服務	create-task-set	CreateTaskSet
註冊任務定義。	the section called “使用主控台建立任務定義”	register-task-definition	RegisterTaskDefinition
建立叢集。	為 Fargate 啟動類型建立 Amazon ECS 叢集	create-cluster	CreateCluster
執行一或多個容器執行個體。	啟動 Amazon ECS Linux 容器執行個體	run-instances	RunInstances

新增標籤至現有資源 (Amazon ECS 主控台)

您可以直接從資源頁面新增或刪除與叢集、服務、工作和工作定義相關聯的標籤。

修改個別資源的標籤

1. 開啟主控台，網址為 https://console.aws.amazon.com/ecs/v2。
2. 從導覽列中，選取 AWS 區域 要使用的。
3. 在導覽窗格中，選取資源類型 (例如，Clusters (叢集))。

4. 從資源清單選取資源，選擇 Tags (標籤) 索引標籤，然後選擇 Manage tags (管理標籤)。
5. 設定標籤。

   [新增標籤] 選擇新增標籤，然後執行下列操作：

   - 在索引鍵中，輸入索引鍵名稱。
   - 在值中，進入索引鍵值。
6. 選擇儲存。

將標籤新增至現有資源 (AWS CLI)

您可以使用或 API 新增或覆寫一 AWS CLI 或多個標籤。

- AWS CLI -標籤資源
- API 動作-TagResource

將標籤新增至 Amazon ECS 容器執行個體

您可以使用以下方法其中之一，將標籤與您的容器執行個體建立關聯：

- 方法 1：當您使用 Amazon EC2 API、CLI 或主控台建立容器執行個體時，使用容器代理程式組態參數 ECS_CONTAINER_INSTANCE_TAGS 將使用者資料傳遞到執行個體，以指定標籤。這樣會建立僅在 Amazon ECS 中與容器執行個體相關聯的標記，使用 Amazon EC2 API 無法將其列出在。如需詳細資訊，請參閱 引導 Amazon ECS Linux 容器執行個體以傳遞資料。

  > **Important**
  >
  > 如果您使用 Amazon EC2 Auto Scaling 群組啟動容器執行個體，則應使用 ECS_CONTAINER_INSTANCE_TAGS 代理程式組態參數來新增標籤。這是由於標籤新增到了使用 Auto Scaling 群組啟動的 Amazon EC2 執行個體。

  以下的使用者資料指令碼範例將標籤與您的容器執行個體建立關聯：

```
#!/bin/bash
cat <<'EOF' >> /etc/ecs/ecs.config
ECS_CLUSTER=MyCluster
```

```
ECS_CONTAINER_INSTANCE_TAGS={"tag_key": "tag_value"}
EOF
```

- 方法 2 – 當您使用 Amazon EC2 API、CLI 或主控台建立容器執行個體時，應先使用 `TagSpecification.N` 參數指定標籤。然後，使用容器代理程式組態參數 `ECS_CONTAINER_INSTANCE_PROPAGATE_TAGS_FROM` 將使用者資料傳遞至執行個體。這樣做可以將它們從 Amazon EC2 傳播到 Amazon ECS。

  以下的使用者資料指令碼範例會傳播與 Amazon EC2 執行個體相關聯的標籤，並向名為 `MyCluster` 的叢集註冊該執行個體。

  ```
  #!/bin/bash
  cat <<'EOF' >> /etc/ecs/ecs.config
  ECS_CLUSTER=MyCluster
  ECS_CONTAINER_INSTANCE_PROPAGATE_TAGS_FROM=ec2_instance
  EOF
  ```

  若要提供存取，以允許容器執行個體標籤從 Amazon EC2 傳播到 Amazon ECS，請將以下許可當作內嵌政策，手動新增至 Amazon ECS 容器執行個體 IAM 角色。如需詳細資訊，請參閱新增和移除 IAM 政策。

  - `ec2:DescribeTags`

  以下為用來新增這些許可的政策範例。

  ```
  {
      "Version": "2012-10-17",
      "Statement": [
          {
              "Effect": "Allow",
              "Action": [
               "ec2:DescribeTags"
              ],
              "Resource": "*"
          }
      ]
  }
  ```

外部容器實例

您可以使用以下其中一種方法，將標籤與您的外部容器執行個體建立關聯。

- 方法 1：在執行安裝指令碼以註冊您的外部執行個體到叢集之前，在 `/etc/ecs/ecs.config` 建立或編輯 Amazon ECS 容器代理程式組態檔案，並且新增 `ECS_CONTAINER_INSTANCE_TAGS` 容器代理程式組態參數。這將會建立與外部執行個體相關聯的標籤。

  以下為範例語法。

  ```
  ECS_CONTAINER_INSTANCE_TAGS={"tag_key": "tag_value"}
  ```

- 方法 2 — 在您的外部執行個體註冊至叢集之後，您可以使用 AWS Management Console 來新增標籤。如需詳細資訊，請參閱 新增標籤至現有資源 (Amazon ECS 主控台)。

Amazon ECS 用量報告

AWS 提供名為「Cost Explorer」的報告工具，可用來分析 Amazon ECS 資源的成本和用量。

您可以使用 Cost Explorer 來檢視用量和成本的圖表。您可以檢視過去 13 個月以來的資料，並預測未來三個月的可能花費。您可以使用 Cost Explorer 來查看在一段時間內的 AWS 資源支出模式。例如，您可以用它來找出需進一步調查的領域，以及查看您可用來了解成本的趨勢。您也可以指定資料的時間範圍，以及根據天或月檢視時間資料。

成本與用量報告中的計量資料會顯示所有 Amazon ECS 任務的用量。針對執行的每項任務，計量資料會包含做為 `vCPU-Hours` 的 CPU 用量以及做為 `GB-Hours` 記憶體用量。資料的呈現方式取決於任務的啟動類型。

針對使用 Fargate 啟動類型的任務，`lineItem/Operation` 欄會顯示 `FargateTask`，而且您會看到與每項任務相關聯的費用。

針對使用 EC2 啟動類型的任務，`lineItem/Operation` 欄會顯示 `ECSTask-EC2`，而且任務不會有與其相關聯的直接費用。報告中顯示的計量資料，如記憶體用量，代表任務在指定計費期間預留的總資源。您可以使用這個資料來判斷 Amazon EC2 執行個體的基礎叢集費用。Amazon EC2 服務下會個別列出 Amazon EC2 執行個體的費用和用量資料。

您也可以使用 Amazon ECS 受管標籤來識別每項任務所屬的服務或叢集。如需詳細資訊，請參閱 使用標籤進行帳單。

> **Important**
>
> 只有在 2018 年 11 月 16 日或之後啟動的任務，才有計量資料可供檢視。在此日期之前啟動的任務不會顯示計量資料。

以下是 Cost Explorer 中可以用來排序成本分配資料的一些欄位範例。

- 叢集名稱
- 服務名稱
- 資源標籤
- 啟動類型
- AWS 區域
- 用量類型

有關「如何創建 AWS 成本和使用報表」的更多內容，敬請參閱《用AWS Billing 戶指南》中的「AWS 成本和使用報表」。

任務層級成本和用量報告

AWS Cost Management 可以在中 AWS Cost and Usage Report 為 Amazon ECS 上的每個任務提供 CPU 和記憶體使用量資料，包括 Fargate 上的任務和 EC2 上的任務。此資料稱為拆分成本分配資料。您可以使用此資料來分析應用程式的成本和用量。此外，您可以拆分成本並將成本分配給具有成本分配標籤和成本類別的個別業務單位和團隊。如需「分割成本配置資料」的詳細資訊，請參閱《 AWS Cost and Usage Report 使用指南》中的瞭解分割成本配置資料。

您可以選擇加入 AWS Cost Management Console帳戶的任務層級拆分成本分配資料。如果您有一個管理 (付款人) 帳戶，您可以從付款人帳戶中選擇加入此組態，並將其套用至每個連結的帳戶。

設定「分割成本配置資料」之後，報表splitLineItem標題下會有額外的欄。如需詳細資訊，請參閱《 AWS Cost and Usage Report 使用指南》中的「分割商品項目

對於 EC2 上的任務，此資料會根據資源用量或保留以及執行個體上的剩餘資源來拆分 EC2 執行個體的成本。

以下是先決者：

- 將 `ECS_DISABLE_METRICS` Amazon ECS 代理程式組態參數設定為`false`。

  當此設定為時`false`，Amazon ECS 代理程式會將指標傳送至 Amazon CloudWatch。在 Linux 上，此設定為預`false`設值，並將量度傳送至 CloudWatch。在 Windows 上，此設定為`true`預設，因此您必須將設定變更為以`false`將量度傳送至以 CloudWatch供 AWS Cost Management 使用。如需 ECS 代理程式組態的詳細資訊，請參閱 Amazon ECS 容器代理程式組態。

- 可靠指標的最低 Docker 版本是 Docker 版本 v20.10.13 及更新版本，該版本隨附於 Amazon ECS 最佳化 AMI 20220607 及更新版本中。

若要使用拆分成本分配資料，您必須建立報表，然後選取 Split cost allocation data (拆分成本分配資料)。如需詳細資訊，請參閱《使用指南》中的「 AWS Cost and Usage Report 建立成本和使用情況報告」

AWS Cost Management 會計算「分割成本配置資料」與工作 CPU 和記憶體使用量。 AWS Cost Management 如果使用量不可用，則可以使用任務 CPU 和內存保留區而不使用使用情況。如果您看到 CUR 正在使用保留項目，請檢查您的容器執行個體是否符合先決條件，並且工作資源使用量度量顯示在中 CloudWatch。

監控 Amazon ECS

監控是維持 Amazon ECS 和 AWS 解決方案的可靠性、可用性和效能的重要組成部分。您應該從 AWS 解決方案的所有部分收集監視資料，以便在發生多點失敗時更輕鬆地偵錯。開始監控 Amazon ECS 之前，請先建立包含下列問題答案的監控計劃：

- 監控目標是什麼？
- 要監控哪些資源？
- 監控這些資源的頻率為何？
- 要使用哪些監控工具？
- 誰將執行監控任務？
- 發生問題時應該通知誰？

提供的指標依據叢集中的任務與服務的啟動類型而定。如果您的服務使用的是 Fargate 啟動類型，將提供 CPU 與記憶體使用率指標，以協助您監控服務。對於 Amazon EC2 啟動類型，您擁有且需要監控構成基礎設施的 EC2 執行個體。叢集、服務和工作提供額外的 CPU 和記憶體保留和使用率指標。

下一步是在各個時間點和不同的負載條件下測量效能，以在您的環境中確立 Amazon ECS 正常效能的基準。當您監控 Amazon ECS 時，請存放歷史記錄監控資料，如此才能與目前的效能資料做比較、辨識正常效能模式和效能異常狀況、規劃問題處理方式。

若要建立基準，您至少必須監控下列項目：

- 您的 Amazon ECS 叢集的 CPU 和記憶體預留及使用率指標
- 您的 Amazon ECS 服務的 CPU 及記憶體使用率指標

  如需詳細資訊，請參閱 檢視 Amazon ECS 指標。

監控 Amazon ECS 的最佳實務

請使用下列最佳實務來監控 Amazon ECS。

- 在小問題成為大問題之前，將監視作為優先事項
- 建立監視計劃，其中包含下列問題的答案
  - 監控目標是什麼？
  - 要監控哪些資源？

  - 監控這些資源的頻率為何？
  - 要使用哪些監控工具？
  - 誰將執行監控任務？
  - 發生問題時應該通知誰？
- 盡可能自動化監控。
- 檢查 Amazon ECS 日誌檔。如需詳細資訊，請參閱 檢視 Amazon ECS 容器代理程式日誌。

Amazon ECS 的監控工具

AWS 提供各種可用來監控 Amazon ECS 的工具。您可以設定其中一些工具來進行監控，但有些工具需要手動介入。建議您盡可能自動化監控任務。

自動化監控工具

您可以使用下列自動化監控工具來監看 Amazon ECS，並在發生錯誤時進行回報：

- Amazon CloudWatch 警示 — 觀看您指定期間內的單一指標，並根據指定臨界值在多個時段內相對於指定閾值的指標值執行一或多個動作。動作是傳送至亞馬遜簡單通知服務 (Amazon SNS) 主題或 Amazon EC2 Auto Scaling 政策的通知。 CloudWatch 警示不會僅因為處於特定狀態而叫用動作；狀態必須已變更並維持指定數目的期間。如需詳細資訊，請參閱 使用監控 Amazon ECS CloudWatch 。

  對於具有使用 Fargate 啟動類型之任務的服務，您可以根據 CloudWatch 指標 (例如 CPU 和記憶體使用率) 使用 CloudWatch 警示來擴展和擴充服務中的任務。如需詳細資訊，請參閱 自動擴展您的 Amazon ECS 服務。

  對於具有使用 EC2 啟動類型的任務或服務的叢集，您可以根據 CloudWatch 指標 (例如叢集記憶體保留) 使用 CloudWatch 警示來擴展和擴展容器執行個體。

  對於使用 Amazon ECS 最佳化 Amazon Linux AMI 啟動的容器執行個體，您可以使用 CloudWatch 日誌在一個方便的位置檢視來自容器執行個體的不同日誌。您必須在容器執行個體上安裝 CloudWatch 代理程式。如需詳細資訊，請參閱 Amazon 使用 CloudWatch 者指南中的使用命令列下載和設定 CloudWatch 代理程式。您還必須將 `ECS-CloudWatchLogs` 政策新增至 `ecsInstanceRole` 角色。如需詳細資訊，請參閱 監視容器執行個體權。
- Amazon CloudWatch 日誌 — 透過在任務定義中指定日誌驅動程式，從 Amazon ECS 任務中的容器監控、存放和存取日`awslogs`誌檔。如需詳細資訊，請參閱 將 Amazon ECS 日誌傳送到 CloudWatch 。

您也可以從 Amazon ECS 容器執行個體監控、存放及存取作業系統和 Amazon ECS 容器代理程式日誌檔案。這種存取日誌的方法可用於使用 EC2 啟動類型的容器。

- Amazon E CloudWatch vents — 匹配事件並將其路由到一個或多個目標函數或串流，以進行變更、擷取狀態資訊並採取糾正措施。有關更多信息，請參閱本指南使用以下方式自動回應 Amazon ECS 錯誤 EventBridge中的以及什麼是 Amazon CloudWatch 活動？ 在 Amazon CloudWatch 活動用戶指南中。
- 容器洞見 — 從容器化應用程式和微服務收集、彙總和摘要指標和記錄。Container Insights 會使用內嵌指標格式將資料收集為效能日誌事件。這些效能記錄事件是使用結構化 JSON 結構描述的項目，允許大規模擷取和儲存高基數資料。根據此資料， CloudWatch 在叢集、工作和服務層級建立彙總指標作為 CloudWatch 指標。Container Insights 收集的指標可在 CloudWatch 自動儀表板中使用，也可在 CloudWatch主控台的「指標」區段中檢視。
- AWS CloudTrail 防護記錄監控 — 在帳戶之間共用記 CloudTrail 錄檔、即時監控記錄檔案，方法是將 CloudWatch 記錄檔傳送至記錄檔、以 Java 撰寫記錄處理應用程式，以及驗證記錄檔在遞送後未變更 CloudTrail。若要取得更多資訊，請參閱本指南使用記錄 Amazon ECS API 呼叫 AWS CloudTrail中的〈〉和〈使用指南〉中的〈AWS CloudTrail 使用 CloudTrail記錄檔〉。
- 執行階段監控 — 偵測您 AWS 環境中叢集和容器的威脅。執行階段監控使用 GuardDuty 安全代理程式，為個別 Amazon ECS 工作負載新增執行時期可見性，例如檔案存取、程序執行和網路連線。

手動監控工具

監控 Amazon ECS 的另一個重要部分是手動監控 CloudWatch 警示未涵蓋的項目。 CloudWatch Trusted Advisor、和其他 AWS 主控台儀表板可提供您 AWS 環境狀態的 at-a-glance 檢視。我們建議您也檢查您容器執行個體上的日誌檔，以及您任務中的容器。

- Amazon ECS 控制台：
  - EC2 啟動類型的叢集指標
  - 服務指標
  - 服務運作狀態
  - 服務部署事件
- CloudWatch 主頁：
  - 目前警示與狀態
  - 警示與資源的圖表
  - 服務運作狀態

此外，您可以使用執行 CloudWatch 以下操作：

- 建立自訂儀表板來監控您關心的服務。
- 用於疑難排解問題以及探索驅勢的圖形指標資料。
- 搜尋並瀏覽所有資 AWS 源指標。
- 建立與編輯要通知發生問題的警示。

- 容器健康狀態檢查-這些是在容器本機上執行並驗證應用程式健康狀態和可用性的命令。您可以在任務定義中為每個容器配置這些內容。
- AWS Trusted Advisor 可協助您監控 AWS 資源，以改善效能、可靠性、安全性和成本效益。所有使用者均可使用四 Trusted Advisor 項檢查；擁有商業或企業支援方案的使用者可使用超過 50 項檢查。如需詳細資訊，請參閱 AWS Trusted Advisor。

  Trusted Advisor 有這些與 Amazon ECS 相關的檢查：

  - 容錯，表示您的服務在單一可用區域中執行。
  - 容錯，表示您尚未針對多個可用區域使用分攤放置策略。
- AWS Compute Optimizer 是分析資 AWS 源組態和使用率指標的服務。這會報告您的資源是否已為最佳化，並產生最佳化建議，以降低成本並改善工作負載的效能。

  如需詳細資訊，請參閱 AWS Compute Optimizer 針對 Amazon ECS 的建議。

使用監控 Amazon ECS CloudWatch

您可以使用 Amazon 監控您的 Amazon ECS 資源 CloudWatch，Amazon 會從 Amazon ECS 收集原始資料並將其處理成可讀且接近即時的指標。這些統計資料記錄會保留兩週，讓您可存取歷史資訊，且能更清楚叢集或服務的執行方式。Amazon ECS 指標資料會 CloudWatch 在 1 分鐘內自動傳送到。如需有關的詳細資訊 CloudWatch，請參閱 Amazon CloudWatch 使用者指南。

Amazon ECS 為叢集和服務提供免費的指標。您只需支付額外費用，就可以針對每個任務指標 (包括 CPU、記憶體和 EBS 檔案系統使用率) 開啟叢集的 Amazon ECS CloudWatch 容器深入解析。如需更多 Container Insights 的相關資訊，請參閱 使用容器洞察來監控 Amazon ECS 容器。

考量事項

使用 Amazon ECS CloudWatch 指標時，應考慮以下事項。

- Fargate 上託管的任何 Amazon ECS 服務都會自動擁有 CloudWatch CPU 和記憶體使用率指標，因此您無需採取任何手動步驟。

- 對於 Amazon EC2 執行個體上託管的任何 Amazon ECS 任務或服務，Amazon EC2 執行個體需要容器代理程式的版本`1.4.01.0.0`或更新版本 (Linux) 或更新版本 (Windows)，才能產生 CloudWatch 指標。不過，我們建議您使用最新版的容器代理程式。如需檢查代理程式版本及更新至最新版本的資訊，請參閱「更新 Amazon ECS 容器代理程式」。
- 可靠 CloudWatch 度量的最低 Docker 版本是 Docker 版本`20.10.13`和更新版本。
- 您的 Amazon EC2 執行個體也需要您啟動 Amazon EC2 執行個體時所使用的 IAM 角色的`ecs:StartTelemetrySession`許可。如果您在 Amazon ECS CloudWatch 指標可用之前建立了 Amazon ECS 容器執行個體 IAM 角色，則可能需要新增此許可。如需容器執行個體 IAM 角色和附加容器執行個體的受管 IAM 政策的相關資訊，請參閱Amazon ECS 容器執行個體 IAM 角色。
- 您可以在 Amazon ECS 容器代理程式組態`ECS_DISABLE_METRICS=true`中設定，在 Amazon EC2 執行個體上停用 CloudWatch 指標收集。如需詳細資訊，請參閱 Amazon ECS 容器代理程式組態。

建議的指標

Amazon ECS 提供可用來監控資源的免費 CloudWatch 指標。整個叢集中的 CPU 和記憶體保留以及 CPU、記憶體和 EBS 檔案系統使用率，以及叢集中服務的 CPU、記憶體和 EBS 檔案系統使用率可以使用這些度量來測量。針對 GPU 工作負載，您可以量測叢集整體的 GPU 保留。

Amazon ECS 任務在叢集中託管的基礎設施會決定哪些指標可用。對於在 Fargate 基礎設施上託管的任務，Amazon ECS 會提供 CPU、記憶體和 EBS 檔案系統使用率指標，以協助監控您的服務。對於 EC2 執行個體上託管的任務，Amazon ECS 會在叢集和服務層級提供 CPU、記憶體和 GPU 保留指標，以及 CPU 和記憶體使用率指標。您需要監控單獨構成基礎設施的 Amazon EC2 執行個體。如需監控 Amazon EC2 執行個體的詳細資訊，請參閱 Amazon EC2 使用者指南中的監控 Amazon EC2。

如需建議搭配 Amazon ECS 使用的警示的相關資訊，請參閱 Amazon CloudWatch 日誌使用者指南中的下列其中一項：

- Amazon ECS
- 具有容器見解的 Amazon ECS

檢視 Amazon ECS 指標

在叢集中執行資源之後，您可以在 Amazon ECS 和 CloudWatch 主控台上檢視指標。Amazon ECS 主控台提供您的叢集和服務指標 24 小時的最小值、最大值和平均值檢視。主 CloudWatch 控台提供精細且可自訂的資源顯示，以及服務中正在執行的工作數目。

Amazon ECS 主控台

Amazon ECS 服務 CPU 和記憶體使用率指標皆可在 Amazon ECS 主控台中取得。提供的服務指標檢視會顯示前 24 小時期間的平均值、最小值、最大值，並提供 5 分鐘間隔的資料點。如需詳細資訊，請參閱 Amazon ECS 服務使用率指標。

1. 開啟主控台，網址為 https://console.aws.amazon.com/ecs/v2。
2. 選取您要檢視其指標的叢集。
3. 決定要檢視的量度。

待辦事項檢視量度	步驟
叢集	在叢集詳細資訊頁面上，選擇「測量結果」頁籤。如果您已開啟「 CloudWatch容器見解」指標，也會提供 CloudWatch 主控台連結，以檢視這些指標。
服務	在叢集詳細資料頁面的服務索引標籤上，選取服務。然後，可在「Health」和「量度」標籤上使用量度。

CloudWatch 控制台

對於 Fargate 啟動類型，也可以在主控台上檢視 Amazon ECS 服務指標。 CloudWatch 主控台提供最詳細的 Amazon ECS 指標檢視，您可以修改這些檢視以符合您的需求。您可以檢視服務使用率和服務 RUNNING 任務計數。

對於 EC2 啟動類型，也可以在 CloudWatch 主控台上檢視 Amazon ECS 叢集和服務指標。主控台提供最詳細的 Amazon ECS 指標檢視，您可以修改這些檢視以符合您的需求。

有關如何檢視指標的詳細資訊，請參閱 Amazon CloudWatch 使用者指南中的檢視可用指標。

Amazon ECS 指標 CloudWatch

您可以使用使 CloudWatch 用量度來提供您帳戶資源使用情況的可見性。使用這些指標，在 CloudWatch 圖形和儀表板上視覺化您目前的服務使用情況。

`CPUReservation`

保留在叢集或服務中的 CPU 單元百分比。

CPU 保留 (篩選依據`ClusterName`) 是以叢集上 Amazon ECS 任務保留的 CPU 單位總計除以叢集中註冊之所有 Amazon EC2 執行個體的 CPU 單位總計。只有處於`ACTIVE`或`DRAINING`狀態的 Amazon EC2 執行個體會影響 CPU 保留指標。只有在 Amazon EC2 執行個體上託管的任務才支援該指標。

有效維度：`ClusterName`。

有用的統計數據：平均，最小值，最大

單位：百分比。

`CPUUtilization`

叢集或服務所使用的 CPU 單位百分比。

叢集層級 CPU 使用率 (篩選依據`ClusterName`) 是以叢集上 Amazon ECS 任務使用的 CPU 單元總計除以叢集中註冊的所有 Amazon EC2 執行個體的 CPU 單位總計。只有處於`ACTIVE`或`DRAINING`狀態的 Amazon EC2 執行個體會影響 CPU 保留指標。叢集層級指標僅支援在 Amazon EC2 執行個體上託管的任務。

服務層次 CPU 使用率 (篩選依據`ClusterName`、`ServiceName`) 是以屬於該服務的工作所使用的 CPU 單位總計，除以為屬於服務的工作保留的 CPU 單位總數。服務層級指標適用於 Amazon EC2 執行個體和 Fargate 上託管的任務。

有效維度：`ClusterName`、`ServiceName`。

有用的統計數據：平均，最小值，最大

單位：百分比。

`MemoryReservation`

由叢集內執行中任務保留的記憶體的百分比。

叢集記憶體保留的計算方式為叢集上 Amazon ECS 任務保留的總記憶體，除以叢集中註冊之所有 Amazon EC2 執行個體的記憶體總量。此測量結果只能依據篩選`ClusterName`。只有處於`ACTIVE`或`DRAINING`狀態的 Amazon EC2 執行個體會影響記憶體保留指標。叢集層級記憶體保留指標僅支援在 Amazon EC2 執行個體上託管的任務。

> **Note**
>
> 計算記憶體使用率時，如果`MemoryReservation`有指定，則會在計算中使用它，而不是總記憶體。

有效維度：`ClusterName`。

有用的統計數據：平均，最小值，最大

單位：百分比。

`MemoryUtilization`

叢集或服務正在使用的記憶體百分比。

叢集層級記憶體使用率 (篩選依據`ClusterName`) 是以叢集上 Amazon ECS 任務使用的總記憶體除以叢集中註冊之所有 Amazon EC2 執行個體的總記憶體。只有處於`ACTIVE`或`DRAINING`狀態的 Amazon EC2 執行個體會影響記憶體使用率指標。叢集層級指標僅支援在 Amazon EC2 執行個體上託管的任務。

服務層級記憶體使用率 (篩選依據`ClusterName`、`ServiceName`) 是以屬於該服務的工作所使用的總記憶體除以保留給屬於服務之工作的總記憶體。服務層級指標適用於 Amazon EC2 執行個體和 Fargate 上託管的任務。

有效維度：`ClusterName`、`ServiceName`。

有用的統計數據：平均，最小值，最大

單位：百分比。

`EBSFilesystemUtilization`

服務中任務所使用的 Amazon EBS 檔案系統百分比。

服務層次 EBS 檔案系統使用率測量結果 (篩選依據`ClusterName`、`ServiceName`) 是以屬於該服務的作業所使用的 EBS 檔案系統總量來測量，除以為屬於該服務之所有作業配置的 EBS 檔案系統

儲存體總數。服務層級 EBS 檔案系統使用率指標僅適用於已連接 EBS 磁碟區的 Amazon EC2 執行個體 (使用容器代理程式版本1.79.0) 和 Fargate (使用平台版本1.4.0) 上託管的任務。

> **Note**
>
> 對於在 Fargate 上託管的任務，磁盤上只有 Fargate 使用的空間。Fargate 使用的空間沒有相關的費用，但是您將使用類似df的工具看到此額外的存儲空間。

有效維度：ClusterName, ServiceName。

有用的統計數據：平均，最小值，最大

單位：百分比。

GPUReservation

由叢集內執行中任務保留的 GPU 可用總數的百分比。

叢集層級 GPU 保留指標的計算方式為叢集上 Amazon ECS 任務保留的 GPU 數量，除以在叢集中註冊 GPU 的所有 Amazon EC2 執行個體上可用的 GPU 總數。只有處於ACTIVE或DRAINING狀態的 Amazon EC2 執行個體會影響 GPU 保留指標。

有效維度：ClusterName。

有用的統計數據：平均，最小值，最大

所有統計數據：平均值，最小值，最大值，總和，樣本數量。

單位：百分比。

ActiveConnectionCount

從用戶端到在共享所選 DiscoveryName 的任務中執行之 Amazon ECS Service Connect Proxy 的作用中同時連線總數。

僅當您已設定 Amazon ECS Service Connect 時，才能使用此指標。

有效維度：DiscoveryName 和 DiscoveryName, ServiceName, ClusterName。

有用的統計數據：平均值，最小值，最大值，總和。

單位：計數。

`NewConnectionCount`

從用戶端到在共享所選 `DiscoveryName` 的任務中執行之 Amazon ECS Service Connect Proxy 的新建立連線總數。

僅當您已設定 Amazon ECS Service Connect 時，才能使用此指標。

有效維度：`DiscoveryName` 和 `DiscoveryName, ServiceName, ClusterName`。

有用的統計數據：平均值，最小值，最大值，總和。

單位：計數。

`ProcessedBytes`

Service Connect Proxy 處理的輸入流量總位元組數。

僅當您已設定 Amazon ECS Service Connect 時，才能使用此指標。

有效維度：`DiscoveryName` 和 `DiscoveryName, ServiceName, ClusterName`。

有用的統計數據：平均值，最小值，最大值，總和。

單位：位元組。

`RequestCount`

Service Connect Proxy 處理的輸入流量請求數。

僅當您已設定 Amazon ECS Service Connect 時，才能使用此指標。

您還需要`appProtocol`在任務定義中配置端口映射。

有效維度：`DiscoveryName` 和 `DiscoveryName, ServiceName, ClusterName`。

有用的統計數據：平均值，最小值，最大值，總和。

單位：計數。

`GrpcRequestCount`

Service Connect Proxy 處理的 gRPC 輸入流量請求數。

僅當您已設定 Amazon ECS Service Connect 且 `appProtocol` 在任務定義的連接埠映射中為 `GRPC` 時，才能使用此指標。

有效維度：`DiscoveryName` 和 `DiscoveryName, ServiceName, ClusterName`。

有用的統計數據：平均值，最小值，最大值，總和。

單位：計數。

`HTTPCode_Target_2XX_Count`

這些任務中應用程式所產生的編號 200 至 299 的 HTTP 回應代碼數。這些任務是目標。此指標只會計算應用程式在這些任務中傳送至 Service Connect Proxy 的回應，而非直接傳送的回應。

僅當您已設定 Amazon ECS Service Connect 且 `appProtocol` 在任務定義的連接埠映射中為 `HTTP` 或 `HTTP2` 時，才能使用此指標。

有效維度：`TargetDiscoveryName` 和 `TargetDiscoveryName, ServiceName, ClusterName`。

有用的統計數據：平均值，最小值，最大值，總和。

單位：計數。

`HTTPCode_Target_3XX_Count`

這些任務中應用程式所產生的編號 300 至 399 的 HTTP 回應代碼數。這些任務是目標。此指標只會計算應用程式在這些任務中傳送至 Service Connect Proxy 的回應，而非直接傳送的回應。

僅當您已設定 Amazon ECS Service Connect 且 `appProtocol` 在任務定義的連接埠映射中為 `HTTP` 或 `HTTP2` 時，才能使用此指標。

有效維度：`TargetDiscoveryName` 和 `TargetDiscoveryName, ServiceName, ClusterName`。

有用的統計數據：平均值，最小值，最大值，總和。

單位：計數。

`HTTPCode_Target_4XX_Count`

這些任務中應用程式所產生的編號 400 至 499 的 HTTP 回應代碼數。這些任務是目標。此指標只會計算應用程式在這些任務中傳送至 Service Connect Proxy 的回應，而非直接傳送的回應。

僅當您已設定 Amazon ECS Service Connect 且 `appProtocol` 在任務定義的連接埠映射中為 `HTTP` 或 `HTTP2` 時，才能使用此指標。

有效維度：`TargetDiscoveryName` 和 `TargetDiscoveryName, ServiceName, ClusterName`。

有用的統計數據：平均，最小值，最大值，總和

單位：計數。

`HTTPCode_Target_5XX_Count`

這些任務中應用程式所產生的編號 500 至 599 的 HTTP 回應代碼數。這些任務是目標。此指標只會計算應用程式在這些任務中傳送至 Service Connect Proxy 的回應，而非直接傳送的回應。

僅當您已設定 Amazon ECS Service Connect 且 `appProtocol` 在任務定義的連接埠映射中為 HTTP 或 HTTP2 時，才能使用此指標。

有用的統計數據：平均值，最小值，最大值，總和。

單位：計數。

`RequestCountPerTarget`

共享所選 `DiscoveryName` 的每個目標接收的平均請求數。

僅當您已設定 Amazon ECS Service Connect 時，才能使用此指標。

有效維度：`TargetDiscoveryName` 和 `TargetDiscoveryName, ServiceName, ClusterName`。

有用的統計數據：平均。

單位：計數。

`TargetProcessedBytes`

Service Connect Proxy 處理的總位元組數。

僅當您已設定 Amazon ECS Service Connect 時，才能使用此指標。

有效維度：`TargetDiscoveryName` 和 `TargetDiscoveryName, ServiceName, ClusterName`。

有用的統計數據：平均值，最小值，最大值，總和。

單位：位元組。

`TargetResponseTime`

應用程式請求處理的延遲。從請求到達目標任務中的 Service Connect Proxy，直到 Proxy 收到目標應用程式回應所經過的時間 (以毫秒為單位)。

僅當您已設定 Amazon ECS Service Connect 時，才能使用此指標。

有效維度：TargetDiscoveryName 和 TargetDiscoveryName, ServiceName, ClusterName。

有用的統計數據：平均，最小值，最大值。

所有統計數據：平均值，最小值，最大值，總和，樣本數量。

單位：毫秒。

ClientTLSNegotiationErrorCount

TLS 連線失敗的總次數。只有在啟用 TLS 時才會使用此量度。

僅當您已設定 Amazon ECS Service Connect 時，才能使用此指標。

有效尺寸：DiscoveryName和DiscoveryName、ServiceName、ClusterName。

有用的統計數據：平均值，最小值，最大值，總和。

單位：計數。

TargetTLSNegotiationErrorCount

TLS 連線因缺少用戶端憑證、驗證失敗或 SAN AWS Private CA 驗證失敗而失敗的總次數。只有在啟用 TLS 時才會使用此量度。

僅當您已設定 Amazon ECS Service Connect 時，才能使用此指標。

有效尺寸：ServiceNameClusterName、TargetDiscoveryName和TargetDiscoveryName。

有用的統計數據：平均值，最小值，最大值，總和。

單位：計數。

Amazon ECS 指標的維度

Amazon ECS 指標使用 AWS/ECS 命名空間，並提供下列維度的指標。Amazon ECS 只會針對 RUNNING 狀態中有任務的資源傳送指標。例如，如果您有一個包含一個服務的叢集，但該服務沒有處於某個RUNNING狀態的任務，則不會有指標傳送至 CloudWatch。如果您有兩個服務，其中一個有執行中的任務，而另一個沒有，則只會傳送有執行中任務之服務的指標。

`ClusterName`

此維度可篩選您為指定叢集中所有資源請求的資料。所有 Amazon ECS 指標皆以 `ClusterName` 篩選。

`ServiceName`

此維度可篩選您為指定叢集中的指定服務的所有資源請求的資料。

`DiscoveryName`

此維度會篩選您為流量指標請求，且傳送至所有 Amazon ECS 叢集內指定 Service Connect 探索名稱的資料。

請注意，執行中容器內的特定連接埠可以擁有多個探索名稱。

`DiscoveryName, ServiceName, ClusterName`

此維度會篩選您為流量指標請求，且傳送至具有此探索名稱並由此叢集中此服務所建立之任務內指定 Service Connect 探索名稱的資料。

如果您在不同命名空間的多個服務中重複使用相同的探索名稱，請使用此維度來查看特定服務的輸入流量指標。

請注意，執行中容器內的特定連接埠可以擁有多個探索名稱。

`TargetDiscoveryName`

此維度會篩選您為流量指標請求，且傳送至所有 Amazon ECS 叢集內指定 Service Connect 探索名稱的資料。

與 `DiscoveryName` 不同，這些流量指標僅會測量傳送至此 `DiscoveryName`，且來自在此命名空間中具有 Service Connect 組態的其他 Amazon ECS 任務的輸入流量。這包括服務使用僅限用戶端或用戶端-伺服器 Service Connect 組態建立的任務。

請注意，執行中容器內的特定連接埠可以擁有多個探索名稱。

`TargetDiscoveryName, ServiceName, ClusterName`

此維度會篩選您為流量指標請求，且傳送至指定 Service Connect 探索名稱的資料，但僅會計算來自此叢集中此服務所建立任務的流量。

使用此維度來查看來自另一服務中特定用戶端的輸入流量指標。

與 DiscoveryName, ServiceName, ClusterName 不同，這些流量指標僅會測量傳送至此 DiscoveryName，且來自在此命名空間中具有 Service Connect 組態的其他 Amazon ECS 任務的輸入流量。這包括服務使用僅限用戶端或用戶端-伺服器 Service Connect 組態建立的任務。

請注意，執行中容器內的特定連接埠可以擁有多個探索名稱。

AWS Fargate 使用量度

您可以使用使 CloudWatch 用量度來提供您帳戶資源使用情況的可見性。使用這些指標，在 CloudWatch 圖形和儀表板上視覺化您目前的服務使用情況。

AWS Fargate 使用量度對應至 AWS 服務配額。您可以設定警示，在您的用量接近服務配額時發出警示。如需 Fargate 的服務配額詳細資訊，請參閱 AWS Fargate 服務配額。

AWS Fargate 會在AWS/Usage命名空間中發佈下列量度。

指標	描述
ResourceCount	您的帳戶中正在執行的特定資源總數。資源由與指標相關聯的維度定義。

以下維度用於強化 AWS Fargate發佈的用量指標。

維度	描述
Service	包含資源的 AWS 服務名稱。對於 AWS Fargate 用量指標，此維度的值為 Fargate。
Type	正在報告的實體類型。目前， AWS Fargate 使用狀況測量結果的唯一有效值為Resource。
Resource	正在執行的資源類型。正在執行的資源類型。目前， AWS Fargate 使用狀況測量結果唯一有效的值vCPU是傳回執行中執行處理的相關資訊。
Class	正在追蹤的資源類別。正在追蹤的資源類別。對於以 vCPU 作為「資源」維度值的 AWS Fargate 使用量度，有效值為 Standard/OnDemand 和Standard/Spot 。

您可以使用 Service Quotas 主控台在圖表上視覺化您的使用情況，並設定警示，以便在使 AWS Fargate 用量接近服務配額時提醒您。如需如何建立 CloudWatch 警示以在接近配額值閾值時通知您的詳細資訊，請參閱Service Quotas 使用者指南中的 Ser vice Quotas 和 Amazon CloudWatch 警示。

.

Amazon ECS 叢集保留指標

相較於叢集中每個作用中之容器執行個體所登錄之彙總的 CPU、記憶體和 GPU，叢集保留指標是依叢集上所有 Amazon ECS 任務保留的 CPU、記憶體和 GPU 百分比計量。只有處於 ACTIVE 或 DRAINING 狀態的容器執行個體會影響叢集保留指標。此指標僅適用於在 EC2 執行個體上託管任務或服務的叢集。在託管工作的叢集上不支援此功能 AWS Fargate。

```
                              (Total CPU units reserved by tasks in cluster) x 100
Cluster CPU reservation =
 ---------------------------------------------------------------
                           (Total CPU units registered by container instances in
 cluster)
```

```
                                 (Total MiB of memory reserved by tasks in cluster x
 100)
Cluster memory reservation =
 ---------------------------------------------------------------------
                              (Total MiB of memory registered by container instances in
 cluster)
```

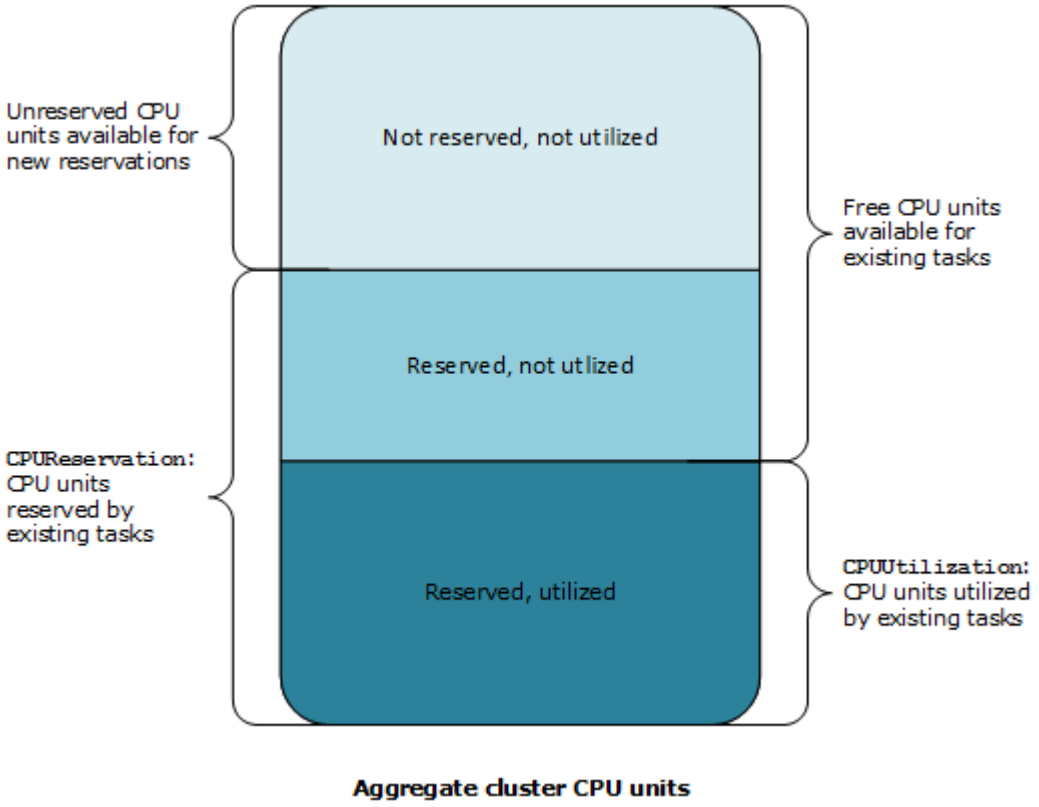
```
                                (Total GPUs reserved by tasks in cluster x 100)
Cluster GPU reservation =
 ---------------------------------------------------------------------
                              (Total GPUs registered by container instances in cluster)
```

當您在叢集中執行任務時，Amazon ECS 會剖析其任務定義，並保留其容器定義中指定的彙總 CPU 單位、記憶體 MiB 和 GPU。Amazon ECS 每分鐘都會計算目前在叢集中，每項執行中之任務保留的 CPU 單位、記憶體 MiB 和 GPU 的數量。系統會計算為叢集上執行的所有工作保留的 CPU、記憶體和 GPU 總量，並將這些數目報告 CloudWatch 為叢集已註冊資源總數的百分比。如果您指定任務定義中的軟性限制 (`memoryReservation`)，則其將用於計算保留的記憶體數量。否則，會使用硬性限制 (`memory`)。叢集中由任務所保留的總 MiB 記憶體也包括暫存檔案系統 (`tmpfs`) 磁碟區大小以及 `sharedMemorySize` (若任務定義中有定義)。如需硬性和軟性限制、共用記憶體大小及 tmpfs 磁碟區大小的詳細資訊，請參閱任務定義參數。

例如，叢集已登錄兩個作用中的容器執行個體：c4.4xlarge 執行個體和 c4.large 執行個體。c4.4xlarge 執行個體在叢集中登錄為 16,384 個 CPU 單位和 30,158 MiB 記憶體。c4.large 執行個體登錄為 2,048 個 CPU 單位和 3,768 MiB 記憶體。此叢集的彙總資源為 18,432 個 CPU 單位和 33,926 MiB 記憶體。

如果任務定義保留 1,024 個 CPU 單位和 2,048 MiB 記憶體，在此叢集上有十項任務使用此任務定義啟動 (目前未執行任何其他任務)，總共保留 10,240 個 CPU 單位和 20,480 MiB 記憶體。報告 CloudWatch 為叢集的 55% CPU 保留區和 60% 記憶體保留區。

下圖顯示叢集中的總登錄 CPU 單位，及其保留和使用率對現有的任務和新任務置放所代表的意義。較低的 (保留、已使用) 和中心 (保留，未使用) 區塊代表為叢集上執行的現有工作或CPUReservation CloudWatch 指標保留的 CPU 單位總計。下方區塊代表執行中工作在叢集上實際使用的保留 CPU 單位，或指CPUUtilization CloudWatch 標。上方的區塊表示現有任務未保留的 CPU 單位，這些 CPU 單位可供新任務置放使用。現有的任務也可以利用這些未保留的 CPU 單位，如果其 CPU 資源需求增加。如需詳細資訊，請參閱「cpu」任務定義參數文件。



Amazon ECS 叢集使用率指標

叢集使用率度量可用於 CPU、記憶體，以及 EBS 磁碟區連接至您的任務時，EBS 檔案系統使用率。這些指標僅適用於在 Amazon EC2 執行個體上託管的任務或服務的叢集。在託管工作的叢集上不支援它們 AWS Fargate。

Amazon ECS 叢集級 CPU 和記憶體使用率指標

CPU 和記憶體使用率是以叢集上所有任務所使用的 CPU 和記憶體百分比來衡量，與註冊到叢集的每個作用中 Amazon EC2 執行個體的彙總 CPU 和記憶體相比。只有處於ACTIVE或DRAINING狀態的 Amazon EC2 執行個體會影響叢集使用率指標。

```
                                  (Total CPU units used by tasks in cluster) x 100
Cluster CPU utilization =
 ------------------------------------------------------------
                           (Total CPU units registered by container instances in
 cluster)
```

```
                                     (Total MiB of memory used by tasks in cluster x
 100)
Cluster memory utilization =
 ------------------------------------------------------------------
                              (Total MiB of memory registered by container instances in
 cluster)
```

每一分鐘，每個 Amazon EC2 執行個體上的 Amazon ECS 容器代理程式都會計算該 Amazon EC2 執行個體上執行的每個任務目前使用的 CPU 單位數和記憶體 MiB 數量，然後將此資訊回報給 Amazon ECS。系統會計算叢集上執行之所有工作所使用的 CPU 和記憶體總量，並將這些數字報告 CloudWatch 為叢集已註冊資源總數的百分比。

例如，叢集有兩個已註冊的作用中 Amazon EC2 執行個體、一個c4.4xlarge執行個體和一個c4.large執行個體。c4.4xlarge執行個體會使用 16,384 CPU 單元和 30,158 MiB 記憶體註冊到叢集中。c4.large執行個體會註冊 2,048 CPU 單元和 3,768 MiB 記憶體。此叢集的彙總資源為 18,432 CPU 單位和 33,926 MiB 記憶體。

如果此叢集上正在執行十項工作，且每項工作都耗用 1,024 CPU 單位和 2,048 MiB 記憶體，則叢集上會使用總計的 10,240 CPU 單位和 20,480 MiB 記憶體。據報告，叢集的 CPU 使用率 CloudWatch 為 55%，記憶體使用率為 60%。

Amazon ECS 叢集級 Amazon EBS 檔案系統使用率

叢集層級 EBS 檔案系統使用率測量結果是以叢集上執行之作業所使用的 EBS 檔案系統總量 (除以為叢集中所有作業配置的 EBS 檔案系統儲存體總量) 進行測量。

```
                                     (Total GB of EBS filesystem used by tasks in
 cluster x 100)
```

```
Cluster EBS filesystem utilization =
 ------------------------------------------------------------
                                      (Total GB of EBS filesystem allocated to tasks
 in cluster)
```

Amazon ECS 服務使用率指標

服務使用率指標可用於 CPU、記憶體，以及 EBS 磁碟區連接至您的任務時，EBS 檔案系統使用率。在 Amazon EC2 執行個體和 Fargate 上託管的任務的服務均支援服務層級指標。

服務層級 CPU 和記憶體使用率

CPU 和記憶體使用率是以叢集上某個服務的 Amazon ECS 任務所使用的 CPU 和記憶體百分比來衡量，與服務任務定義中指定的 CPU 和記憶體相比。

```
                                    (Total CPU units used by tasks in service) x 100
Service CPU utilization =
 ------------------------------------------------------------------------------
                           (Total CPU units specified in task definition) x (number of
 tasks in service)
```

```
                                     (Total MiB of memory used by tasks in service) x
 100
Service memory utilization =
 ------------------------------------------------------------------------------
                              (Total MiB of memory specified in task definition) x
 (number of tasks in service)
```

Amazon ECS 容器代理程式每分鐘都會計算服務擁有的每個任務目前正在使用的 CPU 單元和記憶體 MiB 數量，然後將此資訊回報給 Amazon ECS。系統會計算叢集上執行之服務所擁有之所有工作所使用的 CPU 和記憶體總量，而這些數目會以服務作業定義中 CloudWatch 為服務指定的總資源百分比來回報。如果您指定軟性限制 (`memoryReservation`)，則會用以計算保留的記憶體數量。否則，會使用硬性限制 (`memory`)。如需有關硬限制和軟限制的更多資訊，請參閱任務大小。

例如，服務的任務定義為其所有容器指定總共 512 個 CPU 單位和 1,024 MiB 記憶體 (使用硬性限制 `memory` 參數)。服務的所需計數為 1 項執行中的任務，此服務正使用 1 個 `c4.large` 容器執行個體 (總共 2,048 個 CPU 單位和 3,768 MiB 記憶體) 在叢集上執行，且叢集上目前未執行其他任務。雖然任務指定的是 512 CPU 單位，因為其是在具有 2,048 CPU 單位的容器執行個體上唯一的執行中任務，該任務可以使用指定數量 (2,048/512) 多達四次。然而，指定記憶體 1,024 MiB 是硬性限制且不得被超過，因此在此情況下，服務記憶體使用率不得超過 100%。

如果之前的範例使用軟性限制 memoryReservation 而非硬性限制 memory 參數，則服務的任務如有需要，可以使用超過所指定的 1,024 MiB 記憶體。在這種情況下，服務的記憶體使用率可能超過 100%。

如果您的應用程式在短時間內突然激增記憶體使用率，您將不會看到服務記憶體使用率增加，因為 Amazon ECS 每分鐘收集多個資料點，然後將它們彙總到一個傳送到的資料點。 CloudWatch

如果此任務在某期間內執行 CPU 密集型工作，並使用所有 2,048 個可用的 CPU 單位和 512 MiB 的記憶體，則服務會報告 400% 的 CPU 使用率和 50% 的記憶體使用率。如果任務閒置並使用 128 個 CPU 單位和 128 MiB 記憶體，則服務會回報 25% 的 CPU 使用率和 12.5% 的記憶體使用率。

> **Note**
>
> 在此範例中，只有當 CPU 單位在容器層級定義時，CPU 使用率會超過 100%。若您在任務層級定義 CPU 單位，使用率將不會高於定義的任務層級限制。

服務層級 EBS 檔案系統使用率

服務層級 EBS 檔案系統使用率是以屬於該服務的工作所使用的 EBS 檔案系統總量來衡量，除以為屬於該服務之所有工作配置的 EBS 檔案系統儲存總量。

```
                                    (Total GB of EBS filesystem used by tasks in the
 service x 100)
Service EBS filesystem utilization =
 ------------------------------------------------------------------
                                    (Total GB of EBS filesystem allocated to tasks
 in the service)
```

服務 **RUNNING** 任務計數

您可以使用 CloudWatch 指標來檢視服務中處於RUNNING狀態的工作數目。例如，您可以設定 CloudWatch 此指標的警示，以在服務中執行中的工作數目低於指定值時提醒您。

Amazon ECS CloudWatch 容器洞見中的服務**RUNNING**任務計數

使用 Amazon ECS CloudWatch 容器洞見時，每個叢集和每個服務都可以使用「執行中任務數目」(RunningTaskCount) 指標。您可以針對透過選擇使用containerInsights帳戶設定而建立的所有新叢集使用 Container Insights、在叢集建立期間開啟叢集設定，或使用 UpdateClusterSettings

API 在現有叢集上使用。容器深入解析收 CloudWatch 集的指標會以自訂指標的形式收費。如需有關 CloudWatch 定價的詳細資訊，請參閱CloudWatch定價。

若要檢視此指標，請參閱 Amazon ECS 容器洞見指標 (位於 Amazon CloudWatch 使用者指南)。

使用以下方式自動回應 Amazon ECS 錯誤 EventBridge

使用 Amazon EventBridge，您可以自動化 AWS 服務並自動回應系統事件，例如應用程式可用性問題或資源變更。來自 AWS 服務的事件會以近乎即時 EventBridge 的方式傳送到。您可編寫簡單的規則，來指示您在意的事件，以及當事件符合規則時所要自動執行的動作。可以自動設定的動作如下：

- 將事件新增至 CloudWatch 記錄檔中的記錄群組
- 調用函數 AWS Lambda
- 調用 Amazon EC2 執行命令
- 將事件轉傳至 Amazon Kinesis Data Streams
- 啟動 AWS Step Functions 狀態機
- 通知 Amazon SNS 主題或 Amazon Simple Queue Service (Amazon SQS) 佇列

如需詳細資訊，請參閱 Amazon EventBridge 使用者指南 EventBridge中的 Amazon 入門。

您可以將 Amazon ECS 事件用於 EventBridge 接收有關 Amazon ECS 叢集目前狀態的近乎即時的通知。如果您的任務使用的 EC2 啟動類型，則可以看到容器執行個體的狀態，以及這些容器執行個體上執行之所有任務的目前狀態。如果您的工作使用 Fargate 啟動類型，您可以查看容器執行個體的狀態。

使用時 EventBridge，您可以在 Amazon ECS 之上建立自訂排程器，這些排程器負責跨叢集協調任務，並以近乎即時的方式監控叢集狀態。您可以消除排程和監控程式碼，這些程式碼會持續輪詢 Amazon ECS 服務是否有狀態變更，而是使用任何目標非同步處理 Amazon ECS 狀態變更。EventBridge目標可能包括 AWS Lambda Amazon 簡單佇列服務、亞馬遜簡單通知服務或 Amazon Kinesis 資料串流。

Amazon ECS 事件資料流可確保每個事件會至少交付一次。如果傳送重複的事件，則事件會提供足夠的資訊來識別重複項目。如需詳細資訊，請參閱 處理 Amazon ECS 事件。

由於事件的排序具有相對性，因此您可以清楚得知事件發生的時間 (相對於其他事件)。

主題

- Amazon ECS 事件

- 處理 Amazon ECS 事件

Amazon ECS 事件

Amazon ECS 會追蹤每個任務和服務的狀態。如果任務或服務的狀態發生變化，則會產生事件並將其傳送至 Amazon EventBridge。系統會將這些事件分類為任務狀態變更事件和服務動作事件。下列各節將更詳細地說明這些事件及其可能原因。

Amazon ECS 產生並將下列類型的事件傳送至 EventBridge：容器執行個體狀態變更事件、任務狀態變更事件、服務動作和服務部署狀態變更事件。

- 容器實例狀態變更
- 任務狀態變更
- Deployment state change (部署狀態變更)
- 服務動作

> **Note**
>
> Amazon ECS 日後可能會新增其他事件類型、來源和詳細資訊。如果您要在程式碼中反序列化事件 JSON 資料，請確定應用程式已準備好處理未知屬性，以避免在新增這些額外屬性時發生問題。

在某些情況下，會針對相同的活動產生多個事件。例如，在容器執行個體開始任務時，即會為新的任務產生任務狀態變更事件。系統會針對容器執行個體上可用資源 (例如，CPU、記憶體和可用連接埠) 中的變更，對帳戶產生容器執行個體狀態變更事件。同樣地，如果終止容器執行個體，則會針對容器執行個體、容器代理連線狀態以及在容器執行個體上執行的每個任務產生事件。

容器狀態變更和任務狀態變更事件包含兩個 `version` 欄位：一個位於事件本體，另一個位於事件的 `detail` 物件中。以下說明這兩個欄位之間的差異：

- 事件本體的 `version` 欄位在所有事件中皆設為 `0`。如需有關 EventBridge 參數的詳細資訊，請參閱 Amazon EventBridge 使用者指南中的事件和事件模式。
- 事件之 `detail` 物件中的 `version` 欄位說明相關資源的版本。每次資源變更狀態時，此版本都會遞增。因為事件可以傳送多次，所以此欄位可讓您識別重複的事件。重複事件在 `detail` 物件中具有相同版本。如果您要將 Amazon ECS 容器執行個體和任務狀態複寫在一起 EventBridge，您可以將

Amazon ECS API 報告的資源版本與資源中 EventBridge 報告的版本 (`detail`物件內部) 進行比較，以確認事件串流中的版本是最新的。

服務動作事件只包含本體中的 `version` 欄位。

如需有關如何整合 Amazon ECS 的其他資訊 EventBridge，請參閱整合 Amazon EventBridge 和 Amazon EC S。

Amazon ECS 容器執行個體狀態變更事件

下列案例會引發容器執行個體狀態變更事件：

您可以呼叫 `StartTask`、`RunTask` 或 `StopTask` API 操作 (可直接或使用 AWS Management Console 或軟體開發套件執行此動作)。

在容器執行個體上放置或停用任務會修改容器執行個體上的可用資源 (例如 CPU、記憶體和可用連接埠)。

Amazon ECS 服務排程器會啟動或停止任務。

在容器執行個體上放置或停用任務會修改容器執行個體上的可用資源 (例如 CPU、記憶體和可用連接埠)。

Amazon ECS 容器代理程式會呼叫 `SubmitTaskStateChange` API 操作取得所需狀態為 `RUNNING` 之任務的 `STOPPED` 狀態。

Amazon ECS 容器代理程式會監控容器執行個體上的任務狀態，並報告任何狀態變更。如果應為 `RUNNING` 的任務轉換為 `STOPPED`，則代理程式會釋出配給至已停止任務的資源 (例如 CPU、記憶體和可用連接埠)。

您可以直接或使用或 SDK，透過 `DeregisterContainerInstance` API 作業取消註冊容器執行個體。 AWS Management Console

取消登錄容器代理程式會變更容器執行個體的狀態以及 Amazon ECS 容器代理程式的連線狀態。

在 EC2 執行個體停止時，任務即停止。

當您停止容器執行個體時，在其上執行的任務會轉換為 `STOPPED` 狀態。

Amazon ECS 容器代理程式第一次登錄容器執行個體。

Amazon ECS 容器代理程式第一次登錄容器執行個體時 (在啟動時，或手動初次執行時)，這會建立執行個體的狀態變更事件。

Amazon ECS 容器代理會與 Amazon ECS 連線或中斷連線。

Amazon ECS 容器代理程式與 Amazon ECS 後端連線或中斷連線時，會變更容器執行個體的 `agentConnected` 狀態。

> **Note**
>
> Amazon ECS 容器代理程式在正常操作的過程中，每小時會中斷連線並重新連線數次，因此應預期會發生代理程式連線事件。這些事件並不表示容器代理程式或容器執行個體發生問題。

升級執行個體上的 Amazon ECS 容器代理程式。

容器執行個體詳細資訊包含容器代理版本的物件。如果您升級代理，則此版本資訊會變更並產生事件。

Example 容器執行個體狀態變更事件

容器執行個體狀態變更事件以下列格式交付。以下`detail`部分類似於 Amazon 彈性容器服務 API 參考中從DescribeContainer執行個體 API 作業傳回的ContainerInstance物件。如需有關 EventBridge參數的詳細資訊，請參閱 Amazon EventBridge 使用者指南中的事件和事件模式。

```
{
  "version": "0",
  "id": "8952ba83-7be2-4ab5-9c32-6687532d15a2",
  "detail-type": "ECS Container Instance State Change",
  "source": "aws.ecs",
  "account": "111122223333",
  "time": "2016-12-06T16:41:06Z",
  "region": "us-east-1",
  "resources": [
    "arn:aws:ecs:us-east-1:111122223333:container-instance/
b54a2a04-046f-4331-9d74-3f6d7f6ca315"
  ],
  "detail": {
    "agentConnected": true,
    "attributes": [
      {
        "name": "com.amazonaws.ecs.capability.logging-driver.syslog"
      },
```

```
      {
        "name": "com.amazonaws.ecs.capability.task-iam-role-network-host"
      },
      {
        "name": "com.amazonaws.ecs.capability.logging-driver.awslogs"
      },
      {
        "name": "com.amazonaws.ecs.capability.logging-driver.json-file"
      },
      {
        "name": "com.amazonaws.ecs.capability.docker-remote-api.1.17"
      },
      {
        "name": "com.amazonaws.ecs.capability.privileged-container"
      },
      {
        "name": "com.amazonaws.ecs.capability.docker-remote-api.1.18"
      },
      {
        "name": "com.amazonaws.ecs.capability.docker-remote-api.1.19"
      },
      {
        "name": "com.amazonaws.ecs.capability.ecr-auth"
      },
      {
        "name": "com.amazonaws.ecs.capability.docker-remote-api.1.20"
      },
      {
        "name": "com.amazonaws.ecs.capability.docker-remote-api.1.21"
      },
      {
        "name": "com.amazonaws.ecs.capability.docker-remote-api.1.22"
      },
      {
        "name": "com.amazonaws.ecs.capability.docker-remote-api.1.23"
      },
      {
        "name": "com.amazonaws.ecs.capability.task-iam-role"
      }
    ],
    "clusterArn": "arn:aws:ecs:us-east-1:111122223333:cluster/default",
    "containerInstanceArn": "arn:aws:ecs:us-east-1:111122223333:container-instance/
b54a2a04-046f-4331-9d74-3f6d7f6ca315",
    "ec2InstanceId": "i-f3a8506b",
```

```
    "registeredResources": [
      {
        "name": "CPU",
        "type": "INTEGER",
        "integerValue": 2048
      },
      {
        "name": "MEMORY",
        "type": "INTEGER",
        "integerValue": 3767
      },
      {
        "name": "PORTS",
        "type": "STRINGSET",
        "stringSetValue": [
          "22",
          "2376",
          "2375",
          "51678",
          "51679"
        ]
      },
      {
        "name": "PORTS_UDP",
        "type": "STRINGSET",
        "stringSetValue": []
      }
    ],
    "remainingResources": [
      {
        "name": "CPU",
        "type": "INTEGER",
        "integerValue": 1988
      },
      {
        "name": "MEMORY",
        "type": "INTEGER",
        "integerValue": 767
      },
      {
        "name": "PORTS",
        "type": "STRINGSET",
        "stringSetValue": [
          "22",
```

```
          "2376",
          "2375",
          "51678",
          "51679"
        ]
      },
      {
        "name": "PORTS_UDP",
        "type": "STRINGSET",
        "stringSetValue": []
      }
    ],
    "status": "ACTIVE",
    "version": 14801,
    "versionInfo": {
      "agentHash": "aebcbca",
      "agentVersion": "1.13.0",
      "dockerVersion": "DockerVersion: 1.11.2"
    },
    "updatedAt": "2016-12-06T16:41:06.991Z"
  }
}
```

Amazon ECS 任務狀態變更事件

下列案例會引發任務狀態變更事件：

您可以呼叫 StartTask、RunTask 或 StopTask API 操作 (可直接或使用 AWS Management Console、 AWS CLI或軟體開發套件執行此動作)。

啟動或停止任務會建立新的任務資源，或修改現有任務資源的狀態。

Amazon ECS 服務排程器會啟動或停止任務。

啟動或停止任務會建立新的任務資源，或修改現有任務資源的狀態。

Amazon ECS 容器代理程序會呼叫 SubmitTaskStateChange API 操作。

對於 Amazon EC2 啟動類型，Amazon ECS 容器代理程式會監控容器執行個體上的任務狀態。Amazon ECS 容器代理程式會報告任何狀態變更。狀態變更可能包括從 PENDING 到 RUNNING 或從 RUNNING 到 STOPPED 的變更。

您可以直接或使用或 SDK 強制使用 DeregisterContainerInstance API 作業和force旗標來取消註冊基礎容器執行個體。 AWS Management Console

取消登錄容器代理程式會變更容器執行個體的狀態以及 Amazon ECS 容器代理程式的連線狀態。如果任務是在容器執行個體上執行，則必須將 force 旗標設定為允許取消登錄。這會停止執行個體上的所有任務。

停止或終止基礎容器執行個體。

當您停止或終止容器執行個體時，在其上執行的任務會轉換為 STOPPED 狀態。

任務中的容器變更狀態。

Amazon ECS 容器代理程序會監控任務內的容器狀態。例如，如果任務內的執行容器停止，則此容器狀態變更會產生事件。

使用 Fargate Spot 容量提供者的任務會收到終止通知。

當任務使用 FARGATE_SPOT 容量提供者且因 Spot 中斷而停止時，會產生任務狀態變更事件。

Example 任務狀態變更事件

任務狀態變更事件以下列格式交付。以下detail部分類似於 Amazon 彈性容器服務 API 參考中從DescribeTasksAPI 作業傳回的任務物件。如果您的容器使用 Amazon ECR, 託管的映像，則會傳回 imageDigest 欄位。

> **Note**
>
> createdAt、connectivityAt、pullStartedAt、startedAt、pullStoppedAt 和 updatedAt 欄位的值是 DescribeTasks 動作之回應中的 UNIX 時間戳記，而在任務狀態變更事件中，這些值是 ISO 字串時間戳記。

如需有關 CloudWatch 事件參數的詳細資訊，請參閱 Amazon EventBridge 使用者指南中的事件和事件模式。

如需有關如何設定 Amazon EventBridge 事件規則，該規則僅擷取任務因其中一個基本容器已終止而停止執行的任務事件的資訊，請參閱 發送 Amazon 簡單通知服務提醒 Amazon ECS 任務停止事件

```
{
    "version": "0",
    "id": "3317b2af-7005-947d-b652-f55e762e571a",
    "detail-type": "ECS Task State Change",
```

```
    "source": "aws.ecs",
    "account": "111122223333",
    "time": "2020-01-23T17:57:58Z",
    "region": "us-west-2",
    "resources": [
        "arn:aws:ecs:us-west-2:111122223333:task/FargateCluster/
c13b4cb40f1f4fe4a2971f76ae5a47ad"
    ],
    "detail": {
        "attachments": [
            {
                "id": "1789bcae-ddfb-4d10-8ebe-8ac87ddba5b8",
                "type": "eni",
                "status": "ATTACHED",
                "details": [
                    {
                        "name": "subnetId",
                        "value": "subnet-abcd1234"
                    },
                    {
                        "name": "networkInterfaceId",
                        "value": "eni-abcd1234"
                    },
                    {
                        "name": "macAddress",
                        "value": "0a:98:eb:a7:29:ba"
                    },
                    {
                        "name": "privateIPv4Address",
                        "value": "10.0.0.139"
                    }
                ]
            }
        ],
        "availabilityZone": "us-west-2c",
        "clusterArn": "arn:aws:ecs:us-west-2:111122223333:cluster/FargateCluster",
        "containers": [
            {
                "containerArn": "arn:aws:ecs:us-west-2:111122223333:container/
cf159fd6-3e3f-4a9e-84f9-66cbe726af01",
                "lastStatus": "RUNNING",
                "name": "FargateApp",
                "image": "111122223333.dkr.ecr.us-west-2.amazonaws.com/hello-
repository:latest",
```

```
                "imageDigest":
 "sha256:74b2c688c700ec95a93e478cdb959737c148df3fbf5ea706abe0318726e885e6",
                "runtimeId":
 "ad64cbc71c7fb31c55507ec24c9f77947132b03d48d9961115cf24f3b7307e1e",
                "taskArn": "arn:aws:ecs:us-west-2:111122223333:task/FargateCluster/
c13b4cb40f1f4fe4a2971f76ae5a47ad",
                "networkInterfaces": [
                    {
                        "attachmentId": "1789bcae-ddfb-4d10-8ebe-8ac87ddba5b8",
                        "privateIpv4Address": "10.0.0.139"
                    }
                ],
                "cpu": "0"
            }
        ],
        "createdAt": "2020-01-23T17:57:34.402Z",
        "launchType": "FARGATE",
        "cpu": "256",
        "memory": "512",
        "desiredStatus": "RUNNING",
        "group": "family:sample-fargate",
        "lastStatus": "RUNNING",
        "overrides": {
            "containerOverrides": [
                {
                    "name": "FargateApp"
                }
            ]
        },
        "connectivity": "CONNECTED",
        "connectivityAt": "2020-01-23T17:57:38.453Z",
        "pullStartedAt": "2020-01-23T17:57:52.103Z",
        "startedAt": "2020-01-23T17:57:58.103Z",
        "pullStoppedAt": "2020-01-23T17:57:55.103Z",
        "updatedAt": "2020-01-23T17:57:58.103Z",
        "taskArn": "arn:aws:ecs:us-west-2:111122223333:task/FargateCluster/
c13b4cb40f1f4fe4a2971f76ae5a47ad",
        "taskDefinitionArn": "arn:aws:ecs:us-west-2:111122223333:task-definition/
sample-fargate:1",
        "version": 4,
        "platformVersion": "1.3.0"
    }
}
```

Amazon ECS 服務動作事件

Amazon ECS 會傳送具有 ECS Service Action (ECS 服務動作) 詳細類型的服務動作事件。與容器執行個體和任務狀態變更事件不同，服務動作事件不會在 `details` 回應欄位中包含版本號碼。以下是用來建立 Amazon ECS 服務動作事件 EventBridge 規則的事件模式。如需詳細資訊，請參閱 Amazon EventBridge 使用者指南中的建立 EventBridge規則。

```
{
    "source": [
        "aws.ecs"
    ],
    "detail-type": [
        "ECS Service Action"
    ]
}
```

Amazon ECS 會傳送具有 `INFO`、`WARN` 和 `ERROR` 事件類型的事件。以下是服務動作事件。

具有 **`INFO`** 事件類型的服務動作事件

`SERVICE_STEADY_STATE`

服務運作狀況良好且有所需的任務數量，因而達到穩定狀態。服務排程器會定期報告狀態，因此您可能會多次收到此訊息。

`TASKSET_STEADY_STATE`

任務集運作狀況良好且有所需的任務數量，因而達到穩定的狀態。

`CAPACITY_PROVIDER_STEADY_STATE`

與服務相關聯的容量提供者會達到穩定狀態。

`SERVICE_DESIRED_COUNT_UPDATED`

服務排程器更新適用於服務或任務集的計算所需計數時。當使用者手動更新所需的計數時，不會傳送此事件。

具有 **`WARN`** 事件類型的服務動作事件

`SERVICE_TASK_START_IMPAIRED`

此服務無法成功持續啟動任務。

`SERVICE_DISCOVERY_INSTANCE_UNHEALTHY`

使用服務探索的服務包含運作狀況不良的任務。服務排程器會偵測到服務登錄中的任務運作狀況不良。

具有 **ERROR** 事件類型的服務動作事件

`SERVICE_DAEMON_PLACEMENT_CONSTRAINT_VIOLATED`

使用 DAEMON 服務排程器策略的服務中的任務不再符合此服務的放置限制策略。

`ECS_OPERATION_THROTTLED`

由於 Amazon ECS API 節流限制，服務排程器已經受到節流。

`SERVICE_DISCOVERY_OPERATION_THROTTLED`

由於 AWS Cloud Map API 節流限制，服務調度程序已被限制。這可能會在設定為使用服務探索的服務上發生。

`SERVICE_TASK_PLACEMENT_FAILURE`

服務排程器無法放置任務。原因的描述位於 `reason` 欄位中。

產生此服務事件的常見原因是因為叢集缺乏放置任務的資源。例如，可用的容器執行個體沒有足夠的 CPU 或記憶體容量，或沒有可用的容器執行個體。另一個常見原因是，若容器執行個體上的 Amazon ECS 容器代理程式連線中斷，會造成排程器無法放置任務。

`SERVICE_TASK_CONFIGURATION_FAILURE`

由於組態錯誤，服務排程器無法放置任務。原因的描述位於 `reason` 欄位中。

產生此服務事件的常見原因是因為標籤正在套用至服務，但使用者或角色尚未在區域中選擇使用新的 Amazon Resource Name (ARN) 格式。如需詳細資訊，請參閱 Amazon Resource Name (ARN) 和 ID。另一個常見的原因是 Amazon ECS 無法擔任所提供的任務 IAM 角色。

Example 服務穩定狀態事件

任務穩定狀態事件以下列格式交付。如需有關 EventBridge 參數的詳細資訊，請參閱 Amazon EventBridge 使用者指南中的事件和事件模式。

```
{
    "version": "0",
```

```
    "id": "af3c496d-f4a8-65d1-70f4-a69d52e9b584",
    "detail-type": "ECS Service Action",
    "source": "aws.ecs",
    "account": "111122223333",
    "time": "2019-11-19T19:27:22Z",
    "region": "us-west-2",
    "resources": [
        "arn:aws:ecs:us-west-2:111122223333:service/default/servicetest"
    ],
    "detail": {
        "eventType": "INFO",
        "eventName": "SERVICE_STEADY_STATE",
        "clusterArn": "arn:aws:ecs:us-west-2:111122223333:cluster/default",
        "createdAt": "2019-11-19T19:27:22.695Z"
    }
}
```

Example 容量提供者穩定狀態事件

容量提供者穩定狀態事件以下列格式交付。

```
{
    "version": "0",
    "id": "b9baa007-2f33-0eb1-5760-0d02a572d81f",
    "detail-type": "ECS Service Action",
    "source": "aws.ecs",
    "account": "111122223333",
    "time": "2019-11-19T19:37:00Z",
    "region": "us-west-2",
    "resources": [
        "arn:aws:ecs:us-west-2:111122223333:service/default/servicetest"
    ],
    "detail": {
        "eventType": "INFO",
        "eventName": "CAPACITY_PROVIDER_STEADY_STATE",
        "clusterArn": "arn:aws:ecs:us-west-2:111122223333:cluster/default",
        "capacityProviderArns": [
            "arn:aws:ecs:us-west-2:111122223333:capacity-provider/ASG-tutorial-
capacity-provider"
        ],
        "createdAt": "2019-11-19T19:37:00.807Z"
    }
}
```

Example 服務任務啟動受損事件

服務任務啟動受損事件以下列格式交付。

```
{
    "version": "0",
    "id": "57c9506e-9d21-294c-d2fe-e8738da7e67d",
    "detail-type": "ECS Service Action",
    "source": "aws.ecs",
    "account": "111122223333",
    "time": "2019-11-19T19:55:38Z",
    "region": "us-west-2",
    "resources": [
        "arn:aws:ecs:us-west-2:111122223333:service/default/servicetest"
    ],
    "detail": {
        "eventType": "WARN",
        "eventName": "SERVICE_TASK_START_IMPAIRED",
        "clusterArn": "arn:aws:ecs:us-west-2:111122223333:cluster/default",
        "createdAt": "2019-11-19T19:55:38.725Z"
    }
}
```

Example 服務任務放置失敗事件

服務任務放置失敗事件以下列格式交付。如需有關 EventBridge 參數的詳細資訊，請參閱 Amazon EventBridge 使用者指南中的事件和事件模式。

在下列範例中，此任務嘗試使用 FARGATE_SPOT 容量提供者，但服務排程器無法取得任何 Fargate Spot 容量。

```
{
    "version": "0",
    "id": "ddca6449-b258-46c0-8653-e0e3a6d0468b",
    "detail-type": "ECS Service Action",
    "source": "aws.ecs",
    "account": "111122223333",
    "time": "2019-11-19T19:55:38Z",
    "region": "us-west-2",
    "resources": [
        "arn:aws:ecs:us-west-2:111122223333:service/default/servicetest"
    ],
```

```
    "detail": {
        "eventType": "ERROR",
        "eventName": "SERVICE_TASK_PLACEMENT_FAILURE",
        "clusterArn": "arn:aws:ecs:us-west-2:111122223333:cluster/default",
        "capacityProviderArns": [
            "arn:aws:ecs:us-west-2:111122223333:capacity-provider/FARGATE_SPOT"
        ],
        "reason": "RESOURCE:FARGATE",
        "createdAt": "2019-11-06T19:09:33.087Z"
    }
}
```

在下列 EC2 啟動類型範例中，嘗試在容器容器執行個體 2dd1b186f39845a584488d2ef155c131 上啟動任務，但由於 CPU 不足，服務排程器無法放置任務。

```
{
  "version": "0",
  "id": "ddca6449-b258-46c0-8653-e0e3a6d0468b",
  "detail-type": "ECS Service Action",
  "source": "aws.ecs",
  "account": "111122223333",
  "time": "2019-11-19T19:55:38Z",
  "region": "us-west-2",
  "resources": [
    "arn:aws:ecs:us-west-2:111122223333:service/default/servicetest"
  ],
  "detail": {
    "eventType": "ERROR",
    "eventName": "SERVICE_TASK_PLACEMENT_FAILURE",
    "clusterArn": "arn:aws:ecs:us-west-2:111122223333:cluster/default",
    "containerInstanceArns": [
    "arn:aws:ecs:us-west-2:111122223333:container-instance/
default/2dd1b186f39845a584488d2ef155c131"
    ],
    "reason": "RESOURCE:CPU",
    "createdAt": "2019-11-06T19:09:33.087Z"
  }
}
```

Amazon ECS 服務部署狀態變更事件

Amazon ECS 會傳送具有 ECS Deployment State Change (ECS 部署狀態變更) 詳細類型的服務部署變更狀態事件。以下是用來建立 Amazon ECS 服務部署狀態變更事件 EventBridge 規則的事件模式。如需詳細資訊，請參閱 Amazon EventBridge 使用者指南中的建立 EventBridge規則。

```
{
    "source": [
        "aws.ecs"
    ],
    "detail-type": [
        "ECS Deployment State Change"
    ]
}
```

Amazon ECS 會傳送具有 INFO 和 ERROR 事件類型的事件。以下是服務部署狀態變更事件。

`SERVICE_DEPLOYMENT_IN_PROGRESS`

服務部署正在進行。此事件會被傳送用於初始部署和回復部署。

`SERVICE_DEPLOYMENT_COMPLETED`

服務部署完成。一旦服務在部署後達到穩定狀態，此事件將被傳送。

`SERVICE_DEPLOYMENT_FAILED`

服務部署失敗。此事件會被傳送用於開啟部署斷路器邏輯的服務。

Example 服務部署正在進行事件

服務部署正在進行事件會在初始和回復部署均已開始時交付。兩者之間的區別位於 `reason` 欄位。如需有關 EventBridge 參數的詳細資訊，請參閱 Amazon EventBridge 使用者指南中的事件和事件模式。

下列顯示初始部署開始的輸出範例。

```
{
   "version": "0",
   "id": "ddca6449-b258-46c0-8653-e0e3a6EXAMPLE",
   "detail-type": "ECS Deployment State Change",
   "source": "aws.ecs",
   "account": "111122223333",
```

```
    "time": "2020-05-23T12:31:14Z",
    "region": "us-west-2",
    "resources": [
         "arn:aws:ecs:us-west-2:111122223333:service/default/servicetest"
    ],
    "detail": {
         "eventType": "INFO",
         "eventName": "SERVICE_DEPLOYMENT_IN_PROGRESS",
         "deploymentId": "ecs-svc/123",
         "updatedAt": "2020-05-23T11:11:11Z",
         "reason": "ECS deployment deploymentId in progress."
    }
}
```

下列顯示回復部署開始的輸出範例。reason 欄位提供服務轉返至的部署 ID。

```
{
    "version": "0",
    "id": "ddca6449-b258-46c0-8653-e0e3aEXAMPLE",
    "detail-type": "ECS Deployment State Change",
    "source": "aws.ecs",
    "account": "111122223333",
    "time": "2020-05-23T12:31:14Z",
    "region": "us-west-2",
    "resources": [
         "arn:aws:ecs:us-west-2:111122223333:service/default/servicetest"
    ],
    "detail": {
         "eventType": "INFO",
         "eventName": "SERVICE_DEPLOYMENT_IN_PROGRESS",
         "deploymentId": "ecs-svc/123",
         "updatedAt": "2020-05-23T11:11:11Z",
         "reason": "ECS deployment circuit breaker: rolling back to
 deploymentId deploymentID."
    }
}
```

Example 服務部署完成事件

任務部署完成狀態事件以下列格式交付。如需詳細資訊，請參閱 透過取代任務部署 Amazon ECS 服務。

```
{
```

```
    "version": "0",
    "id": "ddca6449-b258-46c0-8653-e0e3aEXAMPLE",
    "detail-type": "ECS Deployment State Change",
    "source": "aws.ecs",
    "account": "111122223333",
    "time": "2020-05-23T12:31:14Z",
    "region": "us-west-2",
    "resources": [
         "arn:aws:ecs:us-west-2:111122223333:service/default/servicetest"
    ],
    "detail": {
         "eventType": "INFO",
         "eventName": "SERVICE_DEPLOYMENT_COMPLETED",
         "deploymentId": "ecs-svc/123",
         "updatedAt": "2020-05-23T11:11:11Z",
         "reason": "ECS deployment deploymentID completed."
    }
}
```

Example 服務部署失敗事件

任務部署失敗狀態事件以下列格式交付。服務部署失敗狀態事件只會被傳送用於開啟部署斷路器邏輯的服務。如需詳細資訊，請參閱 透過取代任務部署 Amazon ECS 服務。

```
{
    "version": "0",
    "id": "ddca6449-b258-46c0-8653-e0e3aEXAMPLE",
    "detail-type": "ECS Deployment State Change",
    "source": "aws.ecs",
    "account": "111122223333",
    "time": "2020-05-23T12:31:14Z",
    "region": "us-west-2",
    "resources": [
         "arn:aws:ecs:us-west-2:111122223333:service/default/servicetest"
    ],
    "detail": {
         "eventType": "ERROR",
         "eventName": "SERVICE_DEPLOYMENT_FAILED",
         "deploymentId": "ecs-svc/123",
         "updatedAt": "2020-05-23T11:11:11Z",
         "reason": "ECS deployment circuit breaker: task failed to start."
    }
}
```

處理 Amazon ECS 事件

Amazon ECS 會傳送事件至少一次。這意味著您可能會收到給定事件的多個副本。此外，可能不會依事件發生順序將事件交付至事件接聽程式。

為了正確排序事件，每個事件的 `detail` 區段都會包含 `version` 屬性。每次資源變更狀態時，此 `version` 都會遞增。重複事件在 `detail` 物件中具有相同的 `version`。如果您要將 Amazon ECS 容器執行個體和任務狀態複寫在一起 EventBridge，則可以將 Amazon ECS API 報告的資源版本與資源中`version` EventBridge 報告的版本進行比較，以確認事件串流中的版本是最新的。較高版本屬性編號的事件應該視為比較低版本編號的事件晚發生。

範例：處理 AWS Lambda 函式中的事件

下列範例顯示使用 Python 3.9 撰寫的 Lambda 函數，以同時擷取任務和容器執行個體狀態變更事件，並將其儲存至兩個 Amazon DynamoDB 資料表中的其中一個：

- ECS CtrInstance 狀態 — 儲存容器執行個體的最新狀態。資料表 ID 是容器執行個體的 `containerInstanceArn` 值。
- ECS TaskState — 儲存任務的最新狀態。資料表 ID 是任務的 `taskArn` 值。

```
import json
import boto3

def lambda_handler(event, context):
    id_name = ""
    new_record = {}

    # For debugging so you can see raw event format.
    print('Here is the event:')
    print((json.dumps(event)))

    if event["source"] != "aws.ecs":
       raise ValueError("Function only supports input from events with a source type
 of: aws.ecs")

    # Switch on task/container events.
    table_name = ""
    if event["detail-type"] == "ECS Task State Change":
        table_name = "ECSTaskState"
        id_name = "taskArn"
        event_id = event["detail"]["taskArn"]
```

```
    elif event["detail-type"] == "ECS Container Instance State Change":
        table_name = "ECSCtrInstanceState"
        id_name =  "containerInstanceArn"
        event_id = event["detail"]["containerInstanceArn"]
    else:
        raise ValueError("detail-type for event is not a supported type. Exiting
without saving event.")

    new_record["cw_version"] = event["version"]
    new_record.update(event["detail"])

    # "status" is a reserved word in DDB, but it appears in containerPort
    # state change messages.
    if "status" in event:
        new_record["current_status"] = event["status"]
        new_record.pop("status")


    # Look first to see if you have received a newer version of an event ID.
    # If the version is OLDER than what you have on file, do not process it.
    # Otherwise, update the associated record with this latest information.
    print("Looking for recent event with same ID...")
    dynamodb = boto3.resource("dynamodb", region_name="us-east-1")
    table = dynamodb.Table(table_name)
    saved_event = table.get_item(
        Key={
            id_name : event_id
        }
    )
    if "Item" in saved_event:
        # Compare events and reconcile.
        print(("EXISTING EVENT DETECTED: Id " + event_id + " - reconciling"))
        if saved_event["Item"]["version"] < event["detail"]["version"]:
            print("Received event is a more recent version than the stored event -
updating")
            table.put_item(
                Item=new_record
            )
        else:
            print("Received event is an older version than the stored event -
ignoring")
    else:
        print(("Saving new event - ID " + event_id))
```

```
        table.put_item(
            Item=new_record
        )
```

下列 Fargate 範例顯示使用 Python 3.9 撰寫的 Lambda 函數，該函數可擷取任務狀態變更事件，並將它們儲存至下列 Amazon DynamoDB 表格：

```
import json
import boto3

def lambda_handler(event, context):
    id_name = ""
    new_record = {}

    # For debugging so you can see raw event format.
    print('Here is the event:')
    print((json.dumps(event)))

    if event["source"] != "aws.ecs":
       raise ValueError("Function only supports input from events with a source type
 of: aws.ecs")

    # Switch on task/container events.
    table_name = ""
    if event["detail-type"] == "ECS Task State Change":
        table_name = "ECSTaskState"
        id_name = "taskArn"
        event_id = event["detail"]["taskArn"]
    else:
        raise ValueError("detail-type for event is not a supported type. Exiting
 without saving event.")

    new_record["cw_version"] = event["version"]
    new_record.update(event["detail"])

    # "status" is a reserved word in DDB, but it appears in containerPort
    # state change messages.
    if "status" in event:
        new_record["current_status"] = event["status"]
        new_record.pop("status")


    # Look first to see if you have received a newer version of an event ID.
```

```
    # If the version is OLDER than what you have on file, do not process it.
    # Otherwise, update the associated record with this latest information.
    print("Looking for recent event with same ID...")
    dynamodb = boto3.resource("dynamodb", region_name="us-east-1")
    table = dynamodb.Table(table_name)
    saved_event = table.get_item(
        Key={
            id_name : event_id
        }
    )
    if "Item" in saved_event:
        # Compare events and reconcile.
        print(("EXISTING EVENT DETECTED: Id " + event_id + " - reconciling"))
        if saved_event["Item"]["version"] < event["detail"]["version"]:
            print("Received event is a more recent version than the stored event -
updating")
            table.put_item(
                Item=new_record
            )
        else:
            print("Received event is an older version than the stored event -
ignoring")
    else:
        print(("Saving new event - ID " + event_id))

        table.put_item(
            Item=new_record
        )
```

使用容器洞察來監控 Amazon ECS 容器

CloudWatch Container Insights 會從您的容器化應用程式和微服務收集、彙總和摘要指標和記錄。

Container Insights 將探索叢集中所有執行中的容器，並在效能堆疊的每一層收集效能資料。營運資料的收集形式為「效能日誌事件」。這些項目使用的是結構化的 JSON 結構描述，可大規模擷取並存放高基數資料。根據此資料，在叢集、服務和工作層級 CloudWatch 建立較高層級的彙總指標作為 CloudWatch 指標。指標包含 CPU、記憶體、磁碟和網路這類資源的使用率。指標可在 CloudWatch 自動儀表板中使用。如需可用指標的相關資訊，請參閱 Amazon ECS 容器洞察指標 (英文) 中的 Amazon ECS CloudWatch 使用者指南。

> **Important**
>
> 容器深入解析收 CloudWatch 集的指標會以自訂指標的形式收費。如需有關 CloudWatch 定價的詳細資訊，請參閱CloudWatch定價。Amazon ECS 還提供免費的監控指標。如需詳細資訊，請參閱 使用監控 Amazon ECS CloudWatch 。

考量事項

使用 CloudWatch 容器深入解析時，應考慮下列事項。

- CloudWatch 容器見解指標只會反映在指定時間範圍內執行中工作的資源。例如，如果您有一個包含一個服務的叢集，但該服務沒有處於某個RUNNING狀態的任務，則不會有指標傳送至 CloudWatch。如果您有兩個服務，其中一個有執行中的任務，而另一個沒有，則只會傳送有執行中任務之服務的指標。
- 網路指標適用於在 Fargate 上執行的所有任務，以及在 Amazon EC2 執行個體上執行的任務，這些任務使用 `bridge` 或者 `awsvpc` 網路模式。

您可以在 CloudWatch 容器洞見主控台中檢視 Amazon ECS 任務和服務生命週期事件。這有助於您在單一檢視中關聯容器指標、日誌和事件，以便更全面地了解相關操作。

您可以查看的事件是 Amazon ECS 發送到 Amazon EventBridge 的事件。如需詳細資訊，請參閱 Amazon ECS 事件。

您可以選擇設定叢集、任務或服務的效能指標。根據您選擇的資源，將會報告下列事件：

- 容器執行個體狀態變更事件
- 服務動作事件
- 任務狀態變更事件

設定 Amazon ECS 的 CloudWatch 容器洞察

您可以使用 Amazon ECS 主控台、API 和開發套件來 AWS CLI設定容器洞見。

使用下表決定新增容器深入解析時要採取的動作。

Amazon ECS 資源的標記支援

任務	主控台	AWS CLI	API 動作
變更所有使用者的預設值	修改 Amazon ECS 帳戶設置	放入帳戶設置默認	PutAccountSettingDefault
變更特定使用者的預設值	修改 Amazon ECS 帳戶設置	把帳戶設置	PutAccount設定
設定特定叢集的容器深入解析	為 Fargate 啟動類型建立 Amazon ECS 叢集 為 Amazon EC2 啟動類型建立亞馬遜 ECS 叢集 更新 Amazon ECS 叢集	create-cluster UpdateCluster	CreateCluster UpdateCluster

Important

對於內含任務或服務使用 EC2 啟動類型的叢集，您的容器執行個體必須執行版本 1.29.0 或更新版本的 Amazon ECS 代理程式。如需詳細資訊，請參閱 Amazon ECS Linux 容器執行個體管理。

檢視 Amazon ECS 生命週期事件的 CloudWatch 容器洞察所需的許可

您必須設定正確的權限，然後才能在 CloudWatch 容器深入解析主控台中設定和檢視事件。如需詳細資訊，請參閱 Amazon CloudWatch 使用者指南中的容器洞見中的 Amazon ECS 生命週期事件。如需的 IAM 政策的詳細資訊 CloudWatch，請參閱中AWS Identity and Access Management 的 CloudWatch。

設定 Container Insights 以檢視 Amazon ECS 生命週期事件所需的許可

設定生命週期事件時，任務角色需要下列權限：

- 事件：PutRule
- 事件：PutTargets
- 記錄檔:CreateLog群組

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "events:PutRule",
        "events:PutTargets",
        "logs:CreateLogGroup"
      ],
      "Resource": "*"
    }
  ]
}
```

在 Container Insights 中檢視 Amazon ECS 生命週期事件所需的許可

檢視生命週期事件需要以下許可: 將以下許可作為內嵌政策新增至任務執行角色。如需詳細資訊，請參閱新增和移除 IAM 政策。

- 事件：DescribeRule
- 事件：ListTargetsByRule
- 記錄檔:DescribeLog群組

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "events:DescribeRule",
        "events:ListTargetsByRule",
        "logs:DescribeLogGroups"
      ],
      "Resource": "*"
```

```
    }
  ]
}
```

使用容器運作狀態檢查判斷 Amazon ECS 任務運作狀態

建立工作定義時，您可以為您的容器設定健全狀況檢查。Health 狀態檢查是在容器本機上執行，並驗證應用程式健康狀態和可用性的命令。

Amazon ECS 容器代理程式只會監控並報告任務定義中指定的運作狀態檢查。Amazon ECS 不會監控嵌入在容器映像中但未在容器定義中指定的 Docker 運作狀態檢查。容器定義中指定的運作狀態檢查參數，會覆寫任何存在於容器影像中的 Docker 運作狀態檢查。

在工作定義中定義健全狀況檢查時，容器會在容器內執行健全狀況檢查程序，然後評估結束代碼以判斷應用程式健康狀況。

健康狀態檢查由下列參數組成：

- 命令 — 容器執行的命令，以判斷其狀態是否正常。此字串陣列的開頭可以是 CMD，如此能直接執行命令引數；或是 CMD-SHELL，藉以使用容器預設的 shell 來執行命令。
- [間隔] — 每次健全狀況檢查之間的時間 (秒)。
- 逾時 — 等待健全狀況檢查成功之後的時間 (以秒為單位)，再將其視為失敗。
- 重試 — 在容器被視為狀況不良之前，重試失敗的健康狀態檢查的次數。
- 開始期間 — 在失敗健全狀況檢查之前，提供容器啟動時間的選擇性寬限期計入重試次數上限。

如需有關如何在工作定義中指定健康狀態檢查的資訊，請參閱運作狀態檢查。

以下內容說明容器可能的健全狀況狀態值：

- HEALTHY - 容器運作狀態檢查已成功通過。
- UNHEALTHY - 容器運作狀態檢查失敗。
- UNKNOWN - 正在評估容器運作狀態檢查，沒有定義容器運作狀態檢查，或 Amazon ECS 沒有容器的運作狀態。

健康狀態檢查命令會在容器上執行。因此，您必須在容器映像中包含這些命令。

健康狀態檢查會透過容器的回送介面 (位於`localhost`或`127.0.0.1`) 連線至應用程式。的結束代碼`0`表示成功，而非零結束代碼則表示失敗。

使用容器健全狀況檢查時，請考量下列事項

- 容器運作狀態檢查需要 1.17.0 版或更新版本的 Amazon ECS 容器代理程式。
- 如果您使用的是 Linux 平台版本1.1.0或更高版本，或 Windows 平台版本或更高版本，則可支援 Fargate 工作的容器健康狀態檢查 1.1.0

Amazon ECS 如何判斷任務運作狀態

在任務定義中是必要且具有健康狀態檢查命令的容器，是唯一會考慮判斷工作健康狀態的容器。

系統會依序評估下列規則：

1. 如果一個重要容器的狀態為UNHEALTHY，則工作狀態為UNHEALTHY。
2. 如果他的狀態是一個基本容器UNKNOWN，那麼任務狀態是UNKNOWN。
3. 如果所有必要容器的狀態都是HEALTHY，則工作狀態為HEALTHY。

請考慮下列具有 2 個基本容器的工作健康狀態範例。

容器 1 健康	容器 2 健康	任務健康
UNHEALTHY	UNKNOWN	UNHEALTHY
UNHEALTHY	HEALTHY	UNHEALTHY
HEALTHY	UNKNOWN	UNKNOWN
HEALTHY	HEALTHY	HEALTHY

請考慮下列具有 3 個容器的工作健全狀況範例。

容器 1 健康	容器 2 健康	容器 3 健康	任務健康
UNHEALTHY	UNKNOWN	UNKNOWN	UNHEALTHY
UNHEALTHY	UNKNOWN	HEALTHY	UNHEALTHY
UNHEALTHY	HEALTHY	HEALTHY	UNHEALTHY

容器 1 健康	容器 2 健康	容器 3 健康	任務健康
HEALTHY	UNKNOWN	HEALTHY	UNKNOWN
HEALTHY	UNKNOWN	UNKNOWN	UNKNOWN
HEALTHY	HEALTHY	HEALTHY	HEALTHY

代理程式中斷連線對健全狀況檢查的影響

如果 Amazon ECS 容器代理程式與 Amazon ECS 服務中斷連線，這不會導致容器轉換為UNHEALTHY狀態。這是為了確保容器在代理程式重新啟動或暫時無法使用期間保持執行。運作狀態檢查狀態是 Amazon ECS 代理程式的「最後一次聽到」回應，因此，如果在中斷連線HEALTHY之前考慮容器，則該狀態將保留到代理程式重新連線並進行另一個運作狀態檢查為止。沒有對容器運作狀態檢查的狀態做出任何假設。

檢視 Amazon ECS 容器健康狀態

您可以在主控台中檢視容器健全狀況，並在DescribeTasks回應中使用 API。如需詳細資訊，請參閱 Amazon 彈性容器服務 API 參考DescribeTasks中的。

如果您對容器使用日誌記錄 (例如 Amazon Lo CloudWatch gs)，則可以設定運作狀態檢查命令，將容器運作狀態輸出轉寄至日誌。確保使用2&1來 catch stdout 和stderr信息。

```
"command": [
     "CMD-SHELL",
     "curl -f http://localhost/ >> /proc/1/fd/1 2>&1  || exit 1"
   ],
```

監控 Amazon ECS 容器執行個體健康

Amazon ECS 提供容器執行個體運作狀態監控。您可以快速判斷 Amazon ECS 是否偵測到任何可能阻礙您的容器執行個體執行容器的問題。Amazon ECS 對每個具有代理程式版本 1.57.0 或更新版本的正在執行的容器執行個體，執行自動檢查以找出問題。有關驗證容器執行個體代理程式版本的詳細資訊，請參閱 更新 Amazon ECS 容器代理程式。

您必須使用 AWS CLI 版本1.22.3或更新版本或 AWS CLI 版本2.3.6或更新版本。如需有關如何更新的資訊 AWS CLI，請參閱《AWS Command Line Interface 使用指南第 2 版》 AWS CLI中的〈安裝或更新最新版本〉。

狀態檢查會每分鐘執行兩次，並會傳回通過或失敗狀態。如果所有檢查都通過，執行個體的整體狀態即為 OK。若一項或多項檢查失敗，整體狀態便會是 IMPAIRED。狀態檢查內建於 Amazon ECS 容器代理程式中，因此無法關閉或刪除。您可以檢視這些狀態檢查的結果，以找出特定及可偵測的問題。如需詳細資訊，請參閱 the section called “運作狀態檢查”。

執行 DescribeContainerInstances API，並CONTAINER_INSTANCE_HEALTH選擇擷取容器執行個體健康狀況。

```
aws ecs describe-container-instances \
     --cluster cluster_name \
     --container-instances 47279cd2cadb41cbaef2dcEXAMPLE \
     --include CONTAINER_INSTANCE_HEALTH
```

下列為輸出中運作狀態物件的範例。

```
"healthStatus": {
 "overallStatus": "OK",
 "details": [{
  "type": "CONTAINER_RUNTIME",
  "status": "OK",
  "lastUpdated": "2021-11-10T03:30:26+00:00",
  "lastStatusChange": "2021-11-10T03:26:41+00:00"
 }]
}
```

相關主題

- 使用監控 Amazon ECS CloudWatch

使用應用程式追蹤資料識別 Amazon ECS 最佳化機會

Amazon ECS 與發行 AWS 版整合，可從您的 OpenTelemetry 應用程式收集追蹤資料。Amazon ECS 使用適用於 OpenTelemetry 邊車容器的 AWS 發行版，以收集追蹤資料並將其路由到。 AWS X-Ray如需詳細資訊，請參閱在 Amazon ECS 中為 OpenTelemetry 收集器設定發行 AWS 版。然後，您可以使用 AWS X-Ray 來識別錯誤和例外狀況、分析效能瓶頸和回應時間。

為了讓 AWS 發行版 OpenTelemetry 收集器發送跟踪數據 AWS X-Ray，您的應用程序必須配置為創建跟踪數據。如需詳細資訊，請參閱《AWS X-Ray 開發人員指南》中的檢測您的 AWS X-Ray應用程式。

AWS 發行版所需的 IAM 許可以 OpenTelemetry 與之整合 AWS X-Ray

與發行 AWS 版的 Amazon ECS 整合時，您 OpenTelemetry 必須建立任務角色，並在任務定義中指定角色。我們建議您配置 OpenTelemetry 側車的 AWS Distro，以將容器日誌路由到 CloudWatch 日誌。

> **Important**
>
> 如果您還使用 AWS Distro 進行集 OpenTelemetry成收集應用程序指標，請確保您的任務 IAM 角色也包含該集成所需的許可。如需詳細資訊，請參閱 使用應用程式指標關聯 Amazon ECS 應用程式效能。

建立下列原則，然後將其附加至工作執行角色。

若要使用 JSON 政策編輯器來建立政策

1. 登入 AWS Management Console 並開啟身分與存取權管理主控台，網址為 https://console.aws.amazon.com/iam/。
2. 在左側的導覽窗格中，選擇 Policies (政策)。

   如果這是您第一次選擇 Policies (政策)，將會顯示 Welcome to Managed Policies (歡迎使用受管政策) 頁面。選擇 Get Started (開始使用)。
3. 在頁面頂端，選擇 Create policy (建立政策)。
4. 在政策編輯器中，選擇 JSON 選項。
5. 輸入下列 JSON 政策文件：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "logs:PutLogEvents",
                "logs:CreateLogGroup",
                "logs:CreateLogStream",
                "logs:DescribeLogStreams",
```

```
                "logs:DescribeLogGroups",
                "logs:PutRetentionPolicy",
                "xray:PutTraceSegments",
                "xray:PutTelemetryRecords",
                "xray:GetSamplingRules",
                "xray:GetSamplingTargets",
                "xray:GetSamplingStatisticSummaries",
                "ssm:GetParameters"
            ],
            "Resource": "*"
        }
    ]
}
```

6. 選擇下一步。

> **Note**
>
> 您可以隨時切換視覺化與 JSON 編輯器選項。不過，如果您進行變更或在視覺化編輯器中選擇下一步，IAM 就可能會調整您的政策結構，以便針對視覺化編輯器進行最佳化。如需詳細資訊，請參閱 IAM 使用者指南中的調整政策結構。

7. 在檢視與建立頁面上，為您正在建立的政策輸入政策名稱與描述 (選用)。檢視此政策中定義的許可，來查看您的政策所授予的許可。
8. 選擇 Create policy (建立政策) 儲存您的新政策。

指定要 AWS X-Ray 整合到您的任務定 AWS 義中的 OpenTelemetry 附屬發行版

Amazon ECS 主控台使用「使用追蹤收集」選項簡化了為 OpenTelemetry 附屬容器建立發行 AWS 版的程序。如需詳細資訊，請參閱 使用主控台建立 Amazon ECS 任務定義。

如果您沒有使用 Amazon ECS 主控台，您可以將附屬容器的發 OpenTelemetry行 AWS 版新增到您的任務定義中。下面的任務定義代碼片段顯示了添加 AWS Distro 以進行集成的並 OpenTelemetry 行的容器定義。 AWS X-Ray

```
{
 "family": "otel-using-xray",
 "taskRoleArn": "arn:aws:iam::111122223333:role/AmazonECS_OpenTelemetryXrayRole",
 "executionRoleArn": "arn:aws:iam::111122223333:role/ecsTaskExecutionRole",
```

```
 "containerDefinitions": [{
   "name": "aws-otel-emitter",
   "image": "application-image",
   "logConfiguration": {
    "logDriver": "awslogs",
    "options": {
     "awslogs-create-group": "true",
     "awslogs-group": "/ecs/aws-otel-emitter",
     "awslogs-region": "us-east-1",
     "awslogs-stream-prefix": "ecs"
    }
   },
   "dependsOn": [{
    "containerName": "aws-otel-collector",
    "condition": "START"
   }]
  },
  {
   "name": "aws-otel-collector",
   "image": "public.ecr.aws/aws-observability/aws-otel-collector:v0.30.0",
   "essential": true,
   "command": [
    "--config=/etc/ecs/otel-instance-metrics-config.yaml"
   ],
   "logConfiguration": {
    "logDriver": "awslogs",
    "options": {
     "awslogs-create-group": "True",
     "awslogs-group": "/ecs/ecs-aws-otel-sidecar-collector",
     "awslogs-region": "us-east-1",
     "awslogs-stream-prefix": "ecs"
    }
   }
  }
 ],
 "networkMode": "awsvpc",
 "requiresCompatibilities": [
  "FARGATE"
 ],
 "cpu": "1024",
 "memory": "3072"
}
```

使用應用程式指標關聯 Amazon ECS 應用程式效能

Fargate 上的 Amazon ECS 支援從您在 Fargate 上執行的應用程式收集指標，並將其匯出到 Amazon CloudWatch 或 Amazon Prometheus 代管服務。

您可以使用收集的中繼資料，將應用程式效能資料與基礎架構資料建立關聯，從而縮短解決問題的平均時間。

Amazon ECS 使用適用於 OpenTelemetry 邊車容器的 AWS 發行版，以收集您的應用程式指標並將其路由到目的地。Amazon ECS 主控台體驗可簡化建立任務定義時新增此整合的程序。

主題

- 將應用程式指標匯出到 Amazon CloudWatch
- 將應用程式指標匯出至 Amazon Managed Service for Prometheus

將應用程式指標匯出到 Amazon CloudWatch

Fargate 上的 Amazon ECS 支援將您的自訂應用程式指標匯出到 Amazon CloudWatch 做為自訂指標。這是通過將 OpenTelemetry 附屬容器的 AWS Distro 添加到您的任務定義中來完成的。Amazon ECS 主控台透過在建立新任務定義時新增「使用指標收集」選項來簡化此程序。如需詳細資訊，請參閱 使用主控台建立 Amazon ECS 任務定義。

應用程式度量會匯出至具有 CloudWatch 記錄群組名稱的記錄檔，`/aws/ecs/application/metrics`而且可在`ECS/AWSOTel/Application`命名空間中檢視指標。您的應用程式必須使用 OpenTelemetry SDK 進行檢測。有關更多信息，請參閱發行版 OpenTelemetry中的AWS 發行版簡介以獲 OpenTelemetry 取文檔。 AWS

考量事項

在與發行 AWS 版的 Fargate 集成上使用 Amazon ECS 以將應用程序指標發送 OpenTelemetry 到 Amazon 時，應考慮以下事項。 CloudWatch

- 此整合只會將您的自訂應用程式指標傳送至 CloudWatch。如果需要任務層級指標，可以在 Amazon ECS 叢集組態中開啟 Container Insights。如需詳細資訊，請參閱 使用容器洞察來監控 Amazon ECS 容器。
- 對於在 Fargate 託管的 Amazon ECS 工作負載和 Amazon Amazon EC2 實例上託管的亞馬遜 ECS 工作負載支持 OpenTelemetry 整合的發行 AWS 版。目前不支援外部執行個體。

- CloudWatch 每個量度最多支援 30 個維度。預設情況下，Amazon ECS 預設為指標中包含 `TaskARN`、`ClusterARN`、`LaunchType`、`TaskDefinitionFamily`，以及 `TaskDefinitionRevision` 維度。其餘 25 個維度可由您的應用程式定義。如果已設定超過 30 個維度，則 CloudWatch 無法顯示它們。發生這種情況時，應用程式測量結果會顯示在`ECS/AWSOTel/Application` CloudWatch 測量結果命名空間中，但沒有任何維度 您可以檢測應用程式以增加其他維度。有關更多信息，請參閱發行版 OpenTelemetry中的將 CloudWatch AWS 指標與 AWS 發行版一起使用以獲 OpenTelemetry 取文檔。

與 Amazon OpenTelemetry 集成的 AWS 發行版所需的 IAM 許可 CloudWatch

與發行 AWS 版的 Amazon ECS 整合時，您 OpenTelemetry 必須建立任務 IAM 角色，並在任務定義中指定角色。我們建議還將 AWS Distro 配置為將容器日誌路由到 CloudWatch 日誌，這需要在任務定義中創建和指定任務執行 IAM 角色。 OpenTelemetry Amazon ECS 主控台會代表您處理任務執行 IAM 角色，但任務 IAM 角色必須手動建立並新增到您的任務定義中。如需有關任務執行 IAM 角色的詳細資訊，請參閱 Amazon ECS 任務執行 IAM 角色。

> **Important**
>
> 如果您還使用 AWS Distro 收集應用程序跟踪數據進行 OpenTelemetry 集成，請確保您的任務 IAM 角色還包含該集成所需的許可。如需詳細資訊，請參閱 使用應用程式追蹤資料識別 Amazon ECS 最佳化機會。
>
> 如果您的應用程式需要任何其他許可，則應將其新增到此政策中。每個任務定義只能指定一個任務 IAM 角色。例如，如果您使用儲存在 Systems Manager 中的自訂組態檔案，則應將 `ssm:GetParameters` 許可新增至此 IAM 政策。

建立彈性容器服務 (IAM 主控台) 的服務角色

1. 登入 AWS Management Console 並開啟身分與存取權管理主控台，網址為 https://console.aws.amazon.com/iam/。
2. 在 IAM 主控台的導覽窗格中，選擇角色，然後選擇建立角色。
3. 對於 Trusted entity type (信任的實體類型)，請選擇 AWS 服務。
4. 對於服務或使用案例，請選擇彈性容器服務，然後選擇彈性容器服務工作使用案例。
5. 選擇下一步。
6. 在「新增權限」區段中，搜尋 AWSDistroOpenTelemetryPolicyForXray，然後選取原則。
7. (選用) 設定許可界限。這是進階功能，可用於服務角色，而不是服務連結的角色。

a. 開啟 [設定權限界限] 區段，然後選擇 [使用權限界限] 控制最大角色權限。

IAM 在您的帳戶中包含受 AWS 管政策和客戶管理政策的清單。

b. 選取用於許可界限的政策。

8. 選擇下一步。
9. 輸入角色名稱或角色名稱尾碼，以協助您識別角色的用途。

> **Important**
>
> 命名角色時，請注意下列事項：
>
> - 角色名稱在您的內部必須是唯一的 AWS 帳戶，並且不能根據大小寫將其唯一。
>
> 例如，請勿建立同時命名為**PRODROLE**和的角色**prodrole**。當角色名稱用於策略中或作為 ARN 的一部分時，角色名稱會區分大小寫，但是當主控台中的客戶 (例如在登入程序期間) 顯示角色名稱時，角色名稱不區分大小寫。
> - 您無法在建立角色之後編輯該角色的名稱，因為其他實體可能會參照該角色。

10. (選擇性) 在說明中，輸入角色的說明。
11. (選擇性) 若要編輯角色的使用案例和權限，請在步驟 1：選取信任的實體或步驟 2：新增權限區段中，選擇編輯。
12. (選擇性) 若要協助識別、組織或搜尋角色，請將標籤新增為鍵值配對。如需有關在 IAM 中使用標籤的詳細資訊，請參閱《IAM 使用者指南》中的標記 IAM 資源。
13. 檢閱角色，然後選擇 Create role (建立角色)。

在任務定 AWS 義中指定 OpenTelemetry 側車的發行版

Amazon ECS 主控台使用「使用指標收集」選項簡化了建立 OpenTelemetry 附屬容器發行 AWS 版的體驗。如需詳細資訊，請參閱 使用主控台建立 Amazon ECS 任務定義。

如果您沒有使用 Amazon ECS 主控台，您可以手動將附屬容器的 AWS 發 OpenTelemetry 行版新增至您的任務定義。以下任務定義示例顯示了為 Amazon CloudWatch 集成添加用於 OpenTelemetry 側車的 AWS Distro 的容器定義。

```
{
 "family": "otel-using-cloudwatch",
 "taskRoleArn": "arn:aws:iam::111122223333:role/AmazonECS_OpenTelemetryCloudWatchRole",
```

```
"executionRoleArn": "arn:aws:iam::111122223333:role/ecsTaskExecutionRole",
"containerDefinitions": [
   {
  "name": "aws-otel-emitter",
  "image": "application-image",
  "logConfiguration": {
   "logDriver": "awslogs",
   "options": {
    "awslogs-create-group": "true",
    "awslogs-group": "/ecs/aws-otel-emitter",
    "awslogs-region": "us-east-1",
    "awslogs-stream-prefix": "ecs"
   }
  },
  "dependsOn": [{
   "containerName": "aws-otel-collector",
   "condition": "START"
  }]
 },
 {
  "name": "aws-otel-collector",
  "image": "public.ecr.aws/aws-observability/aws-otel-collector:v0.30.0",
  "essential": true,
  "command": [
   "--config=/etc/ecs/ecs-cloudwatch.yaml"
  ],
  "logConfiguration": {
   "logDriver": "awslogs",
   "options": {
    "awslogs-create-group": "True",
    "awslogs-group": "/ecs/ecs-aws-otel-sidecar-collector",
    "awslogs-region": "us-east-1",
    "awslogs-stream-prefix": "ecs"
   }
  }
 }
],
"networkMode": "awsvpc",
"requiresCompatibilities": [
 "FARGATE"
],
"cpu": "1024",
"memory": "3072"
```

```
}
```

將應用程式指標匯出至 Amazon Managed Service for Prometheus

Amazon ECS 支援將您的任務層級 CPU、記憶體、網路和儲存指標，以及自訂應用程式指標匯出至 Amazon Managed Service for Prometheus。這是通過將 OpenTelemetry 附屬容器的 AWS Distro 添加到您的任務定義中來完成的。Amazon ECS 主控台透過在建立新任務定義時新增「使用指標收集」選項來簡化此程序。如需詳細資訊，請參閱 使用主控台建立 Amazon ECS 任務定義。

這些指標將匯出至 Amazon Managed Service for Prometheus，並可以使用 Amazon Managed Grafana 儀表板查看。您的應用程式必須使用 Prometheus 程式庫或 SDK 進行檢測。 OpenTelemetry 有關使用 OpenTelemetry SDK 檢測應用程序的更多信息，請參閱發行版 OpenTelemetry中的發行AWS 版簡介以獲取文檔。 AWS OpenTelemetry

使用 Prometheus 程式庫時，您的應用程式必須公開 `/metrics` 端點，用於抓取指標資料。如需有關使用 Prometheus 程式庫檢測應用程式的詳細資訊，請參閱 Prometheus 文件中的 Prometheus 用戶端程式庫。

考量事項

在與 AWS Distro 整合的 Fargate 上使用 Amazon ECS 時，應考慮以下事項，以便將應用程式指標傳送 OpenTelemetry 到適用於 Prometheus 的 Amazon 託管服務。

- 對於在 Fargate 託管的 Amazon ECS 工作負載和 Amazon Amazon EC2 實例上託管的亞馬遜 ECS 工作負載支持 OpenTelemetry 整合的發行 AWS 版。目前不支援外部執行個體。
- 預設情況下，當匯出到適用於 Prometheus 的 Amazon 受管服務時， AWS Distro for 會 OpenTelemetry 包含應用程式指標的所有可用任務層級維度。您還可以檢測應用程式以增加其他維度。有關更多信息，請參閱發行版中的 Prometheus 適用於 Prometheus 的 Amazon 受管服務的 Prometheus 遠端寫入導出器入門以獲取文檔。 AWS OpenTelemetry

AWS 發行版所需的 IAM 許可，以便與 Prometheus 的 Amazon 託管服務 OpenTelemetry 整合

使用 OpenTelemetry附屬發行 AWS 版與適用於 Prometheus 的 Amazon ECS 管理服務整合時，您必須建立任務 IAM 角色，並在任務定義中指定角色。註冊任務定義之前，必須使用以下步驟手動建立此任務 IAM 角色。

我們建議還將 AWS Distro 配置為將容器日誌路由到 CloudWatch 日誌，這需要在任務定義中創建和指定任務執行 IAM 角色。 OpenTelemetry Amazon ECS 主控台會代表您處理任務執行 IAM 角色，但任

務 IAM 角色必須手動建立。如需有關為任務建立 IAM 角色的詳細資訊，請參閱 Amazon ECS 任務執行 IAM 角色。

> **Important**
>
> 如果您還使用 AWS Distro 收集應用程序跟踪數據進行 OpenTelemetry 集成，請確保您的任務 IAM 角色還包含該集成所需的許可。如需詳細資訊，請參閱 使用應用程式追蹤資料識別 Amazon ECS 最佳化機會。

建立彈性容器服務 (IAM 主控台) 的服務角色

1. 登入 AWS Management Console 並開啟身分與存取權管理主控台，網址為 https://console.aws.amazon.com/iam/。
2. 在 IAM 主控台的導覽窗格中，選擇角色，然後選擇建立角色。
3. 對於 Trusted entity type (信任的實體類型)，請選擇 AWS 服務。
4. 對於服務或使用案例，請選擇彈性容器服務，然後選擇彈性容器服務工作使用案例。
5. 選擇下一步。
6. 在「新增權限」區段中，搜尋「AmazonPrometheusRemoteWrite存取」，然後選取原則。
7. (選用) 設定許可界限。這是進階功能，可用於服務角色，而不是服務連結的角色。
   a. 開啟 [設定權限界限] 區段，然後選擇 [使用權限界限] 控制最大角色權限。

      IAM 在您的帳戶中包含受 AWS 管政策和客戶管理政策的清單。
   b. 選取用於許可界限的政策。
8. 選擇下一步。
9. 輸入角色名稱或角色名稱尾碼，以協助您識別角色的用途。

> **Important**
>
> 命名角色時，請注意下列事項：
>
> - 角色名稱在您的內部必須是唯一的 AWS 帳戶，並且不能根據大小寫將其唯一。
>
>   例如，請勿建立同時命名為**PRODROLE**和的角色**prodrole**。當角色名稱用於策略中或作為 ARN 的一部分時，角色名稱會區分大小寫，但是當主控台中的客戶 (例如在登入程序期間) 顯示角色名稱時，角色名稱不區分大小寫。

- 您無法在建立角色之後編輯該角色的名稱，因為其他實體可能會參照該角色。

10. (選擇性) 在說明中，輸入角色的說明。
11. (選擇性) 若要編輯角色的使用案例和權限，請在步驟 1：選取信任的實體或步驟 2：新增權限區段中，選擇編輯。
12. (選擇性) 若要協助識別、組織或搜尋角色，請將標籤新增為鍵值配對。如需有關在 IAM 中使用標籤的詳細資訊，請參閱《IAM 使用者指南》中的標記 IAM 資源。
13. 檢閱角色，然後選擇 Create role (建立角色)。

在任務定 AWS 義中指定 OpenTelemetry 側車的發行版

Amazon ECS 主控台使用「使用指標收集」選項簡化了建立 OpenTelemetry 附屬容器發行 AWS 版的體驗。如需詳細資訊，請參閱 使用主控台建立 Amazon ECS 任務定義。

如果您沒有使用 Amazon ECS 主控台，您可以手動將附屬容器的 AWS 發 OpenTelemetry 行版新增至您的任務定義。以下任務定義示例顯示了為 Prometheus 集成的 Amazon 受管服務添加適用於附屬發 OpenTelemetry 行 AWS 版的容器定義。

```
{
 "family": "otel-using-cloudwatch",
 "taskRoleArn": "arn:aws:iam::111122223333:role/AmazonECS_OpenTelemetryCloudWatchRole",
 "executionRoleArn": "arn:aws:iam::111122223333:role/ecsTaskExecutionRole",
 "containerDefinitions": [{
   "name": "aws-otel-emitter",
   "image": "application-image",
   "logConfiguration": {
    "logDriver": "awslogs",
    "options": {
     "awslogs-create-group": "true",
     "awslogs-group": "/ecs/aws-otel-emitter",
     "awslogs-region": "aws-region",
     "awslogs-stream-prefix": "ecs"
    }
   },
   "dependsOn": [{
    "containerName": "aws-otel-collector",
    "condition": "START"
   }]
  },
```

```
  {
   "name": "aws-otel-collector",
   "image": "public.ecr.aws/aws-observability/aws-otel-collector:v0.30.0",
   "essential": true,
   "command": [
    "--config=/etc/ecs/ecs-amp.yaml"
   ],
   "environment": [{
    "name": "AWS_PROMETHEUS_ENDPOINT",
    "value": "https://aps-workspaces.aws-region.amazonaws.com/workspaces/
ws-a1b2c3d4-5678-90ab-cdef-EXAMPLE11111/api/v1/remote_write"
   }],
   "logConfiguration": {
    "logDriver": "awslogs",
    "options": {
     "awslogs-create-group": "True",
     "awslogs-group": "/ecs/ecs-aws-otel-sidecar-collector",
     "awslogs-region": "aws-region",
     "awslogs-stream-prefix": "ecs"
    }
   }
  }
 ],
 "networkMode": "awsvpc",
 "requiresCompatibilities": [
  "FARGATE"
 ],
 "cpu": "1024",
 "memory": "3072"
}
```

使用記錄 Amazon ECS API 呼叫 AWS CloudTrail

Amazon ECS 與這項服務整合在一起 AWS CloudTrail，可提供 Amazon ECS 中使用者、角色或服務所採取的動作記錄的 AWS 服務。 CloudTrail 以事件形式擷取 Amazon ECS 的所有 API 呼叫，包括來自 Amazon ECS 主控台的呼叫，以及從程式碼呼叫到 Amazon ECS API 操作的呼叫。為了保護您的 VPC，VPC 端點策略拒絕的請求，但否則將被允許的請求不會記錄在中。 CloudTrail

如果您建立追蹤，您可以啟用持續交付 CloudTrail 事件到 Amazon S3 儲存貯體，包括 Amazon ECS 的事件。如果您未設定追蹤，您仍然可以在 [事件歷程記錄] 中檢視 CloudTrail 主控台中最近的事件。使用收集的資訊 CloudTrail，您可以判斷向 Amazon ECS 發出的請求、提出請求的 IP 地址、提出請求的人員、提出請求的時間以及其他詳細資訊。

如需詳細資訊，請參閱 AWS CloudTrail 使用者指南。

Amazon ECS 信息 CloudTrail

CloudTrail 當您建立 AWS 帳戶時，會在您的帳戶中開啟。當 Amazon ECS 中發生活動時，該活動會與事件歷史記錄中的其他 AWS 服務 CloudTrail 事件一起記錄在事件中。您可以在帳戶中查看，搜索和下載最近的事 AWS 件。如需詳細資訊，請參閱檢視具有事 CloudTrail 件記錄的事件。

如需 AWS 帳戶中持續的事件記錄 (包括 Amazon ECS 的事件)，請建立 CloudTrail 用於將日誌檔傳送到 Amazon S3 儲存貯體的追蹤。根據預設，當您在主控台建立權杖時，權杖會套用到所有區域。追蹤記錄來自 AWS 分區中所有區域的事件，並將日誌檔傳送到您指定的 Amazon S3 儲存貯體。此外，您還可以設定其他 AWS 服務，以進一步分析 CloudTrail 記錄中收集的事件資料並採取行動。如需詳細資訊，請參閱：

- 建立追蹤的概觀
- CloudTrail 支援的服務與整合
- 設定 Amazon SNS 通知 CloudTrail
- 從多個區域接收 CloudTrail 記錄檔並從多個帳戶接收 CloudTrail 記錄檔

所有 Amazon ECS 動作都會記錄下來， CloudTrail 並記錄在 Amazon 彈性容器服務 API 參考中。例如，呼叫`CreateService`、`RunTask`和`DeleteCluster`區段會在 CloudTrail 記錄檔中產生項目。

每一筆事件或日誌專案都會包含產生請求者的資訊。身分資訊可協助您判斷下列事項：

- 該請求是使用根使用者還是使用者憑證提出。
- 提出該請求時，是否使用了特定角色或聯合身分使用者的暫時安全憑證。
- 請求是否由其他 AWS 服務提出。

如需詳細資訊，請參閱CloudTrail 使 userIdentity 元素。

了解 Amazon ECS 日誌檔案項目

追蹤是一種組態，可讓事件以日誌檔的形式傳遞到您指定的 Amazon S3 儲存貯體。 CloudTrail 記錄檔包含一或多個記錄項目。事件代表來自任何來源的單一請求，包括有關請求的操作，動作的日期和時間，請求參數等信息。 CloudTrail 日誌文件不是公共 API 調用的有序堆棧跟踪，因此它們不會以任何特定順序顯示。

> **Note**
>
> 這些範例已格式化，以提升可讀性。在記 CloudTrail 錄檔中，所有項目和事件都會串連成一行。此外，這個範例中受限於單一 Amazon ECS 項目。在真實的 CloudTrail 記錄檔中，您會看到來自多個 AWS 服務的項目和事件。

下列範例顯示示範`CreateCluster`動作的 CloudTrail 記錄項目：

```
{
    "eventVersion": "1.04",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AIDACKCEVSQ6C2EXAMPLE:account_name",
        "arn": "arn:aws:sts::123456789012:user/Mary_Major",
        "accountId": "123456789012",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "sessionContext": {
            "attributes": {
                "mfaAuthenticated": "false",
                "creationDate": "2018-06-20T18:32:25Z"
            },
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AIDACKCEVSQ6C2EXAMPLE",
                "arn": "arn:aws:iam::123456789012:role/Admin",
                "accountId": "123456789012",
                "userName": "Mary_Major"
            }
        }
    },
    "eventTime": "2018-06-20T19:04:36Z",
    "eventSource": "ecs.amazonaws.com",
    "eventName": "CreateCluster",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "203.0.113.12",
    "userAgent": "console.amazonaws.com",
    "requestParameters": {
        "clusterName": "default"
    },
    "responseElements": {
        "cluster": {
```

```
            "clusterArn": "arn:aws:ecs:us-east-1:123456789012:cluster/default",
            "pendingTasksCount": 0,
            "registeredContainerInstancesCount": 0,
            "status": "ACTIVE",
            "runningTasksCount": 0,
            "statistics": [],
            "clusterName": "default",
            "activeServicesCount": 0
        }
    },
    "requestID": "cb8c167e-EXAMPLE",
    "eventID": "e3c6f4ce-EXAMPLE",
    "eventType": "AwsApiCall",
    "recipientAccountId": "123456789012"
}
```

使用 Amazon ECS 中繼資料監控工作負載

您可以使用工作和容器中繼資料來疑難排解工作負載，並根據執行階段環境進行組態變更。

中繼資料包括下列類別：

- 作業層級屬性，提供工作執行位置的相關資訊。
- 提供 Docker ID、名稱和映像詳細資料的容器層級屬性。

  這提供了對容器的可見性。
- 網路設定，例如 IP 位址、子網路和網路模式。

  這有助於進行網路設定和疑難排解。
- 工作狀態與健康狀態

  這可讓您知道工作是否正在執行。

您可以使用下列任一方法檢視中繼資料：

- 容器中繼資料檔案

  從 Amazon ECS 容器代理程式 1.15.0 版開始，容器或主機容器執行個體中提供各種容器中繼資料。透過啟用此功能，您可從容器或主機容器執行個體內查詢關於任務、容器和容器執行個體的資訊。

中繼資料檔案建立於主機執行個體，並在容器中掛載為 Docker 磁碟區，因此當任務託管於 AWS Fargate 時不可用。

- 任務中繼資料端點

  Amazon ECS 容器代理程式會將環境變數插入到每個容器中，稱為任務中繼資料端點，它為容器提供各種任務中繼資料和 Docker 統計資訊。

- 容器內省

  Amazon ECS 容器代理程式提供一種 API 操作，它收集執行代理程式的容器執行個體的詳細資訊，以及在該執行個體上執行之關聯任務的詳細資訊。

Amazon ECS 容器中繼資料檔案

從 Amazon ECS 容器代理程式 1.15.0 版開始，容器或主機容器執行個體中提供各種容器中繼資料。透過啟用此功能，您可從容器或主機容器執行個體內查詢關於任務、容器和容器執行個體的資訊。中繼資料檔案會建立在主機執行個體上，並以 Docker 磁碟區的形式掛載在容器中，因此在 Fargate 上託管工作時無 AWS 法使用。

清除容器時，會清除主機執行個體上的容器中繼資料檔案。您可以使用 `ECS_ENGINE_TASK_CLEANUP_WAIT_DURATION` 容器代理變數，來調整執行此動作的時間。如需詳細資訊，請參閱 自動 Amazon ECS 任務和圖像清理。

主題

- 容器中繼資料檔案位置
- 開啟 Amazon ECS 容器中繼資料
- Amazon ECS 容器元數據文件格式

容器中繼資料檔案位置

容器中繼資料檔案預設會寫入下列主機和容器路徑。

- 對於 Linux 執行個體：
  - 主機路徑：`/var/lib/ecs/data/metadata/`*`cluster_name`*`/`*`task_id`*`/`*`container_name`*`/ecs-container-metadata.json`

> **Note**
>
> Linux 主機路徑假設在啟動代理時，會使用預設資料目錄掛載路徑 (/var/lib/ecs/data)。如果您並非使用 Amazon ECS 最佳化 AMI (或 ecs-init 套件來啟動和維護容器代理程式)，請務必將 ECS_HOST_DATA_DIR 代理程式組態變數設定為容器代理程式狀態檔案所在的主機路徑。如需詳細資訊，請參閱 Amazon ECS 容器代理程式組態。

  - 容器路徑：/opt/ecs/metadata/*random_ID*/ecs-container-metadata.json
- 對於 Windows 執行個體：
  - 主機路徑：C:\ProgramData\Amazon\ECS\data\metadata*task_id**container_name*\ecs-container-metadata.json
  - 容器路徑：C:\ProgramData\Amazon\ECS\metadata*random_ID*\ecs-container-metadata.json

不過，為方便存取，容器中繼資料檔案位置設定為容器內部的 ECS_CONTAINER_METADATA_FILE 環境變數。您可以使用下列命令在容器內部讀取檔案內容：

- 對於 Linux 執行個體：

```
cat $ECS_CONTAINER_METADATA_FILE
```

- 對於視窗執行個體 (PowerShell)：

```
Get-Content -path $env:ECS_CONTAINER_METADATA_FILE
```

開啟 Amazon ECS 容器中繼資料

您可以將 ECS_ENABLE_CONTAINER_METADATA 容器代理變數設定為 true，以在容器執行個體層級開啟容器中繼資料。您可以在 /etc/ecs/ecs.config 組態檔案中設定此變數，並重新啟動代理。啟動代理容器時，您也可以在執行時間將它設定為 Docker 環境變數。如需詳細資訊，請參閱 Amazon ECS 容器代理程式組態。

如果 ECS_ENABLE_CONTAINER_METADATA 在代理程式啟動時設定為 true，則系統會為該時間點後建立的任何容器建立中繼資料檔案。對於在 ECS_ENABLE_CONTAINER_METADATA 容器代理程式變數設定為 true 之前所建立的容器，Amazon ECS 容器代理程式無法為其建立中繼資料檔案。為了確保

所有容器都收到中繼資料檔案，您應該在容器執行個體啟動時設定此代理變數。以下是範例使用者資料指令碼，可設定此變數以及使用叢集來註冊容器執行個體。

```
#!/bin/bash
cat <<'EOF' >> /etc/ecs/ecs.config
ECS_CLUSTER=your_cluster_name
ECS_ENABLE_CONTAINER_METADATA=true
EOF
```

Amazon ECS 容器元數據文件格式

下列資訊存放在容器中繼資料 JSON 檔案中。

`Cluster`

容器任務執行所在叢集的名稱。

`ContainerInstanceARN`

主機容器執行個體的完整 Amazon Resource Name (ARN)。

`TaskARN`

容器所屬任務的完整 Amazon Resource Name (ARN)。

`TaskDefinitionFamily`

容器正在使用的任務定義系列名稱。

`TaskDefinitionRevision`

容器正在使用的任務定義修訂版。

`ContainerID`

容器的 Docker 容器 ID (而不是 Amazon ECS 容器 ID)。

`ContainerName`

容器之 Amazon ECS 任務定義中的容器名稱。

`DockerContainerName`

Docker 常駐程式用於容器的容器名稱 (例如，顯示於 docker ps 命令輸出中的名稱)。

`ImageID`

用來啟動容器之 Docker 映像的 SHA 摘要。

`ImageName`

用來啟動容器之 Docker 映像的映像名稱和標籤。

`PortMappings`

與容器相關聯的任何連接埠映射。

`ContainerPort`

容器上公開的連接埠。

`HostPort`

主機容器執行個體上公開的連接埠。

`BindIp`

Docker 指派給容器的連結 IP 地址。此 IP 地址只能使用 `bridge` 網路模式予以套用，而且只能從容器執行個體存取。

`Protocol`

用於連接埠映射的網路協定。

`Networks`

容器的網路模式和 IP 地址。

`NetworkMode`

容器所屬任務的網路模式。

`IPv4Addresses`

與容器相關聯的 IP 地址。

> **Important**
>
> 如果您的任務使用的是 `awsvpc` 網路模式，就不會傳回容器的 IP 地址。在這種情況下，您可以使用下列命令讀取 /etc/hosts 檔案來擷取 IP 地址：
>
> ```
> tail -1 /etc/hosts | awk '{print $1}'
> ```

`MetadataFileStatus`

中繼資料檔案的狀態。狀態為 READY 時，中繼資料檔案是最新且完整的。如果檔案尚未就緒 (例如，啟動任務時)，則會提供截斷版本的檔案格式。為了避免已啟動容器但尚未寫入中繼資料的可能競爭條件，您可以在依存中繼資料之前剖析中繼資料檔案，並等待這個參數設定為 READY。這通常可在容器啟動的 1 秒內可用。

`AvailabilityZone`

主機容器執行個體所在的可用區域。

`HostPrivateIPv4Address`

容器所屬任務的私有 IP 地址。

`HostPublicIPv4Address`

容器所屬任務的公有 IP 地址。

Example Amazon ECS 容器中繼資料檔案 (**READY**)

下列範例示範狀態為 READY 的容器中繼資料檔案。

```
{
    "Cluster": "default",
    "ContainerInstanceARN": "arn:aws:ecs:us-west-2:012345678910:container-instance/
default/1f73d099-b914-411c-a9ff-81633b7741dd",
    "TaskARN": "arn:aws:ecs:us-west-2:012345678910:task/default/2b88376d-
aba3-4950-9ddf-bcb0f388a40c",
    "TaskDefinitionFamily": "console-sample-app-static",
    "TaskDefinitionRevision": "1",
    "ContainerID": "aec2557997f4eed9b280c2efd7afccdcedfda4ac399f7480cae870cfc7e163fd",
    "ContainerName": "simple-app",
    "CreatedAt": "2023-10-08T20:09:11.44527186Z",
    "StartedAt": "2023-10-08T20:09:11.44527186Z",
    "DockerContainerName": "/ecs-console-sample-app-static-1-simple-app-
e4e8e495e8baa5de1a00",
    "ImageID":
 "sha256:2ae34abc2ed0a22e280d17e13f9c01aaf725688b09b7a1525d1a2750e2c0d1de",
    "ImageName": "httpd:2.4",
    "PortMappings": [
        {
            "ContainerPort": 80,
```

```
            "HostPort": 80,
            "BindIp": "0.0.0.0",
            "Protocol": "tcp"
        }
    ],
    "Networks": [
        {
            "NetworkMode": "bridge",
            "IPv4Addresses": ["192.0.2.0"]
        }
    ],
    "MetadataFileStatus": "READY",
    "AvailabilityZone": "us-east-1b",
    "HostPrivateIPv4Address": "192.0.2.0",
    "HostPublicIPv4Address": "203.0.113.0"
}
```

Example 未完成的 Amazon ECS 容器中繼資料檔案 (尚未 **READY**)

下列範例示範尚未達到 READY 狀態的容器中繼資料檔案。檔案中的資訊僅限於任務定義中已知的數個參數。容器中繼資料檔案應該會在容器啟動的 1 秒內就緒。

```
{
    "Cluster": "default",
    "ContainerInstanceARN": "arn:aws:ecs:us-west-2:012345678910:container-instance/
default/1f73d099-b914-411c-a9ff-81633b7741dd",
    "TaskARN": "arn:aws:ecs:us-west-2:012345678910:task/default/
d90675f8-1a98-444b-805b-3d9cabb6fcd4",
    "ContainerName": "metadata"
}
```

適用於 EC2 上 Amazon ECS 任務的任務中繼資料

Amazon ECS 容器代理程式提供方法來擷取各種任務中繼資料和 Docker 統計資訊。這稱為任務中繼資料端點。以下為可用的版本：

- 任務中繼資料端點版本 4 — 為容器提供各種中繼資料和 Docker 統計資訊。也提供網路速率資料。適用於在執行 Amazon ECS 容器代理程式至少 `1.39.0` 版本的 Amazon EC2 Linux 執行個體上啟動的 Amazon ECS 任務。對於使用 `awsvpc` 網路模式的 Amazon EC2 Windows 執行個體，Amazon ECS 容器代理程式必須至少為 `1.54.0` 版。如需詳細資訊，請參閱 Amazon ECS 任務中繼資料端點版本 4。

- 任務中繼資料端點版本 3 — 為容器提供各種中繼資料和 Docker 統計資訊。適用於在執行 Amazon ECS 容器代理程式至少 `1.21.0` 版本的 Amazon EC2 Linux 執行個體上啟動的 Amazon ECS 任務。對於使用 `awsvpc` 網路模式的 Amazon EC2 Windows 執行個體，Amazon ECS 容器代理程式必須至少為 `1.54.0` 版。如需詳細資訊，請參閱 Amazon ECS 任務中繼資料端點版本 3。
- 任務中繼資料端點版本 2 — 適用於在執行 Amazon ECS 容器代理程式至少 `1.17.0` 版本的 Amazon EC2 Linux 執行個體上啟動的 Amazon ECS 任務。對於使用 `awsvpc` 網路模式的 Amazon EC2 Windows 執行個體，Amazon ECS 容器代理程式必須至少為 `1.54.0` 版。如需詳細資訊，請參閱 Amazon ECS 任務中繼資料端點版本 2。

如果在 Amazon EC2 上託管您的 Amazon ECS 任務，您也可以使用執行個體中繼資料服務 (IMDS) 端點存取任務主機中繼資料。當從託管任務的執行個體內執行時，下列命令會列出主機執行個體的 ID。

```
curl http://169.254.169.254/latest/meta-data/instance-id
```

您可以從端點取得的資訊分為幾個類別，例如 *`instance-id`*。如需使用端點取得不同類別主機執行個體中繼資料的詳細資訊，請參閱執行個體中繼資料類別。

Amazon ECS 任務中繼資料端點版本 4

Amazon ECS 容器代理程式會將環境變數插入到每個容器中，稱為任務中繼資料端點，它為容器提供各種任務中繼資料和 Docker 統計資訊。

工作中繼資料和網路速率統計資料會傳送至 CloudWatch 容器深入解析，並可在中檢視 AWS Management Console。如需詳細資訊，請參閱 使用容器洞察來監控 Amazon ECS 容器。

> **Note**
>
> Amazon ECS 提供任務中繼資料端點的較早版本。為了避免未來需要建立新的任務中繼資料端點版本，可以將其他中繼資料新增至第 4 版輸出。我們不會移除任何現有中繼資料或變更中繼資料欄位名稱。

環境變數會預設插入至 Amazon ECS 任務的容器中，這些任務在執行 Amazon ECS 容器代理程式至少 `1.39.0` 版的 Amazon EC2 Linux 執行個體上啟動。對於使用 `awsvpc` 網路模式的 Amazon EC2 Windows 執行個體，Amazon ECS 容器代理程式必須至少為 `1.54.0` 版。如需詳細資訊，請參閱 Amazon ECS Linux 容器執行個體管理。

> **Note**
>
> 使用較舊版本的 Amazon ECS 容器代理程式可在 Amazon EC2 執行個體上新增此功能的支援，方法是將代理程式更新為最新版本。如需詳細資訊，請參閱 更新 Amazon ECS 容器代理程式。

任務中繼資料端點版本 4 路徑

下列任務中繼資料端點路徑可供容器使用。

`${ECS_CONTAINER_METADATA_URI_V4}`

此路徑傳回容器的中繼資料。

`${ECS_CONTAINER_METADATA_URI_V4}/task`

此路徑傳回任務的中繼資料，包括與任務相關聯之所有容器的容器 ID 和名稱清單。如需此端點之回應的詳細資訊，請參閱「Amazon ECS 任務中繼資料 V4 JSON 回應」。

`${ECS_CONTAINER_METADATA_URI_V4}/taskWithTags`

除了可使用 `ListTagsForResource` API 擷取的任務和容器執行個體標籤之外，此路徑還會傳回包含在 `/task` 端點內的任務的中繼資料。擷取標籤中繼資料時收到的任何錯誤都會包含在回應的 `Errors` 欄位中。

> **Note**
>
> `Errors` 欄位僅位於執行容器代理程式至少 `1.50.0` 版本的 Amazon EC2 Linux 執行個體中託管的任務回應中。對於使用 `awsvpc` 網路模式的 Amazon EC2 Windows 執行個體，Amazon ECS 容器代理程式必須至少為 `1.54.0` 版。
> 此端點需要 `ecs.ListTagsForResource` 許可。

`${ECS_CONTAINER_METADATA_URI_V4}/stats`

此路徑傳回特定容器的 Docker 統計資訊。有關每個返回統計信息的更多信息，請參閱 Docker API 文檔ContainerStats中的。

對於使用 `awsvpc` 或 `bridge` 網路模式、託管於執行容器代理程式至少 `1.43.0` 版本的 Amazon EC2 Linux 執行個體中的 Amazon ECS 任務，回應中會包含額外的網路速率統計資訊。對於所有其他任務，回應只會包含累積的網路統計資訊。

`${ECS_CONTAINER_METADATA_URI_V4}/task/stats`

此路徑傳回與任務相關聯之所有容器的 Docker 統計資訊。附屬容器可使用該資訊來擷取網路指標。有關每個返回統計信息的更多信息，請參閱 Docker API 文檔ContainerStats中的。

對於使用 `awsvpc` 或 `bridge` 網路模式、託管於執行容器代理程式至少 `1.43.0` 版本的 Amazon EC2 Linux 執行個體中的 Amazon ECS 任務，回應中會包含額外的網路速率統計資訊。對於所有其他任務，回應只會包含累積的網路統計資訊。

Amazon ECS 任務中繼資料 V4 JSON 回應

任務中繼資料端點 (`${ECS_CONTAINER_METADATA_URI_V4}/task`) JSON 回應會傳回下列資訊。這包括除了任務內每個容器的中繼資料之外，與任務相關聯的中繼資料。

`Cluster`

任務所屬 Amazon ECS 叢集的 Amazon Resource Name (ARN) 或簡短名稱。

`ServiceName`

工作所屬的服務名稱。 ServiceName 如果任務與服務相關聯，將顯示在亞馬 Amazon EC2 和 Amazon ECS Anywhere 容器執行個體。

> **Note**
>
> 只有在使用 Amazon ECS 容器代理程式版本 `1.63.1` 或更新版本時，才包括該 `ServiceName` 中繼資料。

`VPCID`

Amazon EC2 容器執行個體的 VPC ID。此欄位僅適用於 Amazon EC2 執行個體。

> **Note**
>
> 只有在使用 Amazon ECS 容器代理程式版本 `1.63.1` 或更新版本時，才包括該 `VPCID` 中繼資料。

`TaskARN`

容器所屬任務的完整 Amazon Resource Name (ARN)。

`Family`

任務的 Amazon ECS 任務定義系列。

`Revision`

任務的 Amazon ECS 任務定義修訂。

`DesiredStatus`

Amazon ECS 中任務的所需狀態。

`KnownStatus`

Amazon ECS 中任務的已知狀態。

`Limits`

在任務層級指定的資源限制，例如 CPU (以 vCPU 表示) 和記憶體。如果未定義資源限制，則會省略此參數。

`PullStartedAt`

第一個容器映像提取的開始時間戳記。

`PullStoppedAt`

最後一個容器映像提取的完成時間戳記。

`AvailabilityZone`

任務所在的可用區域。

> **Note**
>
> 可用區域中繼資料僅適用於使用平台第 1.4 版或更新版本 (Linux) 或 1.0.0 (Windows) 的 Fargate 任務。

`LaunchType`

任務使用的啟動類型。使用叢集容量提供者時，這會指出任務是使用 Fargate 還是 EC2 基礎設施。

> **Note**
>
> 只有在使用 Amazon ECS Linux 容器代理程式版本 `1.45.0` 或更新版本 (Linux) 或 1.0.0 或更新版本 (Windows) 時，才包括此 `LaunchType` 中繼資料。

`Containers`

與任務相關聯之每個容器的容器中繼資料清單。

`DockerId`

容器的 Docker ID。

當您使用 Fargate 時，ID 為 32 位十六進制，後跟 10 位數字。

`Name`

任務定義中指定的容器名稱。

`DockerName`

提供給 Docker 的容器名稱。Amazon ECS 容器代理程式會產生容器的唯一名稱，以避免在單一執行個體上執行相同任務定義的多個複本時，發生名稱衝突。

`Image`

容器的映像。

`ImageID`

映像的 SHA-256 摘要。

`Ports`

向容器開放的任何連接埠。如果未開放連接埠，則會省略此參數。

`Labels`

任何套用至容器的標籤。如果未套用標籤，則會省略此參數。

`DesiredStatus`

Amazon ECS 中容器的所需狀態。

`KnownStatus`

Amazon ECS 中容器的已知狀態。

`ExitCode`

容器的結束代碼。如果容器尚未結束，則會省略此參數。

`Limits`

在容器層級指定的資源限制，例如 CPU (以 CPU 單位表示) 和記憶體。如果未定義資源限制，則會省略此參數。

`CreatedAt`

容器的建立時間戳記。如果尚未建立容器，則會省略此參數。

`StartedAt`

容器的啟動時間戳記。如果尚未啟動容器，則會省略此參數。

`FinishedAt`

容器的停止時間戳記。如果尚未停止容器，則會省略此參數。

`Type`

容器的類型。任務定義中指定的容器類型為 `NORMAL`。您可以忽略其他容器類型，這些是 Amazon ECS 容器代理程式用來佈建內部任務資源的容器類型。

`LogDriver`

容器正在使用的日誌驅動程式。

> **Note**
>
> 只有在使用 Amazon ECS Linux 容器代理程式版本 `1.45.0` 或更新版本時，才包括此 `LogDriver` 中繼資料。

`LogOptions`

為容器定義的日誌驅動程式選項。

> **Note**
>
> 只有在使用 Amazon ECS Linux 容器代理程式版本 `1.45.0` 或更新版本時，才包括此 `LogOptions` 中繼資料。

`ContainerARN`

容器的完整 Amazon Resource Name (ARN)。

> **Note**
>
> 只有在使用 Amazon ECS Linux 容器代理程式版本 1.45.0 或更新版本時，才包括此 ContainerARN 中繼資料。

Networks

容器的網路資訊，例如網路模式和 IP 地址。如果未定義網路資訊，則會省略此參數。

ExecutionStoppedAt

任務 DesiredStatus 移至 STOPPED 時的時間戳記。這會在基本容器移至 STOPPED 時發生。

Amazon ECS 任務中繼資料 v4 範例

以下範例顯示來自每個任務中繼資料端點的範例輸出。

容器中繼資料回應範例

查詢 ${ECS_CONTAINER_METADATA_URI_V4} 端點時，只會傳回關於容器本身的中繼資料。以下為範例輸出。

```
{
    "DockerId": "ea32192c8553fbff06c9340478a2ff089b2bb5646fb718b4ee206641c9086d66",
    "Name": "curl",
    "DockerName": "ecs-curltest-24-curl-cca48e8dcadd97805600",
    "Image": "111122223333.dkr.ecr.us-west-2.amazonaws.com/curltest:latest",
    "ImageID":
 "sha256:d691691e9652791a60114e67b365688d20d19940dde7c4736ea30e660d8d3553",
    "Labels": {
        "com.amazonaws.ecs.cluster": "default",
        "com.amazonaws.ecs.container-name": "curl",
        "com.amazonaws.ecs.task-arn": "arn:aws:ecs:us-west-2:111122223333:task/
default/8f03e41243824aea923aca126495f665",
        "com.amazonaws.ecs.task-definition-family": "curltest",
        "com.amazonaws.ecs.task-definition-version": "24"
    },
    "DesiredStatus": "RUNNING",
    "KnownStatus": "RUNNING",
    "Limits": {
        "CPU": 10,
        "Memory": 128
```

```
    },
    "CreatedAt": "2020-10-02T00:15:07.620912337Z",
    "StartedAt": "2020-10-02T00:15:08.062559351Z",
    "Type": "NORMAL",
    "LogDriver": "awslogs",
    "LogOptions": {
        "awslogs-create-group": "true",
        "awslogs-group": "/ecs/metadata",
        "awslogs-region": "us-west-2",
        "awslogs-stream": "ecs/curl/8f03e41243824aea923aca126495f665"
    },
    "ContainerARN": "arn:aws:ecs:us-west-2:111122223333:container/0206b271-
b33f-47ab-86c6-a0ba208a70a9",
    "Networks": [
        {
            "NetworkMode": "awsvpc",
            "IPv4Addresses": [
                "10.0.2.100"
            ],
            "AttachmentIndex": 0,
            "MACAddress": "0e:9e:32:c7:48:85",
            "IPv4SubnetCIDRBlock": "10.0.2.0/24",
            "PrivateDNSName": "ip-10-0-2-100.us-west-2.compute.internal",
            "SubnetGatewayIpv4Address": "10.0.2.1/24"
        }
    ]
}
```

範例任務中繼資料回應

查詢 ${ECS_CONTAINER_METADATA_URI_V4}/task 端點時，除了任務內每個容器的中繼資料外，只會傳回容器所屬任務的中繼資料。以下為範例輸出。

```
{
    "Cluster": "default",
    "TaskARN": "arn:aws:ecs:us-west-2:111122223333:task/
default/158d1c8083dd49d6b527399fd6414f5c",
    "Family": "curltest",
    "ServiceName": "MyService",
    "Revision": "26",
    "DesiredStatus": "RUNNING",
    "KnownStatus": "RUNNING",
    "PullStartedAt": "2020-10-02T00:43:06.202617438Z",
```

```
    "PullStoppedAt": "2020-10-02T00:43:06.31288465Z",
    "AvailabilityZone": "us-west-2d",
    "VPCID": "vpc-1234567890abcdef0",
    "LaunchType": "EC2",
    "Containers": [
        {
            "DockerId":
 "598cba581fe3f939459eaba1e071d5c93bb2c49b7d1ba7db6bb19deeb70d8e38",
            "Name": "~internal~ecs~pause",
            "DockerName": "ecs-curltest-26-internalecspause-e292d586b6f9dade4a00",
            "Image": "amazon/amazon-ecs-pause:0.1.0",
            "ImageID": "",
            "Labels": {
                "com.amazonaws.ecs.cluster": "default",
                "com.amazonaws.ecs.container-name": "~internal~ecs~pause",
                "com.amazonaws.ecs.task-arn": "arn:aws:ecs:us-west-2:111122223333:task/
default/158d1c8083dd49d6b527399fd6414f5c",
                "com.amazonaws.ecs.task-definition-family": "curltest",
                "com.amazonaws.ecs.task-definition-version": "26"
            },
            "DesiredStatus": "RESOURCES_PROVISIONED",
            "KnownStatus": "RESOURCES_PROVISIONED",
            "Limits": {
                "CPU": 0,
                "Memory": 0
            },
            "CreatedAt": "2020-10-02T00:43:05.602352471Z",
            "StartedAt": "2020-10-02T00:43:06.076707576Z",
            "Type": "CNI_PAUSE",
            "Networks": [
                {
                    "NetworkMode": "awsvpc",
                    "IPv4Addresses": [
                        "10.0.2.61"
                    ],
                    "AttachmentIndex": 0,
                    "MACAddress": "0e:10:e2:01:bd:91",
                    "IPv4SubnetCIDRBlock": "10.0.2.0/24",
                    "PrivateDNSName": "ip-10-0-2-61.us-west-2.compute.internal",
                    "SubnetGatewayIpv4Address": "10.0.2.1/24"
                }
            ]
        },
        {
```

```
            "DockerId":
 "ee08638adaaf009d78c248913f629e38299471d45fe7dc944d1039077e3424ca",
            "Name": "curl",
            "DockerName": "ecs-curltest-26-curl-a0e7dba5aca6d8cb2e00",
            "Image": "111122223333.dkr.ecr.us-west-2.amazonaws.com/curltest:latest",
            "ImageID":
 "sha256:d691691e9652791a60114e67b365688d20d19940dde7c4736ea30e660d8d3553",
            "Labels": {
                "com.amazonaws.ecs.cluster": "default",
                "com.amazonaws.ecs.container-name": "curl",
                "com.amazonaws.ecs.task-arn": "arn:aws:ecs:us-west-2:111122223333:task/
default/158d1c8083dd49d6b527399fd6414f5c",
                "com.amazonaws.ecs.task-definition-family": "curltest",
                "com.amazonaws.ecs.task-definition-version": "26"
            },
            "DesiredStatus": "RUNNING",
            "KnownStatus": "RUNNING",
            "Limits": {
                "CPU": 10,
                "Memory": 128
            },
            "CreatedAt": "2020-10-02T00:43:06.326590752Z",
            "StartedAt": "2020-10-02T00:43:06.767535449Z",
            "Type": "NORMAL",
            "LogDriver": "awslogs",
            "LogOptions": {
                "awslogs-create-group": "true",
                "awslogs-group": "/ecs/metadata",
                "awslogs-region": "us-west-2",
                "awslogs-stream": "ecs/curl/158d1c8083dd49d6b527399fd6414f5c"
            },
            "ContainerARN": "arn:aws:ecs:us-west-2:111122223333:container/
abb51bdd-11b4-467f-8f6c-adcfe1fe059d",
            "Networks": [
                {
                    "NetworkMode": "awsvpc",
                    "IPv4Addresses": [
                        "10.0.2.61"
                    ],
                    "AttachmentIndex": 0,
                    "MACAddress": "0e:10:e2:01:bd:91",
                    "IPv4SubnetCIDRBlock": "10.0.2.0/24",
                    "PrivateDNSName": "ip-10-0-2-61.us-west-2.compute.internal",
                    "SubnetGatewayIpv4Address": "10.0.2.1/24"
```

```
                }
            ]
        }
    ]
}
```

具有標籤中繼資料回應的任務範例

查詢 ${ECS_CONTAINER_METADATA_URI_V4}/taskWithTags 端點時，會傳回關於任務的中繼資料，包括任務和容器執行個體標籤。以下為範例輸出。

```
{
    "Cluster": "default",
    "TaskARN": "arn:aws:ecs:us-west-2:111122223333:task/
default/158d1c8083dd49d6b527399fd6414f5c",
    "Family": "curltest",
    "ServiceName": "MyService",
    "Revision": "26",
    "DesiredStatus": "RUNNING",
    "KnownStatus": "RUNNING",
    "PullStartedAt": "2020-10-02T00:43:06.202617438Z",
    "PullStoppedAt": "2020-10-02T00:43:06.31288465Z",
    "AvailabilityZone": "us-west-2d",
    "VPCID": "vpc-1234567890abcdef0",
    "TaskTags": {
        "tag-use": "task-metadata-endpoint-test"
    },
    "ContainerInstanceTags":{
        "tag_key":"tag_value"
    },
    "LaunchType": "EC2",
    "Containers": [
        {
            "DockerId":
 "598cba581fe3f939459eaba1e071d5c93bb2c49b7d1ba7db6bb19deeb70d8e38",
            "Name": "~internal~ecs~pause",
            "DockerName": "ecs-curltest-26-internalecspause-e292d586b6f9dade4a00",
            "Image": "amazon/amazon-ecs-pause:0.1.0",
            "ImageID": "",
            "Labels": {
                "com.amazonaws.ecs.cluster": "default",
                "com.amazonaws.ecs.container-name": "~internal~ecs~pause",
```

```
                "com.amazonaws.ecs.task-arn": "arn:aws:ecs:us-west-2:111122223333:task/
default/158d1c8083dd49d6b527399fd6414f5c",
                "com.amazonaws.ecs.task-definition-family": "curltest",
                "com.amazonaws.ecs.task-definition-version": "26"
            },
            "DesiredStatus": "RESOURCES_PROVISIONED",
            "KnownStatus": "RESOURCES_PROVISIONED",
            "Limits": {
                "CPU": 0,
                "Memory": 0
            },
            "CreatedAt": "2020-10-02T00:43:05.602352471Z",
            "StartedAt": "2020-10-02T00:43:06.076707576Z",
            "Type": "CNI_PAUSE",
            "Networks": [
                {
                    "NetworkMode": "awsvpc",
                    "IPv4Addresses": [
                        "10.0.2.61"
                    ],
                    "AttachmentIndex": 0,
                    "MACAddress": "0e:10:e2:01:bd:91",
                    "IPv4SubnetCIDRBlock": "10.0.2.0/24",
                    "PrivateDNSName": "ip-10-0-2-61.us-west-2.compute.internal",
                    "SubnetGatewayIpv4Address": "10.0.2.1/24"
                }
            ]
        },
        {
            "DockerId":
 "ee08638adaaf009d78c248913f629e38299471d45fe7dc944d1039077e3424ca",
            "Name": "curl",
            "DockerName": "ecs-curltest-26-curl-a0e7dba5aca6d8cb2e00",
            "Image": "111122223333.dkr.ecr.us-west-2.amazonaws.com/curltest:latest",
            "ImageID":
 "sha256:d691691e9652791a60114e67b365688d20d19940dde7c4736ea30e660d8d3553",
            "Labels": {
                "com.amazonaws.ecs.cluster": "default",
                "com.amazonaws.ecs.container-name": "curl",
                "com.amazonaws.ecs.task-arn": "arn:aws:ecs:us-west-2:111122223333:task/
default/158d1c8083dd49d6b527399fd6414f5c",
                "com.amazonaws.ecs.task-definition-family": "curltest",
                "com.amazonaws.ecs.task-definition-version": "26"
            },
```

```
            "DesiredStatus": "RUNNING",
            "KnownStatus": "RUNNING",
            "Limits": {
                "CPU": 10,
                "Memory": 128
            },
            "CreatedAt": "2020-10-02T00:43:06.326590752Z",
            "StartedAt": "2020-10-02T00:43:06.767535449Z",
            "Type": "NORMAL",
            "LogDriver": "awslogs",
            "LogOptions": {
                "awslogs-create-group": "true",
                "awslogs-group": "/ecs/metadata",
                "awslogs-region": "us-west-2",
                "awslogs-stream": "ecs/curl/158d1c8083dd49d6b527399fd6414f5c"
            },
            "ContainerARN": "arn:aws:ecs:us-west-2:111122223333:container/
abb51bdd-11b4-467f-8f6c-adcfe1fe059d",
            "Networks": [
                {
                    "NetworkMode": "awsvpc",
                    "IPv4Addresses": [
                        "10.0.2.61"
                    ],
                    "AttachmentIndex": 0,
                    "MACAddress": "0e:10:e2:01:bd:91",
                    "IPv4SubnetCIDRBlock": "10.0.2.0/24",
                    "PrivateDNSName": "ip-10-0-2-61.us-west-2.compute.internal",
                    "SubnetGatewayIpv4Address": "10.0.2.1/24"
                }
            ]
        }
    ]
}
```

具有標籤和錯誤中繼資料回應的任務範例

查詢 ${ECS_CONTAINER_METADATA_URI_V4}/taskWithTags 端點時，會傳回關於任務的中繼資料，包括任務和容器執行個體標籤。如果擷取標籤資料時發生錯誤，則會在回應中傳回錯誤。以下是與容器執行個體相關聯的 IAM 角色沒有允許 ecs:ListTagsForResource 許可時的輸出範例。

```
{
    "Cluster": "default",
```

```
    "TaskARN": "arn:aws:ecs:us-west-2:111122223333:task/
default/158d1c8083dd49d6b527399fd6414f5c",
    "Family": "curltest",
    "ServiceName": "MyService",
    "Revision": "26",
    "DesiredStatus": "RUNNING",
    "KnownStatus": "RUNNING",
    "PullStartedAt": "2020-10-02T00:43:06.202617438Z",
    "PullStoppedAt": "2020-10-02T00:43:06.31288465Z",
    "AvailabilityZone": "us-west-2d",
    "VPCID": "vpc-1234567890abcdef0",
    "Errors": [
        {
            "ErrorField": "ContainerInstanceTags",
            "ErrorCode": "AccessDeniedException",
            "ErrorMessage": "User: arn:aws:sts::111122223333:assumed-
role/ecsInstanceRole/i-0744a608689EXAMPLE is not authorized to perform:
 ecs:ListTagsForResource on resource: arn:aws:ecs:us-west-2:111122223333:container-
instance/default/2dd1b186f39845a584488d2ef155c131",
            "StatusCode": 400,
            "RequestId": "cd597ef0-272b-4643-9bd2-1de469870fa6",
            "ResourceARN": "arn:aws:ecs:us-west-2:111122223333:container-instance/
default/2dd1b186f39845a584488d2ef155c131"
        },
        {
            "ErrorField": "TaskTags",
            "ErrorCode": "AccessDeniedException",
            "ErrorMessage": "User: arn:aws:sts::111122223333:assumed-
role/ecsInstanceRole/i-0744a608689EXAMPLE is not authorized to perform:
 ecs:ListTagsForResource on resource: arn:aws:ecs:us-west-2:111122223333:task/
default/9ef30e4b7aa44d0db562749cff4983f3",
            "StatusCode": 400,
            "RequestId": "862c5986-6cd2-4aa6-87cc-70be395531e1",
            "ResourceARN": "arn:aws:ecs:us-west-2:111122223333:task/
default/9ef30e4b7aa44d0db562749cff4983f3"
        }
    ],
    "LaunchType": "EC2",
    "Containers": [
        {
            "DockerId":
 "598cba581fe3f939459eaba1e071d5c93bb2c49b7d1ba7db6bb19deeb70d8e38",
            "Name": "~internal~ecs~pause",
            "DockerName": "ecs-curltest-26-internalecspause-e292d586b6f9dade4a00",
```

```
            "Image": "amazon/amazon-ecs-pause:0.1.0",
            "ImageID": "",
            "Labels": {
                "com.amazonaws.ecs.cluster": "default",
                "com.amazonaws.ecs.container-name": "~internal~ecs~pause",
                "com.amazonaws.ecs.task-arn": "arn:aws:ecs:us-west-2:111122223333:task/
default/158d1c8083dd49d6b527399fd6414f5c",
                "com.amazonaws.ecs.task-definition-family": "curltest",
                "com.amazonaws.ecs.task-definition-version": "26"
            },
            "DesiredStatus": "RESOURCES_PROVISIONED",
            "KnownStatus": "RESOURCES_PROVISIONED",
            "Limits": {
                "CPU": 0,
                "Memory": 0
            },
            "CreatedAt": "2020-10-02T00:43:05.602352471Z",
            "StartedAt": "2020-10-02T00:43:06.076707576Z",
            "Type": "CNI_PAUSE",
            "Networks": [
                {
                    "NetworkMode": "awsvpc",
                    "IPv4Addresses": [
                        "10.0.2.61"
                    ],
                    "AttachmentIndex": 0,
                    "MACAddress": "0e:10:e2:01:bd:91",
                    "IPv4SubnetCIDRBlock": "10.0.2.0/24",
                    "PrivateDNSName": "ip-10-0-2-61.us-west-2.compute.internal",
                    "SubnetGatewayIpv4Address": "10.0.2.1/24"
                }
            ]
        },
        {
            "DockerId":
 "ee08638adaaf009d78c248913f629e38299471d45fe7dc944d1039077e3424ca",
            "Name": "curl",
            "DockerName": "ecs-curltest-26-curl-a0e7dba5aca6d8cb2e00",
            "Image": "111122223333.dkr.ecr.us-west-2.amazonaws.com/curltest:latest",
            "ImageID":
 "sha256:d691691e9652791a60114e67b365688d20d19940dde7c4736ea30e660d8d3553",
            "Labels": {
                "com.amazonaws.ecs.cluster": "default",
                "com.amazonaws.ecs.container-name": "curl",
```

```
                "com.amazonaws.ecs.task-arn": "arn:aws:ecs:us-west-2:111122223333:task/
default/158d1c8083dd49d6b527399fd6414f5c",
                "com.amazonaws.ecs.task-definition-family": "curltest",
                "com.amazonaws.ecs.task-definition-version": "26"
            },
            "DesiredStatus": "RUNNING",
            "KnownStatus": "RUNNING",
            "Limits": {
                "CPU": 10,
                "Memory": 128
            },
            "CreatedAt": "2020-10-02T00:43:06.326590752Z",
            "StartedAt": "2020-10-02T00:43:06.767535449Z",
            "Type": "NORMAL",
            "LogDriver": "awslogs",
            "LogOptions": {
                "awslogs-create-group": "true",
                "awslogs-group": "/ecs/metadata",
                "awslogs-region": "us-west-2",
                "awslogs-stream": "ecs/curl/158d1c8083dd49d6b527399fd6414f5c"
            },
            "ContainerARN": "arn:aws:ecs:us-west-2:111122223333:container/
abb51bdd-11b4-467f-8f6c-adcfe1fe059d",
            "Networks": [
                {
                    "NetworkMode": "awsvpc",
                    "IPv4Addresses": [
                        "10.0.2.61"
                    ],
                    "AttachmentIndex": 0,
                    "MACAddress": "0e:10:e2:01:bd:91",
                    "IPv4SubnetCIDRBlock": "10.0.2.0/24",
                    "PrivateDNSName": "ip-10-0-2-61.us-west-2.compute.internal",
                    "SubnetGatewayIpv4Address": "10.0.2.1/24"
                }
            ]
        }
    ]
}
```

容器統計資訊回應範例

查詢 ${ECS_CONTAINER_METADATA_URI_V4}/stats 端點時，會傳回容器的網路指標。對於使用 awsvpc 或 bridge 網路模式、託管於執行容器代理程式至少 1.43.0 版本的 Amazon EC2 執行個體中的 Amazon ECS 任務，回應中會包含額外的網路速率統計資訊。對於所有其他任務，回應只會包含累積的網路統計資訊。

以下是 Amazon EC2 上使用 bridge 網路模式的 Amazon ECS 任務的輸出範例。

```
{
    "read": "2020-10-02T00:51:13.410254284Z",
    "preread": "2020-10-02T00:51:12.406202398Z",
    "pids_stats": {
        "current": 3
    },
    "blkio_stats": {
        "io_service_bytes_recursive": [

        ],
        "io_serviced_recursive": [

        ],
        "io_queue_recursive": [

        ],
        "io_service_time_recursive": [

        ],
        "io_wait_time_recursive": [

        ],
        "io_merged_recursive": [

        ],
        "io_time_recursive": [

        ],
        "sectors_recursive": [

        ]
    },
    "num_procs": 0,
    "storage_stats": {
```

```
    },
    "cpu_stats": {
        "cpu_usage": {
            "total_usage": 360968065,
            "percpu_usage": [
                182359190,
                178608875
            ],
            "usage_in_kernelmode": 40000000,
            "usage_in_usermode": 290000000
        },
        "system_cpu_usage": 13939680000000,
        "online_cpus": 2,
        "throttling_data": {
            "periods": 0,
            "throttled_periods": 0,
            "throttled_time": 0
        }
    },
    "precpu_stats": {
        "cpu_usage": {
            "total_usage": 360968065,
            "percpu_usage": [
                182359190,
                178608875
            ],
            "usage_in_kernelmode": 40000000,
            "usage_in_usermode": 290000000
        },
        "system_cpu_usage": 13937670000000,
        "online_cpus": 2,
        "throttling_data": {
            "periods": 0,
            "throttled_periods": 0,
            "throttled_time": 0
        }
    },
    "memory_stats": {
        "usage": 1806336,
        "max_usage": 6299648,
        "stats": {
            "active_anon": 606208,
            "active_file": 0,
```

```
            "cache": 0,
            "dirty": 0,
            "hierarchical_memory_limit": 134217728,
            "hierarchical_memsw_limit": 268435456,
            "inactive_anon": 0,
            "inactive_file": 0,
            "mapped_file": 0,
            "pgfault": 4185,
            "pgmajfault": 0,
            "pgpgin": 2926,
            "pgpgout": 2778,
            "rss": 606208,
            "rss_huge": 0,
            "total_active_anon": 606208,
            "total_active_file": 0,
            "total_cache": 0,
            "total_dirty": 0,
            "total_inactive_anon": 0,
            "total_inactive_file": 0,
            "total_mapped_file": 0,
            "total_pgfault": 4185,
            "total_pgmajfault": 0,
            "total_pgpgin": 2926,
            "total_pgpgout": 2778,
            "total_rss": 606208,
            "total_rss_huge": 0,
            "total_unevictable": 0,
            "total_writeback": 0,
            "unevictable": 0,
            "writeback": 0
        },
        "limit": 134217728
    },
    "name": "/ecs-curltest-26-curl-c2e5f6e0cf91b0bead01",
    "id": "5fc21e5b015f899d22618f8aede80b6d70d71b2a75465ea49d9462c8f3d2d3af",
    "networks": {
        "eth0": {
            "rx_bytes": 84,
            "rx_packets": 2,
            "rx_errors": 0,
            "rx_dropped": 0,
            "tx_bytes": 84,
            "tx_packets": 2,
            "tx_errors": 0,
```

```
            "tx_dropped": 0
        }
    },
    "network_rate_stats": {
        "rx_bytes_per_sec": 0,
        "tx_bytes_per_sec": 0
    }
}
```

任務統計資訊回應範例

查詢 ${ECS_CONTAINER_METADATA_URI_V4}/task/stats 端點時，會傳回關於容器所屬任務的網路指標。以下為範例輸出。

```
{
    "01999f2e5c6cf4df3873f28950e6278813408f281c54778efec860d0caad4854": {
        "read": "2020-10-02T00:51:32.51467703Z",
        "preread": "2020-10-02T00:51:31.50860463Z",
        "pids_stats": {
            "current": 1
        },
        "blkio_stats": {
            "io_service_bytes_recursive": [

            ],
            "io_serviced_recursive": [

            ],
            "io_queue_recursive": [

            ],
            "io_service_time_recursive": [

            ],
            "io_wait_time_recursive": [

            ],
            "io_merged_recursive": [

            ],
            "io_time_recursive": [

            ],
```

```
            "sectors_recursive": [

            ]
        },
        "num_procs": 0,
        "storage_stats": {

        },
        "cpu_stats": {
            "cpu_usage": {
                "total_usage": 177232665,
                "percpu_usage": [
                    13376224,
                    163856441
                ],
                "usage_in_kernelmode": 0,
                "usage_in_usermode": 160000000
            },
            "system_cpu_usage": 13977820000000,
            "online_cpus": 2,
            "throttling_data": {
                "periods": 0,
                "throttled_periods": 0,
                "throttled_time": 0
            }
        },
        "precpu_stats": {
            "cpu_usage": {
                "total_usage": 177232665,
                "percpu_usage": [
                    13376224,
                    163856441
                ],
                "usage_in_kernelmode": 0,
                "usage_in_usermode": 160000000
            },
            "system_cpu_usage": 13975800000000,
            "online_cpus": 2,
            "throttling_data": {
                "periods": 0,
                "throttled_periods": 0,
                "throttled_time": 0
            }
        },
```

```
        "memory_stats": {
            "usage": 532480,
            "max_usage": 6279168,
            "stats": {
                "active_anon": 40960,
                "active_file": 0,
                "cache": 0,
                "dirty": 0,
                "hierarchical_memory_limit": 9223372036854771712,
                "hierarchical_memsw_limit": 9223372036854771712,
                "inactive_anon": 0,
                "inactive_file": 0,
                "mapped_file": 0,
                "pgfault": 2033,
                "pgmajfault": 0,
                "pgpgin": 1734,
                "pgpgout": 1724,
                "rss": 40960,
                "rss_huge": 0,
                "total_active_anon": 40960,
                "total_active_file": 0,
                "total_cache": 0,
                "total_dirty": 0,
                "total_inactive_anon": 0,
                "total_inactive_file": 0,
                "total_mapped_file": 0,
                "total_pgfault": 2033,
                "total_pgmajfault": 0,
                "total_pgpgin": 1734,
                "total_pgpgout": 1724,
                "total_rss": 40960,
                "total_rss_huge": 0,
                "total_unevictable": 0,
                "total_writeback": 0,
                "unevictable": 0,
                "writeback": 0
            },
            "limit": 4073377792
        },
        "name": "/ecs-curltest-26-internalecspause-a6bcc3dbadfacfe85300",
        "id": "01999f2e5c6cf4df3873f28950e6278813408f281c54778efec860d0caad4854",
        "networks": {
            "eth0": {
                "rx_bytes": 84,
```

```
                "rx_packets": 2,
                "rx_errors": 0,
                "rx_dropped": 0,
                "tx_bytes": 84,
                "tx_packets": 2,
                "tx_errors": 0,
                "tx_dropped": 0
            }
        },
        "network_rate_stats": {
            "rx_bytes_per_sec": 0,
            "tx_bytes_per_sec": 0
        }
    },
    "5fc21e5b015f899d22618f8aede80b6d70d71b2a75465ea49d9462c8f3d2d3af": {
        "read": "2020-10-02T00:51:32.512771349Z",
        "preread": "2020-10-02T00:51:31.510597736Z",
        "pids_stats": {
            "current": 3
        },
        "blkio_stats": {
            "io_service_bytes_recursive": [

            ],
            "io_serviced_recursive": [

            ],
            "io_queue_recursive": [

            ],
            "io_service_time_recursive": [

            ],
            "io_wait_time_recursive": [

            ],
            "io_merged_recursive": [

            ],
            "io_time_recursive": [

            ],
            "sectors_recursive": [
```

```
            ]
        },
        "num_procs": 0,
        "storage_stats": {

        },
        "cpu_stats": {
            "cpu_usage": {
                "total_usage": 379075681,
                "percpu_usage": [
                    191355275,
                    187720406
                ],
                "usage_in_kernelmode": 60000000,
                "usage_in_usermode": 310000000
            },
            "system_cpu_usage": 13977800000000,
            "online_cpus": 2,
            "throttling_data": {
                "periods": 0,
                "throttled_periods": 0,
                "throttled_time": 0
            }
        },
        "precpu_stats": {
            "cpu_usage": {
                "total_usage": 378825197,
                "percpu_usage": [
                    191104791,
                    187720406
                ],
                "usage_in_kernelmode": 60000000,
                "usage_in_usermode": 310000000
            },
            "system_cpu_usage": 13975800000000,
            "online_cpus": 2,
            "throttling_data": {
                "periods": 0,
                "throttled_periods": 0,
                "throttled_time": 0
            }
        },
        "memory_stats": {
            "usage": 1814528,
```

```
            "max_usage": 6299648,
            "stats": {
                "active_anon": 606208,
                "active_file": 0,
                "cache": 0,
                "dirty": 0,
                "hierarchical_memory_limit": 134217728,
                "hierarchical_memsw_limit": 268435456,
                "inactive_anon": 0,
                "inactive_file": 0,
                "mapped_file": 0,
                "pgfault": 5377,
                "pgmajfault": 0,
                "pgpgin": 3613,
                "pgpgout": 3465,
                "rss": 606208,
                "rss_huge": 0,
                "total_active_anon": 606208,
                "total_active_file": 0,
                "total_cache": 0,
                "total_dirty": 0,
                "total_inactive_anon": 0,
                "total_inactive_file": 0,
                "total_mapped_file": 0,
                "total_pgfault": 5377,
                "total_pgmajfault": 0,
                "total_pgpgin": 3613,
                "total_pgpgout": 3465,
                "total_rss": 606208,
                "total_rss_huge": 0,
                "total_unevictable": 0,
                "total_writeback": 0,
                "unevictable": 0,
                "writeback": 0
            },
            "limit": 134217728
        },
        "name": "/ecs-curltest-26-curl-c2e5f6e0cf91b0bead01",
        "id": "5fc21e5b015f899d22618f8aede80b6d70d71b2a75465ea49d9462c8f3d2d3af",
        "networks": {
            "eth0": {
                "rx_bytes": 84,
                "rx_packets": 2,
                "rx_errors": 0,
```

```
                "rx_dropped": 0,
                "tx_bytes": 84,
                "tx_packets": 2,
                "tx_errors": 0,
                "tx_dropped": 0
            }
        },
        "network_rate_stats": {
            "rx_bytes_per_sec": 0,
            "tx_bytes_per_sec": 0
        }
    }
}
```

Amazon ECS 任務中繼資料端點版本 3

> **Important**
>
> 不再積極維護任務中繼資料第 3 版端點。建議您更新任務中繼資料第 4 版端點，以取得最新的中繼資料端點資訊。如需詳細資訊，請參閱 the section called “任務中繼資料端點第 4 版”。如果您使用託管於 AWS Fargate上的 Amazon ECS 任務 AWS Fargate，請參閱 Amazon 彈性容器服務使用者指南中的任務中繼資料端點版本 3。

從 Amazon ECS 容器代理程式 1.21.0 版開始，代理程式會將稱為 `ECS_CONTAINER_METADATA_URI` 的環境變數插入任務中的每個容器。當您查詢任務中繼資料第 3 版端點時，有各種任務中繼資料和 Docker 統計資訊可供任務使用。對於使用 `bridge` 網路模式的工作，查詢 `/stats` 端點時，可使用網路指標。

根據預設，對於在平台版本 1.3.0 或更新版本上使用 Fargate 啟動類型的任務，以及使用 EC2 啟動類型的任務，會啟用任務中繼資料端點第 3 版功能，並在至少執行 Amazon ECS 容器代理程式 1.21.0 版的 Amazon EC2 Linux 基礎設施，或者在至少執行 Amazon ECS 容器代理程式 `1.54.0` 版的 Amazon EC2 Windows 基礎設施上啟動該功能，並使用 `awsvpc` 網路模式。如需詳細資訊，請參閱 Amazon ECS Linux 容器執行個體管理。

您可以將代理更新為最新版本，以在舊容器執行個體上新增此功能的支援。如需詳細資訊，請參閱 更新 Amazon ECS 容器代理程式。

> **Important**
>
> 對於使用 Fargate 啟動類型和低於 1.3.0 版平台版本的任務，支援任務中繼資料第 2 版端點。如需詳細資訊，請參閱 Amazon ECS 任務中繼資料端點版本 2。

工作中繼資料端點第 3 版路徑

下列任務中繼資料端點可供容器使用：

`${ECS_CONTAINER_METADATA_URI}`

此路徑傳回容器的 JSON 中繼資料。

`${ECS_CONTAINER_METADATA_URI}/task`

此路徑傳回任務的中繼資料 JSON，包括與任務相關聯之所有容器的容器 ID 和名稱清單。如需此端點之回應的詳細資訊，請參閱「Amazon ECS 任務元數據 v3 JSON 響應」。

`${ECS_CONTAINER_METADATA_URI}/taskWithTags`

除了可使用 `ListTagsForResource` API 擷取的任務和容器執行個體標籤之外，此路徑還會傳回包含在 `/task` 端點內的任務的中繼資料。

`${ECS_CONTAINER_METADATA_URI}/stats`

此路徑傳回特定 Docker 容器的 Docker 統計資訊 JSON。有關每個返回統計信息的更多信息，請參閱 Docker API 文檔ContainerStats中的。

`${ECS_CONTAINER_METADATA_URI}/task/stats`

此路徑傳回與任務相關聯之所有容器的 Docker 統計資訊 JSON。有關每個返回統計信息的更多信息，請參閱 Docker API 文檔ContainerStats中的。

Amazon ECS 任務元數據 v3 JSON 響應

任務中繼資料端點 (`${ECS_CONTAINER_METADATA_URI}/task`) JSON 回應會傳回下列資訊。

`Cluster`

任務所屬 Amazon ECS 叢集的 Amazon Resource Name (ARN) 或簡短名稱。

`TaskARN`

容器所屬任務的完整 Amazon Resource Name (ARN)。

`Family`

任務的 Amazon ECS 任務定義系列。

`Revision`

任務的 Amazon ECS 任務定義修訂。

`DesiredStatus`

Amazon ECS 中任務的所需狀態。

`KnownStatus`

Amazon ECS 中任務的已知狀態。

`Limits`

在任務層級指定的資源限制，例如 CPU (以 vCPU 表示) 和記憶體。如果未定義資源限制，則會省略此參數。

`PullStartedAt`

第一個容器映像提取的開始時間戳記。

`PullStoppedAt`

最後一個容器映像提取的完成時間戳記。

`AvailabilityZone`

任務所在的可用區域。

> **Note**
>
> 可用區域中繼資料僅適用於使用平台第 1.4 版或更新版本 (Linux) 或 1.0.0 或更新版本 (Windows) 的 Fargate 任務。

`Containers`

與任務相關聯之每個容器的容器中繼資料清單。

`DockerId`

容器的 Docker ID。

`Name`

任務定義中指定的容器名稱。

`DockerName`

提供給 Docker 的容器名稱。Amazon ECS 容器代理程式會產生容器的唯一名稱，以避免在單一執行個體上執行相同任務定義的多個複本時，發生名稱衝突。

`Image`

容器的映像。

`ImageID`

映像的 SHA-256 摘要。

`Ports`

向容器開放的任何連接埠。如果未開放連接埠，則會省略此參數。

`Labels`

任何套用至容器的標籤。如果未套用標籤，則會省略此參數。

`DesiredStatus`

Amazon ECS 中容器的所需狀態。

`KnownStatus`

Amazon ECS 中容器的已知狀態。

`ExitCode`

容器的結束代碼。如果容器尚未結束，則會省略此參數。

`Limits`

在容器層級指定的資源限制，例如 CPU (以 CPU 單位表示) 和記憶體。如果未定義資源限制，則會省略此參數。

`CreatedAt`

容器的建立時間戳記。如果尚未建立容器，則會省略此參數。

`StartedAt`

容器的啟動時間戳記。如果尚未啟動容器，則會省略此參數。

`FinishedAt`

容器的停止時間戳記。如果尚未停止容器，則會省略此參數。

`Type`

容器的類型。任務定義中指定的容器類型為 `NORMAL`。您可以忽略其他容器類型，這些是 Amazon ECS 容器代理程式用來佈建內部任務資源的容器類型。

`Networks`

容器的網路資訊，例如網路模式和 IP 地址。如果未定義網路資訊，則會省略此參數。

`ClockDrift`

關於參考時間和系統時間之間差異的資訊。這適用於 Linux 作業系統。此功能使用 Amazon 時間同步服務來測量時脈準確度，並提供容器的時脈錯誤。如需詳細資訊，請參閱 Amazon EC2 Linux 執行個體使用者指南中的設定 Linux 執行個體的時間。

`ReferenceTime`

時鐘精確度的基礎。Amazon ECS 透過 NTP 使用世界協調時間 (UTC) 的全球標準，例如 `2021-09-07T16:57:44Z`。

`ClockErrorBound`

時鐘誤差的測量值，定義為 UTC 的偏移。此誤差是參考時間和系統時間之間的毫秒差異。

`ClockSynchronizationStatus`

指出系統時間和參考時間之間最近的同步嘗試是否成功。

有效值為 `SYNCHRONIZED` 和 `NOT_SYNCHRONIZED`。

`ExecutionStoppedAt`

任務 `DesiredStatus` 移至 `STOPPED` 時的時間戳記。這會在基本容器移至 `STOPPED` 時發生。

Amazon ECS 任務中繼資料 v3 範例

以下範例顯示來自任務中繼資料端點的範例輸出。

容器中繼資料回應範例

查詢 `${ECS_CONTAINER_METADATA_URI}` 端點時，只會傳回關於容器本身的中繼資料。以下為範例輸出。

```
{
    "DockerId": "43481a6ce4842eec8fe72fc28500c6b52edcc0917f105b83379f88cac1ff3946",
    "Name": "nginx-curl",
    "DockerName": "ecs-nginx-5-nginx-curl-ccccb9f49db0dfe0d901",
    "Image": "nrdlngr/nginx-curl",
    "ImageID":
 "sha256:2e00ae64383cfc865ba0a2ba37f61b50a120d2d9378559dcd458dc0de47bc165",
    "Labels": {
        "com.amazonaws.ecs.cluster": "default",
        "com.amazonaws.ecs.container-name": "nginx-curl",
        "com.amazonaws.ecs.task-arn": "arn:aws:ecs:us-
east-2:012345678910:task/9781c248-0edd-4cdb-9a93-f63cb662a5d3",
        "com.amazonaws.ecs.task-definition-family": "nginx",
        "com.amazonaws.ecs.task-definition-version": "5"
    },
    "DesiredStatus": "RUNNING",
    "KnownStatus": "RUNNING",
    "Limits": {
        "CPU": 512,
        "Memory": 512
    },
    "CreatedAt": "2018-02-01T20:55:10.554941919Z",
    "StartedAt": "2018-02-01T20:55:11.064236631Z",
    "Type": "NORMAL",
    "Networks": [
        {
            "NetworkMode": "awsvpc",
            "IPv4Addresses": [
                "10.0.2.106"
            ]
        }
    ]
}
```

範例任務中繼資料回應

查詢 ${ECS_CONTAINER_METADATA_URI}/task 端點時，只會傳回容器所屬任務的中繼資料。以下為範例輸出。

下列為單一容器任務的 JSON 回應。

```
{
  "Cluster": "default",
```

```
  "TaskARN": "arn:aws:ecs:us-east-2:012345678910:task/9781c248-0edd-4cdb-9a93-
f63cb662a5d3",
  "Family": "nginx",
  "Revision": "5",
  "DesiredStatus": "RUNNING",
  "KnownStatus": "RUNNING",
  "Containers": [
    {
      "DockerId": "731a0d6a3b4210e2448339bc7015aaa79bfe4fa256384f4102db86ef94cbbc4c",
      "Name": "~internal~ecs~pause",
      "DockerName": "ecs-nginx-5-internalecspause-acc699c0cbf2d6d11700",
      "Image": "amazon/amazon-ecs-pause:0.1.0",
      "ImageID": "",
      "Labels": {
        "com.amazonaws.ecs.cluster": "default",
        "com.amazonaws.ecs.container-name": "~internal~ecs~pause",
        "com.amazonaws.ecs.task-arn": "arn:aws:ecs:us-
east-2:012345678910:task/9781c248-0edd-4cdb-9a93-f63cb662a5d3",
        "com.amazonaws.ecs.task-definition-family": "nginx",
        "com.amazonaws.ecs.task-definition-version": "5"
      },
      "DesiredStatus": "RESOURCES_PROVISIONED",
      "KnownStatus": "RESOURCES_PROVISIONED",
      "Limits": {
        "CPU": 0,
        "Memory": 0
      },
      "CreatedAt": "2018-02-01T20:55:08.366329616Z",
      "StartedAt": "2018-02-01T20:55:09.058354915Z",
      "Type": "CNI_PAUSE",
      "Networks": [
        {
          "NetworkMode": "awsvpc",
          "IPv4Addresses": [
            "10.0.2.106"
          ]
        }
      ]
    },
    {
      "DockerId": "43481a6ce4842eec8fe72fc28500c6b52edcc0917f105b83379f88cac1ff3946",
      "Name": "nginx-curl",
      "DockerName": "ecs-nginx-5-nginx-curl-ccccb9f49db0dfe0d901",
      "Image": "nrdlngr/nginx-curl",
```

```
      "ImageID":
 "sha256:2e00ae64383cfc865ba0a2ba37f61b50a120d2d9378559dcd458dc0de47bc165",
      "Labels": {
        "com.amazonaws.ecs.cluster": "default",
        "com.amazonaws.ecs.container-name": "nginx-curl",
        "com.amazonaws.ecs.task-arn": "arn:aws:ecs:us-
east-2:012345678910:task/9781c248-0edd-4cdb-9a93-f63cb662a5d3",
        "com.amazonaws.ecs.task-definition-family": "nginx",
        "com.amazonaws.ecs.task-definition-version": "5"
      },
      "DesiredStatus": "RUNNING",
      "KnownStatus": "RUNNING",
      "Limits": {
        "CPU": 512,
        "Memory": 512
      },
      "CreatedAt": "2018-02-01T20:55:10.554941919Z",
      "StartedAt": "2018-02-01T20:55:11.064236631Z",
      "Type": "NORMAL",
      "Networks": [
        {
          "NetworkMode": "awsvpc",
          "IPv4Addresses": [
            "10.0.2.106"
          ]
        }
      ]
    }
  ],
  "PullStartedAt": "2018-02-01T20:55:09.372495529Z",
  "PullStoppedAt": "2018-02-01T20:55:10.552018345Z",
  "AvailabilityZone": "us-east-2b"
}
```


Amazon ECS 任務中繼資料端點版本 2

> **Important**
>
> 不再積極維護任務中繼資料第 2 版端點。建議您更新任務中繼資料第 4 版端點，以取得最新的中繼資料端點資訊。如需詳細資訊，請參閱 the section called “任務中繼資料端點第 4 版”。

從 Amazon ECS 容器代理程式 1.17.0 版開始，在 Amazon ECS 容器代理提供之 HTTP 端點使用 awsvpc 網路模式的任務，皆可使用各種任務中繼資料和 Docker 統計資訊。

屬於使用 awsvpc 網路模式啟動之任務的所有容器都接收預先定義連結本機地址範圍內的本機 IPv4 地址。當容器查詢中繼資料端點時，Amazon ECS 容器代理程式可以根據任務的唯一 IP 地址，判斷容器屬於哪些任務，並傳回該任務的中繼資料和統計資料。

啟用任務中繼資料

對於下列項目，預設會啟用任務中繼資料第 2 版功能：

- 使用 Fargate 啟動類型且使用平台 1.1.0 版或更新版本的任務。如需詳細資訊，請參閱 適用於 Amazon ECS 的 Fargate Linux 平台版本。
- 使用 EC2 啟動類型的任務，該任務也會使用 awsvpc 網路模式，且在至少執行 Amazon ECS 容器代理程式 1.17.0 版的 Amazon EC2 Linux 基礎設施上啟動，或在至少執行 1.54.0 版 Amazon ECS 容器代理程式的 Amazon EC2 Windows 基礎設施上啟動。如需詳細資訊，請參閱 Amazon ECS Linux 容器執行個體管理。

您可以將代理更新為最新版本，以在舊容器執行個體上新增此功能的支援。如需詳細資訊，請參閱 更新 Amazon ECS 容器代理程式。

任務中繼資料端點路徑

容器可使用下列 API 端點：

`169.254.170.2/v2/metadata`

此端點會傳回任務的中繼資料 JSON，包括與任務相關聯之所有容器的容器 ID 和名稱清單。如需此端點之回應的詳細資訊，請參閱「任務中繼資料 JSON 回應」。

`169.254.170.2/v2/metadata/<container-id>`

此端點會傳回指定 Docker 容器 ID 的中繼資料 JSON。

`169.254.170.2/v2/metadata/taskWithTags`

除了可使用 ListTagsForResource API 擷取的任務和容器執行個體標籤之外，此路徑還會傳回包含在 /task 端點內的任務的中繼資料。

`169.254.170.2/v2/stats`

此端點會傳回與任務相關聯之所有容器的 Docker 統計資訊 JSON。有關每個返回統計信息的更多信息，請參閱 Docker API 文檔ContainerStats中的。

`169.254.170.2/v2/stats/<container-id>`

此端點會傳回指定 Docker 容器 ID 的 Docker 統計資訊 JSON。有關每個返回統計信息的更多信息，請參閱 Docker API 文檔ContainerStats中的。

任務中繼資料 JSON 回應

任務中繼資料端點 (`169.254.170.2/v2/metadata`) JSON 回應會傳回下列資訊。

`Cluster`

任務所屬 Amazon ECS 叢集的 Amazon Resource Name (ARN) 或簡短名稱。

`TaskARN`

容器所屬任務的完整 Amazon Resource Name (ARN)。

`Family`

任務的 Amazon ECS 任務定義系列。

`Revision`

任務的 Amazon ECS 任務定義修訂。

`DesiredStatus`

Amazon ECS 中任務的所需狀態。

`KnownStatus`

Amazon ECS 中任務的已知狀態。

`Limits`

在任務層級指定的資源限制，例如 CPU (以 vCPU 表示) 和記憶體。如果未定義資源限制，則會省略此參數。

`PullStartedAt`

第一個容器映像提取的開始時間戳記。

`PullStoppedAt`

最後一個容器映像提取的完成時間戳記。

`AvailabilityZone`

任務所在的可用區域。

> **Note**
>
> 可用區域中繼資料僅適用於使用平台第 1.4 版或更新版本 (Linux) 或 1.0.0 或更新版本 (Windows) 的 Fargate 任務。

`Containers`

與任務相關聯之每個容器的容器中繼資料清單。

`DockerId`

容器的 Docker ID。

`Name`

任務定義中指定的容器名稱。

`DockerName`

提供給 Docker 的容器名稱。Amazon ECS 容器代理程式會產生容器的唯一名稱，以避免在單一執行個體上執行相同任務定義的多個複本時，發生名稱衝突。

`Image`

容器的映像。

`ImageID`

映像的 SHA-256 摘要。

`Ports`

向容器開放的任何連接埠。如果未開放連接埠，則會省略此參數。

`Labels`

任何套用至容器的標籤。如果未套用標籤，則會省略此參數。

`DesiredStatus`

Amazon ECS 中容器的所需狀態。

`KnownStatus`

Amazon ECS 中容器的已知狀態。

`ExitCode`

容器的結束代碼。如果容器尚未結束，則會省略此參數。

`Limits`

在容器層級指定的資源限制，例如 CPU (以 CPU 單位表示) 和記憶體。如果未定義資源限制，則會省略此參數。

`CreatedAt`

容器的建立時間戳記。如果尚未建立容器，則會省略此參數。

`StartedAt`

容器的啟動時間戳記。如果尚未啟動容器，則會省略此參數。

`FinishedAt`

容器的停止時間戳記。如果尚未停止容器，則會省略此參數。

`Type`

容器的類型。任務定義中指定的容器類型為 `NORMAL`。您可以忽略其他容器類型，這些是 Amazon ECS 容器代理程式用來佈建內部任務資源的容器類型。

`Networks`

容器的網路資訊，例如網路模式和 IP 地址。如果未定義網路資訊，則會省略此參數。

`ClockDrift`

關於參考時間和系統時間之間差異的資訊。這適用於 Linux 作業系統。此功能使用 Amazon 時間同步服務來測量時脈準確度，並提供容器的時脈錯誤。如需詳細資訊，請參閱 Amazon EC2 Linux 執行個體使用者指南中的設定 Linux 執行個體的時間。

`ReferenceTime`

時鐘精確度的基礎。Amazon ECS 透過 NTP 使用世界協調時間 (UTC) 的全球標準，例如 `2021-09-07T16:57:44Z`。

`ClockErrorBound`

時鐘誤差的測量值，定義為 UTC 的偏移。此誤差是參考時間和系統時間之間的毫秒差異。

`ClockSynchronizationStatus`

指出系統時間和參考時間之間最近的同步嘗試是否成功。

有效值為 `SYNCHRONIZED` 和 `NOT_SYNCHRONIZED`。

`ExecutionStoppedAt`

任務 `DesiredStatus` 移至 `STOPPED` 時的時間戳記。這會在基本容器移至 `STOPPED` 時發生。

範例任務中繼資料回應

下列為單一容器任務的 JSON 回應。

```
{
  "Cluster": "default",
  "TaskARN": "arn:aws:ecs:us-east-2:012345678910:task/9781c248-0edd-4cdb-9a93-
f63cb662a5d3",
  "Family": "nginx",
  "Revision": "5",
  "DesiredStatus": "RUNNING",
  "KnownStatus": "RUNNING",
  "Containers": [
    {
      "DockerId": "731a0d6a3b4210e2448339bc7015aaa79bfe4fa256384f4102db86ef94cbbc4c",
      "Name": "~internal~ecs~pause",
      "DockerName": "ecs-nginx-5-internalecspause-acc699c0cbf2d6d11700",
      "Image": "amazon/amazon-ecs-pause:0.1.0",
      "ImageID": "",
      "Labels": {
        "com.amazonaws.ecs.cluster": "default",
        "com.amazonaws.ecs.container-name": "~internal~ecs~pause",
        "com.amazonaws.ecs.task-arn": "arn:aws:ecs:us-
east-2:012345678910:task/9781c248-0edd-4cdb-9a93-f63cb662a5d3",
        "com.amazonaws.ecs.task-definition-family": "nginx",
        "com.amazonaws.ecs.task-definition-version": "5"
      },
      "DesiredStatus": "RESOURCES_PROVISIONED",
      "KnownStatus": "RESOURCES_PROVISIONED",
      "Limits": {
        "CPU": 0,
        "Memory": 0
      },
      "CreatedAt": "2018-02-01T20:55:08.366329616Z",
      "StartedAt": "2018-02-01T20:55:09.058354915Z",
      "Type": "CNI_PAUSE",
      "Networks": [
        {
          "NetworkMode": "awsvpc",
          "IPv4Addresses": [
            "10.0.2.106"
          ]
        }
      ]
```

```
    },
    {
      "DockerId": "43481a6ce4842eec8fe72fc28500c6b52edcc0917f105b83379f88cac1ff3946",
      "Name": "nginx-curl",
      "DockerName": "ecs-nginx-5-nginx-curl-ccccb9f49db0dfe0d901",
      "Image": "nrdlngr/nginx-curl",
      "ImageID":
 "sha256:2e00ae64383cfc865ba0a2ba37f61b50a120d2d9378559dcd458dc0de47bc165",
      "Labels": {
        "com.amazonaws.ecs.cluster": "default",
        "com.amazonaws.ecs.container-name": "nginx-curl",
        "com.amazonaws.ecs.task-arn": "arn:aws:ecs:us-
east-2:012345678910:task/9781c248-0edd-4cdb-9a93-f63cb662a5d3",
        "com.amazonaws.ecs.task-definition-family": "nginx",
        "com.amazonaws.ecs.task-definition-version": "5"
      },
      "DesiredStatus": "RUNNING",
      "KnownStatus": "RUNNING",
      "Limits": {
        "CPU": 512,
        "Memory": 512
      },
      "CreatedAt": "2018-02-01T20:55:10.554941919Z",
      "StartedAt": "2018-02-01T20:55:11.064236631Z",
      "Type": "NORMAL",
      "Networks": [
        {
          "NetworkMode": "awsvpc",
          "IPv4Addresses": [
            "10.0.2.106"
          ]
        }
      ]
    }
  ],
  "PullStartedAt": "2018-02-01T20:55:09.372495529Z",
  "PullStoppedAt": "2018-02-01T20:55:10.552018345Z",
  "AvailabilityZone": "us-east-2b"
}
```

Amazon ECS 任務中繼資料可用於 Fargate 上的任務

Amazon ECS on Fargate 提供一種方法，可擷取關於您容器和任務的各種中繼資料、網路指標，以及 Docker 統計資料。這稱為任務中繼資料端點。以下任務中繼資料端點版本可用於 Amazon ECS on Fargate 任務：

- 任務中繼資料端點第 4 版：適用於使用平台版本 1.4.0 或更新版本的任務。
- 任務中繼資料端點第 3 版：適用於使用平台版本 1.1.0 或更新版本的任務。

屬於使用 awsvpc 網路模式啟動之任務的所有容器都接收預先定義連結本機地址範圍內的本機 IPv4 地址。當容器查詢中繼資料端點時，容器代理程式可以根據任務的唯一 IP 地址，判斷容器屬於哪些任務，並傳回該任務的中繼資料和統計資料。

主題

- 適用於 Fargate 上任務的 Amazon ECS 任務中繼資料端點第 4 版
- 適用於 Fargate 上任務的 Amazon ECS 任務中繼資料端點版本 3

適用於 Fargate 上任務的 Amazon ECS 任務中繼資料端點第 4 版

> **Important**
>
> 如果您使用在 Amazon EC2 執行個體上託管的 Amazon ECS 任務，請參閱 Amazon ECS 任務中繼資料端點。

從 Fargate 平台版本 1.4.0 開始，名為 ECS_CONTAINER_METADATA_URI_V4 的環境變數會插入任務中的每個容器。當您查詢任務中繼資料端點第 4 版時，有各種任務中繼資料和 Docker 統計資訊可供任務使用。

任務中繼資料端點第 4 版功能與第 3 版端點類似，但第 4 版為您的容器和任務提供其他網路中繼資料。查詢 /stats 端點時，也可以使用其他網路指標。

對於使用平台版本1.4.0或更新版本執行的所有 Amazon ECS 任務，預設情況下，工作中繼資料端點處於開啟 AWS Fargate 狀態。

> **Note**
>
> 為了避免未來需要建立新的任務中繼資料端點版本，可以將其他中繼資料新增至第 4 版輸出。我們不會移除任何現有中繼資料或變更中繼資料欄位名稱。

Fargate 任務中繼資料端點第 4 版路徑

下列任務中繼資料端點可供容器使用：

`${ECS_CONTAINER_METADATA_URI_V4}`

此路徑傳回容器的中繼資料。

`${ECS_CONTAINER_METADATA_URI_V4}/task`

此路徑傳回任務的中繼資料，包括與任務相關聯之所有容器的容器 ID 和名稱清單。如需此端點之回應的詳細資訊，請參閱「Amazon ECS 任務中繼資料 v4 適用於 Fargate 上任務的 JSON 回應」。

`${ECS_CONTAINER_METADATA_URI_V4}/stats`

此路徑傳回 Docker 容器的 Docker 統計資訊。有關每個返回統計信息的更多信息，請參閱 Docker API 文檔ContainerStats中的。

> **Note**
>
> 傳回容器統計資料之前，Amazon ECS 任務 AWS Fargate 要求容器執行約 1 秒鐘。

`${ECS_CONTAINER_METADATA_URI_V4}/task/stats`

此路徑傳回與任務相關聯之所有容器的 Docker 統計資訊。有關每個返回統計信息的更多信息，請參閱 Docker API 文檔ContainerStats中的。

> **Note**
>
> 傳回容器統計資料之前，Amazon ECS 任務 AWS Fargate 要求容器執行約 1 秒鐘。

Amazon ECS 任務中繼資料 v4 適用於 Fargate 上任務的 JSON 回應

任務中繼資料端點 (`${ECS_CONTAINER_METADATA_URI_V4}/task`) JSON 回應會傳回下列中繼資料。

`Cluster`

任務所屬 Amazon ECS 叢集的 Amazon Resource Name (ARN) 或簡短名稱。

`VPCID`

Amazon EC2 容器執行個體的 VPC ID。此欄位僅適用於 Amazon EC2 執行個體。

> **Note**
>
> 只有在使用 Amazon ECS 容器代理程式版本 `1.63.1` 或更新版本時，才包括該 `VPCID` 中繼資料。

`TaskARN`

容器所屬任務的完整 Amazon Resource Name (ARN)。

`Family`

任務的 Amazon ECS 任務定義系列。

`Revision`

任務的 Amazon ECS 任務定義修訂。

`DesiredStatus`

Amazon ECS 中任務的所需狀態。

`KnownStatus`

Amazon ECS 中任務的已知狀態。

`Limits`

在任務層級指定的資源限制，例如 CPU (以 vCPU 表示) 和記憶體。如果未定義資源限制，則會省略此參數。

`PullStartedAt`

第一個容器映像提取的開始時間戳記。

`PullStoppedAt`

最後一個容器映像提取的完成時間戳記。

`AvailabilityZone`

任務所在的可用區域。

> **Note**
>
> 可用區域中繼資料僅適用於使用平台第 1.4 版或更新版本 (Linux) 或 1.0.0 (Windows) 的 Fargate 任務。

`LaunchType`

任務使用的啟動類型。使用叢集容量提供者時，這會指出任務是使用 Fargate 還是 EC2 基礎設施。

> **Note**
>
> 只有在使用 Amazon ECS Linux 容器代理程式版本 `1.45.0` 或更新版本 (Linux) 或 1.0.0 或更新版本 (Windows) 時，才包括此 `LaunchType` 中繼資料。

`EphemeralStorageMetrics`

此任務暫時性儲存的保留大小和目前使用量。

> **Note**
>
> Fargate 會在磁盤上保留空間。此空間僅由 Fargate 使用。我們不會向您收費。它不會顯示在這些指標中。但是，您可以在其他工具 (例如 df) 中看到此額外儲存空間。

`Utilized`

此任務目前的暫時性儲存使用量 (MiB)。

`Reserved`

此任務的保留暫時性儲存裝置 (MiB)。在執行中的任務中，無法變更暫時性儲存的大小。您可以在任務定義中指定 `ephermalStorage` 物件來變更暫時性儲存量。`ephermalStorage` 是以

GiB 為單位，而不是以 MiB 為單位。此 ephermalStorage 與 EphemeralStorageMetrics 僅適用於 Fargate Linux 平台版本 1.4.0 或更高版本。

Containers

與任務相關聯之每個容器的容器中繼資料清單。

DockerId

容器的 Docker ID。

當您使用 Fargate 時，ID 為 32 位十六進制，後跟 10 位數字。

Name

任務定義中指定的容器名稱。

DockerName

提供給 Docker 的容器名稱。Amazon ECS 容器代理程式會產生容器的唯一名稱，以避免在單一執行個體上執行相同任務定義的多個複本時，發生名稱衝突。

Image

容器的映像。

ImageID

映像的 SHA-256 摘要。

Ports

向容器開放的任何連接埠。如果未開放連接埠，則會省略此參數。

Labels

任何套用至容器的標籤。如果未套用標籤，則會省略此參數。

DesiredStatus

Amazon ECS 中容器的所需狀態。

KnownStatus

Amazon ECS 中容器的已知狀態。

ExitCode

容器的結束代碼。如果容器尚未結束，則會省略此參數。

`Limits`

在容器層級指定的資源限制，例如 CPU (以 CPU 單位表示) 和記憶體。如果未定義資源限制，則會省略此參數。

`CreatedAt`

容器的建立時間戳記。如果尚未建立容器，則會省略此參數。

`StartedAt`

容器的啟動時間戳記。如果尚未啟動容器，則會省略此參數。

`FinishedAt`

容器的停止時間戳記。如果尚未停止容器，則會省略此參數。

`Type`

容器的類型。任務定義中指定的容器類型為 `NORMAL`。您可以忽略其他容器類型，這些是 Amazon ECS 容器代理程式用來佈建內部任務資源的容器類型。

`LogDriver`

容器正在使用的日誌驅動程式。

> **Note**
>
> 只有在使用 Amazon ECS Linux 容器代理程式版本 `1.45.0` 或更新版本時，才包括此 `LogDriver` 中繼資料。

`LogOptions`

為容器定義的日誌驅動程式選項。

> **Note**
>
> 只有在使用 Amazon ECS Linux 容器代理程式版本 `1.45.0` 或更新版本時，才包括此 `LogOptions` 中繼資料。

`ContainerARN`

容器的完整 Amazon Resource Name (ARN)。

> **Note**
>
> 只有在使用 Amazon ECS Linux 容器代理程式版本 1.45.0 或更新版本時，才包括此 ContainerARN 中繼資料。

Networks

容器的網路資訊，例如網路模式和 IP 地址。如果未定義網路資訊，則會省略此參數。

Snapshotter

containerd 使用 snapshotter 來下載此容器映像。有效值為 overlayfs (為預設值) 和 soci，並且在使用 SOCI 索引延遲載入時使用。此參數僅適用於在 Linux 平台版本 1.4.0 上執行的任務。

ClockDrift

關於參考時間和系統時間之間差異的資訊。此功能使用 Amazon 時間同步服務來測量時脈準確度，並提供容器的時脈錯誤。如需詳細資訊，請參閱 Amazon EC2 Linux 執行個體使用者指南中的設定 Linux 執行個體的時間。

ReferenceTime

時鐘精確度的基礎。Amazon ECS 透過 NTP 使用世界協調時間 (UTC) 的全球標準，例如 2021-09-07T16:57:44Z。

ClockErrorBound

時鐘誤差的測量值，定義為 UTC 的偏移。此誤差是參考時間和系統時間之間的毫秒差異。

ClockSynchronizationStatus

指出系統時間和參考時間之間最近的同步嘗試是否成功。

有效值為 SYNCHRONIZED 和 NOT_SYNCHRONIZED。

ExecutionStoppedAt

任務 DesiredStatus 移至 STOPPED 時的時間戳記。這會在基本容器移至 STOPPED 時發生。

Amazon ECS 任務中繼資料 v4 範例，適用於 Fargate 上的任務

以下範例顯示來自在 AWS Fargate上執行之 Amazon ECS 任務的任務中繼資料端點範例輸出。

在容器中，您可以使用 curl 後跟任務中繼資料端點來查詢端點，例如 curl ${ECS_CONTAINER_METADATA_URI_V4}/task。

容器中繼資料回應範例

查詢 ${ECS_CONTAINER_METADATA_URI_V4} 端點時，只會傳回關於容器本身的中繼資料。以下為範例輸出。

```
{
    "DockerId": "cd189a933e5849daa93386466019ab50-2495160603",
    "Name": "curl",
    "DockerName": "curl",
    "Image": "111122223333.dkr.ecr.us-west-2.amazonaws.com/curltest:latest",
    "ImageID":
 "sha256:25f3695bedfb454a50f12d127839a68ad3caf91e451c1da073db34c542c4d2cb",
    "Labels": {
        "com.amazonaws.ecs.cluster": "arn:aws:ecs:us-west-2:111122223333:cluster/
default",
        "com.amazonaws.ecs.container-name": "curl",
        "com.amazonaws.ecs.task-arn": "arn:aws:ecs:us-west-2:111122223333:task/default/
cd189a933e5849daa93386466019ab50",
        "com.amazonaws.ecs.task-definition-family": "curltest",
        "com.amazonaws.ecs.task-definition-version": "2"
    },
    "DesiredStatus": "RUNNING",
    "KnownStatus": "RUNNING",
    "Limits": {
        "CPU": 10,
        "Memory": 128
    },
    "CreatedAt": "2020-10-08T20:09:11.44527186Z",
    "StartedAt": "2020-10-08T20:09:11.44527186Z",
    "Type": "NORMAL",
    "Networks": [
        {
            "NetworkMode": "awsvpc",
            "IPv4Addresses": [
                "192.0.2.3"
            ],
            "AttachmentIndex": 0,
            "MACAddress": "0a:de:f6:10:51:e5",
            "IPv4SubnetCIDRBlock": "192.0.2.0/24",
            "DomainNameServers": [
```

```
                "192.0.2.2"
            ],
            "DomainNameSearchList": [
                "us-west-2.compute.internal"
            ],
            "PrivateDNSName": "ip-10-0-0-222.us-west-2.compute.internal",
            "SubnetGatewayIpv4Address": "192.0.2.0/24"
        }
    ],
    "ContainerARN": "arn:aws:ecs:us-west-2:111122223333:container/05966557-
f16c-49cb-9352-24b3a0dcd0e1",
    "LogOptions": {
        "awslogs-create-group": "true",
        "awslogs-group": "/ecs/containerlogs",
        "awslogs-region": "us-west-2",
        "awslogs-stream": "ecs/curl/cd189a933e5849daa93386466019ab50"
    },
    "LogDriver": "awslogs",
    "Snapshotter": "overlayfs"
}
```

Amazon ECS 任務中繼資料 v4 範例，適用於 Fargate 上的任務

查詢 ${ECS_CONTAINER_METADATA_URI_V4}/task 端點時，只會傳回容器所屬任務的中繼資料。以下為範例輸出。

```
{
  "Cluster": "arn:aws:ecs:us-east-1:123456789012:cluster/clusterName",
  "TaskARN": "arn:aws:ecs:us-east-1:123456789012:task/MyEmptyCluster/
bfa2636268144d039771334145e490c5",
  "Family": "sample-fargate",
  "Revision": "5",
  "DesiredStatus": "RUNNING",
  "KnownStatus": "RUNNING",
  "Limits": {
    "CPU": 0.25,
    "Memory": 512
  },
  "PullStartedAt": "2023-07-21T15:45:33.532811081Z",
  "PullStoppedAt": "2023-07-21T15:45:38.541068435Z",
  "AvailabilityZone": "us-east-1d",
  "Containers": [
    {
```

```
      "DockerId": "bfa2636268144d039771334145e490c5-1117626119",
      "Name": "curl-image",
      "DockerName": "curl-image",
      "Image": "curlimages/curl",
      "ImageID":
 "sha256:daf3f46a2639c1613b25e85c9ee4193af8a1d538f92483d67f9a3d7f21721827",
      "Labels": {
        "com.amazonaws.ecs.cluster": "arn:aws:ecs:us-east-1:123456789012:cluster/
MyEmptyCluster",
        "com.amazonaws.ecs.container-name": "curl-image",
        "com.amazonaws.ecs.task-arn": "arn:aws:ecs:us-east-1:123456789012:task/
MyEmptyCluster/bfa2636268144d039771334145e490c5",
        "com.amazonaws.ecs.task-definition-family": "sample-fargate",
        "com.amazonaws.ecs.task-definition-version": "5"
      },
      "DesiredStatus": "RUNNING",
      "KnownStatus": "RUNNING",
      "Limits": { "CPU": 128 },
      "CreatedAt": "2023-07-21T15:45:44.91368314Z",
      "StartedAt": "2023-07-21T15:45:44.91368314Z",
      "Type": "NORMAL",
      "Networks": [
        {
          "NetworkMode": "awsvpc",
          "IPv4Addresses": ["172.31.42.189"],
          "AttachmentIndex": 0,
          "MACAddress": "0e:98:9f:33:76:d3",
          "IPv4SubnetCIDRBlock": "172.31.32.0/20",
          "DomainNameServers": ["172.31.0.2"],
          "DomainNameSearchList": ["ec2.internal"],
          "PrivateDNSName": "ip-172-31-42-189.ec2.internal",
          "SubnetGatewayIpv4Address": "172.31.32.1/20"
        }
      ],
      "ContainerARN": "arn:aws:ecs:us-east-1:123456789012:container/MyEmptyCluster/
bfa2636268144d039771334145e490c5/da6cccf7-1178-400c-afdf-7536173ee209",
      "Snapshotter": "overlayfs"
    },
    {
      "DockerId": "bfa2636268144d039771334145e490c5-3681984407",
      "Name": "fargate-app",
      "DockerName": "fargate-app",
      "Image": "public.ecr.aws/docker/library/httpd:latest",
```

```
      "ImageID":
 "sha256:8059bdd0058510c03ae4c808de8c4fd2c1f3c1b6d9ea75487f1e5caa5ececa02",
      "Labels": {
        "com.amazonaws.ecs.cluster": "arn:aws:ecs:us-east-1:123456789012:cluster/
MyEmptyCluster",
        "com.amazonaws.ecs.container-name": "fargate-app",
        "com.amazonaws.ecs.task-arn": "arn:aws:ecs:us-east-1:123456789012:task/
MyEmptyCluster/bfa2636268144d039771334145e490c5",
        "com.amazonaws.ecs.task-definition-family": "sample-fargate",
        "com.amazonaws.ecs.task-definition-version": "5"
      },
      "DesiredStatus": "RUNNING",
      "KnownStatus": "RUNNING",
      "Limits": { "CPU": 2 },
      "CreatedAt": "2023-07-21T15:45:44.954460255Z",
      "StartedAt": "2023-07-21T15:45:44.954460255Z",
      "Type": "NORMAL",
      "Networks": [
        {
          "NetworkMode": "awsvpc",
          "IPv4Addresses": ["172.31.42.189"],
          "AttachmentIndex": 0,
          "MACAddress": "0e:98:9f:33:76:d3",
          "IPv4SubnetCIDRBlock": "172.31.32.0/20",
          "DomainNameServers": ["172.31.0.2"],
          "DomainNameSearchList": ["ec2.internal"],
          "PrivateDNSName": "ip-172-31-42-189.ec2.internal",
          "SubnetGatewayIpv4Address": "172.31.32.1/20"
        }
      ],
      "ContainerARN": "arn:aws:ecs:us-east-1:123456789012:container/MyEmptyCluster/
bfa2636268144d039771334145e490c5/f65b461d-aa09-4acb-a579-9785c0530cbc",
      "Snapshotter": "overlayfs"
    }
  ],
  "LaunchType": "FARGATE",
  "ClockDrift": {
    "ClockErrorBound": 0.446931,
    "ReferenceTimestamp": "2023-07-21T16:09:17Z",
    "ClockSynchronizationStatus": "SYNCHRONIZED"
  },
  "EphemeralStorageMetrics": {
    "Utilized": 261,
    "Reserved": 20496
```

```
  }
}
```

任務統計資訊回應範例

查詢 ${ECS_CONTAINER_METADATA_URI_V4}/task/stats 端點時，會傳回關於容器所屬任務的網路指標。以下為範例輸出。

```
{
  "3d1f891cded94dc795608466cce8ddcf-464223573": {
    "read": "2020-10-08T21:24:44.938937019Z",
    "preread": "2020-10-08T21:24:34.938633969Z",
    "pids_stats": {},
    "blkio_stats": {
      "io_service_bytes_recursive": [
        {
          "major": 202,
          "minor": 26368,
          "op": "Read",
          "value": 638976
        },
        {
          "major": 202,
          "minor": 26368,
          "op": "Write",
          "value": 0
        },
        {
          "major": 202,
          "minor": 26368,
          "op": "Sync",
          "value": 638976
        },
        {
          "major": 202,
          "minor": 26368,
          "op": "Async",
          "value": 0
        },
        {
          "major": 202,
          "minor": 26368,
          "op": "Total",
```

```
          "value": 638976
        }
      ],
      "io_serviced_recursive": [
        {
          "major": 202,
          "minor": 26368,
          "op": "Read",
          "value": 12
        },
        {
          "major": 202,
          "minor": 26368,
          "op": "Write",
          "value": 0
        },
        {
          "major": 202,
          "minor": 26368,
          "op": "Sync",
          "value": 12
        },
        {
          "major": 202,
          "minor": 26368,
          "op": "Async",
          "value": 0
        },
        {
          "major": 202,
          "minor": 26368,
          "op": "Total",
          "value": 12
        }
      ],
      "io_queue_recursive": [],
      "io_service_time_recursive": [],
      "io_wait_time_recursive": [],
      "io_merged_recursive": [],
      "io_time_recursive": [],
      "sectors_recursive": []
    },
    "num_procs": 0,
    "storage_stats": {},
```

```
    "cpu_stats": {
      "cpu_usage": {
        "total_usage": 1137691504,
        "percpu_usage": [
          696479228,
          441212276,
          0,
          0,
          0,
          0,
          0,
          0,
          0,
          0,
          0,
          0,
          0,
          0,
          0,
          0
        ],
        "usage_in_kernelmode": 80000000,
        "usage_in_usermode": 810000000
      },
      "system_cpu_usage": 9393210000000,
      "online_cpus": 2,
      "throttling_data": {
        "periods": 0,
        "throttled_periods": 0,
        "throttled_time": 0
      }
    },
    "precpu_stats": {
      "cpu_usage": {
        "total_usage": 1136624601,
        "percpu_usage": [
          695639662,
          440984939,
          0,
          0,
          0,
          0,
          0,
          0,
          0,
          0,
```

```
          0,
          0,
          0,
          0,
          0,
          0
        ],
        "usage_in_kernelmode": 80000000,
        "usage_in_usermode": 810000000
      },
      "system_cpu_usage": 9373330000000,
      "online_cpus": 2,
      "throttling_data": {
        "periods": 0,
        "throttled_periods": 0,
        "throttled_time": 0
      }
    },
    "memory_stats": {
      "usage": 6504448,
      "max_usage": 8458240,
      "stats": {
        "active_anon": 1675264,
        "active_file": 557056,
        "cache": 651264,
        "dirty": 0,
        "hierarchical_memory_limit": 536870912,
        "hierarchical_memsw_limit": 9223372036854772000,
        "inactive_anon": 0,
        "inactive_file": 3088384,
        "mapped_file": 430080,
        "pgfault": 11034,
        "pgmajfault": 5,
        "pgpgin": 8436,
        "pgpgout": 7137,
        "rss": 4669440,
        "rss_huge": 0,
        "total_active_anon": 1675264,
        "total_active_file": 557056,
        "total_cache": 651264,
        "total_dirty": 0,
        "total_inactive_anon": 0,
        "total_inactive_file": 3088384,
        "total_mapped_file": 430080,
```

```
        "total_pgfault": 11034,
        "total_pgmajfault": 5,
        "total_pgpgin": 8436,
        "total_pgpgout": 7137,
        "total_rss": 4669440,
        "total_rss_huge": 0,
        "total_unevictable": 0,
        "total_writeback": 0,
        "unevictable": 0,
        "writeback": 0
      },
      "limit": 9223372036854772000
    },
    "name": "curltest",
    "id": "3d1f891cded94dc795608466cce8ddcf-464223573",
    "networks": {
      "eth1": {
        "rx_bytes": 2398415937,
        "rx_packets": 1898631,
        "rx_errors": 0,
        "rx_dropped": 0,
        "tx_bytes": 1259037719,
        "tx_packets": 428002,
        "tx_errors": 0,
        "tx_dropped": 0
      }
    },
    "network_rate_stats": {
      "rx_bytes_per_sec": 43.298687872232854,
      "tx_bytes_per_sec": 215.39347269466413
    }
  }
}
```


適用於 Fargate 上任務的 Amazon ECS 任務中繼資料端點版本 3

> **Important**
>
> 不再積極維護任務中繼資料第 3 版端點。建議您更新任務中繼資料第 4 版端點，以取得最新的中繼資料端點資訊。如需詳細資訊，請參閱 the section called “Fargate 上任務的任務中繼資料端點第 4 版”。

從 Fargate 平台版本 1.1.0 開始，名為 ECS_CONTAINER_METADATA_URI 的環境變數會插入任務中的每個容器。當您查詢任務中繼資料第 3 版端點時，有各種任務中繼資料和 Docker 統計資訊可供任務使用。

對於託管於 Fargate 的 Amazon ECS 任務 (使用平台版本 1.1.0 或更新版本)，預設會啟用任務中繼資料端點功能。如需詳細資訊，請參閱 適用於 Amazon ECS 的 Fargate Linux 平台版本。

Fargate 上任務的任務中繼資料端點路徑

容器可使用下列 API 端點：

`${ECS_CONTAINER_METADATA_URI}`

此路徑傳回容器的 JSON 中繼資料。

`${ECS_CONTAINER_METADATA_URI}/task`

此路徑傳回任務的中繼資料 JSON，包括與任務相關聯之所有容器的容器 ID 和名稱清單。如需此端點之回應的詳細資訊，請參閱「Amazon ECS 任務元數據 v3 對 Fargate 任務的 JSON 響應」。

`${ECS_CONTAINER_METADATA_URI}/stats`

此路徑傳回特定 Docker 容器的 Docker 統計資訊 JSON。有關每個返回統計信息的更多信息，請參閱 Docker API 文檔ContainerStats中的。

`${ECS_CONTAINER_METADATA_URI}/task/stats`

此路徑傳回與任務相關聯之所有容器的 Docker 統計資訊 JSON。有關每個返回統計信息的更多信息，請參閱 Docker API 文檔ContainerStats中的。

Amazon ECS 任務元數據 v3 對 Fargate 任務的 JSON 響應

任務中繼資料端點 (`${ECS_CONTAINER_METADATA_URI}/task`) JSON 回應會傳回下列資訊。

`Cluster`

任務所屬 Amazon ECS 叢集的 Amazon Resource Name (ARN) 或簡短名稱。

`TaskARN`

容器所屬任務的完整 Amazon Resource Name (ARN)。

`Family`

任務的 Amazon ECS 任務定義系列。

`Revision`

任務的 Amazon ECS 任務定義修訂。

`DesiredStatus`

Amazon ECS 中任務的所需狀態。

`KnownStatus`

Amazon ECS 中任務的已知狀態。

`Limits`

在任務層級指定的資源限制，例如 CPU (以 vCPU 表示) 和記憶體。如果未定義資源限制，則會省略此參數。

`PullStartedAt`

第一個容器映像提取的開始時間戳記。

`PullStoppedAt`

最後一個容器映像提取的完成時間戳記。

`AvailabilityZone`

任務所在的可用區域。

> **Note**
>
> 可用區域中繼資料僅適用於使用平台第 1.4 版或更新版本 (Linux) 或 1.0.0 或更新版本 (Windows) 的 Fargate 任務。

`Containers`

與任務相關聯之每個容器的容器中繼資料清單。

`DockerId`

容器的 Docker ID。

`Name`

任務定義中指定的容器名稱。

`DockerName`

提供給 Docker 的容器名稱。Amazon ECS 容器代理程式會產生容器的唯一名稱，以避免在單一執行個體上執行相同任務定義的多個複本時，發生名稱衝突。

`Image`

容器的映像。

`ImageID`

映像的 SHA-256 摘要。

`Ports`

向容器開放的任何連接埠。如果未開放連接埠，則會省略此參數。

`Labels`

任何套用至容器的標籤。如果未套用標籤，則會省略此參數。

`DesiredStatus`

Amazon ECS 中容器的所需狀態。

`KnownStatus`

Amazon ECS 中容器的已知狀態。

`ExitCode`

容器的結束代碼。如果容器尚未結束，則會省略此參數。

`Limits`

在容器層級指定的資源限制，例如 CPU (以 CPU 單位表示) 和記憶體。如果未定義資源限制，則會省略此參數。

`CreatedAt`

容器的建立時間戳記。如果尚未建立容器，則會省略此參數。

`StartedAt`

容器的啟動時間戳記。如果尚未啟動容器，則會省略此參數。

`FinishedAt`

容器的停止時間戳記。如果尚未停止容器，則會省略此參數。

Type

容器的類型。任務定義中指定的容器類型為 NORMAL。您可以忽略其他容器類型，這些是 Amazon ECS 容器代理程式用來佈建內部任務資源的容器類型。

Networks

容器的網路資訊，例如網路模式和 IP 地址。如果未定義網路資訊，則會省略此參數。

ClockDrift

關於參考時間和系統時間之間差異的資訊。這適用於 Linux 作業系統。此功能使用 Amazon 時間同步服務來測量時脈準確度，並提供容器的時脈錯誤。如需詳細資訊，請參閱 Amazon EC2 Linux 執行個體使用者指南中的設定 Linux 執行個體的時間。

ReferenceTime

時鐘精確度的基礎。Amazon ECS 透過 NTP 使用世界協調時間 (UTC) 的全球標準，例如 2021-09-07T16:57:44Z。

ClockErrorBound

時鐘誤差的測量值，定義為 UTC 的偏移。此誤差是參考時間和系統時間之間的毫秒差異。

ClockSynchronizationStatus

指出系統時間和參考時間之間最近的同步嘗試是否成功。

有效值為 SYNCHRONIZED 和 NOT_SYNCHRONIZED。

ExecutionStoppedAt

任務 DesiredStatus 移至 STOPPED 時的時間戳記。這會在基本容器移至 STOPPED 時發生。

Amazon ECS 任務中繼資料 v3 範例，適用於 Fargate 上的任務

下列為單一容器任務的 JSON 回應。

```
{
  "Cluster": "default",
  "TaskARN": "arn:aws:ecs:us-east-2:012345678910:task/9781c248-0edd-4cdb-9a93-
f63cb662a5d3",
  "Family": "nginx",
  "Revision": "5",
  "DesiredStatus": "RUNNING",
  "KnownStatus": "RUNNING",
  "Containers": [
```

```
    {
      "DockerId": "731a0d6a3b4210e2448339bc7015aaa79bfe4fa256384f4102db86ef94cbbc4c",
      "Name": "~internal~ecs~pause",
      "DockerName": "ecs-nginx-5-internalecspause-acc699c0cbf2d6d11700",
      "Image": "amazon/amazon-ecs-pause:0.1.0",
      "ImageID": "",
      "Labels": {
        "com.amazonaws.ecs.cluster": "default",
        "com.amazonaws.ecs.container-name": "~internal~ecs~pause",
        "com.amazonaws.ecs.task-arn": "arn:aws:ecs:us-
east-2:012345678910:task/9781c248-0edd-4cdb-9a93-f63cb662a5d3",
        "com.amazonaws.ecs.task-definition-family": "nginx",
        "com.amazonaws.ecs.task-definition-version": "5"
      },
      "DesiredStatus": "RESOURCES_PROVISIONED",
      "KnownStatus": "RESOURCES_PROVISIONED",
      "Limits": {
        "CPU": 0,
        "Memory": 0
      },
      "CreatedAt": "2018-02-01T20:55:08.366329616Z",
      "StartedAt": "2018-02-01T20:55:09.058354915Z",
      "Type": "CNI_PAUSE",
      "Networks": [
        {
          "NetworkMode": "awsvpc",
          "IPv4Addresses": [
            "10.0.2.106"
          ]
        }
      ]
    },
    {
      "DockerId": "43481a6ce4842eec8fe72fc28500c6b52edcc0917f105b83379f88cac1ff3946",
      "Name": "nginx-curl",
      "DockerName": "ecs-nginx-5-nginx-curl-ccccb9f49db0dfe0d901",
      "Image": "nrdlngr/nginx-curl",
      "ImageID":
 "sha256:2e00ae64383cfc865ba0a2ba37f61b50a120d2d9378559dcd458dc0de47bc165",
      "Labels": {
        "com.amazonaws.ecs.cluster": "default",
        "com.amazonaws.ecs.container-name": "nginx-curl",
        "com.amazonaws.ecs.task-arn": "arn:aws:ecs:us-
east-2:012345678910:task/9781c248-0edd-4cdb-9a93-f63cb662a5d3",
```

```
        "com.amazonaws.ecs.task-definition-family": "nginx",
        "com.amazonaws.ecs.task-definition-version": "5"
      },
      "DesiredStatus": "RUNNING",
      "KnownStatus": "RUNNING",
      "Limits": {
        "CPU": 512,
        "Memory": 512
      },
      "CreatedAt": "2018-02-01T20:55:10.554941919Z",
      "StartedAt": "2018-02-01T20:55:11.064236631Z",
      "Type": "NORMAL",
      "Networks": [
        {
          "NetworkMode": "awsvpc",
          "IPv4Addresses": [
            "10.0.2.106"
          ]
        }
      ]
    }
  ],
  "PullStartedAt": "2018-02-01T20:55:09.372495529Z",
  "PullStoppedAt": "2018-02-01T20:55:10.552018345Z",
  "AvailabilityZone": "us-east-2b"
}
```

Amazon ECS 容器內省

Amazon ECS 容器代理程式提供一種 API 操作，它收集執行代理程式的容器執行個體的詳細資訊，以及在該執行個體上執行之關聯任務的詳細資訊。您可以在容器執行個體內使用 curl 命令來查詢 Amazon ECS 容器代理程式 (連接埠 51678)，並傳回容器執行個體的中繼資料或任務資訊。

> **Important**
>
> 您的容器執行個體必須擁有允許存取 Amazon ECS 以擷取中繼資料的 IAM 角色。如需詳細資訊，請參閱 Amazon ECS 容器執行個體 IAM 角色。

若要檢視容器執行個體中繼資料，請透過 SSH 登入您的容器執行個體，並執行以下命令。中繼資料包含容器執行個體 ID、註冊容器執行個體的 Amazon ECS 叢集，以及 Amazon ECS 容器代理程式版本資訊。

```
curl -s http://localhost:51678/v1/metadata | python3 -mjson.tool
```

輸出：

```
{
    "Cluster": "cluster_name",
    "ContainerInstanceArn": "arn:aws:ecs:region:aws_account_id:container-
instance/cluster_name/container_instance_id",
    "Version": "Amazon ECS Agent - v1.30.0 (02ff320c)"
}
```

若要檢視所有在容器執行個體上執行之任務的資訊，請透過 SSH 登入您的容器執行個體，然後執行以下命令：

```
curl http://localhost:51678/v1/tasks
```

輸出：

```
{
    "Tasks": [
        {
            "Arn": "arn:aws:ecs:us-west-2:012345678910:task/default/example5-58ff-46c9-
ae05-543f8example",
            "DesiredStatus": "RUNNING",
            "KnownStatus": "RUNNING",
            "Family": "hello_world",
            "Version": "8",
            "Containers": [
                {
          "DockerId":
 "9581a69a761a557fbfce1d0f6745e4af5b9dbfb86b6b2c5c4df156f1a5932ff1",
          "DockerName": "ecs-hello_world-8-mysql-fcae8ac8f9f1d89d8301",
          "Name": "mysql",
          "CreatedAt": "2023-10-08T20:09:11.445271867Z",
          "StartedAt": "2023-10-08T20:09:11.445271867Z",
          "ImageID":
 "sha256:2ae34abc2ed0a22e280d17e13f9c01aaf725688b09b7a1525d1a2750e2c0d1de"
        },
        {
          "DockerId":
 "bf25c5c5b2d4dba68846c7236e75b6915e1e778d31611e3c6a06831e39814a15",
```

```
            "DockerName": "ecs-hello_world-8-wordpress-e8bfddf9b488dff36c00",
            "Name": "wordpress"
        }
    ]
}
    ]
}
```

您可以檢視在該容器執行個體上執行之特定任務的資訊。若要指定特定任務或容器，請將以下其中一項附加到請求：

- 任務 ARN (?taskarn=*task_arn*)
- 容器的 Docker ID (?dockerid=*docker_id*)

若要使用容器的 Docker ID 取得任務資訊，請透過 SSH 登入您的容器執行個體，然後執行以下命令。

> **Note**
>
> 1.14.2 版之前的 Amazon ECS 容器代理程式需要完整的 Docker 容器 ID 以用於自我檢查 API，而非 docker ps 顯示的簡短版本。您可以藉由在容器執行個體上執行 docker ps --no-trunc 命令，取得容器完整的 Docker ID。

```
curl http://localhost:51678/v1/tasks?dockerid=79c796ed2a7f
```

輸出：

```
{
    "Arn": "arn:aws:ecs:us-west-2:012345678910:task/default/e01d58a8-151b-40e8-
bc01-22647b9ecfec",
    "Containers": [
        {
            "DockerId":
 "79c796ed2a7f864f485c76f83f3165488097279d296a7c05bd5201a1c69b2920",
            "DockerName": "ecs-nginx-efs-2-nginx-9ac0808dd0afa495f001",
            "Name": "nginx",
            "CreatedAt": "2023-10-08T20:09:11.44527186Z",
            "StartedAt": "2023-10-08T20:09:11.44527186Z",
```

```
            "ImageID":
 "sha256:2ae34abc2ed0a22e280d17e13f9c01aaf725688b09b7a1525d1a2750e2c0d1de"
        }
    ],
    "DesiredStatus": "RUNNING",
    "Family": "nginx-efs",
    "KnownStatus": "RUNNING",
    "Version": "2"
}
```

使用執行階段監控識別未經授

Amazon GuardDuty 是一種威脅偵測服務，可協助保護您的帳戶、容器、工作負載和 AWS 環境中的資料。使用機器學習 (ML) 模型，以及異常和威脅偵測功能， GuardDuty 持續監控不同的記錄檔來源和執行階段活動，以識別環境中潛在的安全風險和惡意活動並排定優先順序。

中的運行時監控通過持續監控 AWS 日誌和聯網活動以識別惡意或未經授權的行為來 GuardDuty 保護在 Fargate 和 EC2 容器實例上運行的工作負載。執行階段監控使用輕量型、完全受控的 GuardDuty 安全代理程式來分析主機上的行為，例如檔案存取、程序執行和網路連線。這涵蓋的問題包括權限提升、使用暴露的登入資料，或與惡意 IP 地址、網域通訊，以及 Amazon EC2 執行個體和容器工作負載上存在惡意軟體的情況。如需詳細資訊，請參閱GuardDuty 使用指南中的GuardDuty執行階段監視。

您的安全管理員會針對中的單一或多個帳戶啟 AWS Organizations 用執行階段監視 GuardDuty。他們還會選取是否在您使用 Fargate 時 GuardDuty 自動部署 GuardDuty 安全代理程式。您的所有叢集都會自動受到保護，並代表您 GuardDuty管理安全代理程式。

在下列情況下，您也可以手動設定 GuardDuty 安全代理程式：

- 您使用 EC2 容器執行個體
- 您需要精細控制，才能在叢集層級啟用執行階段監控

若要使用執行階段監控，您必須設定受保護的叢集，以及在 EC2 容器執行個體上安裝和管理安 GuardDuty 全代理程式。

運行時監控如何與 Amazon ECS 一起工作

執行階段監控使用輕量級 GuardDuty 安全代理程式來監控 Amazon ECS 工作負載活動，瞭解應用程式請求的方式、取得存取權和使用基礎系統資源。

對於 Fargate 任務， GuardDuty 安全代理程式會為每項工作作為附屬容器執行。

對於 EC2 容器執行個體， GuardDuty 安全代理程式會在執行個體上以程序的形式執行。

GuardDuty 安全代理程式會從下列資源收集資料，然後將資料傳送 GuardDuty 至處理。您可以在 GuardDuty 主控台中檢視發現項目。您也可以將它們傳送給其他 AWS 服務 (例如 AWS Security Hub，或協力廠商的安全性廠商)，以進行彙總和補救。如需有關如何檢視和管理發現項目的資訊，請參閱 Amazon GuardDuty 使用者指南中的管理 Amazon GuardDuty 發現項目。

- 來自下列 Amazon ECS API 呼叫的回應：
  - DescribeClusters

    當您使用此`--include TAGS`選項時，回應參數包括「執行階段監視」標籤 (設定標籤時)。
  - DescribeTasks

    對於 Fargate 發射類型，響應參數包括 GuardDuty 側車容器。
  - ListAccountSettings

    回應參數包括「執行階段監視」帳戶設定，由您的安全性管理員設定。
- 容器代理程式內部檢查資料。如需詳細資訊，請參閱 Amazon ECS 容器內省。
- 啟動類型的工作中繼資料端點：
  - Amazon ECS 任務中繼資料端點版本 4
  - 適用於 Fargate 上任務的 Amazon ECS 任務中繼資料端點第 4 版

考量事項

使用執行時期監視時，請考量下列事項

- 運行時監視具有與之相關的成本。如需詳細資訊，請參閱 Amazon GuardDuty 定價。
- Amazon ECS 無處不支援執行階段監控。
- Windows 作業系統不支援執行階段監視。
- 在 Fargate 上使用 Amazon ECS Exec 時，您必須指定容器名稱，因為 GuardDuty 安全代理程式會以附屬容器的形式執行。
- 您無法在 GuardDuty 安全代理程式附屬容器上使用 Amazon ECS Exec。
- 在叢集層級控制執行階段監控的 IAM 使用者必須具有適當的 IAM 許可才能進行標記。如需詳細資訊，請參閱 IAM 教學課程：根據 IAM 使用者指南中的標籤定義存取 AWS 資源的許可。

- Fargate 任務必須使用任務執行角色。此角色授與任務權限，以便代表您擷取、更新和管理儲存在 Amazon ECR 私有儲存庫中的 GuardDuty 安全代理程式。

資源使用率

您新增至叢集的標記會計入叢集標記配額。

GuardDuty 代理程式並行容器不會計入每個工作定義配額的容器中。

與大多數安全軟件一樣， GuardDuty. 如需有關 Fargate 記憶體限制的資訊，請參閱《GuardDuty 使用指南》中的 CPU 和記憶體限制。如需 Amazon EC2 記憶體限制的相關資訊，請參閱 GuardDuty 代理程式的 CPU 和記憶體限制。

Amazon ECS Fargate 端工作負載的執行時間監控

如果您使用 EC2 容器執行個體，則必須手動設定執行階段監控。如需詳細資訊，請參閱 Amazon ECS 上 EC2 工作負載的執行時間監控。

您可以在容器執行個體上 GuardDuty 管理安全代理程式。此選項僅適用於 Fargate。此選項 (GuardDuty 代理程式管理) 可在 GuardDuty

使用 GuardDuty 代理程式管理時，請 GuardDuty 執行下列作業：

- 為裝載叢集的每 GuardDuty 個 VPC 建立 VPC 端點。
- 在所有新的獨立 Fargate 任務和新服務部署中擷取並安裝最新的安 GuardDuty 全代理程式作為附屬容器。

  當您第一次啟動服務或使用強制新部署選項更新現有服務時，就會發生新的服務部署。

開啟 Amazon ECS 的執行時間監控

您可以配置 GuardDuty 為自動管理所有 Fargate 叢集的安全代理。

以下是使用「執行時期監視」的先決條件：

- 對於 Linux，Fargate 平台版本必須是`1.4.0`或更高版本。
- Amazon ECS 的 IAM 角色和許可：
  - Fargate 任務必須使用任務執行角色。此角色會授與工作代表您擷取、更新和管理 GuardDuty 安全代理程式的權限。如需更多資訊，請參閱Amazon ECS 任務執行 IAM 角色。

  - 您可以控制具有預先定義標記的叢集的執行階段監視。如果您的存取政策根據標籤限制存取，則必須向 IAM 使用者授予明確許可，以標記叢集。如需詳細資訊，請參閱 IAM 教學課程：根據 IAM 使用者指南中的標籤定義存取 AWS 資源的許可。
- 連接到 Amazon ECR 存儲庫：

  GuardDuty 安全代理程式會儲存在 Amazon ECR 儲存庫中。每個獨立和服務任務都必須具有存取存放庫的權限。您可以使用下列其中一個選項：
  - 對於公有子網路中的任務，您可以使用公有 IP 地址來執行任務，也可以在執行任務的子網路中為 Amazon ECR 建立 VPC 端點。如需詳細資訊，請參閱 《Amazon Elastic Container Registry 使用者指南》中的 Amazon ECR 介面 VPC 端點 (AWS PrivateLink)。
  - 對於私有子網路中的任務，您可以使用網路位址轉譯 (NAT) 閘道，或在執行任務的子網路中為 Amazon ECR 建立 VPC 端點。

    如需詳細資訊，請參閱使用私有子網路和 NAT 閘道。
- 您必須具有的`AWSServiceRoleForAmazonGuardDuty`角色 GuardDuty。如需詳細資訊，請參閱 Amazon GuardDuty 使用者指南 GuardDuty中的服務連結角色許可。
- 您要使用執行階段監視保護的任何檔案都必須可由 root 使用者存取。如果您手動變更檔案的權限，則必須將其設定為`755`。

以下是在 EC2 容器執行個體上使用執行階段監控的先決條件：

- 您必須使用 Amazon ECS-`20230929` AMI 的版本或更高版本。
- 您必須在容器執行個體上執行 Amazon ECS 代理程式至版本`1.77`或更新版本。
- 您必須使用核心版本`5.10`或更新版本。
- 如需有關支援 Linux 作業系統和架構的詳細資訊，請參閱 GuardDuty 執行階段監控支援哪些作業模型和工作負載。
- 您可以使用 Systems Manager 來管理容器執行個體。如需詳細資訊，請參閱AWS Systems Manager Session Manager 使用者指南中的為 EC2 執行個體設定 Systems Manager。

您在中啟用執行階段監視 GuardDuty。如需如何啟用此功能的詳細資訊，請參閱 Amazon GuardDuty 使用者指南中的啟用執行階段監控。

將執行時間監控新增至現有的 Amazon ECS Fargate 任務

當您開啟執行階段監視時，叢集中的所有新獨立工作和新服務部署都會自動受到保護。為了保持不變性約束，現有任務不受影響。

以下是使用「執行時期監視」的先決條件：

- 對於 Linux，Fargate 平台版本必須是`1.4.0`或更高版本。
- Amazon ECS 的 IAM 角色和許可：
  - Fargate 任務必須使用任務執行角色。此角色會授與工作代表您擷取、更新和管理 GuardDuty 安全代理程式的權限。如需更多資訊，請參閱Amazon ECS 任務執行 IAM 角色。
  - 您可以控制具有預先定義標記的叢集的執行階段監視。如果您的存取政策根據標籤限制存取，則必須向 IAM 使用者授予明確許可，以標記叢集。如需詳細資訊，請參閱 IAM 教學課程：根據 IAM 使用者指南中的標籤定義存取 AWS 資源的許可。
- 連接到 Amazon ECR 存儲庫：

  GuardDuty 安全代理程式會儲存在 Amazon ECR 儲存庫中。每個獨立和服務任務都必須具有存取存放庫的權限。您可以使用下列其中一個選項：
  - 對於公有子網路中的任務，您可以使用公有 IP 地址來執行任務，也可以在執行任務的子網路中為 Amazon ECR 建立 VPC 端點。如需詳細資訊，請參閱 《Amazon Elastic Container Registry 使用者指南》中的 Amazon ECR 介面 VPC 端點 (AWS PrivateLink)。
  - 對於私有子網路中的任務，您可以使用網路位址轉譯 (NAT) 閘道，或在執行任務的子網路中為 Amazon ECR 建立 VPC 端點。

    如需詳細資訊，請參閱使用私有子網路和 NAT 閘道。
- 您必須具有的`AWSServiceRoleForAmazonGuardDuty`角色 GuardDuty。如需詳細資訊，請參閱 Amazon GuardDuty 使用者指南 GuardDuty中的服務連結角色許可。
- 您要使用執行階段監視保護的任何檔案都必須可由 root 使用者存取。如果您手動變更檔案的權限，則必須將其設定為`755`。

以下是在 EC2 容器執行個體上使用執行階段監控的先決條件：

- 您必須使用 Amazon ECS-`20230929` AMI 的版本或更高版本。
- 您必須在容器執行個體上執行 Amazon ECS 代理程式至版本`1.77`或更新版本。
- 您必須使用核心版本`5.10`或更新版本。
- 如需有關支援 Linux 作業系統和架構的詳細資訊，請參閱 GuardDuty 執行階段監控支援哪些作業模型和工作負載。
- 您可以使用 Systems Manager 來管理容器執行個體。如需詳細資訊，請參閱AWS Systems Manager Session Manager 使用者指南中的為 EC2 執行個體設定 Systems Manager。

若要立即保護工作，您必須執行下列其中一個動作：

- 對於獨立工作，請停止工作，然後啟動它們。如需詳細資訊，請參閱 停止 Amazon ECS 任務 和 以 Amazon ECS 任務的形式執行應用程式
- 對於屬於服務一部分的任務，請使用「強制新部署」選項更新服務。如需詳細資訊，請參閱 使用主控台更新 Amazon ECS 服務。

從 Amazon ECS 叢集移除執行階段監控

您可能想要從保護中排除某些叢集，例如用於測試的叢集。這會導 GuardDuty 致對叢集中的資源執行下列作業：

- 不再將 GuardDuty 安全代理程式部署到新的獨立 Fargate 工作或新服務部署。

  為了保持不變性限制，啟用「執行時期監視」的現有工作和部署不會受到影響。
- 停止計費，不再接受任務的運行時事件。

執行下列步驟，從叢集移除「執行階段監視」。

1. 使用 Amazon ECS 主控台或 AWS CLI 將叢集上的`GuardDutyManaged`標籤金鑰設定為`false`。如需詳細資訊，請參閱使用 CLI 或 API 更新叢集或使用標籤。對標籤使用下列值。

   > **Note**
   >
   > 「鍵」和「值」區分大小寫，且必須完全符合字串。

   鍵 =`GuardDutyManaged`，值 = `false`
2. 刪除叢集的 GuardDuty VPC 端點。如需有關如何刪除 VPC 端點的詳細資訊，請參閱AWS PrivateLink 使用手冊中的刪除介面端點。

從帳戶移除 Amazon ECS 的執行階段監控

如果您不想再使用執行階段監視，請停用中的功能 GuardDuty。如需如何停用此功能的詳細資訊，請參閱 Amazon GuardDuty 使用者指南中的啟用執行階段監控。

GuardDuty 執行以下操作：

- 針對裝載叢集的每 GuardDuty 個 VPC 刪除 VPC 端點。
- 不再將 GuardDuty 安全代理程式部署到新的獨立 Fargate 工作或新服務部署。

  為了保留不變性限制，現有的工作和部署在停止、複寫或調整規模之前不會受到影響。
- 停止計費，不再接受工作的執行時間事件。

Amazon ECS 上 EC2 工作負載的執行時間監控

當您針對容量使用 EC2 執行個體，或需要在 Fargate 上叢集層級對執行階段監控進行精細控制時，請使用此選項。

您可以新增預先定義的標記來佈建叢集以進行執行階段監視。

對於 EC2 容器執行個體，您可以下載、安裝和管理 GuardDuty 安全代理程式。

對於 Fargate，請代表您 GuardDuty 管理安全代理程式。

開啟 Amazon ECS 的執行時間監控

您可以針對 EC2 執行個體的叢集開啟執行階段監控，或者當您需要在 Fargate 上的叢集層級對執行階段監控進行精細控制時。

以下是使用「執行時期監視」的先決條件：

- 對於 Linux，Fargate 平台版本必須是`1.4.0`或更高版本。
- Amazon ECS 的 IAM 角色和許可：
  - Fargate 任務必須使用任務執行角色。此角色會授與工作代表您擷取、更新和管理 GuardDuty 安全代理程式的權限。如需更多資訊，請參閱Amazon ECS 任務執行 IAM 角色。
  - 您可以控制具有預先定義標記的叢集的執行階段監視。如果您的存取政策根據標籤限制存取，則必須向 IAM 使用者授予明確許可，以標記叢集。如需詳細資訊，請參閱 IAM 教學課程：根據 IAM 使用者指南中的標籤定義存取 AWS 資源的許可。
- 連接到 Amazon ECR 存儲庫：

  GuardDuty 安全代理程式會儲存在 Amazon ECR 儲存庫中。每個獨立和服務任務都必須具有存取存放庫的權限。您可以使用下列其中一個選項：
  - 對於公有子網路中的任務，您可以使用公有 IP 地址來執行任務，也可以在執行任務的子網路中為 Amazon ECR 建立 VPC 端點。如需詳細資訊，請參閱 《Amazon Elastic Container Registry 使用者指南》中的 Amazon ECR 介面 VPC 端點 (AWS PrivateLink)。

- 對於私有子網路中的任務，您可以使用網路位址轉譯 (NAT) 閘道，或在執行任務的子網路中為 Amazon ECR 建立 VPC 端點。

  如需詳細資訊，請參閱使用私有子網路和 NAT 閘道。
- 您必須具有的AWSServiceRoleForAmazonGuardDuty角色 GuardDuty。如需詳細資訊，請參閱 Amazon GuardDuty 使用者指南 GuardDuty中的服務連結角色許可。
- 您要使用執行階段監視保護的任何檔案都必須可由 root 使用者存取。如果您手動變更檔案的權限，則必須將其設定為755。

以下是在 EC2 容器執行個體上使用執行階段監控的先決條件：

- 您必須使用 Amazon ECS-20230929 AMI 的版本或更高版本。
- 您必須在容器執行個體上執行 Amazon ECS 代理程式至版本1.77或更新版本。
- 您必須使用核心版本5.10或更新版本。
- 如需有關支援 Linux 作業系統和架構的詳細資訊，請參閱 GuardDuty 執行階段監控支援哪些作業模型和工作負載。
- 您可以使用 Systems Manager 來管理容器執行個體。如需詳細資訊，請參閱AWS Systems Manager Session Manager 使用者指南中的為 EC2 執行個體設定 Systems Manager。

您可以在中開啟執行階段監視 GuardDuty。如需如何啟用此功能的詳細資訊，請參閱 Amazon GuardDuty 使用者指南中的啟用執行階段監控。

新增 Amazon ECS 叢集的執行時間監控

為叢集設定執行階段監控，然後在 EC2 容器執行個體上安裝安 GuardDuty 全代理程式。

以下是使用「執行時期監視」的先決條件：

- 對於 Linux，Fargate 平台版本必須是1.4.0或更高版本。
- Amazon ECS 的 IAM 角色和許可：
  - Fargate 任務必須使用任務執行角色。此角色會授與工作代表您擷取、更新和管理 GuardDuty 安全代理程式的權限。如需更多資訊，請參閱Amazon ECS 任務執行 IAM 角色。
  - 您可以控制具有預先定義標記的叢集的執行階段監視。如果您的存取政策根據標籤限制存取，則必須向 IAM 使用者授予明確許可，以標記叢集。如需詳細資訊，請參閱 IAM 教學課程：根據 IAM 使用者指南中的標籤定義存取 AWS 資源的許可。
- 連接到 Amazon ECR 存儲庫：

GuardDuty 安全代理程式會儲存在 Amazon ECR 儲存庫中。每個獨立和服務任務都必須具有存取存放庫的權限。您可以使用下列其中一個選項：

- 對於公有子網路中的任務，您可以使用公有 IP 地址來執行任務，也可以在執行任務的子網路中為 Amazon ECR 建立 VPC 端點。如需詳細資訊，請參閱 《Amazon Elastic Container Registry 使用者指南》中的 Amazon ECR 介面 VPC 端點 (AWS PrivateLink)。
- 對於私有子網路中的任務，您可以使用網路位址轉譯 (NAT) 閘道，或在執行任務的子網路中為 Amazon ECR 建立 VPC 端點。

  如需詳細資訊，請參閱使用私有子網路和 NAT 閘道。

- 您必須具有的`AWSServiceRoleForAmazonGuardDuty`角色 GuardDuty。如需詳細資訊，請參閱 Amazon GuardDuty 使用者指南 GuardDuty中的服務連結角色許可。
- 您要使用執行階段監視保護的任何檔案都必須可由 root 使用者存取。如果您手動變更檔案的權限，則必須將其設定為`755`。

以下是在 EC2 容器執行個體上使用執行階段監控的先決條件：

- 您必須使用 Amazon ECS-`20230929` AMI 的版本或更高版本。
- 您必須在容器執行個體上執行 Amazon ECS 代理程式至版本`1.77`或更新版本。
- 您必須使用核心版本`5.10`或更新版本。
- 如需支援 Linux 作業系統和架構的相關資訊，請參閱 GuardDuty 執行階段監控支援哪些作業模型和工作負載。
- 您可以使用 Systems Manager 來管理容器執行個體。如需詳細資訊，請參閱AWS Systems Manager Session Manager 使用者指南中的為 EC2 執行個體設定 Systems Manager。

執行下列作業，將「執行時期監視」新增至叢集。

1. 為 GuardDuty 每個叢集 VPC 建立 VPC 端點。如需詳細資訊，請參閱GuardDuty 使用者指南中的手動建立 Amazon VPC 端點。
2. 設定 EC2 容器執行個體。

   a. 將 Amazon ECS 代理程式更新為叢集中 EC2 容器執行個體的版本`1.77`或更新版本。如需更多資訊，請參閱更新 Amazon ECS 容器代理程式。

   b. 在叢集中的 EC2 容器執行個體上安裝安 GuardDuty 全代理程式。如需詳細資訊，請參閱GuardDuty 使用者指南中的手動管理 Amazon EC2 執行個體上的安全代理程式。

所有新的和現有的任務以及部署都會立即受到保護，因為 GuardDuty 安全代理程式會在 EC2 容器執行個體上以程序的形式執行。

3. 使用 Amazon ECS 主控台或 AWS CLI 將叢集上的`GuardDutyManaged`標籤金鑰設定為`true`。如需詳細資訊，請參閱使用 CLI 或 API 更新叢集或使用標籤。對標籤使用下列值。

> **Note**
>
> 「鍵」和「值」區分大小寫，且必須完全符合字串。

鍵 =`GuardDutyManaged`，值 = `true`

將執行時間監控新增至現有的 Amazon ECS 任務

當您開啟執行階段監視時，叢集中的所有新獨立工作和新服務部署都會自動受到保護。為了保持不變性約束，現有任務不受影響。

以下是使用「執行時期監視」的先決條件：

- 對於 Linux，Fargate 平台版本必須是`1.4.0`或更高版本。
- Amazon ECS 的 IAM 角色和許可：
  - Fargate 任務必須使用任務執行角色。此角色會授與工作代表您擷取、更新和管理 GuardDuty 安全代理程式的權限。如需更多資訊，請參閱Amazon ECS 任務執行 IAM 角色。
  - 您可以控制具有預先定義標記的叢集的執行階段監視。如果您的存取政策根據標籤限制存取，則必須向 IAM 使用者授予明確許可，以標記叢集。如需詳細資訊，請參閱 IAM 教學課程：根據 IAM 使用者指南中的標籤定義存取 AWS 資源的許可。
- 連接到 Amazon ECR 存儲庫：

  GuardDuty 安全代理程式會儲存在 Amazon ECR 儲存庫中。每個獨立和服務任務都必須具有存取存放庫的權限。您可以使用下列其中一個選項：

  - 對於公有子網路中的任務，您可以使用公有 IP 地址來執行任務，也可以在執行任務的子網路中為 Amazon ECR 建立 VPC 端點。如需詳細資訊，請參閱 《Amazon Elastic Container Registry 使用者指南》中的 Amazon ECR 介面 VPC 端點 (AWS PrivateLink)。
  - 對於私有子網路中的任務，您可以使用網路位址轉譯 (NAT) 閘道，或在執行任務的子網路中為 Amazon ECR 建立 VPC 端點。

如需詳細資訊，請參閱使用私有子網路和 NAT 閘道。

- 您必須具有的AWSServiceRoleForAmazonGuardDuty角色 GuardDuty。如需詳細資訊，請參閱 Amazon GuardDuty 使用者指南 GuardDuty中的服務連結角色許可。
- 您要使用執行階段監視保護的任何檔案都必須可由 root 使用者存取。如果您手動變更檔案的權限，則必須將其設定為755。

以下是在 EC2 容器執行個體上使用執行階段監控的先決條件：

- 您必須使用 Amazon ECS-20230929 AMI 的版本或更高版本。
- 您必須在容器執行個體上執行 Amazon ECS 代理程式至版本1.77或更新版本。
- 您必須使用核心版本5.10或更新版本。
- 如需支援 Linux 作業系統和架構的相關資訊，請參閱 GuardDuty 執行階段監控支援哪些作業模型和工作負載。
- 您可以使用 Systems Manager 來管理容器執行個體。如需詳細資訊，請參閱AWS Systems Manager Session Manager 使用者指南中的為 EC2 執行個體設定 Systems Manager。

若要立即保護工作，您必須執行下列其中一個動作：

- 對於獨立工作，請停止工作，然後啟動它們。如需詳細資訊，請參閱 停止 Amazon ECS 任務 和 以 Amazon ECS 任務的形式執行應用程式
- 對於屬於服務一部分的任務，請使用「強制新部署」選項更新服務。如需詳細資訊，請參閱 使用主控台更新 Amazon ECS 服務。

從 Amazon ECS 叢集移除執行階段監控

您可以從叢集中移除執行階段監視。這會導 GuardDuty 致停止監視叢集中的所有資源。

從叢集中移除執行階段監視。

1. 使用 Amazon ECS 主控台或 AWS CLI 將叢集上的GuardDutyManaged標籤金鑰設定為false。如需詳細資訊，請參閱使用 CLI 或 API 更新叢集或使用標籤。

> **Note**
>
> 「鍵」和「值」區分大小寫，且必須完全符合字串。

鍵 =GuardDutyManaged，值 = false

2. 解除 GuardDuty 安裝叢集中 EC2 容器執行個體上的安全代理程式。

   如需詳細資訊，請參閱《GuardDuty 使用指南》中的「手動解除安裝 Security Agent」。

3. 刪除每個叢集 GuardDuty VPC 的 VPC 端點。如需有關如何刪除 VPC 端點的詳細資訊，請參閱AWS PrivateLink 使用手冊中的刪除介面端點。

更新 Amazon ECS 容器執行個體上的 GuardDuty 安全代理程式

如需如何更新 EC2 容器執行個體上的 GuardDuty 安全代理程式的詳細資訊，請參閱 Amazon GuardDuty 使用者指南中的更新 GuardDuty 安全代理程式。

從帳戶移除 Amazon ECS 的執行階段監控

如果您不想再使用執行階段監視，請停用中的功能 GuardDuty。如需如何停用此功能的詳細資訊，請參閱 Amazon GuardDuty 使用者指南中的啟用執行階段監控。

從所有叢集移除執行階段監視。如需詳細資訊，請參閱 從 Amazon ECS 叢集移除執行階段監控。

運行時監視疑難排解

您可能需要疑難排解或確認執行階段監視已啟用，並在工作和容器上執行。

主題

- 如何判斷我的帳戶上的運行時監視是否處於活動狀態？
- 如何判斷群集上的運行時監視是否處於活動狀態？
- 我如何判斷 GuardDuty 安全代理是否在 Fargate 任務上運行？
- 如何判斷 GuardDuty 安全代理程式是否在 EC2 容器執行個體上執行？
- 如果叢集上執行的工作沒有任務執行角色，會發生什麼情況？
- 如何判斷我是否具有標記叢集以進行執行階段監視的正確權限？
- 當沒有連接 Amazon ECR 會發生什麼？
- 啟用運行時監視後，如何解決 Fargate 任務中的內存不足錯誤？

如何判斷我的帳戶上的運行時監視是否處於活動狀態？

在 Amazon ECS 主控台中，資訊位於 [帳戶設定] 頁面上。

您也可以list-account-settings使用該effective-settings選項運行。

```
aws ecs list-account-settings --effective-settings
```

輸出

名稱設定為guardDutyActivate且值設定為的設定on表示帳戶已設定。您必須洽詢您的系統管理 GuardDuty 員，以瞭解管理是自動還是手動管理。

```
{
    "setting": {
        "name": "guardDutyActivate",
        "value": "enabled",
        "principalArn": "arn:aws:iam::123456789012:root",
        "type": "aws-managed"
    }
}
```

如何判斷群集上的運行時監視是否處於活動狀態？

在 Amazon ECS 主控台中，資訊位於叢集詳細資料頁面的標籤索引標籤上。

您也可以describe-clusters使用該TAGS選項運行。

下列範例顯示預設叢集的輸出

```
aws ecs describe-clusters --cluster default --include TAGS
```

輸出

「金鑰」設為GuardDutyManaged且「值」設定為的標籤true表示叢集已設定為「執行階段監視」。

```
{
    "clusters": [
        {
            "clusterArn": "arn:aws:ecs:us-east-1:1234567890:cluster/default",
            "clusterName": "default",
            "status": "ACTIVE",
            "registeredContainerInstancesCount": 0,
            "runningTasksCount": 1,
            "pendingTasksCount": 0,
```

```
            "activeServicesCount": 0,
            "statistics": [],
            "tags": [
                {
                    "key": "GuardDutyManaged",
                    "value": "true"
                }
            ],
            "settings": [],
            "capacityProviders": [],
            "defaultCapacityProviderStrategy": []
        }
    ],
    "failures": []
}
```

我如何判斷 GuardDuty 安全代理是否在 Fargate 任務上運行？

GuardDuty 安全代理程式會以附屬容器的方式執行 Fargate 工作。

在 Amazon ECS 主控台中，並行會顯示在「任務詳細資訊」頁面的「容器」下方。

您可以運行`describe-tasks`並查找名稱設置為`aws-gd-agent`和 LastStatus 設置為的容器。`RUNNING`

下列範例顯示工作預設叢集的輸出`aws:ecs:us-east-1:123456789012:task/0b69d5c0-d655-4695-98cd-5d2d5EXAMPLE`。

```
aws ecs describe-tasks --cluster default --tasks aws:ecs:us-
east-1:123456789012:task/0b69d5c0-d655-4695-98cd-5d2d5EXAMPLE
```

輸出

名為`gd-agent`的容器處於狀`RUNNING`態。

```
"containers": [
      {
        "containerArn": "arn:aws:ecs:us-east-1:123456789012:container/4df26bb4-
f057-467b-a079-96167EXAMPLE",
        "taskArn": "arn:aws:ecs:us-east-1:123456789012:task/0b69d5c0-
d655-4695-98cd-5d2d5EXAMPLE",
        "lastStatus": "RUNNING",
```

```
        "healthStatus": "UNKNOWN",
        "memory": "string",
        "name": "aws-gd-agent"
      }
    ]
```

如何判斷 GuardDuty 安全代理程式是否在 EC2 容器執行個體上執行？

執行下列命令以檢視狀態：

```
sudo systemctl status amazon-guardduty-agent
```

記錄檔位於下列位置：

```
/var/log/amzn-guardduty-agent
```

如果叢集上執行的工作沒有任務執行角色，會發生什麼情況？

對於 Fargate 任務，任務在沒有 GuardDuty 安全代理程序側車容器的情況下啟動。 GuardDuty 儀表板會在涵蓋範圍統計資料儀表板中顯示工作缺少保護。

如何判斷我是否具有標記叢集以進行執行階段監視的正確權限？

若要標記叢集，您必須同時執行`CreateCluster`和的`ecs:TagResource`動作`UpdateCluster`。

以下是一個示例策略的片段。

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
         "ecs:TagResource"
      ],
      "Resource": "*",
      "Condition": {
         "StringEquals": {
             "ecs:CreateAction" : "CreateCluster",
             "ecs:CreateAction" : "UpdateCluster",
          }
       }
```

```
    }
  ]
}
```

當沒有連接 Amazon ECR 會發生什麼？

對於 Fargate 任務，任務在沒有 GuardDuty 安全代理程序側車容器的情況下啟動。 GuardDuty 儀表板會在涵蓋範圍統計資料儀表板中顯示工作缺少保護。

啟用運行時監視後，如何解決 Fargate 任務中的內存不足錯誤？

GuardDuty 安全性代理程式是輕量型程序。但是，該過程仍然會根據工作負載的大小消耗資源。我們建議您使用容器資源追蹤工具，例如 Amazon CloudWatch 容器洞察，在叢集中分段 GuardDuty部署。這些工具可協助您探索應用程式 GuardDuty 安全代理程式的使用設定檔。然後，您可以根據需要調整 Fargate 任務大小，以避免潛在的內存不足情況。

使用 ECS 執行監控 Amazon ECS 容器

使用 Amazon ECS Exec，您可以直接與容器互動，而無需先與主機容器作業系統互動、開啟傳入連接埠或管理 SSH 金鑰。您可以使用 ECS Exec 執行命令，或為在 Amazon EC2 執行個體或 AWS Fargate上執行的容器取得 shell。如此一來就可以更輕鬆地收集診斷資訊並快速對錯誤進行故障診斷。例如，在開發環境中，您可以使用 ECS Exec 輕鬆地與容器中的各種程序互動，並對應用程式進行故障診斷。在生產場景中，您可以使用它來獲得容器的破碎玻璃訪問權限，以調試問題。

您可以使用來自 Amazon ECS API、 AWS Command Line Interface (AWS CLI)、 AWS 開發套件或副駕駛 CLI 的 ECS Exec，在執行中的 Linux 或視窗容器中執行命令。 AWS 如需使用 ECS Exec 的詳細資訊，以及使用 C AWS opilot CLI 的影片逐步解說，請參閱 Copilot 文件。GitHub

您也可以使用 ECS Exec 來維護更嚴格的存取控制原則。透過選擇性地開啟此功能，您可以控制執行命令的人員，以及他們可以執行這些命令的任務。透過每個命令及其輸出的記錄，您可以使用 ECS Exec 來檢視已執行的工作，並可用 CloudTrail 來稽核存取容器的人員。

考量事項

在本主題中，您應該熟悉下列使用 ECS Exec 的相關方面：

- 目前不支援 ECS 執行器使用. AWS Management Console
- 對於在下列基礎設施上執行的任務支援 ECS Exec：
  - 任何 Amazon ECS 最佳化 AMI 中 Amazon EC2 上的 Linux 容器，包括 Bottlerocket

  - 外部執行個體上的 Linux 和視窗容器 (Amazon ECS Anywhere)
  - AWS Fargate 上的 Linux 和視窗容器
  - 下列 Windows Amazon ECS 最佳化 AMI 中 Amazon EC2 上的 Windows 容器 (使用容器代理程式版本 `1.56` 或更新版本)：
    - Amazon ECS 最佳化 Windows Server 2022 Full AMI
    - Amazon ECS 最佳化 Windows Server 2022 Core AMI
    - Amazon ECS 最佳化 Windows Server 2019 Full AMI
    - Amazon ECS 最佳化 Windows Server 2019 Core AMI
    - Amazon ECS 最佳化 Windows Server 20H2 Core AMI
- ECS 執行和 Amazon VPC
  - 如果您將界面 Amazon VPC 端點與 Amazon ECS 搭配使用，則必須為 Systems Manager 工作階段管理工具 (`ssmmessages`) 建立界面 Amazon VPC 端點。如需有關 Systems Manager VPC 端點的詳細資訊，請參閱《使 AWS PrivateLink 用指南》中的「AWS Systems Manager 用於為工作階段管理員設定 VPC 端點」。
  - 如果您將界面 Amazon VPC 端點與 Amazon ECS 搭配使用，並將 AWS KMS key 用於加密，則必須為 AWS KMS key建立界面 Amazon VPC 端點。如需詳細資訊，請參閱《AWS Key Management Service 開發人員指南》中的透過 VPC 端點連線至 AWS KMS key。
  - 當您有在 Amazon EC2 執行個體上執行的任務時，請使用`awsvpc`聯網模式。如果您無法存取網際網路 (例如未設定為使用 NAT 閘道)，則必須為系統管理員工作階段管理員建立介面 Amazon VPC 端點 (`ssmmessages`)。如需有關 `awsvpc` 網路模式考量事項的詳細資訊，請參閱考量事項。如需有關 Systems Manager VPC 端點的詳細資訊，請參閱《使 AWS PrivateLink 用指南》中的「AWS Systems Manager 用於為工作階段管理員設定 VPC 端點」。
- ECS 執行及 SSM
  - 當使用者使用 ECS Exec 在容器上執行命令時，這些命令會以 `root` 使用者執行。即使您指定容器的使用者 ID，SSM 代理程式及其子處理程序也會以根身分執行。
  - SSM 代理程式要求可以寫入容器檔案系統，才能建立所需的目錄和檔案。因此，不支援使用 `readonlyRootFilesystem` 任務定義參數或任何其他方法將根檔案系統設為唯讀。
  - 可以啟動 `execute-command` 動作以外的 SSM 工作階段時，這會導致工作階段未被記錄，會計入工作階段限制。建議您限制此存取，方法是使用 IAM 政策拒絕 `ssm:start-session` 動作。如需詳細資訊，請參閱 限制存取「啟動工作階段」動作。
- 以下功能作為邊車容器運行。因此，您必須指定要在其上執行命令的容器名稱。
  - 執行期監控
  - Service Connect

- 使用者可以執行容器內容中所有可用的命令。下列動作可能會導致孤立和廢止程序：終止容器的主要程序、終止命令代理程式，以及刪除相依性。為了清理廢止程序，建議將 `initProcessEnabled` 旗標新增至您的任務定義。
- ECS 執行器使用一些 CPU 和內存。在任務定義中指定 CPU 和記憶體資源配置時，您需要適應這一點。
- 您必須使用 AWS CLI 版本`1.22.3`或更新版本或 AWS CLI 版本`2.3.6`或更新版本。如需有關如何更新的資訊 AWS CLI，請參閱《AWS Command Line Interface 使用指南第 2 版》 AWS CLI中的〈安裝或更新最新版本〉。
- 每個程序 ID (PID) 命名空間僅能有一個 ECS Exec 工作階段。如果您在任務中共享 PID 命名空間，則僅能在一個容器中啟動 ECS Exec 工作階段。
- ECS Exec 工作階段的閒置逾時值為 20 分鐘。此值無法變更。
- 您無法針對現有任務開啟 ECS Exec。只能針對新任務開啟。
- 當您使用在叢集上啟動工作時，無法使用 `run-task` ECS Exec，而該叢集使用具有非同步放置的受管擴充功能 (在沒有執行個體的情況下啟動工作)。
- 您無法針對 Microsoft 納米服務器容器運行 ECS Exec。

必要條件

開始使用 ECS Exec 之前，請確定您已完成下列動作：

- 安裝及設定 AWS CLI。如需詳細資訊，請參閱 AWS CLI。
- 安裝的工作階段管理員外掛程式 AWS CLI。如需詳細資訊，請參閱為 AWS CLI安裝工作階段管理工具外掛程式。
- 您必須使用具有 ECS Exec 適當許可的任務角色。如需詳細資訊，請參閱任務 IAM 角色。
- ECS Exec 具有版本要求，取決於您的任務是否託管於 Amazon EC2 或 AWS Fargate：
  - 如果您使用的是 Amazon EC2，則必須使用在 2021 年 1 月 20 日之後發行的 Amazon ECS 最佳化 AMI，且代理程式版本為 1.50.2 或更高版本。如需詳細資訊，請參閱 Amazon ECS 最佳化 AMI。
  - 如果您使用的是 AWS Fargate，您必須使用平台版本`1.4.0`或更高版本 (Linux) 或 `1.0.0` (視窗)。如需詳細資訊，請參閱 AWS Fargate 平台版本。

架構

ECS Exec 利用 AWS Systems Manager (SSM) 工作階段管理員建立與執行中容器的連線，並使用 AWS Identity and Access Management (IAM) 政策來控制對執行中容器中執行命令的存取。將必要的 SSM 代理程式二進位檔繫結掛載至容器，即可達成此目的。Amazon ECS 或 AWS Fargate 代理程式負責在容器內與應用程式程式碼一起啟動 SSM 核心代理程式。如需詳細資訊，請參閱 Systems Manager 工作階段管理員。

您可以使用中的`ExecuteCommand`事件稽核哪些使用者存取容器， AWS CloudTrail 並將每個命令 (及其輸出) 記錄到 Amazon S3 或 Amazon CloudWatch 日誌。若要使用您自己的加密金鑰加密本機用戶端和容器之間的資料，您必須提供 AWS Key Management Service (AWS KMS) 金鑰。

使用 ECS Exec

可選任務定義變更

如果您將工作定義參數設`initProcessEnabled`定為`true`，這會啟動容器內的 init 程序。這會移除找到的任何殭屍 SSM 代理程式子處理程序。以下是範例。

```
{
    "taskRoleArn": "ecsTaskRole",
    "networkMode": "awsvpc",
    "requiresCompatibilities": [
        "EC2",
        "FARGATE"
    ],
    "executionRoleArn": "ecsTaskExecutionRole",
    "memory": ".5 gb",
    "cpu": ".25 vcpu",
    "containerDefinitions": [
        {
            "name": "amazon-linux",
            "image": "amazonlinux:latest",
            "essential": true,
            "command": ["sleep","3600"],
            "linuxParameters": {
                "initProcessEnabled": true
            }
        }
    ],
    "family": "ecs-exec-task"
```

```
}
```

為您的任務和服務開啟 ECS Exec

您可以在使用下列其中一個指 AWS CLI 令時指定--enable-execute-command旗標，為服務和獨立工作開啟 ECS Exec 功能：create-service、update-servicestart-task、或。run-task

例如，如果您執行下列指令，則會為在 Fargate 上執行的新建立服務開啟 ECS Exec 功能。如需建立服務的詳細資訊，請參閱 create-service。

```
aws ecs create-service \
    --cluster cluster-name \
    --task-definition task-definition-name \
    --enable-execute-command \
    --service-name service-name \
    --launch-type FARGATE \
     --network-configuration
 "awsvpcConfiguration={subnets=[subnet-12344321],securityGroups=[sg-12344321],assignPublicIp=EN
 \
    --desired-count 1
```

為任務開啟 ECS Exec 之後，您可以執行下列命令來確認任務已可供使用。如果 ExecuteCommandAgent 的 lastStatus 屬性會列為 RUNNING 且 enableExecuteCommand 屬性會設為 true，則您的任務已準備就緒。

```
aws ecs describe-tasks \
    --cluster cluster-name \
    --tasks task-id
```

以下輸出程式碼片段為您可能會看到的內容範例。

```
{
    "tasks": [
        {
            ...
            "containers": [
                {
                    ...
                    "managedAgents": [
                        {
                            "lastStartedAt": "2021-03-01T14:49:44.574000-06:00",
```

```
                            "name": "ExecuteCommandAgent",
                            "lastStatus": "RUNNING"
                        }
                    ]
                }
            ],
            ...
            "enableExecuteCommand": true,
            ...
        }
    ]
}
```

使用 ECS Exec 執行命令

在確認 `ExecuteCommandAgent` 正在執行後，您可以使用以下命令在容器上開啟交互式 Shell。如果您的任務包含多個容器，您必須使用 `--container` 旗標指定容器名稱。Amazon ECS 僅支援啟動互動式工作階段，因此您必須使用 `--interactive` 旗標。

########### ID ##################/bin/sh####

ID 是任務的 Amazon 資源名稱（ARN）。

```
aws ecs execute-command --cluster cluster-name \
    --task task-id \
    --container container-name \
    --interactive \
    --command "/bin/sh"
```

使用 ECS 執行程式進行記錄

開啟任務和服務的登入

> **Important**
>
> 如需有關 CloudWatch 定價的詳細資訊，請參閱CloudWatch 定價。Amazon ECS 還提供免費的監控指標。如需詳細資訊，請參閱 使用監控 Amazon ECS CloudWatch 。

Amazon ECS 為使用 ECS Exec 執行的記錄命令提供預設組態，方法是使用任務定義中設定的 CloudWatch 日`awslogs`誌驅動程式將日誌傳送至日誌。如果您想要提供自訂組態， AWS CLI 支援

create-cluster 和 update-cluster 命令的 --configuration 旗標。同樣重要的是要知道容器映像需cat要script並安裝，以便將命令日誌正確上傳到 Amazon S3 或 CloudWatch 日誌。如需建立叢集的詳細資訊，請參閱 create-cluster。

> **Note**
>
> 此組態只會處理 execute-command 工作階段的日誌記錄。它不會影響應用程式的日誌記錄。

下列範例會建立叢集，然後將輸出記錄到名為的 CloudWatch 日誌cloudwatch-log-group-name和 LogGroup 名為的 Amazon S3 儲存貯體s3-bucket-name。

將CloudWatchEncryptionEnabled選項設定為時，您必須使用 AWS KMS 客戶管理金鑰來加密記錄群組true。如需如何加密記錄群組的詳細資訊，請參閱《使用 AWS Key Management Service指南》中的〈 CloudWatch 記錄檔方式〉中的〈加密記錄檔資料〉。Amazon CloudWatch Logs

```
aws ecs create-cluster \
    --cluster-name cluster-name \
    --configuration executeCommandConfiguration="{ \
        kmsKeyId=string, \
        logging=OVERRIDE, \
        logConfiguration={ \
            cloudWatchLogGroupName=cloudwatch-log-group-name, \
            cloudWatchEncryptionEnabled=true, \
            s3BucketName=s3-bucket-name, \
            s3EncryptionEnabled=true, \
            s3KeyPrefix=demo \
        } \
    }"
```

logging 屬性決定 ECS Exec 的日誌記錄功能行為：

- NONE：記錄已關閉。
- DEFAULT：日誌被發送到配置的awslogs驅動程序。如果未設定驅動程式，則不會儲存任何記錄。
- OVERRIDE：日誌被發送到提供的 Amazon CloudWatch 日誌 LogGroup，Amazon S3 存儲桶或兩者。

Amazon CloudWatch 日誌或 Amazon S3 日誌所需的 IAM 許可

若要啟用日誌記錄，任務定義中參考的 Amazon ECS 任務角色需要有額外的許可。這些額外的許可可以作為政策新增至任務角色。根據您是否將日誌導向到亞馬遜日 CloudWatch 誌或 Amazon S3，它們會有所不同。

Amazon CloudWatch Logs

下列範例政策會新增必要的 Amazon CloudWatch 日誌許可。

```
{
   "Version": "2012-10-17",
   "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogGroups"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
               "logs:CreateLogStream",
               "logs:DescribeLogStreams",
               "logs:PutLogEvents"
            ],
            "Resource": "arn:aws:logs:region:account-id:log-group:/aws/
ecs/cloudwatch-log-group-name:*"
        }
   ]
}
```

Amazon S3

下列政策範例會新增必要 Amazon S3 許可。

```
{
   "Version": "2012-10-17",
   "Statement": [
        {
            "Effect": "Allow",
```

```
            "Action": [
               "s3:GetBucketLocation"
            ],
            "Resource": "*"
        },
        {
           "Effect": "Allow",
           "Action": [
               "s3:GetEncryptionConfiguration"
           ],
           "Resource": "arn:aws:s3:::s3-bucket-name"
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:PutObject"
            ],
            "Resource": "arn:aws:s3:::s3-bucket-name/*"
        }
    ]
   }
```

使用您自己的加密所需的 IAM 許可 AWS KMS key (KMS 金鑰)

根據預設，在本機用戶端和容器之間傳輸的資料會使用 AWS 提供的 TLS 1.2 加密。若要使用您自己的 KMS 金鑰進一步加密資料，您必須建立 KMS 金鑰並新增 `kms:Decrypt` 許可至您的任務 IAM 角色。您的容器使用此許可來解密資料。如需建立 KMS 金鑰的詳細資訊，請參閱建立金鑰。

您可以將下列內嵌政策新增至需要 AWS KMS 許可的任務 IAM 角色。如需詳細資訊，請參閱 ECS 執行權限。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "kms:Decrypt"
            ],
            "Resource": "kms-key-arn"
        }
    ]
```

```
}
```

若要使用您自己的 KMS 金鑰加密資料，則必須向使用 `execute-command` 動作的使用者或群組授予 `kms:GenerateDataKey` 許可。

下列使用者或群組的範例政策包含使用您自己的 KMS 金鑰的必要許可。您必須指定 KMS 金鑰的 Amazon Resource Name (ARN)。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "kms:GenerateDataKey"
            ],
            "Resource": "kms-key-arn"
        }
    ]
}
```

使用 IAM 政策限制 ECS Exec 的存取

您可以使用下列一或多個 IAM 政策條件金鑰來限制使用者存取執行命令 API 動作：

- `aws:ResourceTag/clusterTagKey`
- `ecs:ResourceTag/clusterTagKey`
- `aws:ResourceTag/taskTagKey`
- `ecs:ResourceTag/taskTagKey`
- `ecs:container-name`
- `ecs:cluster`
- `ecs:task`
- `ecs:enable-execute-command`

使用下列範例 IAM 政策，使用者可以在任務內執行的容器中執行命令，任務標籤具有 `environment` 鍵和 `development` 值，並位於名為 `cluster-name` 的叢集中。

```
{
```

```
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ecs:ExecuteCommand",
                "ecs:DescribeTasks"
            ],
         "Resource": [
                "arn:aws:ecs:region:aws-account-id:task/cluster-name/*",
                "arn:aws:ecs:region:aws-account-id:cluster/*"
            ],
            "Condition": {
                "StringEquals": {
                    "ecs:ResourceTag/environment": "development"
                }
            }
        }
    ]
}
```

在下列 IAM 政策範例中，當容器名稱為 `production-app` 時，使用者無法使用 `execute-command` API。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": [
                "ecs:ExecuteCommand"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "ecs:container-name": "production-app"
                }
            }
        }
    ]
}
```

使用下列 IAM 政策，使用者只能在 ECS Exec 關閉時啟動任務。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ecs:RunTask",
                "ecs:StartTask",
                "ecs:CreateService",
                "ecs:UpdateService"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "ecs:enable-execute-command": "false"
                }
            }
        }
    ]
}
```

> **Note**
>
> 由於 `execute-command` API 動作只包含請求中的任務和叢集資源，所以只會評估叢集和任務標籤。

如需有關 IAM 政策條件索引鍵的詳細資訊，請參閱《服務授權參考》中的 Amazon Elastic Container Service 的動作、資源與條件索引鍵。

限制存取「啟動工作階段」動作

雖然可以在 ECS Exec 之外的容器上啟動 SSM 工作階段，但這可能會導致工作階段未被記錄。在 ECS Exec 以外啟動的工作階段也會計入工作階段配額。建議您透過直接拒絕針對使用 IAM 政策之 Amazon ECS 任務的 `ssm:start-session` 動作來限制此存取。您可以根據使用的標籤拒絕存取所有 Amazon ECS 任務或特定任務。

以下為範例 IAM 政策，它會針對具有特定叢集名稱之所有區域中的任務，拒絕存取 `ssm:start-session` 動作。您可以選擇包含具有 *`cluster-name`* 的萬用字元。

```
{
```

```
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": "ssm:StartSession",
            "Resource": [
                "arn:aws:ecs:region:aws-account-id:task/cluster-name/*",
                "arn:aws:ecs:region:aws-account-id:cluster/*"
            ]
        }
    ]
}
```

以下為範例 IAM 政策，它會針對所有具有標籤鍵 `Task-Tag-Key` 和標籤值 `Exec-Task` 的區域中的資源，拒絕對 `ssm:start-session` 動作的存取。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": "ssm:StartSession",
            "Resource": "arn:aws:ecs:*:*:task/*",
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/Task-Tag-Key": "Exec-Task"
                }
            }
        }
    ]
}
```

如需使用 Amazon ECS Exec 時可能遇到的任何問題的說明，請參閱 Exec 的疑難排解問題。

AWS Compute Optimizer 針對 Amazon ECS 的建議

AWS Compute Optimizer 針對 Amazon ECS 任務和容器大小產生建議。如需詳細資訊，請參閱《 AWS Compute Optimizer使用者指南》中的什麼是AWS Compute Optimizer ？。

Amazon ECS 服務的任務和容器大小建議 AWS Fargate

AWS Compute Optimizer 在 AWS Fargate上產生有關 Amazon ECS 服務的建議。 AWS Compute Optimizer 建議工作 CPU 和工作記憶體大小以及容器 CPU、容器記憶體和容器記憶體保留區大小。這些建議會顯示在 Compute Optimizer 主控台的以下頁面中。

- 針對 Fargate 上 Amazon ECS 服務的建議頁面
- Fargate 上的 Amazon ECS 服務詳細資訊頁面

如需詳細資訊，請參閱《AWS Compute Optimizer 使用者指南》中的檢視針對 Fargate 上 Amazon ECS 服務的建議。

Amazon ECS 故障診斷

您可能需要疑難排解負載平衡器、工作、服務或容器執行個體的問題。本章會協助您從 Amazon ECS 容器代理程式、容器執行個體的 Docker 常駐程式，以及 Amazon ECS 主控台的服務事件日誌中尋找診斷資訊。

如需已停止工作的相關資訊，請參閱下列其中一項。

動作	進一步了解
解決停止的工作錯誤。	查看 Amazon ECS 停止的任務錯誤
檢視已停止的工作錯誤。	解決 Amazon ECS 停止的任務錯誤
檢閱已停止的工作錯誤代碼。	Amazon ECS 停止任務錯誤消息
檢閱 CannotPullContainer 工作錯誤。	CannotPullContainer Amazon ECS 中的任務錯誤
檢視工作 IAM 角色請求。	查看 Amazon ECS 任務的 IAM 角色請求

如需有關服務錯誤的資訊，請參閱下列其中一項。

動作	進一步了解
檢視服務事件訊息。	檢視 Amazon ECS 服務事件訊息
檢閱服務事件訊息。	Amazon ECS 服務事件消息
檢閱負載平衡器問題。	疑難排解 Amazon ECS 中的服務負載平衡器

動作	進一步了解	
檢閱服務 auto 調整規模問題。	疑難排解 Amazon ECS 中的服務 auto 擴展	

如需有關作業定義錯誤的資訊，請參閱下列其中一項。

動作	進一步了解	
解決工作定義記憶體錯誤。	疑難排解 Amazon ECS 任務定義無效的 CPU 或記憶體錯誤	

如需 Amazon ECS 代理程式錯誤的相關資訊，請參閱下列其中一項。

動作	進一步了解	
檢視 Amazon ECS 容器代理程式日誌。	檢視 Amazon ECS 容器代理程式日誌	
了解如何收集 Amazon ECS 日誌。	使用 Amazon ECS 日誌收集器收集容器日誌	
使用 Amazon ECS 代理程式擷取診斷詳細資訊。	使用代理程式自檢功能擷取 Amazon ECS 診斷詳細資料	

如需 Docker 錯誤的相關資訊，請參閱下列其中一項。

動作	進一步了解	
使用泊塢視窗診斷。	Amazon ECS 中的碼頭診斷	
開啟碼頭偵錯模式。	在 Amazon ECS 中從 Docker 守護進程配置詳細輸出	

動作	進一步了解	
解決碼頭工人 API 錯誤 500.	Amazon ECS 中的泊塢視API error (500): devmapper 窗疑難排解	

如需 ECS Exec 和 Amazon ECS Anywhere 所不在錯誤的相關資訊，請參閱下列其中一項。

動作	進一步了解	
ECS 執行程式疑難排解	Amazon ECS 執行問題疑難排解	
隨時隨地進行亞馬遜 ECS 疑難	疑難排解 Amazon ECS Anywhere 可見	

如需節流問題的相關資訊，請參閱下列其中一項。

動作	進一步了解	
瞭解 Fargate 節流配額。	AWS Fargate 節流配額	
了解 Amazon ECS 節流的最佳實務。	處理 Amazon ECS 節流問題	

如需 API 錯誤的相關資訊，請參閱下列其中一項。

動作	進一步了解	
解決 API 錯誤。	Amazon ECS API 故障原因	

解決 Amazon ECS 停止的任務錯誤

當您的工作無法啟動時，您會在主控台和`describe-tasks`輸出參數中看到錯誤訊息 (stoppeDreAon 和)。 StoppedCode下列各節提供有關如何解決已停止工作問題的其他資訊。

下列頁面提供已停止工作的相關資訊。

- 瞭解已停止工作錯誤訊息的變更。

  Amazon ECS 停止任務錯誤消息更新
- 檢視已停止的工作，以便取得原因的相關資訊。

  查看 Amazon ECS 停止的任務錯誤
- 瞭解已停止的工作錯誤訊息以及錯誤的可能原因。

  Amazon ECS 停止任務錯誤消息
- 瞭解如何驗證已停止的工作連線並修正錯誤。

  驗證 Amazon ECS 停止任務連接

Amazon ECS 停止任務錯誤消息更新

從 2024 年 6 月 14 日開始，Amazon ECS 團隊正在變更已停止的任務錯誤訊息，如下表所述。不`stopCode`會改變。如果您的應用程式依賴於確切的錯誤訊息字串，則必須使用新字串更新應用程式。有關問題或問題的幫助，請聯繫 AWS Support。

> **Note**
>
> 我們建議您不要依賴自動化操作的錯誤訊息，因為錯誤訊息可能會變更。

CannotPullContainerError

舊的錯誤訊息	新錯誤訊息	
CannotPullContainerError: ########## ########## ############docker ##########roleARN	CannotPullContainerError：任務無法拉出圖像。檢查角色是否具有從登錄提取映像的權限。協助程式的錯誤回應：儲存#的提取存取遭拒、儲存庫不存在或可能需要「docker 登入」：拒絕：使用者：role	

舊的錯誤訊息	新錯誤訊息	
	ARN ####：ecr: BatchGetImage on resource: *image*，因為沒有以身分識別為基礎的政策允許執行 ecr: 動作。BatchGetImage	
	CannotPullContainerError：任務無法拉出圖像。檢查圖像是否存在。守護進程的錯誤響應：存##的提取訪問被拒絕，存儲庫不存在或可能需要「docker 登錄」：拒絕：對資源的請求訪問被拒絕。	
CannotPullContainerError：來自守護程序的錯誤響應：獲取## *URI*：net/http：等待連接時取消請求	CannotPullContainerError：任務無法拉出圖像。檢查您的網路組態。來自守護進程的錯誤響應：獲取##：net/http：等待連接時取消請求（等待標題時超過客戶端 .Timeout）	

ResourceNotFoundException

舊的錯誤訊息	新錯誤訊息	
從區域 us-west-2 AWS Secrets Manager 中獲取秘密數據：秘密 *SercreTarn* :: Secrets Manager 找不到指定的密碼。 ResourceNotFoundException	ResourceNotFoundException：該任務無法從中檢索 ARN '#統' 的秘密。 AWS Secrets Manager檢查秘密是否存在於指定的區域中。 ResourceNotFoundException: 從區域#域 AWS Secrets Manager 中獲取秘密數據:秘密 *SercreTarn* ：ResourceN	

舊的錯誤訊息	新錯誤訊息	
	otFoundException: Secrets Manager 找不到指定的密鑰。	

查看 Amazon ECS 停止的任務錯誤

如果啟動任務時有問題，您的任務可能會因為應用程式或組態錯誤而停止。例如，您執行任務，而任務顯示 PENDING 狀態，然後就消失。

如果您的任務是由 Amazon ECS 服務建立，Amazon ECS 為維護服務所採取的動作會發佈在服務事件中。您可以在、、 AWS 開發套件 AWS Management Console AWS CLI、Amazon ECS API 或使用開發套件和 API 的工具中檢視事件。這些事件包括 Amazon ECS 停止並取代任務，因為任務中的容器已停止執行，或多次未通過 Elastic Load Balancing 的運作狀態檢查。

如果您的任務在 Amazon EC2 或外部電腦上的容器執行個體上執行，您也可以查看容器執行階段和 Amazon ECS 代理程式的日誌。這些日誌位於主機 Amazon EC2 執行個體或外部電腦上。如需詳細資訊，請參閱 檢視 Amazon ECS 容器代理程式日誌。

程序

Console

AWS Management Console

以下步驟可用於使用 new 來檢查已停止的任務是否存在錯誤 AWS Management Console。

1. 開啟主控台，網址為 https://console.aws.amazon.com/ecs/v2。
2. 在導覽窗格中，選擇叢集。
3. 在叢集頁面上，選擇叢集。
4. 在 Cluster : ***name*** (叢集：名稱) 頁面上，選擇 Tasks (任務) 索引標籤。
5. 設定篩選條件以顯示已停止的任務。針對篩選所需狀態，選擇 Stopped (已停止) 或 Any desired status (任何所需狀態)。

   Stopped (已停止) 選項會顯示已停止的任務，而 Any desired status (任何所需狀態) 會顯示所有任務。
6. 選擇要檢查的已停止任務。
7. 在已停止任務的列中，在 Last Status (上次狀態) 欄中，選擇 Stopped (已停止)。

彈出窗口顯示停止的原因。

AWS CLI

1. 列出叢集中停止的任務。輸出包含任務的 Amazon Resource Name (ARN)，您需要 Amazon Resource Name (ARN) 來描述任務。

```
aws ecs list-tasks \
     --cluster cluster_name \
     --desired-status STOPPED \
     --region region
```

2. 描述已停止的工作以擷取資訊。如需詳細資訊，請參閱《參考資料》中的描述任務。 AWS Command Line Interface

```
aws ecs describe-tasks \
     --cluster cluster_name \
     --tasks arn:aws:ecs:region:account_id:task/cluster_name/task_ID \
     --region region
```

使用下列輸出參數。

- `stopCode`-停止代碼指出任務停止的原因，例如 ResourceInitializationError
- `StoppedReason`-任務停止的原因。
- `reason`（在`containers`結構中）-原因提供有關停止容器的其他詳細信息。

後續步驟

檢視已停止的工作，以便取得原因的相關資訊。如需詳細資訊，請參閱 Amazon ECS 停止任務錯誤消息。

Amazon ECS 停止任務錯誤消息

以下是當工作意外停止時，您可能會收到的可能錯誤訊息。

若要使用檢查已停止的工作是否有錯誤訊息 AWS Management Console，請參閱查看 Amazon ECS 停止的任務錯誤。

分離的任務錯誤代碼具有與其相關聯的類別，例如「ResourceInitialization錯誤」。要獲取有關每個類別的更多信息，請參閱以下內容：

類別	進一步了解
TaskFailedToStart	Amazon ECS TaskFailedToStart 錯誤疑難排解
ResourceInitialization錯誤	Amazon ECS ResourceInitializationError 錯誤疑難排解
ResourceNotFoundException	Amazon ECS ResourceNotFoundException 錯誤疑難排解
SpotInterruption錯誤	Amazon ECS SpotInterruption 錯誤疑難排解
InternalError	Amazon ECS InternalError 錯誤疑難排解
OutOfMemoryError	Amazon ECS OutOfMemoryError 錯誤疑難排解
ContainerRuntime錯誤	Amazon ECS ContainerRuntimeError 錯誤疑難排解
ContainerRuntimeTimeoutError	Amazon ECS ContainerRuntimeTimeoutError 錯誤疑難排解
CannotStartContainerError	Amazon ECS CannotStartContainerError 錯誤疑難排解
CannotStopContainerError	Amazon ECS CannotStopContainerError 錯誤疑難排解

類別	進一步了解	
CannotInspectContainerError	Amazon ECS CannotInspectContainerError 錯誤疑難排解	
CannotCreateVolumeError	Amazon ECS CannotCreateVolumeError 錯誤疑難排解	
CannotPull容器	CannotPullContainer Amazon ECS 中的任務錯誤	

Amazon ECS TaskFailedToStart 錯誤疑難排解

以下是您可以採取的一些`TaskFailedToStart`錯誤訊息和動作來修正錯誤。

嘗試在#網 '子網 ID 中啟用公共 IP 分配創建網絡接口時出現意外的 EC2 錯誤

當 Fargate 工作使用網路模式並在具有公用 IP 位址的子`awsvpc`網路中執行，且子網路沒有足夠的 IP 位址時，就會發生這種情況。

可用 IP 位址的數量可在 Amazon EC2 主控台的子網路詳細資料頁面上取得，或使用`describe-subnets`。如需詳細資訊，請參閱 Amazon VPC 使用者指南中的檢視子網路。

若要修正此問題，您可以建立新的子網路來執行您的工作。

InternalError: <reason>

在請求 ENI 附件時，會發生此錯誤。Amazon EC2 以非同步方式處理 ENI 的佈建。佈建程序需要時間。Amazon ECS 會出現逾時，導致等待時間較長或未報告的故障。有時會佈建 ENI，但報告會在故障逾時後傳送給 Amazon ECS。在這種情況下，Amazon ECS 會透過使用中的 ENI 查看報告的任務失敗。

選取的工作定義與選取的運算策略不相容

當您選擇的啟動類型不符合叢集容量類型的作業定義時，就會發生這個錯誤。如需詳細資訊，請參閱 Amazon ECS 啟動類型。您需要選取符合指派給叢集之容量提供者的作業定義。

Amazon ECS ResourceInitializationError 錯誤疑難排解

以下是您可以採取的一些`ResourceInitialization`錯誤訊息和動作來修正錯誤。

無法提取密碼或註冊表身份驗證：該任務無法從 Amazon ECR 提取註冊表身份驗證

當您的任務無法提取任務定義中定義的圖像時，就會發生此錯誤。

這個問題是由下列其中一個原因所造成：

錯誤原因..	執行此作業...
Amazon ECR 虛擬私人雲端端點與任務之間的網路連線問題。 當您在錯誤訊息中看到下列任何字串時，問題是網路問題： • 撥打 TCP • 撥號 UDP • <ip>:<port>: I/O超時 • 網/HTTP：TLS 握手超時 • 讀取：連線逾時 • 等待標頭時用戶端超過逾時 • net/http：等待連接時取消請求 • 信號：被殺 • 超過前後關聯期	驗證任務與 Amazon VPC 端點之間的連線能力：驗證 Amazon ECS 停止任務連接
任務定義中定義的角色沒有 Amazon ECR 的許可。	將必要的權限新增至工作執行角色。 此工作會使用下列其中一個角色： • 對於具有 Fargate 啟動類型的工作，這是工作執行角色。如需相關資訊，請參閱 透過界面端點許可提取 Amazon ECR 映像的 Fargate 端任務。

錯誤原因..	執行此作業...	
	• 對於具有 EC2 啟動類型的任務，這是容器執行個體角色。如需相關資訊，請參閱 Amazon ECR 許可。	

錯誤原因..	執行此作業...	
圖像 ARN 不存在	檢視影像，然後驗證下列項目： 如需檢視影像的相關資訊，請參閱 Amazon 彈性容器登錄使用者指南中的在 Amazon ECR 中檢視影像詳細資料。 • 影像與工作位於相同的區域中。 將圖像推入正確的區域。然後，使用新映像 ARN 更新任務。 如需有關推送映像的資訊，請參閱 Amazon ECR 使用者指南中的將映像推送到 Amazon ECR 儲存庫 如需更新任務定義的相關資訊，請參閱 Amazon 彈性容器服務 API 參考中的使用主控台更新 Amazon ECS 任務定義或RegisterTask定義。 • 工作定義具有不正確的影像 ARN。 更新作業定義。如需更新任務定義的相關資訊，請參閱 Amazon 彈性容器服務 API 參考中的使用主控台更新 Amazon ECS 任務定義或RegisterTask定義。	

無法提取密碼或註冊表身份驗證：無法從 ssm 檢索密碼：任務無法從 Systems Manager 中提取秘密「Secret ***Name***」

當您的工作無法使用 Systems Manager 中的認證提取工作定義中定義的映像時，就會發生此錯誤。

這個問題是由下列其中一個原因所造成：

錯誤原因..	執行此作業...	
Systems Manager VPC 端點與工作之間的網路連線問題。 當您在錯誤訊息中看到下列任何字串時，問題是網路問題： • 撥打 TCP • 撥號 UDP • <ip>:<port>: I/O超時 • 網/HTTP：TLS 握手超時 • 讀取：連線逾時 • 等待標頭時用戶端超過逾時 • net/http：等待連接時取消請求 • 信號：被殺 • 超過前後關聯期	驗證工作與 Systems Manager 端點之間的連線：驗證 Amazon ECS 停止任務連接。	
任務定義中定義的角色沒有 Secrets Manager 的權限。	將必要的 Systems Manager 權限新增至工作執行角色。如需詳細資訊，請參閱 Secrets Manager 或 Systems Manager 權限。	
秘密 ARN 不存在	檢查 ARN 是否存在。若要取得資訊，請參閱使用指南中的〈搜尋 Systems Manager 參數 AWS Systems Manager〉。	

無法提取密碼或註冊表身份驗證：無法從 asm 檢索密碼：任務無法從密碼管理器中提取秘密 'Secret *ARN*'

當您的 Fargate 工作無法使用 Secrets Manager 中的認證提取工作定義中定義的映像時，就會發生這個錯誤。

這個問題是由下列其中一個原因所造成：

錯誤原因..	執行此作業...	
Secrets Manager 虛擬私人雲端端點與工作之間的網路連線問題。 當您在錯誤訊息中看到下列任何字串時，問題是網路問題： • 撥打 TCP • 撥號 UDP • <ip>:<port>: I/O超時 • 網/HTTP：TLS 握手超時 • 讀取：連線逾時 • 等待標頭時用戶端超過逾時 • net/http：等待連接時取消請求 • 信號：被殺 • 超過前後關聯期	驗證工作與 Secrets Manager 端點之間的連線。如需詳細資訊，請參閱 驗證 Amazon ECS 停止任務連接。	
任務定義中定義的任務執行角色沒有 Secrets Manager 的權限。	將 Secrets Manager 的必要權限新增至工作執行角色。如需詳細資訊，請參閱 Secrets Manager 或 Systems Manager 權限。	
秘密 ARN 不存在	檢查 ARN 是否存在於 Secrets Manager 中。如需檢視圖片的相關資訊，請參閱秘密管理員	

錯誤原因..	執行此作業...	
	開發人員指南中的「在 Secrets Manager 中尋找密碼」。	

無法提取密碼或註冊表身份驗證：該任務無法從秘密管理器中提取秘密 'Secret ***ARN***'

當您的任務無法使用 Secrets Manager 中的認證提取任務定義中定義的映像時，就會發生此錯誤。

該錯誤表示 Systems Manager VPC 端點和任務之間存在網絡連接問題。

如需如何驗證工作與端點之間連線的相關資訊，請參閱驗證 Amazon ECS 停止任務連接。

無法下載 env 文件：任務無法從 Amazon S3 下載環境變量文件

當您的任務無法從 Amazon S3 下載環境檔案時，就會發生此錯誤。

錯誤原因..	執行此作業...	
任務和 Amazon S3 之間的網絡連接問題。	驗證任務與 Amazon S3 端點之間的連線能驗證 Amazon ECS 停止任務連接力：	
任務定義中定義的角色沒有 Amazon S3 的許可。	將 Amazon S3 許可添加到該角色。如需詳細資訊，請參閱 Amazon S3 檔案儲存許可。	

無法驗證記錄器 args: 工作找不到工作定義中定義的記 CloudWatch 錄*####*。任務和之間存在連接問題 CloudWatch。

當您的任務找不到您在任務定義中定義的 CloudWatch 日誌群組時，就會發生這個錯誤。

錯誤表示任務定義中的 CloudWatch 群組不存在。

您可以執行下列其中一個選項來修正此問題：

若要使用此選項...	執行此作業...	
更新工作定義，將記錄群組組態包含在容器定義中。	如需更新任務定義的相關資訊，請參閱 Amazon 彈性容	

若要使用此選項...	執行此作業...	
	器服務 API 參考中的使用主控台更新 Amazon ECS 任務定義或RegisterTask定義。	
在中建立記錄群組 CloudWatch	a. 執行下列命令以取得記錄群組名稱。 `aws ecs describe-task-definition \ --task-definition task-def-name | jq -r.taskDefinition.containerDefinitions[].logConfiguration` b. 建立記錄群組。如需詳細資訊，請參閱 Amazon 日誌使用者指南中的 CloudWatch 日誌中的建立 CloudWatch 日誌群組。	

無法初始化記錄驅動程式

當您的任務找不到您在任務定義中定義的 CloudWatch 日誌群組時，就會發生這個錯誤。

錯誤表示任務定義中的 CloudWatch 群組不存在。

您可以執行下列其中一個選項來修正此問題：

若要使用此選項...	執行此作業...	
更新工作定義，將記錄群組組態包含在容器定義中。	如需更新任務定義的相關資訊，請參閱 Amazon 彈性容器服務 API 參考中的使用主控台更新 Amazon ECS 任務定義或RegisterTask定義。	

若要使用此選項...	執行此作業...	
在中建立記錄群組 CloudWatch	a. 執行下列命令以取得記錄群組名稱。 `aws ecs describe-task-definition \ --task-definition task-def-name | jq -r.taskDefinition.containerDefinitions[].logConfiguration` b. 建立記錄群組。如需詳細資訊，請參閱 Amazon 日誌使用者指南中的 CloudWatch 日誌中的建立 CloudWatch 日誌群組。	

無法呼叫 EFS 公用程式命令來設定 EFS 磁碟區

下列問題可能會導致您無法在提問中掛接 Amazon EFS 磁碟區：

- Amazon EFS 檔案系統未正確設定。
- 工作沒有必要的權限。
- 存在與網路和 VPC 組態相關的問題。

如需如何偵錯和修正此問題的相關資訊，請參閱為什麼我無法在 AWS Re: post 上的 AWS Fargate 任務上掛接 Amazon EFS 磁碟區。

Amazon ECS ResourceNotFoundException 錯誤疑難排解

以下是您可以採取的一些 `ResourceNotFoundException`錯誤訊息和動作來修正錯誤。

該任務無法從中檢索 ARN '**##**器' 的秘密。 AWS Secrets Manager檢查秘密是否存在於指定的區域中。

當工作無法從密碼管理員擷取密碼時，就會發生這個錯誤。這表示在工作定義中指定的密碼 (以及包含在錯誤訊息中) 不存在於 Secrets Manager 中。

「地區」位於錯誤訊息中。

從 AWS Secrets Manager 區域#域擷取秘密資料:秘密 *SercreTarn*: ResourceNotFoundException: Secrets Manager 找不到指定的密碼。

如需有關尋找密碼的資訊，請參閱《AWS Secrets Manager 使用指南》AWS Secrets Manager中的〈尋找密碼〉。

請使用下表判斷並解決錯誤。

問題	動作	
秘密位於與任務定義不同的「區域」中。	a. 在與任務相同的區域中創建密碼。如需詳細資訊，請參閱建立 AWS Secrets Manager 密碼。 b. 使用新密碼更新作業定義。如需詳細資訊，請參閱 Amazon 彈性容器服務 API 參考中的使用主控台更新 Amazon ECS 任務定義或RegisterTask定義。	
任務定義具有不正確的秘密 ARN。密碼管理員中存在正確的密碼。	使用正確的密碼更新作業定義。如需詳細資訊，請參閱 Amazon 彈性容器服務 API 參考中的使用主控台更新 Amazon ECS 任務定義或RegisterTask定義。	
秘密不再存在。	a. 在與任務相同的區域中創建密碼。如需詳細資訊，請參閱建立 AWS Secrets Manager 密碼。 b. 使用新密碼更新作業定義。如需詳細資訊，請參閱 Amazon 彈性容器服務 API 參考中的使用主控台	

問題	動作	
	更新 Amazon ECS 任務定義或RegisterTask定義。	

Amazon ECS SpotInterruption 錯誤疑難排解

該`SpotInterruption`錯誤對於 Fargate 和 EC2 啟動類型有不同的原因。

Fargate 啟動類型

當沒有遠門競價型容量或當 Fargate `SpotInterruption` 收回 Spot 容量時，就會發生錯誤。

您可以在多個可用區域中執行工作，以提供更多容量。

EC2 啟動類型

當沒有可用的 Spot 執行個體或 EC2 收回 Spot 執行個體容量時，就會發生此錯誤。

您可以讓執行個體在多個可用區域中執行，以提供更多容量。

Amazon ECS InternalError 錯誤疑難排解

適用於：Fargate 啟動類型

代理程式遇到非預期、非執行階段相關的內部`InternalError`錯誤時發生的錯誤。

只有在使用平台版本 `1.4` 或更新版本時，才會發生此錯誤。

如需如何偵錯和修正此問題的相關資訊，請參閱如何疑難排解 Re: POST 上 AWS ECS 叢集中無法啟動的 Amazon ECS 任務。

Amazon ECS OutOfMemoryError 錯誤疑難排解

以下是您可以採取的一些 OutOfMemoryError 錯誤訊息和動作來修正錯誤。

容器因內存使用而死亡

當容器結束是由於容器中的處理序消耗的記憶體比任務定義中分配的記憶體更多時，就會發生此錯誤。

Amazon ECS ContainerRuntimeError 錯誤疑難排解

以下是您可以採取的一些 ContainerRuntimeError 錯誤訊息和動作來修正錯誤。

ContainerRuntime錯誤

當代理程式從 containerd 接收到特定執行時間操作的意外錯誤時，會發生此錯誤。此錯誤通常是由代理程式或 containerd 執行時間的內部失敗所造成。

只有在使用平台版本 1.4.0 或更新版本 (Linux)，或 1.0.0 或更新版本 (Windows) 時，才會發生此錯誤。

如需有關如何偵錯和修正此問題的詳細資訊，請參閱為什麼我的 Amazon ECS 任務在 AWS Re: post 上停止。

Amazon ECS ContainerRuntimeTimeoutError 錯誤疑難排解

以下是您可以採取的一些 ContainerRuntimeTimeoutError 錯誤訊息和動作來修正錯誤。

無法轉換為執行中；等待 1m 或 Docker 逾時錯誤後逾時

當容器在逾時期間無法轉換成 RUNNING 或 STOPPED 狀態時，會發生此錯誤。錯誤訊息中會提供原因和逾時值。

Amazon ECS CannotStartContainerError 錯誤疑難排解

以下是您可以採取的一些 CannotStartContainerError 錯誤訊息和動作來修正錯誤。

無法取得容器狀態：<reason>

無法啟動容器時，會發生此錯誤。

Amazon ECS CannotStopContainerError 錯誤疑難排解

以下是您可以採取的一些 CannotStopContainerError 錯誤訊息和動作來修正錯誤。

CannotStopContainerError

無法停用容器時，會發生此錯誤。

如需有關如何偵錯和修正此問題的詳細資訊，請參閱為什麼我的 Amazon ECS 任務在 AWS Re: post 上停止。

Amazon ECS CannotInspectContainerError 錯誤疑難排解

以下是您可以採取的一些 CannotInspectContainerError 錯誤訊息和動作來修正錯誤。

CannotInspectContainerError

當容器代理程式無法透過容器執行時間描述容器時，會發生此錯誤。

使用平台版本1.3或更早版本時，Amazon ECS 代理程式會從 Docker 傳回原因。

使用平台版本1.4.0或更新版本 (Linux) 1.0.0 或更新版本 (Windows) 時，Fargate 代理程式會傳回原因。containerd

如需有關如何偵錯和修正此問題的詳細資訊，請參閱為什麼我的 Amazon ECS 任務在 AWS Re: post 上停止。

Amazon ECS CannotCreateVolumeError 錯誤疑難排解

以下是您可以採取的一些 CannotCreateVolumeError 錯誤訊息和動作來修正錯誤。

CannotCreateVolumeError

當代理程式無法建立任務定義中指定的磁碟區掛載時，會發生此錯誤。

只有在使用平台版本 1.4.0 或更新版本 (Linux)，或 1.0.0 或更新版本 (Windows) 時，才會發生此錯誤。

如需有關如何偵錯和修正此問題的詳細資訊，請參閱為什麼我的 Amazon ECS 任務在 AWS Re: post 上停止。

CannotPullContainer Amazon ECS 中的任務錯誤

下列錯誤表示工作無法啟動，因為 Amazon ECS 無法擷取指定的容器映像。

> **Note**
>
> 1.4 Fargate 平台版本會截斷長的錯誤訊息。

錯誤

- 工作無法提取影像。檢查角色是否具有從登錄提取映像的權限
- 工作無法提取影像。檢查您的網路組態
- API 錯誤（500）：獲取 https://111122223333.dkr.ecr.us-east-1.amazonaws.com/v2/：網絡/HTTP：在等待連接時取消請求
- API 錯誤

- 寫入 /var/lib/碼頭/tmp/ Blob1111111：設備上沒有剩GetImage餘空間
- 錯誤：過多請求：請求過多或您已達到提取率限制。
- 來自守護進程的錯誤響應：獲取網址：net/http：等待連接時取消請求
- ref pull 已重試 1 次:複製失敗:: 開啟失敗 httpReaderSeeker:非預期的狀態碼
- 提取訪問被拒絕
- pull 命令失敗：恐慌：運行時錯誤：無效的內存地址或零指針解除引用
- 拉圖像conf/提取圖像配置錯誤
- 內容已取消

工作無法提取影像。檢查角色是否具有從登錄提取映像的權限

此錯誤表示工作無法提取工作定義中指定的影像，因為權限問題。錯誤訊息中還有其他資訊，可提供影像或造成問題的角色。

「守護進程的錯誤響應：存*##*的提取訪問被拒絕，或者可能需要 'docker 登錄'：拒絕：用戶：role *AR* N 無權執行：ecr：在資源：*##BatchGetImage* 上，因為沒有基於身份的策略允許 ecr：操作。」BatchGetImage

若要解決此問題：

1. 檢查圖像是否存在於產品*#*。如需檢視影像的相關資訊，請參閱 Amazon 彈性容器登錄使用者指南中的在 Amazon ECR 中檢視影像詳細資料。
2. 驗證*##arn* 具有提取映像的正確權限。

   有關如何查看和修改角色的更多信息，請參閱《AWS Identity and Access Management 使用指南》中的修改角色。

   此工作會使用下列其中一個角色：

   - 對於具有 Fargate 啟動類型的工作，這是工作執行角色。如需 Amazon ECR 其他許可的相關資訊，透過界面端點許可提取 Amazon ECR 映像的 Fargate 端任務請參閱.
   - 對於具有 EC2 啟動類型的任務，這是容器執行個體角色。如需 Amazon ECR 其他許可的相關資訊，Amazon ECR 許可請參閱.

工作無法提取影像。檢查您的網路組態

此錯誤表示該任務無法連接到 Amazon ECR。

如需如何驗證和解決問題的相關資訊，請參閱驗證 Amazon ECS 停止任務連接。

API 錯誤（500）：獲取 https://111122223333.dkr.ecr.us-east-1.amazonaws.com/v2/：網絡/HTTP：在等待連接時取消請求

此錯誤表示連線逾時，因為網際網路的路由不存在。

要解決此問題，則可以採取下列方式：

- 針對公有子網路中的任務，您必須在啟動任務時，將 Auto-assign public IP (自動指派公有 IP) 指定為 ENABLED (啟用)。如需詳細資訊，請參閱 以 Amazon ECS 任務的形式執行應用程式。
- 針對在私有子網路中的任務，則必須在啟動任務時，將 Auto-assign public IP (自動指派公有 IP) 指定為 DISABLED (停用)，並設定 VPC 中的 NAT 閘道，以便將請求路由至網際網路。如需詳細資訊，請參閱 Amazon VPC 使用者指南中的 NAT 閘道。

API 錯誤

此錯誤表示 Amazon ECR 端點發生連線問題。

如需有關如何解決此問題的詳細資訊，請參閱如何解決網站上 Amazon ECS 中的 Amazon ECR 錯誤「CannotPullContainerError：API 錯誤」。 AWS Support

/var/lib/##/tmp/ Blob1111111#######GetImage###

此錯誤表示磁碟空間不足。

若要解決這個問題，請釋放磁碟空間。

如果您使用的是 Amazon ECS 最佳化 AMI，可以使用下列命令擷取檔案系統上 20 個最大的檔案：

```
du -Sh / | sort -rh | head -20
```

輸出範例：

```
5.7G    /var/lib/docker/
containers/50501b5f4cbf90b406e0ca60bf4e6d4ec8f773a6c1d2b451ed8e0195418ad0d2
1.2G    /var/log/ecs
594M    /var/lib/docker/devicemapper/mnt/
c8e3010e36ce4c089bf286a623699f5233097ca126ebd5a700af023a5127633d/rootfs/data/logs
...
```

在某些情況下，根磁碟區可能會由執行中的容器填滿。如果容器使用預設的 json-file 日誌驅動程式，而不具有 max-size 限制，則該日誌檔案可能佔用大部分空間。您可以使用 docker ps 命令，透過將目錄名稱從上面的輸出映射至容器 ID，來驗證哪個容器正在使用該空間。例如：

```
CONTAINER ID   IMAGE                                COMMAND             CREATED
 STATUS              PORTS                               NAMES
50501b5f4cbf   amazon/amazon-ecs-agent:latest   "/agent"            4 days ago
 Up 4 days                                              ecs-agent
```

在預設情況下，當使用 json-file 日誌驅動程式時，Docker 會擷取所有容器的標準輸出 (和標準錯誤)，並使用 JSON 格式將其寫入檔案中。您可以將 max-size 設定為日誌驅動程式選項，以防止日誌檔案佔用過多空間。如需詳細資訊，請參閱 Docker 文件中的設定日誌驅動程式。

下列是一個容器定義片段，顯示如何使用此選項：

```
{
    "log-driver": "json-file",
    "log-opts": {
        "max-size": "256m"
    }
}
```

另一種方法是，如果您的容器日誌佔用了太多的磁盤空間，則是使用日awslogs誌驅動程序。記awslogs錄驅動程式會將記錄檔傳送至 CloudWatch，以釋放磁碟空間，否則會用於容器執行個體上的容器記錄檔。如需詳細資訊，請參閱 將 Amazon ECS 日誌傳送到 CloudWatch 。

錯誤：過多請求：請求過多或您已達到提取率限制。

此錯誤表示存在 Docker 集線器速率限制。

如果收到下列其中一項錯誤訊息，您很可能會達到 Docker Hub 速率限制：

如需 Docker Hub 速率限制的詳細資訊，請參閱了解 Docker Hub 速率限制。

如果您提高了 Docker Hub 速率限制，並且需要針對容器執行個體驗證 Docker 提取，請參閱容器執行個體的專用登錄驗證。

來自守護進程的錯誤響應：獲取網#：net/http：等待連接時取消請求

此錯誤表示連線逾時，因為網際網路的路由不存在。

要解決此問題，則可以採取下列方式：

- 針對公有子網路中的任務，您必須在啟動任務時，將 Auto-assign public IP (自動指派公有 IP) 指定為 ENABLED (啟用)。如需詳細資訊，請參閱 以 Amazon ECS 任務的形式執行應用程式。
- 針對在私有子網路中的任務，則必須在啟動任務時，將 Auto-assign public IP (自動指派公有 IP) 指定為 DISABLED (停用)，並設定 VPC 中的 NAT 閘道，以便將請求路由至網際網路。如需詳細資訊，請參閱 Amazon VPC 使用者指南中的 NAT 閘道。

ref pull 已重試 1 次:複製失敗:: 開啟失敗 httpReaderSeeker:非預期的狀態碼

此錯誤表示複製影像時發生失敗。

若要解決此問題，請檢閱下列其中一篇文章：

- 如需 Fargate 任務，請參閱如何解決 Fargate 上 Amazon ECS 任務的 "cannotpullcontainererror" 錯誤。
- 如需其他任務，請參閱如何解決 Fargate 上 Amazon ECS 任務的 "cannotpullcontainererror" 錯誤。

提取訪問被拒絕

此錯誤表示無法存取影像。

若要解決此問題，您可能需要使用 Amazon ECR 驗證 Docker 用戶端如需詳細資訊，請參閱 Amazon ECR 使用者指南中的私人登錄身份驗證。

pull 命令失敗：恐慌：運行時錯誤：無效的內存地址或零指針解除引用

此錯誤表明由於無效的內存地址或零指針解除引用而無法訪問圖像。

若要解決此問題：

- 檢查您是否擁有可連接 Amazon S3 的安全群組規則。
- 使用閘道端點時，必須在路由表中新增路由，才能存取端點。

拉圖像conf/提取圖像配置錯誤

此錯誤表示已達到速率限制或發生網路錯誤：

若要解決此問題，請參閱如何解決 Amazon ECS EC2 啟動類型任務中的 CannotPull ContainerError「」錯誤。

內容已取消

此錯誤表示前後關聯已取消。

此錯誤的原因通常是因為您的任務所使用的 VPC 沒有路由，而無法從 Amazon ECR 提取容器映像。

驗證 Amazon ECS 停止任務連接

有時候工作會因為網路連線問題而停止。這可能是間歇性問題，但最有可能是因為工作無法連線到端點而造成。

測試任務連接

您可以使用 `AWSSupport-TroubleshootECSTaskFailedToStart` runbook 來測試任務連接。當您使用 runbook 時，您需要下列資源資訊：

- 任務識別碼

  使用最近失敗工作的 ID。
- 工作所在的叢集

如需有關如何使用 runnbook 的資訊，請參閱AWS Systems Manager 自動化工作流程簿參考資料[AWSSupport-TroubleshootECSTaskFailedToStart](#)中的。

工作流程簿會分析工作。您可以在「輸出」區段中檢視結果，以瞭解下列可能導致工作無法啟動的問題：

- 與已設定容器登錄的網路連線
- VPC 端點連線
- 安全性群組規則組態

修正 VPC 端點問題

當 `AWSSupport-TroubleshootECSTaskFailedToStart` runbook 結果指出 VPC 端點問題時，請檢查下列組態：

- 您在其中建立端點的 VPC 需要使用私有 DNS。
- 請確定您有工作無法在與工作相同 VPC 中連線的服務 AWS PrivateLink 端點。如需詳細資訊，請參閱下列其中一項：

服務	服務的 VPC 端點資訊	
Amazon ECR	Amazon ECR 接口 VPC 端端點 ()AWS PrivateLink	
Systems Manager	建立虛擬私人雲端端點	
Secrets Manager	使用 AWS Systems Manager VPC 端點	
CloudWatch	CloudWatch VPC 端點	
Amazon S3	AWS PrivateLink 對於 Amazon S3	

- 為工作子網路設定輸出規則，允許在連接埠 443 DNS (UDP 和 TCP) 流量上使用 HTTPS。如需詳細資訊，請參閱 Amazon 彈性運算雲端使用者指南中的將規則新增至安全群組。
- 如果子網路具有網路 ACL，則需要下列 ACL 規則：
  - 允許在通訊埠 1024-65535 上允許流量的輸出規則。
  - 允許連接埠 443 上的 TCP 流量的輸入規則。

  如需如何設定規則的詳細資訊，請參閱 Amazon Virtual Private Cloud 使用者指南中的使用網路 ACL 控制到子網路的流量。

修正網路問題

當 `AWSSupport-TroubleshootECSTaskFailedToStart` runbook 結果指出網路問題時，請檢查下列組態：

在公有子網路中使用 awsvpc 網路模式的工作

根據工作流程簿執行下列組態：

- 針對公有子網路中的任務，您必須在啟動任務時，將 Auto-assign public IP (自動指派公有 IP) 指定為 ENABLED (啟用)。如需詳細資訊，請參閱 以 Amazon ECS 任務的形式執行應用程式。
- 您需要一個網關來處理互聯網流量。工作子網路的路由表必須具有通往閘道的流量的路由。

  如需詳細資訊，請參閱 Amazon Virtual Private Cloud 使用者指南中的從路由表新增和移除路由表。

閘道類型	路線表目的地	路線表目標
NAT	0.0.0.0/0	NAT 閘道 ID
網際網路閘道	0.0.0.0/0	網際網路閘道 ID

- 如果工作子網路具有網路 ACL，則需要下列 ACL 規則：
  - 允許在通訊埠 1024-65535 上允許流量的輸出規則。
  - 允許連接埠 443 上的 TCP 流量的輸入規則。

  如需如何設定規則的詳細資訊，請參閱 Amazon Virtual Private Cloud 使用者指南中的使用網路 ACL 控制到子網路的流量。

在私有子網路中使用 awsvpc 網路模式的工作

根據工作流程簿執行下列組態：

- 啟動工作時，針對自動指派公用 IP 選擇 [已停用]。
- 在 VPC 中設定 NAT 閘道，將要求路由到網際網路。如需詳細資訊，請參閱 Amazon VPC 使用者指南中的 NAT 閘道。
- 工作子網路的路由表必須具有連至 NAT 閘道之流量的路由。

  如需詳細資訊，請參閱 Amazon Virtual Private Cloud 使用者指南中的從路由表新增和移除路由表。

閘道類型	路線表目的地	路線表目標
NAT	0.0.0.0/0	NAT 閘道 ID

- 如果工作子網路具有網路 ACL，則需要下列 ACL 規則：
  - 允許在通訊埠 1024-65535 上允許流量的輸出規則。
  - 允許連接埠 443 上的 TCP 流量的輸入規則。

  如需如何設定規則的詳細資訊，請參閱 Amazon Virtual Private Cloud 使用者指南中的使用網路 ACL 控制到子網路的流量。

在公共子網中不使用 awsvpc 網絡模式的任務

根據工作流程簿執行下列組態：

- 建立叢集時，在 Amazon EC2 執行個體的聯網下選擇開啟自動指派 IP。

  此選項會將公用 IP 位址指派給執行個體主要網路介面。
- 您需要一個網關來處理互聯網流量。執行個體子網路的路由表必須具有通往閘道的流量的路由。

  如需詳細資訊，請參閱 Amazon Virtual Private Cloud 使用者指南中的從路由表新增和移除路由表。

閘道類型	路線表目的地	路線表目標
NAT	0.0.0.0/0	NAT 閘道 ID
網際網路閘道	0.0.0.0/0	網際網路閘道 ID

- 如果執行個體子網路具有網路 ACL，則需要下列 ACL 規則：
  - 允許在通訊埠 1024-65535 上允許流量的輸出規則。
  - 允許連接埠 443 上的 TCP 流量的輸入規則。

  如需如何設定規則的詳細資訊，請參閱 Amazon Virtual Private Cloud 使用者指南中的使用網路 ACL 控制到子網路的流量。

在私有子網路中使用 awsvpc 網路模式的工作

根據工作流程簿執行下列組態：

- 建立叢集時，在 Amazon EC2 執行個體的聯網下選擇關閉自動指派 IP。
- 在 VPC 中設定 NAT 閘道，將要求路由到網際網路。如需詳細資訊，請參閱 Amazon VPC 使用者指南中的 NAT 閘道。
- 執行個體子網路的路由表必須具有 NAT 閘道流量的路由。

  如需詳細資訊，請參閱 Amazon Virtual Private Cloud 使用者指南中的從路由表新增和移除路由表。

閘道類型	路線表目的地	路線表目標
NAT	0.0.0.0/0	NAT 閘道 ID

- 如果工作子網路具有網路 ACL，則需要下列 ACL 規則：
  - 允許在通訊埠 1024-65535 上允許流量的輸出規則。
  - 允許連接埠 443 上的 TCP 流量的輸入規則。

  如需如何設定規則的詳細資訊，請參閱 Amazon Virtual Private Cloud 使用者指南中的使用網路 ACL 控制到子網路的流量。

查看 Amazon ECS 任務的 IAM 角色請求

當您在 IAM 角色中使用提供者作為任務登入資料時，提供者會請求儲存在稽核記錄中。稽核紀錄會繼承與容器代理程式紀錄相同的紀錄輪換設定。ECS_LOG_ROLLOVER_TYPE、ECS_LOG_MAX_FILE_SIZE_MB 和 ECS_LOG_MAX_ROLL_COUNT 容器代理程式組態變數可供設定來影響稽核日誌的行為。如需詳細資訊，請參閱 Amazon ECS 容器代理程式日誌組態參數。

針對 1.36.0 和更新版本的容器代理程式，稽核日誌會位在 `/var/log/ecs/audit.log`。紀錄輪換時，會在日誌檔名稱的結尾新增 *YYYY*-*MM*-*DD*-*HH* 格式的時間戳記。

針對 1.35.0 和更舊版本的容器代理程式，稽核日誌會位在 `/var/log/ecs/audit.log.`*YYYY*-*MM*-*DD*-*HH*。

日誌項目的格式如下：

- 時間戳記
- HTTP 回應代碼
- 請求來源的 IP 地址和連接埠號碼
- 登入資料提供者的相對 URI
- 提出請求的使用者代理
- 提出請求容器所屬之任務的 ARN
- `GetCredentials` API 名稱和版本編號
- 容器執行個體註冊對象的 Amazon ECS 叢集名稱
- 容器執行個體 ARN

您可以使用下列命令來檢視日誌檔案。

```
cat /var/log/ecs/audit.log.2016-07-13-16
```

輸出：

```
2016-07-13T16:11:53Z 200 172.17.0.5:52444 "/v1/credentials" "python-requests/2.7.0
 CPython/2.7.6 Linux/4.4.14-24.50.amzn1.x86_64" TASK_ARN GetCredentials
 1 CLUSTER_NAME CONTAINER_INSTANCE_ARN
```

檢視 Amazon ECS 服務事件訊息

對服務問題進行故障診斷時，您第一個應該檢查以取得診斷資訊的位置是服務事件日誌。您可以使用 `DescribeServices` API 檢視服務事件 AWS CLI，或使用 AWS Management Console.

使用 Amazon ECS API 檢視服務事件訊息時，只會傳回來自服務排程器的事件。其中包括最近的任務放置和執行個體運作狀態事件。不過，Amazon ECS 主控台會顯示下列來源的服務事件。

- 來自 Amazon ECS 服務排程器的任務放置和執行個體運作狀態事件。這些事件具有服務前綴 **### ###**。為了確保此事件檢視有幫助，我們只會顯示 100 個最近事件，在解決原因或經過六個小時之前，會忽略重複事件訊息。如果原因未在六小時內解決，您會收到另一則有關該原因的服務事件訊息。
- Service Auto Scaling 事件。這些事件的前置詞為「訊息」。系統會顯示 10 個最近擴展事件。只有在使用 Application Auto Scaling 擴展政策設定服務時，才會發生這些事件。

請依照下列步驟檢視您目前的服務事件訊息。

Console

1. 開啟主控台，網址為 https://console.aws.amazon.com/ecs/v2。
2. 在導覽窗格中，選擇叢集。
3. 在叢集頁面上，選擇叢集。
4. 選擇要檢查的服務。
5. 選擇 Deployments and events (部署和事件)，然後在 Events (事件) 下檢視訊息。

AWS CLI

使用 describe-services 命令來檢視指定服務的服務事件訊息。

下列 AWS CLI 範例說明##叢集中的服務##服務，此服務將提供最新的服務事件訊息。

```
aws ecs describe-services \
```

```
    --cluster default \
    --services service-name \
    --region us-west-2
```

Amazon ECS 服務事件消息

您會在 Amazon ECS 主控台中看到以下服務事件訊息範例：

服務 (*service-name*) 已達到穩定狀態。

服務排程器會在`service (`*service-name*`) has reached a steady state.`服務狀況良好且處於所需的工作數目時傳送服務事件，進而達到穩定狀態。

服務排程器會定期報告狀態，因此您可能會多次收到此訊息。

服務 (*service-name*) 無法放置任務，因為沒有容器執行個體符合其所有要求。

當服務排程器找不到可用資源來新增其他工作時，會傳送此事件訊息。可能的原因如下：

您的叢集中找不到任何容器執行個體

如果您嘗試在其中執行工作的叢集中沒有註冊容器執行個體，就會收到此錯誤。您應該將容器執行個體新增到您的叢集。如需詳細資訊，請參閱 啟動 Amazon ECS Linux 容器執行個體。

連接埠不足

如果您的任務使用固定的主機連接埠映射 (例如，您的任務為 Web 伺服器使用主機的連接埠 80)，則您每項任務至少必須擁有一個容器執行個體，因為一個容器一次只能使用單一個主機連接埠。您應該新增容器執行個體到您的叢集，或降低所需任務數量。

註冊的連接埠太多

任務放置最接近的相符容器執行個體不得超過每個容器執行個體 100 個主機連接埠允許的最大保留連接埠限制。使用動態主機連接埠映射可以修復此問題。

連接埠已在使用中

此工作的作業定義會在其連接埠對應中使用相同的連接埠，作為已在所選容器執行個體上執行的工作。服務事件訊息會將具有所選容器執行個體 ID 作為以下訊息的一部分。

```
The closest matching container-instance is already using a port required by your
 task.
```

記憶體不足

如果您的任務定義指定 1000 MiB 的記憶體，而您叢集中的容器執行個體每個都有 1024 MiB 的記憶體，則您每個容器執行個體只能執行一個此任務複本。您可以在任務定義中試驗使用較少的記憶體，以便您每個容器執行個體可以啟動多項任務，或在您的叢集中啟動更多的容器執行個體。

> **Note**
>
> 若您嘗試盡可能為特定執行個體類型的任務提供最多的記憶體，以將資源使用率最大化，請參閱「保留 Amazon ECS Linux 容器執行個體記憶體」。

CPU 不足

容器執行個體的每個 CPU 核心都有 1,024 CPU 單位。如果您的任務定義指定 1,000 CPU 單位，而您叢集中的容器執行個體每個都有 1,024 CPU 單位，您每個容器執行個體只能執行一個此任務複本。您可以在任務定義中試驗使用較少的 CPU 單位，以便您每個容器執行個體可以啟動多項任務，或在您的叢集中啟動更多的容器執行個體。

可用的 ENI 連接點不足

使用 `awsvpc` 網路模式的每項任務都會收到自己的彈性網路介面 (ENI)，連接到裝載它的容器執行個體。Amazon EC2 執行個體有可以連接至其的 ENI 數目限制，而且有 ENI 容量可用的叢集中沒有容器執行個體。

個別容器執行個體的 ENI 限制取決於以下條件：

- 如果您尚未選擇使用 `awsvpcTrunking` 帳戶設定，則每個容器執行個體的 ENI 限制取決於執行個體類型。如需詳細資訊，請參閱《Amazon EC2 使用者指南》中的每個執行個體類型每個網路介面的 IP 地址。
- 如果您已選擇加入`awsvpcTrunking`帳戶設定，但在選擇加入後並未使用支援的執行個體類型啟動新的容器執行個體，則每個容器執行個體的 ENI 限制仍為預設值。如需詳細資訊，請參閱《Amazon EC2 使用者指南》中的每個執行個體類型每個網路介面的 IP 地址。
- 如果您已選擇使用 `awsvpcTrunking` 帳戶設定，且已在選擇使用後，使用支援的執行個體類型啟動新的容器執行個體，則會有額外的 ENI 可用。如需詳細資訊，請參閱 支援增加 Amazon ECS 容器網路界面的執行個體。

如需選擇使用的 `awsvpcTrunking` 帳戶設定詳細資訊，請參閱 增加 Amazon ECS Linux 容器實例網絡界面。

您可以將容器執行個體新增到您的叢集，以提供更多可用的網路轉接器。

容器執行個體遺漏必要的屬性

有些任務定義參數需要在容器執行個體上安裝特定的 Docker 遠端 API 版本。其他，例如記錄驅動程式選項，需要容器執行個體向 ECS_AVAILABLE_LOGGING_DRIVERS 代理組態變數註冊這些日誌驅動程式。如果您的工作定義包含需要特定容器執行個體屬性的參數，而且您沒有任何可用的容器執行個體可以滿足此需求，則無法放置工作。

造成此錯誤的常見原因是，如果您的服務使用使用awsvpc網路模式和 EC2 啟動類型的工作。您指定的叢集沒有在建立服務awsvpcConfiguration時所指定的相同子網路中註冊容器執行個體。

有關特定任務定義參數和代理組態變數需要哪些必要屬性的詳細資訊，請參閱「Amazon ECS 任務定義參數」和「Amazon ECS 容器代理程式組態」。

服務 (*service-name*) 無法放置任務，因為沒有容器執行個體符合其所有要求。最接近的相符容器執行個體 (*###### id*) 可用 CPU 單位不足。

用於任務放置的最接近的匹配容器實例不包含足夠的 CPU 單元，無法滿足任務定義中的需求。檢閱任務定義之任務大小和容器定義參數的 CPU 需求。

服務 (*service-name*) 無法放置任務，因為沒有容器執行個體符合其所有要求。最接近的相符容器執行個體 *container-instance-id* 發生「代理」錯誤。

適合放置任務之最接近相符容器執行個體中的 Amazon ECS 容器代理中斷連線。如果您可以使用 SSH 連線到容器執行個體，您即可檢查代理日誌，如需詳細資訊，請參閱「Amazon ECS 容器代理程式日誌組態參數」。您也應該確認代理是否在執行個體上執行。如果您使用的是 Amazon ECS 最佳化 AMI，您可以使用下列命令嘗試停用並重新啟動代理。

- 對於 Amazon ECS 優化的 Amazon Linux 2 AMI 和 Amazon ECS 優化 Amazon Linux 2023 AMI

```
sudo systemctl restart ecs
```

- 對於 Amazon ECS 最佳化 Amazon Linux AMI

```
sudo stop ecs && sudo start ecs
```

服務 (*####*) (執行個體*#### ID*) 在 (elb *elb-name*) 中運作狀態不良，原因是執行個體至少連續執行狀態檢查次數失敗。) UnhealthyThreshold

此服務已在負載平衡器註冊，但負載平衡器的運作狀態檢查失敗。如需詳細資訊，請參閱 疑難排解 Amazon ECS 中的服務負載平衡器。

服務 (*service-name*) 無法成功持續啟動任務。

此服務包含在連續嘗試後無法啟動的任務。此時，服務排程器開始逐漸增加重試之間的時間。您應該對任務無法啟動的原因進行故障診斷。如需詳細資訊，請參閱 Amazon ECS 服務節流邏輯。

服務更新之後，例如使用更新的任務定義，服務排程器就會恢復正常的行為。

服務 (*service-name*) 操作正在受到調節。將稍後再試。

由於 API 調節限制，此服務無法啟動更多任務。一旦服務排程器能夠啟動更多任務，就會繼續執行。

若要請求提高 API 比率限制配額，請開啟 AWS Support Center (中心) 頁面，並視需要登入，然後選擇 Create case (建立案例)。選擇 Service limit increase (提高服務限制)。填妥並提交表格。

服務 (*service-name*) 在部署期間因服務部署組態而無法停止或啟動任務。請更新 minimumHealthyPercent 或「最大百分比」值，然後再試一次。

由於部署組態，此服務無法在服務部署期間停止或啟動任務。部署組態包含`minimumHealthyPercent`和`maximumPercent`值，這些值是在建立服務時定義的。這些值也可以在現有服務上進行更新。

`minimumHealthyPercent`表示在部署期間或容器執行個體耗盡時，應針對服務執行的工作數目下限。這是服務所需任務數量的百分比。此值會向上捨入。例如，如果健全狀況下限百分比為，`50`且所需的工作計數為四，則排程器可以先停止兩個現有工作，然後再開始兩個新工作。同樣地，如果最小運作狀態良好的百分比為 75%，而所需的任務計數為 2，則排程器無法停止任何任務，因為產生的值也是 2。

`maximumPercent`表示在部署期間或容器執行個體耗盡時，應針對服務執行的工作數目上限。這是服務所需任務數量的百分比。此值會向下捨去。例如，如果最大百分比為，`200`且所需的工作計數為四，則排程器可以在停止四個現有工作之前啟動四個新工作。同樣地，如果最大百分比為 125，且所需的任務計數為 3，則排程器無法停止任何任務，因為產生的值也是 3。

設定最小最小運作狀態良好的百分比或最大百分比時，您應確定排程器在觸發部署時，至少可以停止或啟動一項任務。

服務 (*service-name*) 無法放置任務。原因：您已達到可以同時執行的任務數量限制

您可以請求提升導致錯誤的資源配額。如需詳細資訊，請參閱 Service Quotas。若要請求增加配額，請參閱《Service Quotas 使用者指南》中的請求提高配額。

服務 (*service-name*) 無法放置任務。原因：內部錯誤。

下列為導致此錯誤的可能原因：

- 由於子網路位於不受支持的可用區域中，該服務無法開始任務。

  如需支援的區域的 Fargate 和可用區域的相關資訊，請參閱 the section called “AWS Fargate 地區”。

  如需如何檢視子網路可用區域的相關資訊，請參閱《Amazon VPC 使用者指南》中的檢視您的子網路。
- 您正在嘗試在 Fargate 點上運行使用 ARM 架構的任務定義。

服務 (*service-name*) 無法放置任務。原因：請求的 CPU 組態超過您的限制。

您可以請求提升導致錯誤的資源配額。如需詳細資訊，請參閱 Service Quotas。若要請求增加配額，請參閱《Service Quotas 使用者指南》中的請求提高配額。

服務 (*service-name*) 無法放置任務。原因：請求的 MEMORY 組態超過您的限制。

您可以請求提升導致錯誤的資源配額。如需詳細資訊，請參閱 Service Quotas。若要請求增加配額，請參閱《Service Quotas 使用者指南》中的請求提高配額。

服務 (*service-name*) 無法放置任務。原因：您已達到可以同時執行的 vCPU 數量限制

AWS Fargate 正在從以工作計數為基礎的配額轉換為以 vCPU 為基礎的配額。

對於 Fargate vCPU 型配額，您可請求提高配額。如需詳細資訊，請參閱 Service Quotas。若要請求提高 Fargate 配額，請參閱《Service Quotas 使用者指南》中的請求提高配額。

服務 (*service-name*) 無法達到穩定狀態，因為任務集 (*taskSet-ID*) 無法縮減。原因：受保護任務的數量超過所需任務的計數。

服務具有的受保護任務數超過所需任務計數。您可以執行下列任一動作：

- 等待目前任務的保護到期，然後終止這些任務。
- 判斷哪些工作可以停止，並使用 UpdateTaskProtection API，並將選protectionEnabled項設定false為取消設定這些工作的保護。
- 將服務的所需任務計數提高到超過受保護任務的數目。

服務 (*service-name*) 無法達到穩定狀態。原因：在容量提供者中找不到容器執行個體。

當服務排程器找不到可用資源來新增其他工作時，會傳送此事件訊息。可能的原因如下：

沒有與叢集相關聯的容量提供者

用describe-services於確認您有與叢集相關聯的容量提供者您可以更新服務的容量提供者策略。

確認容量提供者中有可用容量。在 EC2 啟動類型的情況下，請確定容器執行個體符合任務定義要求。

您的叢集中找不到任何容器執行個體

如果您嘗試在其中執行工作的叢集中沒有註冊容器執行個體，就會收到此錯誤。您應該將容器執行個體新增到您的叢集。如需詳細資訊，請參閱 啟動 Amazon ECS Linux 容器執行個體。

連接埠不足

如果您的工作使用固定主機連接埠對應 (例如，您的工作使用 Web 伺服器主機上的連接埠 80)，則每個工作必須至少有一個容器執行個體。一次只能有一個容器使用單一主機連接埠。您應該新增容器執行個體到您的叢集，或降低所需任務數量。

註冊的連接埠太多

任務放置最接近的相符容器執行個體不得超過每個容器執行個體 100 個主機連接埠允許的最大保留連接埠限制。使用動態主機連接埠映射可以修復此問題。

連接埠已在使用中

此工作的作業定義會在其連接埠對應中使用相同的連接埠，作為已在所選容器執行個體上執行的工作。服務事件訊息會將具有所選容器執行個體 ID 作為以下訊息的一部分。

```
The closest matching container-instance is already using a port required by your
 task.
```

記憶體不足

如果您的任務定義指定 1000 MiB 的記憶體，而您叢集中的容器執行個體每個都有 1024 MiB 的記憶體，則您每個容器執行個體只能執行一個此任務複本。您可以在任務定義中試驗使用較少的記憶體，以便您每個容器執行個體可以啟動多項任務，或在您的叢集中啟動更多的容器執行個體。

> **Note**
>
> 若您嘗試盡可能為特定執行個體類型的任務提供最多的記憶體，以將資源使用率最大化，請參閱「保留 Amazon ECS Linux 容器執行個體記憶體」。

可用的 ENI 連接點不足

使用 awsvpc 網路模式的每項任務都會收到自己的彈性網路介面 (ENI)，連接到裝載它的容器執行個體。Amazon EC2 執行個體對可連接的 ENI 數量有限制，而且叢集中沒有可用 ENI 容量的容器執行個體。

個別容器執行個體的 ENI 限制取決於以下條件：

- 如果您尚未選擇使用 awsvpcTrunking 帳戶設定，則每個容器執行個體的 ENI 限制取決於執行個體類型。如需詳細資訊，請參閱《Amazon EC2 使用者指南》中的每個執行個體類型每個網路介面的 IP 地址。
- 如果您已選擇加入awsvpcTrunking帳戶設定，但在選擇加入後並未使用支援的執行個體類型啟動新的容器執行個體，則每個容器執行個體的 ENI 限制仍為預設值。如需詳細資訊，請參閱《Amazon EC2 使用者指南》中的每個執行個體類型每個網路介面的 IP 地址。
- 如果您已選擇使用 awsvpcTrunking 帳戶設定，且已在選擇使用後，使用支援的執行個體類型啟動新的容器執行個體，則會有額外的 ENI 可用。如需詳細資訊，請參閱 支援增加 Amazon ECS 容器網路界面的執行個體。

如需選擇使用的 awsvpcTrunking 帳戶設定詳細資訊，請參閱 增加 Amazon ECS Linux 容器實例網絡界面。

您可以將容器執行個體新增到您的叢集，以提供更多可用的網路轉接器。

容器執行個體遺漏必要的屬性

有些任務定義參數需要在容器執行個體上安裝特定的 Docker 遠端 API 版本。其他，例如記錄驅動程式選項，需要容器執行個體向 ECS_AVAILABLE_LOGGING_DRIVERS 代理組態變數註冊這些日誌驅動程式。如果您的工作定義包含需要特定容器執行個體屬性的參數，而且您沒有任何可用的容器執行個體可以滿足此需求，則無法放置工作。

造成此錯誤的常見原因是，如果您的服務正在使用使用awsvpc網路模式和 EC2 啟動類型的工作，而您指定的叢集並未在建立服務awsvpcConfiguration時指定的相同子網路中向其註冊容器執行個體。

有關特定任務定義參數和代理組態變數需要哪些必要屬性的詳細資訊，請參閱「Amazon ECS 任務定義參數」和「Amazon ECS 容器代理程式組態」。

服務 (*service-name*) 無法放置任務。原因：目前無法使用容量。請稍後再試一次，或在不同的可用區域中嘗試。

目前沒有可用的容量可用來執行您的服務。

您可以執行下列任一動作：

- 等到 Fargate 容量或 EC2 容器執行個體可以使用。
- 重新啟動服務並指定其他子網路。

服務 (服###) 部署失敗:工作啟動失敗。

服務中的工作無法啟動。

如需有關如何偵錯已停止之工作的資訊。請參閱Amazon ECS 停止任務錯誤消息。

服務 (####) 逾時等待 Amazon ECS 代理程式啟動。請檢查 /var/log/ecs/ecs-agent.log 中的日誌「。

適合放置任務之最接近相符容器執行個體中的 Amazon ECS 容器代理中斷連線。如果您可以使用 SSH 連線至容器執行個體，則可以檢查代理程式記錄。如需詳細資訊，請參閱 Amazon ECS 容器代理程式日誌組態參數。您也應該確認代理是否在執行個體上執行。如果您使用的是 Amazon ECS 最佳化 AMI，您可以使用下列命令嘗試停用並重新啟動代理。

- 對於 Amazon ECS 最佳化 Amazon Linux 2 AMI

```
sudo systemctl restart ecs
```

- 對於 Amazon ECS 最佳化 Amazon Linux AMI

```
sudo stop ecs && sudo start ecs
```

ID## targetGroup#####-ARN############TARGET GROUP IS NOT FOUND

為服務設定的工作未通過健全狀況檢查，因為找不到目標群組。您應該刪除並重新建立服務。除非已刪除對應的 Amazon ECS 服務，否則請勿刪除任何 Elastic Load Balancing 目標群組。

ID## targetGroup#####-ARN############TARGET IS NOT FOUND

為服務設定的工作未通過健康狀態檢查，因為找不到目標。

疑難排解 Amazon ECS 中的服務負載平衡器

Amazon ECS 服務可在 Elastic Load Balancing 負載平衡器註冊任務。負載平衡器組態錯誤是常見的任務停止原因。如果您的停止任務是服務使用負載平衡器所啟動，請考慮以下可能的原因。

Amazon ECS 服務連結角色不存在

Amazon ECS 服務連結的角色可讓 Amazon ECS 服務向 Elastic Load Balancing 負載平衡器註冊容器執行個體。必須在您的帳戶中建立服務連結角色。如需詳細資訊，請參閱 使用 Amazon ECS 的服務連結角色。

容器實例安全性群組

如果您的容器映射到您容器執行個體的連接埠 80，則您的容器執行個體安全群組必須允許連接埠 80 傳入流量，以讓負載平衡器的運作狀態檢查通過。

未針對所有可用區域設定 Elastic Load Balancing 器

您的負載平衡器應該設定使用一個區域中的所有可用區域，或至少是您容器執行個體所在的所有可用區域。如果服務使用負載平衡器，並在位於未設定負載平衡器使用的可用區域中的容器執行個體上啟動工作，則工作永遠不會通過健全狀況檢查。這會導致任務被殺死。

Elastic Load Balancing 負載平衡器健康檢查設定錯誤

負載平衡器的運作狀態檢查參數可能太過侷限或指向不存在的資源。如果容器執行個體被判定為健康狀態不佳，就會從負載平衡器中移除該執行個體。請務必確認為您的服務負載平衡器正確設定以下參數。

Ping 連接埠

負載平衡器運作狀態檢查的 Ping Port (Ping 連接埠) 值，是負載平衡器檢查判斷運作狀態是否良好之容器執行個體的連接埠。如果此連接埠設定錯誤，負載平衡器可能會從本身取消註冊您的容器執行個體。此連接埠應設定使用您處理運作狀態檢查之服務任務定義容器的 `hostPort` 值。

Ping 路徑

這是負載平衡器健康狀態檢查的一部分。它是應用程式上的端點，可在應用程式狀態良好時重新啟動成功的狀態碼 (例如 200)。此值通常設為 `index.html`，但如果您的服務不回應該請求，則運作狀態檢查會失敗。如果您的容器沒有 `index.html` 檔案，您可以將此設定為 `/`，以容器執行個體的基本 URL 為目標。

回應逾時

這是您的容器必須傳回運作狀態檢查 ping 回應的時間。如果此值低於回應所需的時間，則運作狀態檢查會失敗。

運作狀態檢查間隔

這是運作狀態檢查 ping 之間的時間。運作狀態檢查的間隔愈短，您的容器執行個體就愈快到達 Unhealthy Threshold (狀態不良閾值)。

狀態不良閾值

這是您的運作狀態檢查失敗前，您的容器執行個體視為運作狀態不良的次數。如果您的狀況不良臨界值為 2，且健康狀態檢查間隔為 30 秒，則您的工作有 60 秒的時間來回應健康狀態檢查 ping，然後再將其假定為不健康狀態。您可以提高運作狀態不良閾值或運作狀態檢查間隔，讓您的任務有更多的時間回應。

無法更新服務**####**：工作定義中的負載平衡器容器名稱或連接埠已變更

如果您的服務使用負載平衡器，您可以使用 AWS CLI 或 SDK 來修改負載平衡器組態。如需如何修改組態的相關資訊，請參閱 Amazon 彈性容器服務 API 參考UpdateService中的。如果您為服務更新任務定義，則在負載平衡器組態中指定的容器名稱和容器連接埠必須保留在任務定義中。

您已達到可同時執行的工作數目上限。

若為新帳戶，您的配額可能低於服務配額。您可以在 Service Quotas 主控台檢視您的帳戶的服務配額。若要請求提高配額，請參閱《Service Quotas 使用者指南》中的請求提高配額。

疑難排解 Amazon ECS 中的服務 auto 擴展

應用程式 Auto Scaling 會在 Amazon ECS 部署進行時關閉擴充程序，並在部署完成後繼續進行。不過，除非在部署期間暫停，否則向外擴展程序會繼續發生。如需詳細資訊，請參閱暫停和繼續擴展 Application Auto Scaling。

疑難排解 Amazon ECS 任務定義無效的 CPU 或記憶體錯誤

使用 Amazon ECS API 註冊任務定義時 AWS CLI，或者如果指定無效cpu或memory值，則會傳回下列錯誤。

```
An error occurred (ClientException) when calling the RegisterTaskDefinition operation:
 Invalid 'cpu' setting for task.
```

> **Note**
>
> 使用 Terraform 時，可能會返回以下錯誤。
>
> ```
> Error: ClientException: No Fargate configuration exists for given values.
> ```

若要解決這個問題，您必須為您任務定義中的任務 CPU 和記憶體指定受支援的值。此cpu值可以在任務定義中以 CPU 單位或 vCPUs 表示。註冊任務定義時，它會轉換為指示 CPU 單位的整數。此memory值可以在工作定義中以 MiB 或 GB 表示。註冊任務定義時，它會轉換為表示 MIB 的整數。

對於僅為 `requiresCompatibilities` 參數指定 EC2 的任務定義，則支援的 CPU 值介於 256 個 CPU 單位 (0.25 vCPU) 和 16384 個 CPU 單位 (16 vCPU) 之間。記憶體值必須是整數，且限制取決於您使用的基礎 Amazon EC2 執行個體上的可用記憶體數量。

對於為requiresCompatibilities參數指定FARGATE的工作定義 (即使EC2也已指定)，您必須使用下表中的其中一個值。這些值會決定您支援的 CPU 和記憶體參數值範圍。

對於在 Fargate 上託管的任務，下表顯示了有效的 CPU 和記憶體組合。JSON 檔案中的記憶體值是以 MiB 為單位。您可以將 GB 值乘以 1024，將 GB值 轉換為 MiB。例如 1 GB = 1024 MiB。

CPU 數值	記憶體數值	AWS Fargate 支援的作業系統
256 (.25 vCPU)	512 MiB、1 GB、2 GB	Linux

CPU 數值	記憶體數值	AWS Fargate 支援的作業系統
512 (.5 vCPU)	1 GB、2 GB、3 GB、4 GB	Linux
1024 (1 vCPU)	2 GB、3 GB、4 GB、5 GB、6 GB、7 GB、8 GB	Linux、Windows
2048 (2 vCPU)	介於 4 GB 與 16 GB 之間，以 1 GB 為單位遞增	Linux、Windows
4096 (4 vCPU)	介於 8 GB 與 30 GB 之間，以 1 GB 為單位遞增	Linux、Windows
8192 (8 vCPU) **Note** 此選項需要 Linux 平台 `1.4.0` 或更新版本。	介於 16 GB 與 60 GB 之間，以 4 GB 為單位遞增	Linux
16384 (16vCPU) **Note** 此選項需要 Linux 平台 `1.4.0` 或更新版本。	介於 32 GB 與 120 GB 之間，以 8 GB 為單位遞增	Linux

對於在 Amazon EC2 上託管的任務，支援的任務 CPU 值介於 0.25 個 vCPUs 和 192 個 vCPUs 之間。

Note

Windows 容器會忽略任務層級的 CPU 和記憶體參數。

檢視 Amazon ECS 容器代理程式日誌

Amazon ECS 會將日誌存放在您容器執行個體的 /var/log/ecs 資料夾中。您可以從 Amazon ECS 容器代理程式和 ecs-init 服務中取得日誌，該服務控制容器執行個體上的代理程式狀態 (開始/停止)。您可以使用 SSH 連線到容器執行個體來檢視這些日誌檔。

> **Note**
>
> 如果您不確定如何收集您容器執行個體中的所有日誌，您可以使用 Amazon ECS 日誌收集器。如需詳細資訊，請參閱 使用 Amazon ECS 日誌收集器收集容器日誌 。

Linux 作業系統

ecs-init 程序會將日誌存放在 /var/log/ecs/ecs-init.log。

此ecs-init.log檔案包含有關 Continer 代理程式生命週期管理、設定和啟動載入的資訊。

您可以使用下列命令來檢視日誌檔案。

```
cat /var/log/ecs/ecs-init.log
```

輸出：

```
2018-02-16T18:13:54Z [INFO] pre-start
2018-02-16T18:13:56Z [INFO] start
2018-02-16T18:13:56Z [INFO] No existing agent container to remove.
2018-02-16T18:13:56Z [INFO] Starting Amazon Elastic Container Service Agent
```

Windows 作業系統

您可以使用適用於視窗的 Amazon ECS 日誌收集器。如需詳細資訊，請參閱 Github 上的 Amazon ECS 視窗日誌收集器。

1. 連線到您的執行個體。
2. 開啟， PowerShell 然後以系統管理權限執行下列命令。這些命令會下載指令碼並收集記錄檔。

```
Invoke-WebRequest -OutFile ecs-logs-collector.ps1 https://
raw.githubusercontent.com/awslabs/aws-ecs-logs-collector-for-windows/master/ecs-
logs-collector.ps1
```

```
.\ecs-logs-collector.ps1
```

您可以開啟 Amazon ECS 代理程式和 Docker 精靈的偵錯記錄功能。此選項可讓指令碼在開啟偵錯模式之前收集記錄檔。指令碼會重新啟動 Docker 精靈和 Amazon ECS 代理程式，然後終止執行個體上執行的所有容器。執行下列命令之前，請先排空容器執行個體，並將任何重要工作移至其他容器執行個體。

執行下列命令以開啟記錄功能。

```
.\ecs-logs-collector.ps1 -RunMode debug
```

使用 Amazon ECS 日誌收集器收集容器日誌

如果您不確定如何收集您容器執行個體中的所有各種日誌，您可以使用 Amazon ECS 日誌收集器。它可以在 GitHub Linux 和視窗上使用。此指令碼會收集一般作業系統日誌以及 Docker 和 Amazon ECS 容器代理程式日誌，這對於疑難排解 AWS Support 案例很有幫助。然後，它會將所收集的資訊壓縮並封存為單一檔案，輕鬆共用於診斷目的。也支援為 Docker 常駐程式和 Amazon Linux 變體上的 Amazon ECS 容器代理程式 (例如 Amazon ECS 最佳化 AMI) 啟用除錯模式。目前，Amazon ECS 日誌收集器支援下列作業系統：

- Amazon Linux
- Red Hat Enterprise Linux 7
- Debian 8
- Ubuntu 14.04
- Ubuntu 16.04
- Ubuntu 18.04
- Windows Server 2016

> **Note**
>
> Amazon ECS 日誌收集器的原始程式碼可在 Linux 和視窗上 GitHub 使用。我們鼓勵您為想要進行的變更提交提取請求。然而，Amazon Web Services 目前不支援執行此軟體經修改的複本。

下載並執行 Linux 版的 Amazon ECS 日誌收集器

1. 連線到您的容器執行個體。
2. 下載 Amazon ECS 日誌收集器指令碼。

```
curl -O https://raw.githubusercontent.com/awslabs/ecs-logs-collector/master/ecs-logs-collector.sh
```

3. 執行指令碼以收集日誌並建立封存。

> **Note**
>
> 若要啟用 Docker 精靈和 Amazon ECS 容器代理程式的除錯模式，請將`--mode=enable-debug`選項新增至下列命令。這可能會重新啟動 Docker 常駐程式，這會終止執行個體上執行的所有容器。請考慮耗盡容器執行個體，並將任何重要的任務先移到其他容器執行個體，再啟用除錯模式。如需詳細資訊，請參閱 排空 Amazon ECS 容器實例。

```
[ec2-user ~]$ sudo bash ./ecs-logs-collector.sh
```

在您執行指令碼之後，您可以在指令碼建立的 `collect` 資料夾中檢查收集到的日誌。
該`collect.tgz`文件是所有日誌的壓縮歸檔，您可以與其共享以 AWS Support 獲取診斷幫助。

下載並執行 Windows 版的 Amazon ECS 日誌收集器

1. 連線到您的容器執行個體。如需詳細資訊，請參閱 Amazon EC2 使用者指南中的連線到 Windows 執行個體。
2. 使用下載 Amazon ECS 日誌收集器 PowerShell腳本。

```
Invoke-WebRequest -OutFile ecs-logs-collector.ps1 https://raw.githubusercontent.com/awslabs/aws-ecs-logs-collector-for-windows/master/ecs-logs-collector.ps1
```

3. 執行指令碼以收集日誌並建立封存。

> **Note**
>
> 若要啟用 Docker 精靈和 Amazon ECS 容器代理程式的除錯模式，請將-RunMode debug選項新增至下列命令。這會重新啟動 Docker 常駐程式，它會終止在執行個體上執行的所有容器。請考慮耗盡容器執行個體，並將任何重要的任務先移到其他容器執行個體，再啟用除錯模式。如需詳細資訊，請參閱 排空 Amazon ECS 容器實例。

```
.\ecs-logs-collector.ps1
```

在您執行指令碼之後，您可以在指令碼建立的 `collect` 資料夾中檢查收集到的日誌。此`collect.tgz`檔案是所有記錄檔的壓縮封存檔，您可以與 Sup AWS port 人員共用以取得診斷協助。

使用代理程式自檢功能擷取 Amazon ECS 診斷詳細資料

Amazon ECS 代理程式內省 API 提供 Amazon ECS 代理程式和容器執行個體的整體狀態相關資訊。

您可以使用代理程式內部檢查 API 來取得工作中容器的 Docker ID。您可以使用 SSH 連線到容器執行個體，使用代理自我檢查 API。

> **Important**
>
> 您的容器執行個體必須擁有允許存取 Amazon ECS 以連接自我檢查 API 的 IAM 角色。如需詳細資訊，請參閱 Amazon ECS 容器執行個體 IAM 角色。

下列範例會顯示兩個工作，一個工作目前正在執行，另一個工作已停止。

> **Note**
>
> 以下命令通過管道進行傳輸，以python -mjson.tool提高可讀性。

```
curl http://localhost:51678/v1/tasks | python -mjson.tool
```

輸出：

```
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  1095  100  1095    0     0   117k      0 --:--:-- --:--:-- --:--:--  133k
{
    "Tasks": [
        {
            "Arn": "arn:aws:ecs:us-west-2:aws_account_id:task/090eff9b-1ce3-4db6-848a-
a8d14064fd24",
            "Containers": [
                {
                    "DockerId":
 "189a8ff4b5f04affe40e5160a5ffadca395136eb5faf4950c57963c06f82c76d",
                    "DockerName": "ecs-console-sample-app-static-6-simple-
app-86caf9bcabe3e9c61600",
                    "Name": "simple-app"
                },
                {
                    "DockerId":
 "f7f1f8a7a245c5da83aa92729bd28c6bcb004d1f6a35409e4207e1d34030e966",
                    "DockerName": "ecs-console-sample-app-static-6-busybox-
ce83ce978a87a890ab01",
                    "Name": "busybox"
                }
            ],
            "Family": "console-sample-app-static",
            "KnownStatus": "STOPPED",
            "Version": "6"
        },
        {
            "Arn": "arn:aws:ecs:us-west-2:aws_account_id:task/1810e302-eaea-4da9-
a638-097bea534740",
            "Containers": [
                {
                    "DockerId":
 "dc7240fe892ab233dbbcee5044d95e1456c120dba9a6b56ec513da45c38e3aeb",
                    "DockerName": "ecs-console-sample-app-static-6-simple-app-
f0e5859699a7aecfb101",
                    "Name": "simple-app"
                },
                {
                    "DockerId":
 "096d685fb85a1ff3e021c8254672ab8497e3c13986b9cf005cbae9460b7b901e",
```

```
                    "DockerName": "ecs-console-sample-app-static-6-
busybox-92e4b8d0ecd0cce69a01",
                    "Name": "busybox"
                }
            ],
            "DesiredStatus": "RUNNING",
            "Family": "console-sample-app-static",
            "KnownStatus": "RUNNING",
            "Version": "6"
        }
    ]
}
```

在上述範例中，已停止的工作 (*090 ## -1ce3-4db6-848-* a8d14064fd24) 有兩個容器。您可以使用 docker inspect ***container-ID*** 以檢視每個容器的詳細資訊。如需詳細資訊，請參閱 Amazon ECS 容器內省。

Amazon ECS 中的碼頭診斷

Docker 提供數種診斷工具，可協助您對容器和任務問題進行故障診斷。如需所有可用 Docker 命令列公用程式的詳細資訊，請參閱 Docker 文件的 Docker 命令列主題。您可以使用 SSH 連線到容器執行個體，存取 Docker 命令列公用程式。

Docker 容器回報的結束代碼，也可以提供一些診斷資訊 (例如，結束代碼 137 表示容器收到 SIGKILL 訊號)。如需詳細資訊，請參閱 Docker 文件中的「結束狀態」。

列出 Amazon ECS 中的碼頭集裝箱

您可以在您的容器執行個體中使用 docker ps 命令列出執行中的容器。在下列範例中，只有 Amazon ECS 容器代理程式正在執行。如需詳細資訊，請參閱 Docker 文件中的 docker ps。

```
docker ps
```

輸出：

```
CONTAINER ID        IMAGE                            COMMAND             CREATED
      STATUS              PORTS                          NAMES
cee0d6986de0        amazon/amazon-ecs-agent:latest   "/agent"            22 hours ago
      Up 22 hours         127.0.0.1:51678->51678/tcp   ecs-agent
```

您可以使用 docker ps -a 命令查看所有容器 (甚至是停止或終止的容器)。這有助於列出未預期停止的容器。在下列範例中，容器 f7f1f8a7a245 已在 9 秒前結束，所以 -a 旗標會出現在 docker ps 輸出中。

```
docker ps -a
```

輸出：

```
CONTAINER ID        IMAGE                                                COMMAND
 CREATED             STATUS                          PORTS                          NAMES
db4d48e411b1        amazon/ecs-emptyvolume-base:autogenerated   "not-applicable"
  19 seconds ago                                                                     ecs-
console-sample-app-static-6-internalecs-emptyvolume-source-c09288a6b0cba8a53700
f7f1f8a7a245        busybox:buildroot-2014.02                    "\"sh -c '/bin/sh -c
  22 hours ago        Exited (137) 9 seconds ago                                     ecs-
console-sample-app-static-6-busybox-ce83ce978a87a890ab01
189a8ff4b5f0        httpd:2                                      "httpd-foreground"
  22 hours ago        Exited (137) 40 seconds ago                                    ecs-
console-sample-app-static-6-simple-app-86caf9bcabe3e9c61600
0c7dca9321e3        amazon/ecs-emptyvolume-base:autogenerated   "not-applicable"
  22 hours ago                                                                       ecs-
console-sample-app-static-6-internalecs-emptyvolume-source-90fefaa68498a8a80700
cee0d6986de0        amazon/amazon-ecs-agent:latest               "/agent"
  22 hours ago        Up 22 hours                     127.0.0.1:51678->51678/tcp   ecs-
agent
```

在 Amazon ECS 中查看碼頭日誌

您可以使用 docker logs 命令檢視容器的 STDOUT 和 STDERR 串流。在此範例中，會顯示 *dc7240fe892a* 容器的日誌，並為簡潔起見使用 head 命令以管道輸送。如需詳細資訊，請前往 Docker 文件中的 docker logs。

> **Note**
>
> 若您使用預設的 json 日誌驅動程式，則 Docker 日誌僅適用於容器執行個體。如果您已將工作設定為使用記awslogs錄驅動程式，則您的容器記錄可在 Lo CloudWatch gs 中取得。如需詳細資訊，請參閱 將 Amazon ECS 日誌傳送到 CloudWatch 。

```
docker logs dc7240fe892a | head
```

輸出：

```
AH00558: httpd: Could not reliably determine the server's fully qualified domain name,
 using 172.17.0.11. Set the 'ServerName' directive globally to suppress this message
AH00558: httpd: Could not reliably determine the server's fully qualified domain name,
 using 172.17.0.11. Set the 'ServerName' directive globally to suppress this message
[Thu Apr 23 19:48:36.956682 2015] [mpm_event:notice] [pid 1:tid 140327115417472]
 AH00489: Apache/2.4.12 (Unix) configured -- resuming normal operations
[Thu Apr 23 19:48:36.956827 2015] [core:notice] [pid 1:tid 140327115417472] AH00094:
 Command line: 'httpd -D FOREGROUND'
10.0.1.86 - - [23/Apr/2015:19:48:59 +0000] "GET / HTTP/1.1" 200 348
10.0.0.154 - - [23/Apr/2015:19:48:59 +0000] "GET / HTTP/1.1" 200 348
10.0.1.86 - - [23/Apr/2015:19:49:28 +0000] "GET / HTTP/1.1" 200 348
10.0.0.154 - - [23/Apr/2015:19:49:29 +0000] "GET / HTTP/1.1" 200 348
10.0.1.86 - - [23/Apr/2015:19:49:50 +0000] "-" 408 -
10.0.0.154 - - [23/Apr/2015:19:49:50 +0000] "-" 408 -
10.0.1.86 - - [23/Apr/2015:19:49:58 +0000] "GET / HTTP/1.1" 200 348
10.0.0.154 - - [23/Apr/2015:19:49:59 +0000] "GET / HTTP/1.1" 200 348
10.0.1.86 - - [23/Apr/2015:19:50:28 +0000] "GET / HTTP/1.1" 200 348
10.0.0.154 - - [23/Apr/2015:19:50:29 +0000] "GET / HTTP/1.1" 200 348
time="2015-04-23T20:11:20Z" level="fatal" msg="write /dev/stdout: broken pipe"
```

在 Amazon ECS 中檢查碼頭集裝箱

如果您有容器的 Docker ID，您可以使用 docker inspect 命令來檢查該 Docker ID。檢查容器可提供最詳細的容器啟動環境檢視。如需詳細資訊，請參閱 Docker 文件中的 docker inspect。

```
docker inspect dc7240fe892a
```

輸出：

```
[{
    "AppArmorProfile": "",
    "Args": [],
    "Config": {
        "AttachStderr": false,
        "AttachStdin": false,
        "AttachStdout": false,
        "Cmd": [
            "httpd-foreground"
        ],
        "CpuShares": 10,
```

```
        "Cpuset": "",
        "Domainname": "",
        "Entrypoint": null,
        "Env": [
            "PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/
local/apache2/bin",
            "HTTPD_PREFIX=/usr/local/apache2",
            "HTTPD_VERSION=2.4.12",
            "HTTPD_BZ2_URL=https://www.apache.org/dist/httpd/httpd-2.4.12.tar.bz2"
        ],
        "ExposedPorts": {
            "80/tcp": {}
        },
        "Hostname": "dc7240fe892a",
...
```

在 Amazon ECS 中從 Docker 守護進程配置詳細輸出

如果您在使用 Docker 容器或映像時遇到問題，則可以在 Docker 守護程序上打開調試模式。使用偵錯可從精靈提供更詳細的輸出，您可以使用此選項擷取從容器登錄 (例如 Amazon ECR) 傳送的錯誤訊息。

> **Important**
>
> 此程序是針對 Amazon ECS 最佳化 Amazon Linux AMI 所寫入。對於其他操作系統，請參閱 Docker 文檔systemd中的啟用調試和控制和配置Docker。

在 Amazon ECS 優化 Amazon Linux AMI 上使用碼頭守護進程調試模式

1. 連線到您的容器執行個體。
2. 使用文字編輯器開啟 Docker 選項檔案，例如 vi。對於 Amazon ECS 最佳化 Amazon Linux AMI，Docker 選項檔案位於 `/etc/sysconfig/docker`。
3. 尋找 Docker 選項陳述式，並在字串中新增以引號括住的 `-D` 選項。

> **Note**
>
> 如果 Docker 選項陳述式以 # 開頭，請移除該字元以取消陳述式註解並啟用選項。

對於 Amazon ECS 最佳化 Amazon Linux AMI，Docker 選項陳述式稱為 OPTIONS。例如：

```
# Additional startup options for the Docker daemon, for example:
# OPTIONS="--ip-forward=true --iptables=true"
# By default we limit the number of open files per container
OPTIONS="-D --default-ulimit nofile=1024:4096"
```

4. 儲存檔案並結束您的文字編輯器。
5. 重新啟動 Docker 常駐程式。

```
sudo service docker restart
```

其輸出如下：

```
Stopping docker:                                           [  OK  ]
Starting docker: .                                    [  OK  ]
```

6. 重新啟動 Amazon ECS 代理。

```
sudo service ecs restart
```

您的 Docker 日誌現在應該會顯示更多的詳細輸出。

```
time="2015-12-30T21:48:21.907640838Z" level=debug msg="Unexpected response from
 server: \"{\\\"errors\\\":[{\\\"code\\\":\\\"DENIED\\\",\\\"message\\\":\\\"User:
 arn:aws:sts::1111:assumed-role/ecrReadOnly/i-abcdefg is not authorized to perform:
 ecr:InitiateLayerUpload on resource: arn:aws:ecr:us-east-1:1111:repository/nginx_test
\\\"}]}\\n\" http.Header{\"Connection\":[]string{\"keep-alive\"}, \"Content-Type\":
[]string{\"application/json; charset=utf-8\"}, \"Date\":[]string{\"Wed, 30 Dec 2015
 21:48:21 GMT\"}, \"Docker-Distribution-Api-Version\":[]string{\"registry/2.0\"},
 \"Content-Length\":[]string{\"235\"}}"
```

Amazon ECS 中的泊塢視API error (500): devmapper窗疑難排解

以下 Docker 錯誤指出，您容器執行個體上的精簡集區儲存已滿，Docker 常駐程式無法建立新的容器：

```
CannotCreateContainerError: API error (500): devmapper: Thin Pool has 4350 free data
 blocks which is less than minimum required 4454 free data blocks. Create more free
 space in thin pool or use dm.min_free_space option to change behavior
```

根據預設，2015.09.d 版和更新版本的 Amazon ECS 最佳化 Amazon Linux AMI 會使用作業系統的 8 GiB 磁碟區啟動，此作業系統連接於 /dev/xvda，而且掛載為檔案系統的根目錄。另有一個 22 GiB 的磁碟區連接在 /dev/xvdcz，Docker 用於儲存映像和中繼資料。如果這個儲存空間已滿，則 Docker 常駐程式無法建立新的容器。

要將儲存體新增到您的容器執行個體，最簡單方式是終止現有的執行個體，並啟動有較大資料儲存體磁碟區的新執行個體。不過，如果您無法執行此作業，您可以遵循 Amazon ECS 優化的 Linux AMI 中的程序，在 Docker 所用的磁碟區群組新增儲存，並擴展其邏輯磁碟區。

如果您的容器執行個體儲存體太快填滿，您可以採取幾個動作降低此效果：

- 若要檢視精簡輪詢資訊，請在您的容器執行個體上執行下列命令：

  ```
  docker info
  ```

- (Amazon ECS 容器代理程式 1.8.0 及更新版本) 您可以減少容器執行個體上停止或退出容器保留的時間量。ECS_ENGINE_TASK_CLEANUP_WAIT_DURATION 代理組態變數會將時間設為從任務停止等到 Docker 容器移除 (此值預設為 3 小時)。這會移除 Docker 容器的資料。如果此值設定得太低，您可能無法檢查已停止的容器或在移除記錄之前檢視記錄檔。如需詳細資訊，請參閱 Amazon ECS 容器代理程式組態。
- 您可以從容器執行個體中移除未執行的容器和未使用的映像檔。您可以使用以下範例命令，手動移除已停止的容器和未使用的映像。容器在刪除之後即無法檢查，而已刪除的映像必須得再次提取，才能從中啟動新的容器。

  若要移除非執行中的容器，請在您的容器執行個體上執行以下命令：

  ```
  docker rm $(docker ps -aq)
  ```

  若要移除未使用的映像，請在您的容器執行個體上執行以下命令：

  ```
  docker rmi $(docker images -q)
  ```

- 您可以移除容器中未使用的資料區塊。您可以使用下列命令在任何執行中的容器上執行 fstrim，並捨棄容器檔案系統未使用的任何資料區塊。

```
sudo sh -c "docker ps -q | xargs docker inspect --format='{{ .State.Pid }}' | xargs -IZ fstrim /proc/Z/root/"
```

Amazon ECS 執行問題疑難排解

以下是故障診斷注意事項，可協助診斷您在使用 ECS Exec 時可能會出現錯誤的原因。

使用執行檢查程式進行驗證

ECS Exec 檢查程式指令碼提供一種方法來驗證和驗證您的 Amazon ECS 叢集和任務是否符合使用 ECS Exec 功能的先決條件。ECS Exec 檢查程式指令碼會代表您呼叫各種 API，來驗證您的 AWS CLI 環境和叢集，以及任務已準備好可供 ECS Exec 使用。此工具需要最新版本的， AWS CLI 且該版jq本可用。如需詳細資訊，請參閱開啟的 ECS 執行檢查程式。 GitHub

呼叫 **execute-command** 時發生錯誤

如果發生 `The execute command failed` 錯誤，則可能的原因如下。

- 此任務沒有必要的許可。確認用來啟動任務的任務定義已定義任務 IAM 角色，且該角色具有必要的許可。如需詳細資訊，請參閱 ECS 執行權限。
- SSM 代理程式未安裝或未執行。
- Amazon ECS 有一個接口 Amazon VPC 端點，但沒有一個用於系統管理器會話管理器。

疑難排解 Amazon ECS Anywhere 可見

Amazon ECS Anywhere 不在提供將外部執行個體 (例如現場部署伺服器或虛擬機器 (VM) 註冊到 Amazon ECS 叢集的支援。以下是您可能會遇到的常見問題，及其一般故障診斷建議。

主題

- 外部執行個體註冊問題
- 外部執行個體網路問題
- 在外部執行個體上執行任務的問題

外部執行個體註冊問題

在 Amazon ECS 叢集註冊外部執行個體時，必須符合以下要求：

- 必須擷取包含啟動 ID 和啟動碼的啟動。 AWS Systems Manager 您可以使用它來將外部執行個體註冊為 Systems Manager 受管執行個體。當要求啟用 Systems Manager 時，請指定註冊限制和到期日。註冊限制指定使用啟用時可註冊的執行個體數目上限。註冊限制的預設值是1執行個體。過期日期指定啟用的過期時間。預設值為 24 小時。如果您用來註冊外部執行個體的 Systems Manager 啟用無效，則請求新的執行個體。如需詳細資訊，請參閱 將外部執行個體註冊到 Amazon ECS 叢集。
- IAM 政策用於為外部執行個體提供與 AWS API 操作通訊所需的許可。如果未正確建立此受管政策，且不包含必要的許可，則外部執行個體註冊會失敗。如需詳細資訊，請參閱 Amazon ECS Anywhere IAM 角色。
- Amazon ECS 提供安裝指令碼，可在外部執行個體上安裝 Docker、Amazon ECS 容器代理程式和 Systems Manager Agent。如果安裝指令碼失敗，則指令碼很可能無法在不發生錯誤的情形下在同一個執行個體上再次執行。如果發生這種情況，請按照清理程序清除執行個體的 AWS 資源，以便您可以再次執行安裝指令碼。如需詳細資訊，請參閱 取消註冊 Amazon ECS 外部執行個體。

  > **Note**
  >
  > 請注意，如果成功請求安裝指令碼並使用 Systems Manager 啟用，則第二次執行安裝指令碼會再次使用 Systems Manager 啟用。這轉而可能會導致您達到該啟用的註冊限制。如果達到此限制，則您必須建立新的啟用。

- 在 GPU 工作負載的外部執行個體上執行安裝指令碼時，如果未偵測到或正確設定 NVIDIA 驅動程式，就會發生錯誤。安裝指令碼使用 `nvidia-smi` 命令來確認 NVIDIA 驅動程式的存在。

外部執行個體網路問題

若要傳達任何變更，您的外部執行個體需要 AWS網路連線。如果您的外部執行個體失去與的網路連線 AWS，除非手動停止，否則在執行個體上執行的工作仍會繼續執行。恢復連線後，Amazon ECS 容器代理程式和外部執行個體上的系統管理員代理程式使用的 AWS 登入資料會自動更新。 AWS 如需有關用於外部執行個體之間通訊的 AWS 網域的詳細資訊 AWS，請參閱聯網 。

在外部執行個體上執行任務的問題

如果您的任務或容器無法在外部執行個體上執行，最常見的原因是網路或許可相關問題。如果您的容器從 Amazon ECR 提取其映像，或設定為將容器日誌傳送到日 CloudWatch 誌，則您的任務定義必須指

定有效的任務執行 IAM 角色。如果沒有有效的任務執行 IAM 角色，則您的容器將無法啟動。如需網路相關問題的詳細資訊，請參閱 外部執行個體網路問題。

> **Important**
>
> Amazon ECS 提供了 Amazon ECS 日誌收集工具。您可以將其用於從外部執行個體收集日誌，以進行故障診斷。如需詳細資訊，請參閱 使用 Amazon ECS 日誌收集器收集容器日誌 。

AWS Fargate 節流配額

AWS Fargate 針對每個區域使用每個 AWS 帳戶的權杖儲存貯體演算法，將 Amazon ECS 任務和 Amazon EKS 網繭啟動速率限制為配額 (以前稱為限制)。使用此演算法，您的帳戶擁有儲存特定數量字符的儲存貯體。儲存貯體中的字符數量代表您在任何指定秒數的速率配額。每個客戶帳戶都有一個任務和 Pod 字符儲存貯體，會根據客戶帳戶啟動的任務和 Pod 的數量耗盡。此字符儲存貯體具有允許您定期提出更多請求的儲存貯體最大值，以及允許您在需要時保持穩定的請求速率的重新填滿速率。

例如，Fargate 客戶帳戶的任務和 Pod 字符儲存貯體大小為 100 個字符，重新填滿速率為每秒 20 個字符。因此，您可以立即啟動 Amazon ECS 任務和 Amazon EKS Pod (每個客戶帳戶最多 100 個)，持續啟動速率為每秒 20 個 Amazon ECS 任務和 Amazon EKS Pod。

動作	儲存貯體容量上限 (或爆量率)	儲存貯體重新填滿速率 (或持續速率)
隨需型 Amazon ECS 任務和 Amazon EKS Pod 的 Fargate 資源速率配額[1]	100	20
Spot Amazon ECS 任務的 Fargate 資源速率配額	100	20

[1]使用 Amazon EKS 平台版本中提到的平台版本時，僅啟動 Amazon EKS Pod 的帳戶爆量率為 20，Pod 的持續啟動速率為每秒 20 次。

在 Fargate 中節流 **RunTask** API

此外，Fargate 在以單獨配額使用 Amazon ECS RunTask API 啟動任務時會限制請求速率。Fargate 針對每個區域限制每個 AWS 帳戶的 Amazon ECS RunTask API 請求。您每提出一個請求，就會從

儲存貯體中刪除一個字符。我們這樣做是為了協助服務效能，並確保所有 Fargate 客戶的公平使用。無論 API 呼叫來自 Amazon Elastic Container Service 主控台、命令列工具還是第三方應用程式，API 呼叫都會受限於請求配額。呼叫 Amazon ECS RunTask API 的速率配額是每秒 20 次呼叫 (爆量和持續)。不過，每次呼叫此 API 最多可啟動 10 個任務。這表示您可以在一秒鐘內對此 API 進行 10 次呼叫，請求在每次呼叫中啟動 10 個任務，從而實現在一秒鐘內啟動 100 個任務。同理，您也可以對此 API 進行 20 次呼叫，請求在每次呼叫中啟動 5 個任務。如需有關 Amazon ECS API 的 API 節流的詳細資訊，請參閱 Amazon ECS RunTask API 參考中的 API 請求節流。

實際上，任務和 Pod 啟動速率還取決於其他注意事項，例如要下載和解壓縮的容器映像、運作狀態檢查以及啟用的其他整合 (例如向負載平衡器註冊任務或 Pod)。根據客戶啟用的功能，客戶會看到工作和網繭啟動率與先前表示的配額相比較之下的差異。

調整 Fargate 中的費率配額

您可為 AWS 帳戶請求增加 Fargate 速率調節配額。若要請求調節配額，請聯絡 AWS Support 中心。

處理 Amazon ECS 節流問題

節流錯誤分為兩大類：同步節流和非同步節流。

同步節流

發生同步節流時，您會立即收到來自 Amazon ECS 的錯誤回應。當您在執行任務或建立服務時呼叫 Amazon ECS API 時，通常會發生此類節流。如需有關節流和相關節流限制的詳細資訊，請參閱 Amazon ECS API 的要求節流。

當您的應用程式啟動 API 要求時 (例如，使用 AWS CLI 或 AWS SDK)，您可以修復 API 節流。您可以透過架構應用程式來處理錯誤，或透過針對 API 呼叫的重試邏輯實作指數輪詢和抖動策略來執行此操作。如需詳細資訊，請參閱逾時、重試和含抖動的輪詢。

如果您使用 AWS SDK，則自動重試邏輯已內建且可設定。

Amazon ECS 中的異步節流

因為 Amazon ECS 或 AWS CloudFormation 可能代表您呼叫 API 來佈建資源的非同步工作流程，所以會發生非同步節流。重要的是要知道 Amazon ECS 代表您調用哪些 AWS API。例如，會針對`CreateNetworkInterface`使用`awsvpc`網路模式的工作叫用 `DescribeTargetHealth` API，並在針對向負載平衡器註冊的工作執行健康狀態檢查時叫用 API。

當您的工作負載達到相當大的規模時，這些 API 作業可能會受到限制。也就是說，它們可能會被限制到足以違反 Amazon ECS 強制執行的限制或被調用的 AWS 服務 限制。例如，如果您部署數百個服務，每個服務同時具有數百個使用awsvpc網路模式的任務，Amazon ECS 會分別叫用 Amazon EC2 API 操作，以CreateNetworkInterface及 Elastic Load Balancing API 操作 (例如RegisterTarget或DescribeTargetHealth) 註冊 elastic network interface 和負載平衡器。這些 API 呼叫可能會超出 API 限制，進而導致節流錯誤。以下是服務事件訊息中包含的 Elastic Load Balancing 節流錯誤的範例。

```
{
   "userIdentity":{
      "arn":"arn:aws:sts::111122223333:assumed-role/AWSServiceRoleForECS/ecs-service-
scheduler",
      "eventTime":"2022-03-21T08:11:24Z",
      "eventSource":"elasticloadbalancing.amazonaws.com",
      "eventName":" DescribeTargetHealth ",
      "awsRegion":"us-east-1",
      "sourceIPAddress":"ecs.amazonaws.com",
      "userAgent":"ecs.amazonaws.com",
      "errorCode":"ThrottlingException",
      "errorMessage":"Rate exceeded",
      "eventID":"0aeb38fc-229b-4912-8b0d-2e8315193e9c"
   }
}
```

當這些 API 呼叫與您帳戶中的其他 API 流量共用限制時，即使這些 API 呼叫是以服務事件形式發出，也可能難以監控。

監控節流

確定哪些 API 請求被限制以及誰發出這些請求非常重要。您可以使用 AWS CloudTrail 哪些監控節流，並與 CloudWatch Amazon Athena 和 Amazon 集成。 EventBridge您可以設定 CloudTrail 將特定事件傳送至 CloudWatch 記錄檔。 CloudWatch 日誌日誌洞察解析和分析事件。這會識別節流事件的詳細資訊，例如進行呼叫的使用者或 IAM 角色，以及進行的 API 呼叫數量。如需詳細資訊，請參閱使用記 CloudTrail 錄監視 CloudWatch 記錄檔。

如需有關 CloudWatch 日誌深入解析和如何查詢記錄檔指示的詳細資訊，請參閱使用日誌深入解析分析 CloudWatch 記錄資料。

使用 Amazon Athena，您可以使用標準 SQL 建立查詢和分析資料。例如，您可以建立 Athena 資料表來剖析 CloudTrail 事件。如需詳細資訊，請參閱使用 CloudTrail 主控台為 CloudTrail 記錄建立 Athena 表格。

建立 Athena 資料表之後，您可以使用簡單的 SQL 查詢 (例如下列查詢) 來調查ThrottlingException錯誤。

```
select eventname, errorcode,eventsource,awsregion, useragent,COUNT(*) count
FROM cloudtrail-table-name
where errorcode = 'ThrottlingException'
AND eventtime between '2022-01-14T03:00:08Z' and '2022-01-23T07:15:08Z'
group by errorcode, awsregion, eventsource, username, eventname
order by count desc;
```

Amazon ECS 還向 Amazon 發送事件通知。 EventBridge有資源狀態變更事件和服務動作事件。它們包括 API 節流事件，ECS_OPERATION_THROTTLED例如和. SERVICE_DISCOVERY_OPERATION_THROTTLED 如需詳細資訊，請參閱 Amazon ECS 服務動作事件。

這些事件可以由服務消耗，例 AWS Lambda 如執行響應動作。如需詳細資訊，請參閱 處理 Amazon ECS 事件。

如果您執行獨立工作，某些 API 作業 (例如RunTask非同步)，並且不會自動執行重試作業。在這種情況下，您可以使用服務 (例如 EventBridge 整 AWS Step Functions 合) 來重試限制或失敗的作業。如需詳細資訊，請參閱管理容器任務 (Amazon ECS、Amazon SNS)。

用 CloudWatch 於監控節流

CloudWatch 在「按 AWS 資源」下的Usage命名空間提供 API 使用監視。這些指標會以類型 API 和指標名稱記錄CallCount。您可以建立警示，以便在這些指標達到特定閾值時啟動。如需詳細資訊，請參閱視覺化服務配額和設定警示。

CloudWatch 還提供異常檢測。此功能使用機器學習，根據您啟用該功能之測量結果的特定行為來分析和建立基準線。如果有異常的 API 活動，您可以將此功能與 CloudWatch 警報一起使用。如需詳細資訊，請參閱使用 CloudWatch 異常偵測。

透過主動監控節流錯誤，您可以聯絡 AWS Support 以提高相關節流限制，並獲得針對您獨特應用程式需求的指引。

Amazon ECS API 故障原因

當您透過 Amazon ECS API、主控台或 AWS CLI 結束時顯示failures錯誤訊息觸發的 API 動作，以下內容可能有助於排除原因。失敗會傳回與失敗關聯之資源的原因和 Amazon 資源名稱 (ARN)。

許多資源是區域限定的，所以在使用主控台時，請務必為您的資源設定正確的區域。使用時 AWS CLI，請確定您的 AWS CLI 命令已使用`--region` *region*參數傳送至正確的 Region。

如需有關 `Failure` 資料類型結構的詳細資訊，請參閱《Amazon Elastic Container Service API 參考》中的故障。

以下是您在執行 API 命令時可能會收到的失敗訊息範例。

API 動作	失敗原因或停止原因	原因
`DescribeClusters`	`MISSING`	找不到指定的叢集。請確認叢集名稱的拼字。
`DescribeInstances`	`MISSING`	找不到指定的容器執行個體。確認您指定的叢集容器執行個體已註冊，且容器執行個體 ARN 或 ID 都是正確的。
`DescribeServices`	`MISSING`	找不到指定的服務。請確認已指定正確的叢集或區域，且服務 ARN 或名稱有效。
`DescribeTasks`	`MISSING`	找不到指定的任務。請確認已指定正確的叢集或區域，而且任務 ARN 或 ID 皆有效。
`DescribeTasks`	`TaskFailedToStart: RESOURCE:*`	如果 `RESOURCE:CPU` 發生錯誤，無法在容器執行個體上取得任務請求的 CPU 數量。當任務定義中的 CPU 單元需求大於對應至容量提供者的 Auto Scaling 群組中定義之 Amazon EC2 執行個體的 CPU 大小時，通常會發生這種情況。您需要檢查容量提供者組態。 如果 `RESOURCE:MEMORY` 發生錯誤，無法在容器執行個體上取得任務請求的記憶體數量

API 動作	失敗原因或停止原因	原因
		。當任務定義中的記憶體量需求大於對應至容量提供者的 Auto Scaling 群組中定義的 Amazon EC2 執行個體上支援的記憶體時，通常會發生這種情況。您需要檢查容量提供者組態。
	`TaskFailedToStart: AGENT`	您嘗試啟動任務所在之容器執行個體上的代理程式目前中斷連線。為避免延長任務置放等待時間，請求遭到拒絕。 如需如何對已中斷連線的代理程式進行故障診斷的相關資訊，請參閱 How do I troubleshoot a disconnected Amazon ECS agent (如何對已中斷連線的 Amazon ECS 代理程式進行故障診斷)。
	`TaskFailedToStart: MemberOf placement constraint unsatisfied`	沒有符合任務定義中定義的放置限制的容器執行個體。

API 動作	失敗原因或停止原因	原因
	TaskFailedToStart: ATTRIBUTE	您的任務定義包含需要特定容器執行個體屬性的參數，但您的容器執行個體不提供此屬性。例如，如果您的任務使用 awsvpc 網路模式，但您使用 ecs.capability.task-eni 屬性指定的子網路中沒有任何執行個體。如需特定任務定義參數和代理組態變數需要哪些必要屬性的詳細資訊，請參閱「Amazon ECS 任務定義參數」和「Amazon ECS 容器代理程式組態」。
	TaskFailedToStart: NO ACTIVE INSTANCES	容量提供者中沒有作用中的執行個體。如需有關如何管理 Auto Scaling 群組的相關資訊，請參閱《Amazon EC2 Auto Scaling 使用者指南》中的 Auto Scaling 群組。
	TaskFailedToStart: EMPTY CAPACITY PROVIDER	您的叢集中沒有執行個體。這很可能是因為空白容量提供者，或因為容量提供者中的執行個體未註冊至叢集。如需有關如何管理 Auto Scaling 群組的相關資訊，請參閱《Amazon EC2 Auto Scaling 使用者指南》中的 Auto Scaling 群組。
GetTaskProtection	MISSING	找不到指定的任務。確認叢集名稱或 ARN 以及任務 ARN 或 ID 是否有效。

API 動作	失敗原因或停止原因	原因
	TASK_NOT_VALID	指定的任務不是 Amazon ECS 服務的一部分。只有 Amazon ECS 服務受管任務才能受到保護。請確認任務 ARN 或 ID，然後再試一次。
RunTask 或 StartTask	RESOURCE:*	無法在叢集的容器執行個體上取得任務請求的一或多項資源。如果資源是 CPU、記憶體、連接埠或彈性網路介面，您可能需要新增其他容器執行個體到您的叢集。 若為 RESOURCE:ENI 錯誤，您的叢集不會有任何可用的彈性網路介面連接點，但這對使用 awsvpc 網路模式的任務是必要的。Amazon EC2 執行個體有可以連接到它們的網路界面數量限制，主要網路界面包含在內。如需每種執行個體類型支援多少個網路界面的詳細資訊，請參閱 Amazon EC2 使用者指南中每個執行個體類型的每個網路界面的 IP 地址。 對於 RESOURCE:GPU 錯誤，任務請求的 GPU 數目不可用，而且您可能需要將啟用 GPU 的容器執行個體新增至叢集。如需詳細資訊，請參閱 GPU 工作負載的 Amazon ECS 任務定義。

API 動作	失敗原因或停止原因	原因
	AGENT	您嘗試啟動任務所在之容器執行個體上的代理程式目前中斷連線。為避免延長任務置放等待時間，請求遭到拒絕。 如需如何對已中斷連線的代理程式進行故障診斷的相關資訊，請參閱 How do I troubleshoot a disconnected Amazon ECS agent (如何對已中斷連線的 Amazon ECS 代理程式進行故障診斷)。
	LOCATION	您嘗試啟動任務的容器執行個體，位於與您在 awsVpcConfiguration 中指定的子網路不同的可用區域。
	ATTRIBUTE	您的任務定義包含需要特定容器執行個體屬性的參數，但您的容器執行個體不提供此屬性。例如，如果您的任務使用 awsvpc 網路模式，但您使用 ecs.capability.task-eni 屬性指定的子網路中沒有任何執行個體。如需特定任務定義參數和代理組態變數需要哪些必要屬性的詳細資訊，請參閱「Amazon ECS 任務定義參數」和「Amazon ECS 容器代理程式組態」。

API 動作	失敗原因或停止原因	原因
StartTask	MISSING	找不到您嘗試啟動工作的容器執行個體。檢查是否指定了錯誤的群集或區域，或者容器實例 ARN 或 ID 拼寫錯誤。
	INACTIVE	您嘗試啟動任務所在之容器執行個體，之前已向 Amazon ECS 取消註冊，無法使用。
UpdateTaskProtection	DEPLOYMENT_BLOCKED	無法設置任務保護，因為一個或多個受保護的任務阻止服務部署到達穩定狀態。取消現有任務的任務保護設定，或等到任務保護到期。
	MISSING	找不到指定的任務。確認叢集名稱或 ARN 以及任務 ARN 或 ID 是否有效。
	TASK_NOT_VALID	指定的任務不是 Amazon ECS 服務的一部分。只有 Amazon ECS 服務受管任務才能受到保護。請確認任務 ARN 或 ID，然後再試一次。

Note

除了此處描述的故障情況之外，API 操作也可能由於異常而失敗，從而導致錯誤響應。如需此類例外狀況的清單，請參閱常見錯誤。

Amazon Elastic Container Service 的安全性

雲安全 AWS 是最高的優先級。身為 AWS 客戶，您可以從資料中心和網路架構中獲益，該架構專為滿足對安全性最敏感的組織的需求而打造。

安全是 AWS 與您之間共同承擔的責任。共同責任模型 將此描述為雲端的安全和雲端內的安全：

- 雲端的安全性 — AWS 負責保護在 AWS 雲端中執行 AWS 服務的基礎架構。 AWS 還為您提供可以安全使用的服務。在 AWS 合規計畫中，第三方稽核員會定期測試並驗證我們的安全功效。若要了解適用於 Amazon Elastic Container Service 的合規計劃，請參閱合規計劃範圍內的AWS 服務。
- 雲端中的安全性 — 您的責任取決於您使用的 AWS 服務。您也必須對其他因素負責，包括資料的機密性、您公司的要求和適用法律和法規。

本文件有助於您了解如何在使用 Amazon ECS 時套用共同責任模型。下列主題說明如何設定 Amazon ECS 以符合您的安全與合規目標。您也會學到如何使用其他可 AWS 協助您監控和保護 Amazon ECS 資源的服務。

主題

- Amazon Elastic Container Service 的身分和存取管理
- 在 Amazon Elastic Container Service 中記錄和監控
- Amazon Elastic Container Service 的合規驗證
- AWS Fargate 美國聯邦資訊處理標準 (FIPS-140)
- Amazon Elastic Container Service 的基礎設施安全性
- Amazon ECS 任務和容器安全最佳實務

Amazon Elastic Container Service 的身分和存取管理

AWS Identity and Access Management (IAM) 可協助系統管理員安全地控制 AWS 資源存取權。 AWS 服務 IAM 管理員可以控制應對誰進行身分驗證 (已登入) 和授權 (具有許可) 以使用 Amazon ECS 資源。IAM 是您可以使用的 AWS 服務 ，無需額外付費。

主題

- 物件
- 使用身分驗證

- 使用政策管理存取權
- Amazon Elastic Container Service 如何與 IAM 搭配使用
- Amazon Elastic Container Service 身分型政策範例
- AWS Amazon 彈性容器服務的受管政策
- 使用 Amazon ECS 的服務連結角色
- Amazon ECS 的 IAM 角色
- Amazon ECS 主控台必要的許可
- Amazon ECS 服務 auto 擴展所需的 IAM 許可
- 在建立時授予標記資源的許可
- 對 Amazon Elastic Container Service 身分和存取進行故障診斷
- Amazon ECS 的 IAM 最佳做法

物件

您使用 AWS Identity and Access Management (IAM) 的方式會因您在 Amazon ECS 中所做的工作而有所不同。

服務使用者 – 如果您使用 Amazon ECS 服務執行任務，您的管理員會為您提供您需要的憑證和許可。隨著您為了執行作業而使用的 Amazon ECS 功能數量變多，您可能會需要額外的許可。了解存取許可的管理方式可協助您向管理員請求正確的許可。如果您無法存取 Amazon ECS 中的功能，請參閱對 Amazon Elastic Container Service 身分和存取進行故障診斷。

服務管理員：若您在公司負責管理 Amazon ECS 資源，您應該擁有 Amazon ECS 的完整存取權。您的任務是判斷服務使用者應存取的 Amazon ECS 功能和資源。接著，您必須將請求提交給您的 IAM 管理員，來變更您服務使用者的許可。檢閱此頁面上的資訊，了解 IAM 的基本概念。若要進一步了解貴公司可搭配 Amazon ECS 使用 IAM 的方式，請參閱Amazon Elastic Container Service 如何與 IAM 搭配使用。

IAM 管理員：如果您是 IAM 管理員，建議您掌握如何撰寫政策以管理 Amazon ECS 存取權的詳細資訊。若要檢視您可以在 IAM 中使用的基於 Amazon ECS 身分的政策範例，請參閱Amazon Elastic Container Service 身分型政策範例。

使用身分驗證

驗證是您 AWS 使用身分認證登入的方式。您必須以 IAM 使用者身分或假設 IAM 角色進行驗證 (登入 AWS)。 AWS 帳戶根使用者

您可以使用透過 AWS 身分識別來源提供的認證，以聯合身分識別身分登入。 AWS IAM Identity Center (IAM 身分中心) 使用者、貴公司的單一登入身分驗證，以及您的 Google 或 Facebook 登入資料都是聯合身分識別的範例。您以聯合身分登入時，您的管理員先前已設定使用 IAM 角色的聯合身分。當您使 AWS 用同盟存取時，您會間接擔任角色。

根據您的使用者類型，您可以登入 AWS Management Console 或 AWS 存取入口網站。如需有關登入的詳細資訊 AWS，請參閱《AWS 登入 使用指南》 AWS 帳戶中的如何登入您的。

如果您 AWS 以程式設計方式存取，請 AWS 提供軟體開發套件 (SDK) 和命令列介面 (CLI)，以使用您的認證以加密方式簽署要求。如果您不使用 AWS 工具，則必須自行簽署要求。如需使用建議的方法自行簽署請求的詳細資訊，請參閱 IAM 使用者指南中的簽署 AWS API 請求。

無論您使用何種身分驗證方法，您可能都需要提供額外的安全性資訊。例如， AWS 建議您使用多重要素驗證 (MFA) 來增加帳戶的安全性。如需更多資訊，請參閱 AWS IAM Identity Center 使用者指南中的多重要素驗證和 IAM 使用者指南中的在 AWS中使用多重要素驗證 (MFA)。

AWS 帳戶 根使用者

當您建立時 AWS 帳戶，您會從一個登入身分開始，該身分可完整存取該帳戶中的所有資源 AWS 服務和資源。此身分稱為 AWS 帳戶 root 使用者，可透過使用您用來建立帳戶的電子郵件地址和密碼登入來存取。強烈建議您不要以根使用者處理日常任務。保護您的根使用者憑證，並將其用來執行只能由根使用者執行的任務。如需這些任務的完整清單，了解需以根使用者登入的任務，請參閱 IAM 使用者指南中的需要根使用者憑證的任務。

聯合身分

最佳作法是要求人類使用者 (包括需要系統管理員存取權的使用者) 使用與身分識別提供者的同盟，才能使用臨時認證 AWS 服務 來存取。

聯合身分識別是來自企業使用者目錄的使用者、Web 身分識別提供者、Identi ty Center 目錄，或使用透過身分識別來源提供的認證進行存取 AWS 服務 的任何使用者。 AWS Directory Service同盟身分存取時 AWS 帳戶，他們會假設角色，而角色則提供臨時認證。

對於集中式存取權管理，我們建議您使用 AWS IAM Identity Center。您可以在 IAM Identity Center 中建立使用者和群組，也可以連線並同步到自己身分識別來源中的一組使用者和群組，以便在所有應用程式 AWS 帳戶 和應用程式中使用。如需 IAM Identity Center 的相關資訊，請參閱 AWS IAM Identity Center 使用者指南中的什麼是 IAM Identity Center？。

IAM 使用者和群組

IAM 使用者是您內部的身分，具 AWS 帳戶 有單一人員或應用程式的特定許可。建議您盡可能依賴暫時憑證，而不是擁有建立長期憑證 (例如密碼和存取金鑰) 的 IAM 使用者。但是如果特定使用案例需要擁有長期憑證的 IAM 使用者，建議您輪換存取金鑰。如需更多資訊，請參閱 IAM 使用者指南中的為需要長期憑證的使用案例定期輪換存取金鑰。

IAM 群組是一種指定 IAM 使用者集合的身分。您無法以群組身分簽署。您可以使用群組來一次為多名使用者指定許可。群組可讓管理大量使用者許可的程序變得更為容易。例如，您可以擁有一個名為 IAMAdmins 的群組，並給予該群組管理 IAM 資源的許可。

使用者與角色不同。使用者只會與單一人員或應用程式建立關聯，但角色的目的是在由任何需要它的人員取得。使用者擁有永久的長期憑證，但角色僅提供暫時憑證。如需進一步了解，請參閱 IAM 使用者指南中的建立 IAM 使用者 (而非角色) 的時機。

IAM 角色

IAM 角色是您 AWS 帳戶 內部具有特定許可的身分。它類似 IAM 使用者，但不與特定的人員相關聯。您可以切換角色，在中暫時擔任 IAM 角色。 AWS Management Console 您可以透過呼叫 AWS CLI 或 AWS API 作業或使用自訂 URL 來擔任角色。如需使用角色的方法更多相關資訊，請參閱 IAM 使用者指南中的使用 IAM 角色。

使用暫時憑證的 IAM 角色在下列情況中非常有用：

- 聯合身分使用者存取 – 若要向聯合身分指派許可，請建立角色，並為角色定義許可。當聯合身分進行身分驗證時，該身分會與角色建立關聯，並獲授予由角色定義的許可。如需有關聯合角色的相關資訊，請參閱 IAM 使用者指南中的為第三方身分提供者建立角色。如果您使用 IAM Identity Center，則需要設定許可集。為控制身分驗證後可以存取的內容，IAM Identity Center 將許可集與 IAM 中的角色相關聯。如需有關許可集的資訊，請參閱 AWS IAM Identity Center 使用者指南中的許可集。
- 暫時 IAM 使用者許可 – IAM 使用者或角色可以擔任 IAM 角色來暫時針對特定任務採用不同的許可。
- 跨帳戶存取權 – 您可以使用 IAM 角色，允許不同帳戶中的某人 (信任的委託人) 存取您帳戶中的資源。角色是授予跨帳戶存取權的主要方式。但是，對於某些策略 AWS 服務，您可以將策略直接附加到資源（而不是使用角色作為代理）。若要了解跨帳戶存取權角色和資源型政策間的差異，請參閱 IAM 使用者指南中的 IAM 角色與資源類型政策的差異。
- 跨服務訪問 — 有些 AWS 服務 使用其他 AWS 服務功能。例如，當您在服務中進行呼叫時，該服務通常會在 Amazon EC2 中執行應用程式或將物件儲存在 Amazon Simple Storage Service (Amazon S3) 中。服務可能會使用呼叫主體的許可、使用服務角色或使用服務連結角色來執行此作業。

- 轉寄存取工作階段 (FAS) — 當您使用 IAM 使用者或角色在中執行動作時 AWS，您會被視為主體。使用某些服務時，您可能會執行某個動作，進而在不同服務中啟動另一個動作。FAS 會使用主體呼叫的權限 AWS 服務，並結合要求 AWS 服務 向下游服務發出要求。只有當服務收到需要與其 AWS 服務 他資源互動才能完成的請求時，才會發出 FAS 請求。在此情況下，您必須具有執行這兩個動作的許可。如需提出 FAS 請求時的政策詳細資訊，請參閱《轉發存取工作階段》。
- 服務角色 – 服務角色是服務擔任的 IAM 角色，可代表您執行動作。IAM 管理員可以從 IAM 內建立、修改和刪除服務角色。如需更多資訊，請參閱 IAM 使用者指南中的建立角色以委派許可給 AWS 服務。
- 服務連結角色 — 服務連結角色是連結至. AWS 服務服務可以擔任代表您執行動作的角色。服務連結角色會顯示在您的中， AWS 帳戶 且屬於服務所有。IAM 管理員可以檢視，但不能編輯服務連結角色的許可。

- 在 Amazon EC2 上執行的應用程式 — 您可以使用 IAM 角色來管理在 EC2 執行個體上執行的應用程式以及發出 AWS CLI 或 AWS API 請求的臨時登入資料。這是在 EC2 執行個體內儲存存取金鑰的較好方式。若要將 AWS 角色指派給 EC2 執行個體並提供給其所有應用程式，請建立連接至執行個體的執行個體設定檔。執行個體設定檔包含該角色，並且可讓 EC2 執行個體上執行的程式取得暫時憑證。如需更多資訊，請參閱 IAM 使用者指南中的利用 IAM 角色來授予許可給 Amazon EC2 執行個體上執行的應用程式。

若要了解是否要使用 IAM 角色或 IAM 使用者，請參閱 IAM 使用者指南中的建立 IAM 角色 (而非使用者) 的時機。

使用政策管理存取權

您可以透 AWS 過建立原則並將其附加至 AWS 身分識別或資源來控制中的存取。原則是一個物件 AWS ，當與身分識別或資源相關聯時，會定義其權限。 AWS 當主參與者 (使用者、root 使用者或角色工作階段) 提出要求時，評估這些原則。政策中的許可決定是否允許或拒絕請求。大多數原則會 AWS 以 JSON 文件的形式儲存在中。如需 JSON 政策文件結構和內容的更多相關資訊，請參閱 IAM 使用者指南中的 JSON 政策概觀。

管理員可以使用 AWS JSON 政策來指定誰可以存取哪些內容。也就是說，哪個主體在什麼條件下可以對什麼資源執行哪些動作。

預設情況下，使用者和角色沒有許可。若要授予使用者對其所需資源執行動作的許可，IAM 管理員可以建立 IAM 政策。然後，管理員可以將 IAM 政策新增至角色，使用者便能擔任這些角色。

IAM 政策定義該動作的許可，無論您使用何種方法來執行操作。例如，假設您有一個允許 `iam:GetRole` 動作的政策。具有該原則的使用者可以從 AWS Management Console AWS CLI、或 AWS API 取得角色資訊。

身分型政策

身分型政策是可以附加到身分 (例如 IAM 使用者、使用者群組或角色) 的 JSON 許可政策文件。這些政策可控制身分在何種條件下能對哪些資源執行哪些動作。若要了解如何建立身分類型政策，請參閱 IAM 使用者指南中的建立 IAM 政策。

身分型政策可進一步分類成內嵌政策或受管政策。內嵌政策會直接內嵌到單一使用者、群組或角色。受管理的策略是獨立策略，您可以將其附加到您的 AWS 帳戶. 受管政策包括 AWS 受管政策和客戶管理的策略。若要了解如何在受管政策及內嵌政策間選擇，請參閱 IAM 使用者指南中的在受管政策和內嵌政策間選擇。

資源型政策

資源型政策是連接到資源的 JSON 政策文件。資源型政策的最常見範例是 IAM 角色信任政策和 Amazon S3 儲存貯體政策。在支援資源型政策的服務中，服務管理員可以使用它們來控制對特定資源的存取權限。對於附加政策的資源，政策會定義指定的主體可以對該資源執行的動作以及在何種條件下執行的動作。您必須在資源型政策中指定主體。主參與者可以包括帳戶、使用者、角色、同盟使用者或。 AWS 服務

資源型政策是位於該服務中的內嵌政策。您無法在以資源為基礎的政策中使用 IAM 的 AWS 受管政策。

存取控制清單 (ACL)

存取控制清單 (ACL) 可控制哪些委託人 (帳戶成員、使用者或角色) 擁有存取某資源的許可。ACL 類似於資源型政策，但它們不使用 JSON 政策文件格式。

Amazon S3 和 Amazon VPC 是支援 ACL 的服務範例。 AWS WAF若要進一步了解 ACL，請參閱 Amazon Simple Storage Service 開發人員指南中的存取控制清單 (ACL) 概觀。

其他政策類型

AWS 支援其他較不常見的原則類型。這些政策類型可設定較常見政策類型授予您的最大許可。

- 許可界限 – 許可範圍是一種進階功能，可供您設定身分型政策能授予 IAM 實體 (IAM 使用者或角色) 的最大許可。您可以為實體設定許可界限。所產生的許可會是實體的身分型政策和其許可界限的交集。會在 `Principal` 欄位中指定使用者或角色的資源型政策則不會受到許可界限限制。所有這類政

策中的明確拒絕都會覆寫該允許。如需許可範圍的更多相關資訊，請參閱 IAM 使用者指南中的 IAM 實體許可範圍。

- 服務控制策略（SCP）— SCP 是 JSON 策略，用於指定中組織或組織單位（OU）的最大權限。AWS Organizations AWS Organizations 是一種用於分組和集中管理您企業擁 AWS 帳戶 有的多個服務。若您啟用組織中的所有功能，您可以將服務控制政策 (SCP) 套用到任何或所有帳戶。SCP 限制成員帳戶中實體的權限，包括每個 AWS 帳戶根使用者帳戶。如需組織和 SCP 的更多相關資訊，請參閱 AWS Organizations 使用者指南中的 SCP 運作方式。
- 工作階段政策 – 工作階段政策是一種進階政策，您可以在透過編寫程式的方式建立角色或聯合使用者的暫時工作階段時，作為參數傳遞。所產生工作階段的許可會是使用者或角色的身分型政策和工作階段政策的交集。許可也可以來自資源型政策。所有這類政策中的明確拒絕都會覆寫該允許。如需更多資訊，請參閱 IAM 使用者指南中的工作階段政策。

多種政策類型

將多種政策類型套用到請求時，其結果形成的許可會更為複雜、更加難以理解。要了解如何在涉及多個政策類型時 AWS 確定是否允許請求，請參閱《IAM 使用者指南》中的政策評估邏輯。

Amazon Elastic Container Service 如何與 IAM 搭配使用

在您使用 IAM 管理對 Amazon ECS 的存取權之前，請瞭解哪些 IAM 功能可以與 Amazon ECS 搭配使用。

您可以搭配 Amazon Elastic Container Service 使用的 IAM 功能

IAM 功能	Amazon ECS 支援
身分型政策	是
資源型政策	否
政策動作	是
政策資源	部分
政策條件索引鍵	是
ACL	否

IAM 功能	Amazon ECS 支援
ABAC (政策中的標籤)	是
臨時憑證	是
轉送存取工作階段 (FAS)	是
服務角色	是
服務連結角色	是

若要深入瞭解 Amazon ECS 和其他 AWS 服務如何與大多數 IAM 功能搭配運作，請參閱 IAM 使用者指南中的搭配 IAM 使用的AWS 服務。

Amazon ECS 身分型政策

支援身分型政策	是

身分型政策是可以連接到身分 (例如 IAM 使用者、使用者群組或角色) 的 JSON 許可政策文件。這些政策可控制身分在何種條件下能對哪些資源執行哪些動作。若要了解如何建立身分類型政策，請參閱《IAM 使用者指南》中的建立 IAM 政策。

使用 IAM 身分型政策，您可以指定允許或拒絕的動作和資源，以及在何種條件下允許或拒絕動作。您無法在身分型政策中指定主體，因為這會套用至連接的使用者或角色。如要了解您在 JSON 政策中使用的所有元素，請參閱《IAM 使用者指南》中的 IAM JSON 政策元素參考。

Amazon ECS 的身分型政策範例

若要檢視 Amazon ECS 身分型政策的範例，請參閱Amazon Elastic Container Service 身分型政策範例。

Amazon ECS 中的資源型政策

支援以資源基礎的政策	否

資源型政策是附加到資源的 JSON 政策文件。資源型政策的最常見範例是 IAM 角色信任政策和 Amazon S3 儲存貯體政策。在支援資源型政策的服務中，服務管理員可以使用它們來控制對特定資源的存取權限。對於附加政策的資源，政策會定義指定的主體可以對該資源執行的動作以及在何種條件下執行的動作。您必須在資源型政策中指定主體。主參與者可以包括帳戶、使用者、角色、同盟使用者或。 AWS 服務

若要啟用跨帳戶存取，您可以指定在其他帳戶內的所有帳戶或 IAM 實體，作為資源型政策的主體。新增跨帳戶主體至資源型政策，只是建立信任關係的一半。當主體和資源不同時 AWS 帳戶，受信任帳戶中的 IAM 管理員也必須授與主體實體 (使用者或角色) 權限，才能存取資源。其透過將身分型政策連接到實體來授與許可。不過，如果資源型政策會為相同帳戶中的主體授予存取，這時就不需要額外的身分型政策。如需詳細資訊，請參閱《IAM 使用者指南》中的 IAM 角色與資源型政策有何差異。

Amazon ECS 的政策動作

支援政策動作	是

管理員可以使用 AWS JSON 政策來指定誰可以存取哪些內容。也就是說，哪個主體在什麼條件下可以對什麼資源執行哪些動作。

JSON 政策的 `Action` 元素描述您可以用來允許或拒絕政策中存取的動作。原則動作通常與關聯的 AWS API 作業具有相同的名稱。有一些例外狀況，例如沒有相符的 API 操作的僅限許可動作。也有一些作業需要政策中的多個動作。這些額外的動作稱為相依動作。

政策會使用動作來授予執行相關聯動作的許可。

若要查看 Amazon ECS 動作的清單，請參閱《服務授權參考》中的 Amazon Elastic Container Service 定義的動作。

Amazon ECS 中的政策動作會在動作之前使用下列字首：

```
ecs
```

如需在單一陳述式中指定多個動作，請用逗號分隔。

```
"Action": [
      "ecs:action1",
      "ecs:action2"
         ]
```

您也可以使用萬用字元 (*) 來指定多個動作。例如，若要指定開頭是 Describe 文字的所有動作，請包含以下動作：

```
"Action": "ecs:Describe*"
```

若要檢視 Amazon ECS 身分型政策的範例，請參閱Amazon Elastic Container Service 身分型政策範例。

Amazon ECS 的政策資源

支援政策資源	部分

管理員可以使用 AWS JSON 政策來指定誰可以存取哪些內容。也就是說，哪個主體在什麼條件下可以對什麼資源執行哪些動作。

Resource JSON 政策元素可指定要套用動作的物件。陳述式必須包含 Resource 或 NotResource 元素。最佳實務是使用其 Amazon Resource Name (ARN) 來指定資源。您可以針對支援特定資源類型的動作 (稱為資源層級許可) 來這麼做。

對於不支援資源層級許可的動作 (例如列出操作)，請使用萬用字元 (*) 來表示陳述式適用於所有資源。

```
"Resource": "*"
```

若要查看 Amazon ECS 資源類型及其 ARN 的清單，請參閱《服務授權參考》中的 Amazon Elastic Container Service 定義的資源。若要瞭解您可以使用哪些動作指定每個資源的 ARN，請參閱 Amazon Elastic Container Service 定義的動作。

部分 Amazon ECS API 動作支援多個資源。例如，呼叫 DescribeClusters API 動作時可以參考多個叢集。若要在單一陳述式中指定多項資源，請使用逗號分隔 ARN。

```
"Resource": [
      "EXAMPLE-RESOURCE-1",
      "EXAMPLE-RESOURCE-2"
```

例如，Amazon ECS 叢集資源具有以下 ARN：

```
arn:${Partition}:ecs:${Region}:${Account}:cluster/${clusterName}
```

若要在您的陳述式中指定 my-cluster-1 和 my-cluster-2 叢集，請使用以下 ARN：

```
"Resource": [
        "arn:aws:ecs:us-east-1:123456789012:cluster/my-cluster-1",
        "arn:aws:ecs:us-east-1:123456789012:cluster/my-cluster-2"
```

若要指定所有屬於特定帳戶的叢集，請使用萬用字元 (*)：

```
"Resource": "arn:aws:ecs:us-east-1:123456789012:cluster/*"
```

對於任務定義，您可以指定最新版本或特定修訂版。

若要指定任務定義的所有修訂版本，請使用萬用字元 (*)：

```
"Resource:arn:${Partition}:ecs:${Region}:${Account}:task-definition/
${TaskDefinitionFamilyName}:*"
```

若要指定特定的任務定義修訂版本，請使用 $ {TaskDefinitionRevisionNumber}：

```
"Resource:arn:${Partition}:ecs:${Region}:${Account}:task-definition/
${TaskDefinitionFamilyName}:${TaskDefinitionRevisionNumber}"
```

若要檢視 Amazon ECS 身分型政策的範例，請參閱Amazon Elastic Container Service 身分型政策範例。

Amazon ECS 的政策條件索引鍵

支援服務特定政策條件金鑰	是

管理員可以使用 AWS JSON 政策來指定誰可以存取哪些內容。也就是說，哪個主體在什麼條件下可以對什麼資源執行哪些動作。

Condition 元素 (或 Condition 區塊) 可讓您指定使陳述式生效的條件。Condition 元素是選用項目。您可以建立使用條件運算子的條件運算式 (例如等於或小於)，來比對政策中的條件和請求中的值。

若您在陳述式中指定多個 Condition 元素，或是在單一 Condition 元素中指定多個索引鍵， AWS 會使用邏輯 AND 操作評估他們。如果您為單一條件索引鍵指定多個值，請使用邏輯OR運算來 AWS 評估條件。必須符合所有條件，才會授與陳述式的許可。

您也可以在指定條件時使用預留位置變數。例如，您可以只在使用者使用其 IAM 使用者名稱標記時，將存取資源的許可授予該 IAM 使用者。如需更多資訊，請參閱 IAM 使用者指南中的 IAM 政策元素：變數和標籤。

AWS 支援全域條件金鑰和服務特定條件金鑰。若要查看所有 AWS 全域條件金鑰，請參閱《IAM 使用者指南》中的AWS 全域條件內容金鑰。

Amazon ECS 支援以下服務特定條件索引鍵，您可以使用這些條件索引鍵來為您的 IAM 政策提供精細篩選條件：

條件索引鍵	描述	評估類型
AWS：RequestTag/$ {} TagKey	內容金鑰的格式為 "aws:RequestTag/ *tag-key*":"*tag-value* "，其中 *tag-key* 和 *tag-value* 是標籤金鑰和值對。 檢查標籤鍵值配對是否存在於請求中 AWS 。例如，您可以檢查請求是否包含標籤金鑰 "Dept" 並且它具有值 "Accounting" 。	字串
AWS：ResourceTag/$ {} TagKey	內容金鑰的格式為 "aws:ResourceTag/ *tag-key*":"*tag-value* "，其中 *tag-key* 和 *tag-value* 是標籤金鑰和值對。 檢查連接至身分資源 (使用者或角色) 的標籤是否符合指定的鍵名稱和值。	字串
AWS：TagKeys	此內容鍵的格式為 "aws:TagKeys":" *tag-key*"，其中 *tag-key* 是沒有值的標籤鍵清單 (例如，["Dept","Cost-Center"])。 檢查存在於 AWS 請求中的標籤鍵。	字串
ECS: ResourceTag /$ {} TagKey	內容金鑰的格式為 "ecs:ResourceTag/ *tag-key*":"*tag-value* "，其中 *tag-key* 和 *tag-value* 是標籤金鑰和值對。 檢查連接至身分資源 (使用者或角色) 的標籤是否符合指定的鍵名稱和值。	字串

條件索引鍵	描述	評估類型
ecs:cluster	內容金鑰的格式為 "ecs:cluster":" *cluster-arn* "，其中 *cluster-arn* 是 Amazon ECS 叢集的 ARN。	ARN, Null
ecs:container-instances	內容金鑰的格式為 "ecs:container-instances":" *container-instance-arns* "，其中 *container-instance-arns* 為一或多個容器執行個體的 ARN。	ARN, Null
ecs:container-name	內容索引鍵的格式為 "ecs:container-name":" *container-name* "，其中 *container-instance-* 是在任務定義中定義的 Amazon ECS 容器名稱。	字串
ecs:enable-execute-command	內文索引鍵已被格式化為 "ecs:enable-execute-command":" *value*"，其中 *value-* 是 "true" 或 "false"。	字串
ecs:enable-service-connect	內容索引鍵的格式為 "ecs:enable-service-connect":" *value*"，其中 *value* 是 "true" 或 "false"。	字串
ec: 啟用 EB-磁碟區	內容索引鍵的格式為 "ecs:enable-ebs-volumes":" *value*"，其中 *value* 是 "true" 或 "false"。	字串
ecs:namespace	內容索引鍵的格式為 "ecs:namespace":" *namespace-arn* "，其中 *namespace-arn* 是 AWS Cloud Map 命名空間的 ARN。	ARN, Null
ecs:service	內容金鑰的格式為 "ecs:service":" *service-arn* "，其中 *service-arn* 是 Amazon ECS 服務的 ARN。	ARN, Null
ecs:task-definition	內容金鑰的格式為 "ecs:task-definition":" *task-definition-arn* "，其中 *task-definition-arn* 為 Amazon ECS 任務定義的 ARN。	ARN, Null

條件索引鍵	描述	評估類型
ecs:account-setting	內容索引鍵的格式為 "ecs:account-setting":" *account-setting* "，其中 *account-setting* 為 Amazon ECS 帳戶設定的名稱。	字串

若要查看 Amazon ECS 條件索引鍵的清單，請參閱《服務授權參考》中的 Amazon Elastic Kubernetes Service 的條件索引鍵。若要了解您可以搭配哪些動作和資源使用條件索引鍵，請參閱 Amazon Elastic Container Service 定義的動作。

若要檢視 Amazon ECS 身分型政策的範例，請參閱Amazon Elastic Container Service 身分型政策範例。

Amazon ECS 中的存取控制清單 (ACL)

支援 ACL	否

存取控制清單 (ACL) 可控制哪些主體 (帳戶成員、使用者或角色) 擁有存取某資源的許可。ACL 類似於資源型政策，但它們不使用 JSON 政策文件格式。

搭配 Amazon ECS 的屬性型存取控制 (ABAC)

> **Important**
>
> Amazon ECS 支援所有 Amazon ECS 資源的屬性型存取控制。若要判斷是否可以使用屬性來限定動作範圍，請使用《服務授權參考》中由 Amazon ECS 表格定義的動作。首先驗證 Resource (資源) 欄位中是否有資源。然後，使用 Condition keys (條件索引鍵) 欄位查看動作/資源組合的索引鍵。

支援 ABAC (政策中的標籤)	是

屬性型存取控制 (ABAC) 是一種授權策略，可根據屬性來定義許可。在中 AWS，這些屬性稱為標籤。您可以將標籤附加到 IAM 實體 (使用者或角色) 和許多 AWS 資源。為實體和資源加上標籤是 ABAC 的第一步。您接著要設計 ABAC 政策，允許在主體的標籤與其嘗試存取的資源標籤相符時操作。

ABAC 在成長快速的環境中相當有幫助，並能在政策管理變得繁瑣時提供協助。

若要根據標籤控制存取，請使用 aws:ResourceTag/*key-name*、aws:RequestTag/*key-name* 或 aws:TagKeys 條件金鑰，在政策的條件元素中，提供標籤資訊。

如果服務支援每個資源類型的全部三個條件金鑰，則對該服務而言，值為 Yes。如果服務僅支援某些資源類型的全部三個條件金鑰，則值為 Partial。

如需 ABAC 的詳細資訊，請參閱《IAM 使用者指南》中的什麼是 ABAC?。如要查看含有設定 ABAC 步驟的教學課程，請參閱《IAM 使用者指南》中的使用屬性型存取控制 (ABAC)。

如需標記 Amazon ECS 資源的詳細資訊，請參閱 標記 Amazon ECS 資源。

若要檢視身分型政策範例，以根據該資源上的標籤來限制存取資源，請參閱根據標籤描述 Amazon ECS 服務。

將暫時憑證與 Amazon ECS 搭配使用

支援臨時憑證	是

當您使用臨時憑據登錄時，某些 AWS 服務 不起作用。如需其他資訊，包括哪些 AWS 服務 與臨時登入資料搭配AWS 服務 使用，請參閱 IAM 使用者指南中的 IAM。

如果您使用除了使用者名稱和密碼以外的任何方法登入，則您正在 AWS Management Console 使用臨時認證。例如，當您 AWS 使用公司的單一登入 (SSO) 連結存取時，該程序會自動建立暫時認證。當您以使用者身分登入主控台，然後切換角色時，也會自動建立臨時憑證。如需切換角色的詳細資訊，請參閱《IAM 使用者指南》中的切換至角色 (主控台)。

您可以使用 AWS CLI 或 AWS API 手動建立臨時登入資料。然後，您可以使用這些臨時登入資料來存取 AWS。 AWS 建議您動態產生臨時登入資料，而非使用長期存取金鑰。如需詳細資訊，請參閱 IAM 中的暫時性安全憑證。

Amazon ECS 的轉發存取工作階段

支援轉寄存取工作階段 (FAS)	是

當您使用 IAM 使用者或角色在中執行動作時 AWS，您會被視為主體。使用某些服務時，您可能會執行某個動作，進而在不同服務中啟動另一個動作。FAS 會使用主體呼叫的權限 AWS 服務，並結合要求

AWS 服務 向下游服務發出要求。只有當服務收到需要與其 AWS 服務 他資源互動才能完成的請求時，才會發出 FAS 請求。在此情況下，您必須具有執行這兩個動作的許可。如需提出 FAS 請求時的政策詳細資訊，請參閱《轉發存取工作階段》。

Amazon ECS 的服務角色

支援服務角色	是

服務角色是服務擔任的 IAM 角色，可代您執行動作。IAM 管理員可以從 IAM 內建立、修改和刪除服務角色。如需更多資訊，請參閱 IAM 使用者指南中的建立角色以委派許可給 AWS 服務。

> **Warning**
>
> 變更服務角色的許可可能會中斷 Amazon ECS 功能。只有在 Amazon ECS 提供指引時，才能編輯服務角色。

Amazon ECS 的服務連結角色

支援服務連結角色	是

服務連結角色是連結至. AWS 服務服務可以擔任代表您執行動作的角色。服務連結角色會顯示在您的中， AWS 帳戶 且屬於服務所有。IAM 管理員可以檢視，但不能編輯服務連結角色的許可。

如需建立或管理 Amazon ECS 服務連結角色的詳細資訊，請參閱使用 Amazon ECS 的服務連結角色。

Amazon Elastic Container Service 身分型政策範例

根據預設，使用者和角色不具備建立或修改 Amazon ECS 資源的許可。他們也無法使用 AWS Management Console、 AWS Command Line Interface (AWS CLI) 或 AWS API 來執行工作。若要授予使用者對其所需資源執行動作的許可，IAM 管理員可以建立 IAM 政策。然後，管理員可以將 IAM 政策新增至角色，使用者便能擔任這些角色。

若要了解如何使用這些範例 JSON 政策文件建立 IAM 身分型政策，請參閱《IAM 使用者指南》中的建立 IAM 政策。

如需 Amazon ECS 所定義之動作和資源類型的詳細資訊，包括每種資源類型的 ARN 格式，請參閱服務授權參考中的《適用於 Amazon Elastic Container Service 的動作、資源和條件索引鍵》。

主題

- Amazon ECS 政策最佳實務
- 允許 Amazon ECS 使用者檢視自己的許可
- Amazon ECS 集群示例
- Amazon ECS 容器執行個體範例
- Amazon ECS 任務定義示例
- 運行 Amazon ECS 任務示例
- 開始 Amazon ECS 任務示例
- 列出並說明 Amazon ECS 任務範例
- 建立 Amazon ECS 服務範例
- 更新 Amazon ECS 服務示例
- 根據標籤描述 Amazon ECS 服務
- 拒絕 Amazon ECS 服務 Connect 命名空間覆寫範例

Amazon ECS 政策最佳實務

身分型政策會判斷您帳戶中的某個人員是否可以建立、存取或刪除 Amazon ECS 資源。這些動作可能會讓您的 AWS 帳戶產生費用。當您建立或編輯身分型政策時，請遵循下列準則及建議事項：

- 開始使用 AWS 受管原則並邁向最低權限權限 — 若要開始授與使用者和工作負載的權限，請使用可授與許多常見使用案例權限的AWS 受管理原則。它們在您的 AWS 帳戶. 建議您透過定義特定於您使用案例的 AWS 客戶管理政策，進一步降低使用權限。如需更多資訊，請參閱 IAM 使用者指南中的 AWS 受管政策或任務職能的AWS 受管政策。
- 套用最低許可許可 – 設定 IAM 政策的許可時，請僅授予執行任務所需的權限。為實現此目的，您可以定義在特定條件下可以對特定資源採取的動作，這也稱為最低權限許可。如需使用 IAM 套用許可的更多相關資訊，請參閱 IAM 使用者指南中的 IAM 中的政策和許可。
- 使用 IAM 政策中的條件進一步限制存取權 – 您可以將條件新增至政策，以限制動作和資源的存取。例如，您可以撰寫政策條件，指定必須使用 SSL 傳送所有請求。您也可以使用條件來授與對服務動作的存取權 (如透過特定) 使用這些動作 AWS 服務，例如 AWS CloudFormation。如需更多資訊，請參閱 IAM 使用者指南中的 IAM JSON 政策元素：條件。

- 使用 IAM Access Analyzer 驗證 IAM 政策，確保許可安全且可正常運作 – IAM Access Analyzer 驗證新政策和現有政策，確保這些政策遵從 IAM 政策語言 (JSON) 和 IAM 最佳實務。IAM Access Analyzer 提供 100 多項政策檢查及切實可行的建議，可協助您編寫安全且實用的政策。如需更多資訊，請參閱 IAM 使用者指南中的 IAM Access Analyzer 政策驗證。
- 需要多因素身份驗證 (MFA) — 如果您的案例需要 IAM 使用者或根使用者 AWS 帳戶，請開啟 MFA 以獲得額外的安全性。若要在呼叫 API 作業時請求 MFA，請將 MFA 條件新增至您的政策。如需更多資訊，請參閱 IAM 使用者指南中的設定 MFA 保護的 API 存取。

如需 IAM 中最佳實務的相關資訊，請參閱 IAM 使用者指南中的 IAM 安全最佳實務。

允許 Amazon ECS 使用者檢視自己的許可

此範例會示範如何建立政策，允許 IAM 使用者檢視連接到他們使用者身分的內嵌及受管政策。此原則包含在主控台上或以程式設計方式使用 AWS CLI 或 AWS API 完成此動作的權限。

```
{
    "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": "*"
        }
    ]
}
```

Amazon ECS 集群示例

以下 IAM 政策允許建立和列出叢集的許可。CreateCluster 和 ListClusters 動作不接受任何資源，因此資源定義已針對所有資源設為 *。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ecs:CreateCluster",
                "ecs:ListClusters"
            ],
            "Resource": ["*"]
        }
    ]
}
```

以下 IAM 政策允許說明和刪除指定叢集的許可。DescribeClusters 和 DeleteCluster 動作接受叢集 ARN 做為資源。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ecs:DescribeClusters",
                "ecs:DeleteCluster"
            ],
            "Resource": ["arn:aws:ecs:us-east-1:<aws_account_id>:cluster/
<cluster_name>"]
        }
    ]
```

```
}
```

以下 IAM 政策可連接到使用者或群組，只允許該使用者或群組在特定叢集上執行操作。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "ecs:Describe*",
                "ecs:List*"
            ],
            "Effect": "Allow",
            "Resource": "*"
        },
        {
            "Action": [
                "ecs:DeleteCluster",
                "ecs:DeregisterContainerInstance",
                "ecs:ListContainerInstances",
                "ecs:RegisterContainerInstance",
                "ecs:SubmitContainerStateChange",
                "ecs:SubmitTaskStateChange"
            ],
            "Effect": "Allow",
            "Resource": "arn:aws:ecs:us-east-1:<aws_account_id>:cluster/default"
        },
        {
            "Action": [
                "ecs:DescribeContainerInstances",
                "ecs:DescribeTasks",
                "ecs:ListTasks",
                "ecs:UpdateContainerAgent",
                "ecs:StartTask",
                "ecs:StopTask",
                "ecs:RunTask"
            ],
            "Effect": "Allow",
            "Resource": "*",
            "Condition": {
                "ArnEquals": {"ecs:cluster": "arn:aws:ecs:us-
east-1:<aws_account_id>:cluster/default"}
            }
```

```
        }
    ]
}
```

Amazon ECS 容器執行個體範例

容器執行個體的註冊由 Amazon ECS 代理程式處理，但有時候您可能會想要讓使用者能從叢集手動取消註冊執行個體。也許容器執行個體意外註冊到錯誤的叢集，或執行個體上尚有任務執行中時遭到終止。

以下 IAM 政策可讓使用者列出和取消註冊指定叢集中的容器執行個體：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ecs:DeregisterContainerInstance",
                "ecs:ListContainerInstances"
            ],
            "Resource": ["arn:aws:ecs:<region>:<aws_account_id>:cluster/
<cluster_name>"]
        }
    ]
}
```

以下 IAM 政策可讓使用者說明指定叢集中的指定容器執行個體。若要開啟此許可給叢集中的所有容器執行個體，您可以使用 * 取代容器執行個體的 UUID。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": ["ecs:DescribeContainerInstances"],
            "Condition": {
                "ArnEquals": {"ecs:cluster":
 "arn:aws:ecs:<region>:<aws_account_id>:cluster/<cluster_name>"}
            },
            "Resource": ["arn:aws:ecs:<region>:<aws_account_id>:container-instance/
<cluster_name>/<container_instance_UUID>"]
```

```
        }
    ]
}
```

Amazon ECS 任務定義示例

任務定義 IAM 政策不支援資源層級許可，但以下 IAM 政策允許使用者註冊、列出和說明任務定義：

如果您使用主控台，則必須新增 `CloudFormation: CreateStack` 作為 `Action`。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ecs:RegisterTaskDefinition",
                "ecs:ListTaskDefinitions",
                "ecs:DescribeTaskDefinition"
            ],
            "Resource": ["*"]
        }
    ]
}
```

運行 Amazon ECS 任務示例

`RunTask` 的資源為任務定義。若要限制使用者可執行任務定義的叢集，您可以在 `Condition` 區塊中加以指定。優點是您不需要在您的資源中列出任務定義和叢集，也能允許適當的存取。您可以套用其中一項，或同時套用兩者。

以下 IAM 政策允許在特定叢集上執行任何特定任務定義之修訂的許可：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": ["ecs:RunTask"],
            "Condition": {
                "ArnEquals": {"ecs:cluster":
 "arn:aws:ecs:<region>:<aws_account_id>:cluster/<cluster_name>"}
```

```
            },
            "Resource": ["arn:aws:ecs:<region>:<aws_account_id>:task-definition/
<task_family>:*"]
        }
    ]
}
```

開始 Amazon ECS 任務示例

StartTask 的資源為任務定義。若要限制使用者可啟動任務定義的叢集和容器執行個體，您可以在 Condition 區塊中加以指定。優點是您不需要在您的資源中列出任務定義和叢集，也能允許適當的存取。您可以套用其中一項，或同時套用兩者。

以下 IAM 政策允許在特定叢集與特定容器執行個體上，啟動任何特定任務定義修訂的許可。

> **Note**
>
> 在此範例中，當您使用 AWS CLI 或其他 AWS SDK 呼叫 StartTask API 時，您必須指定任務定義修訂版本，以使Resource對應符合。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": ["ecs:StartTask"],
            "Condition": {
                "ArnEquals": {
                    "ecs:cluster": "arn:aws:ecs:<region>:<aws_account_id>:cluster/
<cluster_name>",
                    "ecs:container-instances":
 ["arn:aws:ecs:<region>:<aws_account_id>:container-instance/<cluster_name>/
<container_instance_UUID>"]
                }
            },
            "Resource": ["arn:aws:ecs:<region>:<aws_account_id>:task-definition/
<task_family>:*"]
        }
    ]
}
```

列出並說明 Amazon ECS 任務範例

以下 IAM 政策允許使用者列出指定叢集的任務：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": ["ecs:ListTasks"],
            "Condition": {
                "ArnEquals": {"ecs:cluster":
 "arn:aws:ecs:<region>:<aws_account_id>:cluster/<cluster_name>"}
            },
            "Resource": ["arn:aws:ecs:<region>:<aws_account_id>:cluster/
<cluster_name>"]
        }
    ]
}
```

以下 IAM 政策可讓使用者說明指定叢集中的指定任務：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": ["ecs:DescribeTasks"],
            "Condition": {
                "ArnEquals": {"ecs:cluster":
 "arn:aws:ecs:<region>:<aws_account_id>:cluster/<cluster_name>"}
            },
            "Resource": ["arn:aws:ecs:<region>:<aws_account_id>:task/<cluster_name>/
<task_UUID>"]
        }
    ]
}
```

建立 Amazon ECS 服務範例

以下 IAM 政策可讓使用者在 AWS Management Console中建立 Amazon ECS 服務：

```
{
```

```
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "application-autoscaling:Describe*",
                "application-autoscaling:PutScalingPolicy",
                "application-autoscaling:RegisterScalableTarget",
                "cloudwatch:DescribeAlarms",
                "cloudwatch:PutMetricAlarm",
                "ecs:List*",
                "ecs:Describe*",
                "ecs:CreateService",
                "elasticloadbalancing:Describe*",
                "iam:GetPolicy",
                "iam:GetPolicyVersion",
                "iam:GetRole",
                "iam:ListAttachedRolePolicies",
                "iam:ListRoles",
                "iam:ListGroups",
                "iam:ListUsers"
            ],
            "Resource": ["*"]
        }
    ]
}
```

更新 Amazon ECS 服務示例

以下 IAM 政策可讓使用者在 AWS Management Console中更新 Amazon ECS 服務：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "application-autoscaling:Describe*",
                "application-autoscaling:PutScalingPolicy",
                "application-autoscaling:DeleteScalingPolicy",
                "application-autoscaling:RegisterScalableTarget",
                "cloudwatch:DescribeAlarms",
                "cloudwatch:PutMetricAlarm",
```

```
                "ecs:List*",
                "ecs:Describe*",
                "ecs:UpdateService",
                "iam:GetPolicy",
                "iam:GetPolicyVersion",
                "iam:GetRole",
                "iam:ListAttachedRolePolicies",
                "iam:ListRoles",
                "iam:ListGroups",
                "iam:ListUsers"
            ],
            "Resource": ["*"]
        }
    ]
}
```

根據標籤描述 Amazon ECS 服務

您可以在基於身分的政策中使用條件，根據標籤控制 Amazon ECS 資源的存取權。此範例會示範如何建立會允許描述您的服務的政策。但是，只有在服務標籤 Owner 的值是該使用者的使用者名稱時，才會授予該許可。此政策也會授予在主控台完成此動作的必要許可。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "DescribeServices",
            "Effect": "Allow",
            "Action": "ecs:DescribeServices",
            "Resource": "*"
        },
        {
            "Sid": "ViewServiceIfOwner",
            "Effect": "Allow",
            "Action": "ecs:DescribeServices",
            "Resource": "arn:aws:ecs:*:*:service/*",
            "Condition": {
                "StringEquals": {"ecs:ResourceTag/Owner": "${aws:username}"}
            }
        }
    ]
}
```

您可以將此政策連接到您帳戶中的 IAM 使用者。如果名為 `richard-roe` 的使用者嘗試描述 Amazon ECS 服務，則該服務必須標記為 `Owner=richard-roe` 或 `owner=richard-roe`。否則他便會被拒絕存取。條件標籤鍵 `Owner` 符合 `Owner` 和 `owner`，因為條件索引鍵名稱不區分大小寫。如需詳細資訊，請參閱《IAM 使用者指南》中的 IAM JSON 政策元素：條件。

拒絕 Amazon ECS 服務 Connect 命名空間覆寫範例

下列 IAM 政策會拒絕使用者覆寫服務組態中的預設 Service Connect 命名空間。預設命名空間在叢集中設定。不過，您可以在服務組態中覆寫命名空間。為了保持一致性，請考慮將所有新服務設定為使用相同的命名空間。使用下列內容索引鍵可要求服務使用特定的命名空間。在以下範例中，將 `<region>`、`<aws_account_id>`、`<cluster_name>` 和 `<namespace_id>` 取代為自訂文字。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ecs:CreateService",
                "ecs:UpdateService"
            ],
            "Condition": {
                "ArnEquals": {
                    "ecs:cluster": "arn:aws:ecs:<region>:<aws_account_id>:cluster/
<cluster_name>",
                    "ecs:namespace":
 "arn:aws:servicediscovery:<region>:<aws_account_id>:namespace/<namespace_id>"
                }
            },
            "Resource": "*"
        }
    ]
}
```

AWS Amazon 彈性容器服務的受管政策

若要新增使用者、群組和角色的權限，使用 AWS 受管理的原則比自己撰寫原則更容易。建立 IAM 客戶受管政策需要時間和專業知識，而受管政策可為您的團隊提供其所需的許可。若要快速開始使用，

您可以使用我們的 AWS 受管政策。這些政策涵蓋常見使用案例，並可在您的 AWS 帳戶中使用。如需 AWS 受管政策的詳細資訊，請參閱 IAM 使用者指南中的AWS 受管政策。

AWS 服務會維護和更新 AWS 受管理的策略。您無法變更 AWS 受管理原則中的權限。服務有時會將其他權限新增至受 AWS 管理的策略，以支援新功能。此類型的更新會影響已連接政策的所有身分識別 (使用者、群組和角色)。當新功能啟動或新作業可用時，服務最有可能更新 AWS 受管理的策略。服務不會從 AWS 受管理的政策移除權限，因此政策更新不會破壞您現有的權限。

此外，還 AWS 支援跨多個服務之工作職能的受管理原則。例如，ReadOnly存取 AWS 管理原則會提供所有 AWS 服務和資源的唯讀存取權。當服務啟動新功能時，會為新作業和資源新 AWS 增唯讀權限。如需任務職能政策的清單和說明，請參閱 IAM 使用者指南中有關任務職能的AWS 受管政策。

Amazon ECS 和 Amazon ECR 提供多個受管政策和信任關係，您可將其連接至使用者、群組、角色、Amazon EC2 執行個體和 Amazon ECS 任務，允許對資源和 API 操作進行不同層級的控制。您可以直接套用這些政策，也可以使用它們開始建立您自己的政策。如需 Amazon ECR 受管政策的詳細資訊，請參閱 Amazon ECR 受管政策。

亞馬遜 FullAccess

您可將 `AmazonECS_FullAccess` 政策連接到 IAM 身分。

此政策授予對 Amazon ECS 資源的管理存取權，並授予 IAM 身分識別 (例如使用者、群組或角色) 存取 Amazon ECS 與之整合的 AWS 服務，以便使用所有 Amazon ECS 功能。使用此政策可存取 AWS Management Console中可用的所有 Amazon ECS 功能。

許可詳細資訊

`AmazonECS_FullAccess` 受管 IAM 政策包含下列許可。遵循授予最低權限的最佳實務，您可以使用 `AmazonECS_FullAccess` 受管政策作為範本，以建立您自己的自訂政策。如此一來，您就可以根據特定需求，從受管政策中取消或新增許可。

- `ecs`— 允許主體完全存取所有 Amazon ECS API 操作。
- `application-autoscaling` - 允許委託人建立、描述及管理 Application Auto Scaling 資源。為 Amazon ECS 服務啟用服務自動擴展時，這是必要項。
- `appmesh` - 允許委託人列出 App Mesh 服務網格和虛擬節點，並描述 App Mesh 虛擬節點。將 Amazon ECS 服務與 App Mesh 整合時，這為必要項。
- `autoscaling` - 允許委託人建立、管理和描述 Amazon EC2 Auto Scaling 資源。使用叢集自動擴展功能時，在管理 Amazon EC2 Auto Scaling 群組時，這是必需的。
- `cloudformation`— 允許主參與者建立和管理 AWS CloudFormation 堆疊。在使用 AWS Management Console 建立 Amazon ECS 叢集以及後續管理這些叢集時，這為必要項。

- cloudwatch— 允許校長建立、管理和描述 Amazon CloudWatch 警報。
- codedeploy— 可讓主參與者建立及管理應用程式部署，以及檢視其組態、修訂版本及部署目標。
- sns - 允許委託人檢視 Amazon SNS 主題清單。
- lambda - 允許委託人檢視 AWS Lambda 函數清單及其版本特定組態。
- ec2— 允許主體執行 Amazon EC2 執行個體，以及建立和管理路由、路由表、網際網路閘道、啟動群組、安全群組、虛擬私有雲端、Spot 叢集和子網路。
- elasticloadbalancing - 允許委託人建立、描述及刪除 Elastic Load Balancing 負載平衡器。主體也可以向負載平衡器的新建立目標群組、接聽程式和接聽程式規則新增標籤。
- events— 允許主體建立、管理和刪除 Amazon EventBridge 規則及其目標。
- iam - 允許委託人列出 IAM 角色及其連接的政策。委託人還可列出 Amazon EC2 執行個體可用的執行個體設定檔。
- logs— 允許主體建立和描述 Amazon CloudWatch 日誌記錄群組。委託人也可以列出這些日誌群組的日誌事件。
- route53 - 允許委託人建立、管理和刪除 Amazon Route 53 託管區域。委託人也可以檢視 Amazon Route 53 運作狀態檢查組態和資訊。如需有關託管區域的詳細資訊，請參閱使用託管區域。
- servicediscovery— 允許主參與者建立、管理和刪除 AWS Cloud Map 服務，以及建立私人 DNS 命名空間。

AmazonECS_FullAccess 政策範例如下。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "application-autoscaling:DeleteScalingPolicy",
                "application-autoscaling:DeregisterScalableTarget",
                "application-autoscaling:DescribeScalableTargets",
                "application-autoscaling:DescribeScalingActivities",
                "application-autoscaling:DescribeScalingPolicies",
                "application-autoscaling:PutScalingPolicy",
                "application-autoscaling:RegisterScalableTarget",
                "appmesh:DescribeVirtualGateway",
                "appmesh:DescribeVirtualNode",
                "appmesh:ListMeshes",
                "appmesh:ListVirtualGateways",
```

```
                "appmesh:ListVirtualNodes",
                "autoscaling:CreateAutoScalingGroup",
                "autoscaling:CreateLaunchConfiguration",
                "autoscaling:DeleteAutoScalingGroup",
                "autoscaling:DeleteLaunchConfiguration",
                "autoscaling:Describe*",
                "autoscaling:UpdateAutoScalingGroup",
                "cloudformation:CreateStack",
                "cloudformation:DeleteStack",
                "cloudformation:DescribeStack*",
                "cloudformation:UpdateStack",
                "cloudwatch:DeleteAlarms",
                "cloudwatch:DescribeAlarms",
                "cloudwatch:GetMetricStatistics",
                "cloudwatch:PutMetricAlarm",
                "codedeploy:BatchGetApplicationRevisions",
                "codedeploy:BatchGetApplications",
                "codedeploy:BatchGetDeploymentGroups",
                "codedeploy:BatchGetDeployments",
                "codedeploy:ContinueDeployment",
                "codedeploy:CreateApplication",
                "codedeploy:CreateDeployment",
                "codedeploy:CreateDeploymentGroup",
                "codedeploy:GetApplication",
                "codedeploy:GetApplicationRevision",
                "codedeploy:GetDeployment",
                "codedeploy:GetDeploymentConfig",
                "codedeploy:GetDeploymentGroup",
                "codedeploy:GetDeploymentTarget",
                "codedeploy:ListApplicationRevisions",
                "codedeploy:ListApplications",
                "codedeploy:ListDeploymentConfigs",
                "codedeploy:ListDeploymentGroups",
                "codedeploy:ListDeployments",
                "codedeploy:ListDeploymentTargets",
                "codedeploy:RegisterApplicationRevision",
                "codedeploy:StopDeployment",
                "ec2:AssociateRouteTable",
                "ec2:AttachInternetGateway",
                "ec2:AuthorizeSecurityGroupIngress",
                "ec2:CancelSpotFleetRequests",
                "ec2:CreateInternetGateway",
                "ec2:CreateLaunchTemplate",
                "ec2:CreateRoute",
```

```
                "ec2:CreateRouteTable",
                "ec2:CreateSecurityGroup",
                "ec2:CreateSubnet",
                "ec2:CreateVpc",
                "ec2:DeleteLaunchTemplate",
                "ec2:DeleteSubnet",
                "ec2:DeleteVpc",
                "ec2:Describe*",
                "ec2:DetachInternetGateway",
                "ec2:DisassociateRouteTable",
                "ec2:ModifySubnetAttribute",
                "ec2:ModifyVpcAttribute",
                "ec2:RequestSpotFleet",
                "ec2:RunInstances",
                "ecs:*",
                "elasticfilesystem:DescribeAccessPoints",
                "elasticfilesystem:DescribeFileSystems",
                "elasticloadbalancing:CreateListener",
                "elasticloadbalancing:CreateLoadBalancer",
                "elasticloadbalancing:CreateRule",
                "elasticloadbalancing:CreateTargetGroup",
                "elasticloadbalancing:DeleteListener",
                "elasticloadbalancing:DeleteLoadBalancer",
                "elasticloadbalancing:DeleteRule",
                "elasticloadbalancing:DeleteTargetGroup",
                "elasticloadbalancing:DescribeListeners",
                "elasticloadbalancing:DescribeLoadBalancers",
                "elasticloadbalancing:DescribeRules",
                "elasticloadbalancing:DescribeTargetGroups",
                "events:DeleteRule",
                "events:DescribeRule",
                "events:ListRuleNamesByTarget",
                "events:ListTargetsByRule",
                "events:PutRule",
                "events:PutTargets",
                "events:RemoveTargets",
                "fsx:DescribeFileSystems",
                "iam:ListAttachedRolePolicies",
                "iam:ListInstanceProfiles",
                "iam:ListRoles",
                "lambda:ListFunctions",
                "logs:CreateLogGroup",
                "logs:DescribeLogGroups",
                "logs:FilterLogEvents",
```

```
                "route53:CreateHostedZone",
                "route53:DeleteHostedZone",
                "route53:GetHealthCheck",
                "route53:GetHostedZone",
                "route53:ListHostedZonesByName",
                "servicediscovery:CreatePrivateDnsNamespace",
                "servicediscovery:CreateService",
                "servicediscovery:DeleteService",
                "servicediscovery:GetNamespace",
                "servicediscovery:GetOperation",
                "servicediscovery:GetService",
                "servicediscovery:ListNamespaces",
                "servicediscovery:ListServices",
                "servicediscovery:UpdateService",
                "sns:ListTopics"
            ],
            "Resource": ["*"]
        },
        {
            "Effect": "Allow",
            "Action": [
                "ssm:GetParameter",
                "ssm:GetParameters",
                "ssm:GetParametersByPath"
            ],
            "Resource": "arn:aws:ssm:*:*:parameter/aws/service/ecs*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:DeleteInternetGateway",
                "ec2:DeleteRoute",
                "ec2:DeleteRouteTable",
                "ec2:DeleteSecurityGroup"
            ],
            "Resource": ["*"],
            "Condition": {
                "StringLike": {"ec2:ResourceTag/aws:cloudformation:stack-name":
"EC2ContainerService-*"}
            }
        },
        {
            "Action": "iam:PassRole",
            "Effect": "Allow",
```

```
            "Resource": ["*"],
            "Condition": {
                "StringLike": {"iam:PassedToService": "ecs-tasks.amazonaws.com"}
            }
        },
        {
            "Action": "iam:PassRole",
            "Effect": "Allow",
            "Resource": ["arn:aws:iam::*:role/ecsInstanceRole*"],
            "Condition": {
                "StringLike": {
                    "iam:PassedToService": [
                        "ec2.amazonaws.com",
                        "ec2.amazonaws.com.cn"
                    ]
                }
            }
        },
        {
            "Action": "iam:PassRole",
            "Effect": "Allow",
            "Resource": ["arn:aws:iam::*:role/ecsAutoscaleRole*"],
            "Condition": {
                "StringLike": {
                    "iam:PassedToService": [
                        "application-autoscaling.amazonaws.com",
                        "application-autoscaling.amazonaws.com.cn"
                    ]
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": "iam:CreateServiceLinkedRole",
            "Resource": "*",
            "Condition": {
                "StringLike": {
                    "iam:AWSServiceName": [
                        "autoscaling.amazonaws.com",
                        "ecs.amazonaws.com",
                        "ecs.application-autoscaling.amazonaws.com",
                        "spot.amazonaws.com",
                        "spotfleet.amazonaws.com"
                    ]
```

```
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": ["elasticloadbalancing:AddTags"],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "elasticloadbalancing:CreateAction": [
                        "CreateTargetGroup",
                        "CreateRule",
                        "CreateListener",
                        "CreateLoadBalancer"
                    ]
                }
            }
        }
    ]
}
```

AmazonECS 卷 InfrastructureRole PolicyFor

受AmazonECSInfrastructureRolePolicyForVolumes管身分與存取權管理政策授予 Amazon ECS 代表您進行 AWS API 呼叫所需的許可。您可以在啟動 Amazon ECS 任務和服務時，將此政策附加到隨磁碟區組態提供的 IAM 角色。該角色可讓 Amazon ECS 管理連接到您的任務的磁碟區。如需詳細資訊，請參閱 Amazon ECS 基礎設施 IAM 角色。

許可詳細資訊

AmazonECSInfrastructureRolePolicyForVolumes 受管 IAM 政策包含下列許可。遵循授與最少權限的標準安全性建議，您可以使用AmazonECSInfrastructureRolePolicyForVolumes受管理的原則做為範本，以建立僅包含您所需權限的自訂原則。

- ec2:CreateVolume— 允許主體建立 Amazon EBS 磁碟區，且僅當它們被標記為AmazonECSCreated和標AmazonECSManaged籤時。需要此權限才能建立附加至 Amazon ECS 任務的 Amazon EBS 磁碟區，並將此政策提供給 Amazon ECS 的許可最小化。
- ec2:CreateTags— 允許主體將標籤新增至 Amazon EBS 磁碟區，做為其中的一部ec2:CreateVolume分。Amazon ECS 需要此權限，才能將客戶指定的標籤新增至代表您建立的 Amazon EBS 磁碟區。

- ec2:AttachVolume— 允許主體將 Amazon EBS 磁碟區連接到 Amazon EC2 執行個體。Amazon ECS 需要此許可才能將 Amazon EBS 磁碟區連接到託管相關聯的 Amazon ECS 任務的亞馬遜 EC2 執行個體。
- ec2:DescribeVolume— 允許主體擷取有關 Amazon EBS 磁碟區的資訊。管理 Amazon EBS 磁碟區的生命週期需要此權限。
- ec2:DescribeAvailabilityZones— 允許主體擷取您帳戶中可用區域的相關資訊。這是管理 EBS 磁碟區生命週期的必要條件。
- ec2:DetachVolume— 允許主體從亞馬遜 EC2 執行個體分離 Amazon EBS 磁碟區。Amazon ECS 需要此權限才能在任務終止時從託管相關 Amazon ECS 任務的 Amazon EC2 執行個體中分離 Amazon EBS 磁碟區。
- ec2:DeleteVolume— 允許主體刪除 Amazon EBS 磁碟區。Amazon ECS 需要此權限才能刪除 Amazon ECS 任務不再使用的 Amazon EBS 磁碟區。
- ec2:DeleteTags— 允許主體從 Amazon EBS 磁碟區刪除AmazonECSManaged標籤。在 Amazon EBS 磁碟區不再與 Amazon ECS 工作負載產生關聯之後，Amazon ECS 需要此權限才能移除該磁碟區的存取權。只有在任務關閉後未刪除 Amazon EBS 磁碟區時，才適用此選項。

AmazonECSInfrastructureRolePolicyForVolumes 政策範例如下。

```
{
 "Version": "2012-10-17",
 "Statement": [
  {
   "Sid": "CreateEBSManagedVolume",
   "Effect": "Allow",
   "Action": "ec2:CreateVolume",
   "Resource": "arn:aws:ec2:*:*:volume/*",
   "Condition": {
    "ArnLike": {
     "aws:RequestTag/AmazonECSCreated": "arn:aws:ecs:*:*:task/*"
    },
    "StringEquals": {
     "aws:RequestTag/AmazonECSManaged": "true"
    }
   }
  },
  {
   "Sid": "TagOnCreateVolume",
   "Effect": "Allow",
   "Action": "ec2:CreateTags",
```

```
    "Resource": "arn:aws:ec2:*:*:volume/*",
    "Condition": {
     "ArnLike": {
      "aws:RequestTag/AmazonECSCreated": "arn:aws:ecs:*:*:task/*"
     },
     "StringEquals": {
      "ec2:CreateAction": "CreateVolume",
      "aws:RequestTag/AmazonECSManaged": "true"
     }
    }
   },
   {
    "Sid": "DescribeVolumesForLifecycle",
    "Effect": "Allow",
    "Action": [
     "ec2:DescribeVolumes",
     "ec2:DescribeAvailabilityZones"
    ],
    "Resource": "*"
   },
   {
    "Sid": "ManageEBSVolumeLifecycle",
    "Effect": "Allow",
    "Action": [
     "ec2:AttachVolume",
     "ec2:DetachVolume"
    ],
    "Resource": "arn:aws:ec2:*:*:volume/*",
    "Condition": {
     "StringEquals": {
      "aws:ResourceTag/AmazonECSManaged": "true"
     }
    }
   },
   {
    "Sid": "ManageVolumeAttachmentsForEC2",
    "Effect": "Allow",
    "Action": [
     "ec2:AttachVolume",
     "ec2:DetachVolume"
    ],
    "Resource": "arn:aws:ec2:*:*:instance/*"
   },
   {
```

```
    "Sid": "DeleteEBSManagedVolume",
    "Effect": "Allow",
    "Action": "ec2:DeleteVolume",
    "Resource": "arn:aws:ec2:*:*:volume/*",
    "Condition": {
     "ArnLike": {
      "aws:ResourceTag/AmazonECSCreated": "arn:aws:ecs:*:*:task/*"
     },
     "StringEquals": {
      "aws:ResourceTag/AmazonECSManaged": "true"
     }
    }
   }
  ]
}
```

亚马逊角ContainerServicefor色

Amazon ECS 會將此政策連接到服務角色，讓 Amazon ECS 能夠代表您對 Amazon EC2 執行個體或外部執行個體執行動作。

此政策授予管理許可，允許 Amazon ECS 容器執行個體代表您撥 AWS 打電話。如需詳細資訊，請參閱 Amazon ECS 容器執行個體 IAM 角色。

考量事項

使用 AmazonEC2ContainerServiceforEC2Role 受管 IAM 政策時，您應考慮下列建議和考量。

- 遵循授予最低權限的標準安全建議，您可以修改 AmazonEC2ContainerServiceforEC2Role 受管政策，以符合您的特定需求。如果您的使用案例不需要受管政策中授予的任何許可，請建立自訂政策並僅新增您需要的許可。例如，專為 Spot 執行個體耗盡而提供的 UpdateContainerInstancesState 許可。如果您的使用案例不需要該許可，請使用自訂政策將其排除。如需詳細資訊，請參閱 許可詳細資訊。
- 在您的容器執行個體上執行的容器可透過「執行個體中繼資料」，存取所有提供給容器執行個體角色的許可。我們建議您將容器執行個體角色中的許可限制為受管 AmazonEC2ContainerServiceforEC2Role 政策所提供的最小許可清單。若您任務中的容器需要未列出的額外許可，我們建議您使用其專屬的 IAM 角色提供這些任務。如需詳細資訊，請參閱 Amazon ECS 任務 IAM 角色。

  您可以防止 docker0 橋接器上的容器存取容器執行個體角色提供的許可。您可以執行此操作，同時仍允許 Amazon ECS 任務 IAM 角色 提供的許可，方法是在容器執行個體上執行下列 iptables 命

令。此規則生效時，容器無法查詢執行個體中繼資料。此命令假設使用預設 Docker 橋接器組態，並且將無法使用採用 host 網路模式的容器。如需詳細資訊，請參閱 網路模式。

```
sudo yum install -y iptables-services; sudo iptables --insert DOCKER USER 1 --in-interface docker+ --destination 169.254.169.254/32 --jump DROP
```

您必須將此 iptables 規則儲存在您的容器執行個體上，才能避免在重新開機時遭刪除。對於 Amazon ECS 最佳化 AMI，請使用下列命令。至於其他作業系統，請參閱該作業系統文件。

- 對於 Amazon ECS 最佳化 Amazon Linux 2 AMI：

```
sudo iptables-save | sudo tee /etc/sysconfig/iptables && sudo systemctl enable --now iptables
```

- 對於 Amazon ECS 最佳化 Amazon Linux AMI：

```
sudo service iptables save
```

許可詳細資訊

AmazonEC2ContainerServiceforEC2Role 受管 IAM 政策包含下列許可。遵循授予最低權限的標準安全建議，AmazonEC2ContainerServiceforEC2Role 受管政策可作為指南。如果您的使用案例不需要受管政策中授予的任何許可，請建立自訂政策並僅新增您需要的許可。

- ec2:DescribeTags - 允許委託人描述與 Amazon EC2 執行個體相關聯的標籤。Amazon ECS 容器代理程式會使用此許可，以支援資源標籤傳播。如需詳細資訊，請參閱 如何標記資源。
- ecs:CreateCluster - 允許委託人建立 Amazon ECS 叢集。Amazon ECS 容器代理程式會使用此許可來建立 default 叢集 (如果叢集尚不存在)。
- ecs:DeregisterContainerInstance - 允許委託人從叢集中取消註冊 Amazon ECS 容器執行個體。Amazon ECS 容器代理程式不會呼叫此 API 作業，但此權限仍有助於確保向後相容性。
- ecs:DiscoverPollEndpoint - 此動作會傳回 Amazon ECS 容器代理程式用來輪詢更新的端點。
- ecs:Poll - 允許 Amazon ECS 容器代理程式與 Amazon ECS 控制平面通訊，以報告任務狀態變更。
- ecs:RegisterContainerInstance - 允許委託人向叢集註冊容器執行個體。Amazon ECS 容器代理程式會使用此權限向叢集註冊 Amazon EC2 執行個體，並支援資源標籤傳播。
- ecs:StartTelemetrySession - 允許 Amazon ECS 容器代理程式與 Amazon ECS 控制平面通訊，以報告每個容器和任務的運作狀態資訊和指標。

- ecs:TagResource – 允許 Amazon ECS 容器代理程式在建立叢集時標記叢集，並在將容器執行個體註冊到叢集時加以標記。
- ecs:UpdateContainerInstancesState - 允許委託人修改 Amazon ECS 容器執行個體的狀態。Amazon ECS 容器代理程式使用此許可，用於 Spot 執行個體耗盡。
- ecs:Submit* - 這包括 SubmitAttachmentStateChanges、SubmitContainerStateChange 以及 SubmitTaskStateChange API 動作。Amazon ECS 容器代理程式會使用它們，將每個資源的狀態變更報告給 Amazon ECS 控制平面。Amazon ECS 容器代理程式不再使用此SubmitContainerStateChange權限，但仍有助於確保向後相容性。
- ecr:GetAuthorizationToken - 允許委託人擷取授權字符。授權字符代表您的 IAM 身分驗證憑證，且可用來存取 IAM 委託人有存取權限的任何 Amazon ECR 登錄檔。收到的授權字符的有效期為 12 小時。
- ecr:BatchCheckLayerAvailability - 將容器映像推送至 Amazon ECR 私有儲存庫時，會檢查每個映像層以驗證它是否已推送。如果已推送，則會略過映像層。
- ecr:GetDownloadUrlForLayer - 從 Amazon ECR 私有儲存庫提取容器映像時，會針對尚未快取的每個映像層呼叫此 API 一次。
- ecr:BatchGetImage - 從 Amazon ECR 私有儲存庫提取容器映像時，會呼叫此 API 一次以擷取映像資訊清單。
- logs:CreateLogStream— 允許主體為指定的 CloudWatch 記錄群組建立記錄檔資料流。
- logs:PutLogEvents - 允許委託人將批次日誌事件上傳到指定的日誌串流。

AmazonEC2ContainerServiceforEC2Role 政策範例如下。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ec2:DescribeTags",
                "ecs:CreateCluster",
                "ecs:DeregisterContainerInstance",
                "ecs:DiscoverPollEndpoint",
                "ecs:Poll",
                "ecs:RegisterContainerInstance",
                "ecs:StartTelemetrySession",
                "ecs:UpdateContainerInstancesState",
```

```
                "ecs:Submit*",
                "ecr:GetAuthorizationToken",
                "ecr:BatchCheckLayerAvailability",
                "ecr:GetDownloadUrlForLayer",
                "ecr:BatchGetImage",
                "logs:CreateLogStream",
                "logs:PutLogEvents"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "ecs:TagResource",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "ecs:CreateAction": [
                        "CreateCluster",
                        "RegisterContainerInstance"
                    ]
                }
            }
        }
    ]
}
```

亚马逊 ContainerService EventsRole

此政策授予允許 Amazon EventBridge (以前稱為「 CloudWatch 事件」) 代表您執行任務的許可。此政策可連接至在建立排程任務時指定的 IAM 角色。如需詳細資訊，請參閱 Amazon ECS EventBridge IAM 角色。

許可詳細資訊

此政策包含以下許可。

- ecs— 允許服務中的主體呼叫 Amazon ECS RunTask API。允許服務中的主體在呼叫 Amazon ECS RunTask API 時新增標籤 (TagResource)。
- iam - 允許將任何 IAM 服務角色傳遞給任何 Amazon ECS 任務。

AmazonEC2ContainerServiceEventsRole 政策範例如下。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": ["ecs:RunTask"],
            "Resource": ["*"]
        },
        {
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": ["*"],
            "Condition": {
                "StringLike": {"iam:PassedToService": "ecs-tasks.amazonaws.com"}
            }
        },
        {
            "Effect": "Allow",
            "Action": "ecs:TagResource",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "ecs:CreateAction": ["RunTask"]
                }
            }
        }
    ]
}
```

AmazonECS TaskExecution RolePolicy

受AmazonECSTaskExecutionRolePolicy管 IAM 政策授予 Amazon ECS 容器代理程式和 AWS Fargate 容器代理程式所需的許可，以代表您進行 AWS API 呼叫。此政策可新增至您的任務執行 IAM 角色。如需詳細資訊，請參閱 Amazon ECS 任務執行 IAM 角色。

許可詳細資訊

AmazonECSTaskExecutionRolePolicy 受管 IAM 政策包含下列許可。遵循授予最低權限的標準安全建議，AmazonECSTaskExecutionRolePolicy 受管政策可作為指南。如果您的使用案例不需要受管政策中授予的任何許可，請建立自訂政策並僅新增您需要的許可。

- `ecr:GetAuthorizationToken` - 允許委託人擷取授權字符。授權字符代表您的 IAM 身分驗證憑證，且可用來存取 IAM 委託人有存取權限的任何 Amazon ECR 登錄檔。收到的授權字符的有效期為 12 小時。
- `ecr:BatchCheckLayerAvailability` - 將容器映像推送至 Amazon ECR 私有儲存庫時，會檢查每個映像層以驗證它是否已推送。如果已推送，則會跳過映像層。
- `ecr:GetDownloadUrlForLayer` - 從 Amazon ECR 私有儲存庫提取容器映像時，會針對尚未快取的每個映像層呼叫此 API 一次。
- `ecr:BatchGetImage` - 從 Amazon ECR 私有儲存庫提取容器映像時，會呼叫此 API 一次以擷取映像資訊清單。
- `logs:CreateLogStream`— 允許主體為指定的 CloudWatch 記錄群組建立記錄檔資料流。
- `logs:PutLogEvents` - 允許委託人將批次日誌事件上傳到指定的日誌串流。

`AmazonECSTaskExecutionRolePolicy` 政策範例如下。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ecr:GetAuthorizationToken",
                "ecr:BatchCheckLayerAvailability",
                "ecr:GetDownloadUrlForLayer",
                "ecr:BatchGetImage",
                "logs:CreateLogStream",
                "logs:PutLogEvents"
            ],
            "Resource": "*"
        }
    ]
}
```

AmazonECS 政策 ServiceRole

`AmazonECSServiceRolePolicy` 受管 IAM 政策可讓 Amazon Elastic Container Service 管理您的叢集。此政策可新增至您的任務執行 IAM 角色。如需詳細資訊，請參閱 Amazon ECS 任務執行 IAM 角色。

許可詳細資訊

AmazonECSServiceRolePolicy 受管 IAM 政策包含下列許可。遵循授予最低權限的標準安全建議，AmazonECSServiceRolePolicy 受管政策可作為指南。如果您的使用案例不需要受管政策中授予的任何許可，請建立自訂政策並僅新增您需要的許可。

- autoscaling - 允許委託人建立、管理和描述 Amazon EC2 Auto Scaling 資源。使用叢集自動擴展功能時，在管理 Amazon EC2 Auto Scaling 群組時，這是必需的。
- autoscaling-plans – 允許主體建立、刪除和描述自動調整計畫。
- cloudwatch— 允許校長建立、管理和描述 Amazon CloudWatch 警報。
- ec2— 允許主體執行至 Amazon EC2 執行個體，以及建立和管理網路界面和標籤。
- elasticloadbalancing - 允許委託人建立、描述及刪除 Elastic Load Balancing 負載平衡器。主參與者也可以新增和描述目標群組。
- logs— 允許主體建立和描述 Amazon CloudWatch 日誌記錄群組。委託人也可以列出這些日誌群組的日誌事件。
- route53 - 允許委託人建立、管理和刪除 Amazon Route 53 託管區域。委託人也可以檢視 Amazon Route 53 運作狀態檢查組態和資訊。如需有關託管區域的詳細資訊，請參閱使用託管區域。
- servicediscovery— 允許主參與者建立、管理和刪除 AWS Cloud Map 服務，以及建立私人 DNS 命名空間。
- events— 允許主體建立、管理和刪除 Amazon EventBridge 規則及其目標。

AmazonECSServiceRolePolicy 政策範例如下。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ECSTaskManagement",
            "Effect": "Allow",
            "Action": [
                "ec2:AttachNetworkInterface",
                "ec2:CreateNetworkInterface",
                "ec2:CreateNetworkInterfacePermission",
                "ec2:DeleteNetworkInterface",
                "ec2:DeleteNetworkInterfacePermission",
                "ec2:Describe*",
                "ec2:DetachNetworkInterface",
                "elasticloadbalancing:DeregisterInstancesFromLoadBalancer",
```

```
                "elasticloadbalancing:DeregisterTargets",
                "elasticloadbalancing:Describe*",
                "elasticloadbalancing:RegisterInstancesWithLoadBalancer",
                "elasticloadbalancing:RegisterTargets",
                "route53:ChangeResourceRecordSets",
                "route53:CreateHealthCheck",
                "route53:DeleteHealthCheck",
                "route53:Get*",
                "route53:List*",
                "route53:UpdateHealthCheck",
                "servicediscovery:DeregisterInstance",
                "servicediscovery:Get*",
                "servicediscovery:List*",
                "servicediscovery:RegisterInstance",
                "servicediscovery:UpdateInstanceCustomHealthStatus"
            ],
            "Resource": "*"
        },
        {
            "Sid": "AutoScaling",
            "Effect": "Allow",
            "Action": [
                "autoscaling:Describe*"
            ],
            "Resource": "*"
        },
        {
            "Sid": "AutoScalingManagement",
            "Effect": "Allow",
            "Action": [
                "autoscaling:DeletePolicy",
                "autoscaling:PutScalingPolicy",
                "autoscaling:SetInstanceProtection",
                "autoscaling:UpdateAutoScalingGroup",
                "autoscaling:PutLifecycleHook",
                "autoscaling:DeleteLifecycleHook",
                "autoscaling:CompleteLifecycleAction",
                "autoscaling:RecordLifecycleActionHeartbeat"
            ],
            "Resource": "*",
            "Condition": {
                "Null": {
                    "autoscaling:ResourceTag/AmazonECSManaged": "false"
                }
```

```
            }
        },
        {
            "Sid": "AutoScalingPlanManagement",
            "Effect": "Allow",
            "Action": [
                "autoscaling-plans:CreateScalingPlan",
                "autoscaling-plans:DeleteScalingPlan",
                "autoscaling-plans:DescribeScalingPlans",
                "autoscaling-plans:DescribeScalingPlanResources"
            ],
            "Resource": "*"
        },
        {
            "Sid": "EventBridge",
            "Effect": "Allow",
            "Action": [
                "events:DescribeRule",
                "events:ListTargetsByRule"
            ],
            "Resource": "arn:aws:events:*:*:rule/ecs-managed-*"
        },
        {
            "Sid": "EventBridgeRuleManagement",
            "Effect": "Allow",
            "Action": [
                "events:PutRule",
                "events:PutTargets"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "events:ManagedBy": "ecs.amazonaws.com"
                }
            }
        },
        {
            "Sid": "CWAlarmManagement",
            "Effect": "Allow",
            "Action": [
                "cloudwatch:DeleteAlarms",
                "cloudwatch:DescribeAlarms",
                "cloudwatch:PutMetricAlarm"
            ],
```

```
            "Resource": "arn:aws:cloudwatch:*:*:alarm:*"
        },
        {
            "Sid": "ECSTagging",
            "Effect": "Allow",
            "Action": [
                "ec2:CreateTags"
            ],
            "Resource": "arn:aws:ec2:*:*:network-interface/*"
        },
        {
            "Sid": "CWLogGroupManagement",
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogGroup",
                "logs:DescribeLogGroups",
                "logs:PutRetentionPolicy"
            ],
            "Resource": "arn:aws:logs:*:*:log-group:/aws/ecs/*"
        },
        {
            "Sid": "CWLogStreamManagement",
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogStream",
                "logs:DescribeLogStreams",
                "logs:PutLogEvents"
            ],
            "Resource": "arn:aws:logs:*:*:log-group:/aws/ecs/*:log-stream:*"
        },
        {
            "Sid": "ExecuteCommandSessionManagement",
            "Effect": "Allow",
            "Action": [
                "ssm:DescribeSessions"
            ],
            "Resource": "*"
        },
        {
            "Sid": "ExecuteCommand",
            "Effect": "Allow",
            "Action": [
                "ssm:StartSession"
            ],
```

```
            "Resource": [
                "arn:aws:ecs:*:*:task/*",
                "arn:aws:ssm:*:*:document/AmazonECS-ExecuteInteractiveCommand"
            ]
        },
        {
            "Sid": "CloudMapResourceCreation",
            "Effect": "Allow",
            "Action": [
                "servicediscovery:CreateHttpNamespace",
                "servicediscovery:CreateService"
            ],
            "Resource": "*",
            "Condition": {
                "ForAllValues:StringEquals": {
                    "aws:TagKeys": [
                        "AmazonECSManaged"
                    ]
                }
            }
        },
        {
            "Sid": "CloudMapResourceTagging",
            "Effect": "Allow",
            "Action": "servicediscovery:TagResource",
            "Resource": "*",
            "Condition": {
                "StringLike": {
                    "aws:RequestTag/AmazonECSManaged": "*"
                }
            }
        },
        {
            "Sid": "CloudMapResourceDeletion",
            "Effect": "Allow",
            "Action": [
                "servicediscovery:DeleteService"
            ],
            "Resource": "*",
            "Condition": {
                "Null": {
                    "aws:ResourceTag/AmazonECSManaged": "false"
                }
            }
```

```
        },
        {
            "Sid": "CloudMapResourceDiscovery",
            "Effect": "Allow",
            "Action": [
                "servicediscovery:DiscoverInstances",
                "servicediscovery:DiscoverInstancesRevision"
            ],
            "Resource": "*"
        }
    ]
}
```

AmazonECSInfrastructureRolePolicyForServiceConnectTransportLayerSecur

提供代表您管理 Amazon ECS AWS 服務 Connect TLS 功能所需的管理存取權 AWS Private Certificate Authority、Secrets Manager 和其他服務。

許可詳細資訊

AmazonECSInfrastructureRolePolicyForServiceConnectTransportLayerSecurity 受管 IAM 政策包含下列許可。遵循授予最低權限的標準安全建議，AmazonECSInfrastructureRolePolicyForServiceConnectTransportLayerSecurity 受管政策可作為指南。如果您的使用案例不需要受管政策中授予的任何許可，請建立自訂政策並僅新增您需要的許可。

- secretsmanager:CreateSecret— 允許主參與者建立密碼。這是服務 Connect TLS 所必需的，Amazon ECS 會將客戶的私密金鑰保存在客戶的 Secrets Manager 中。
- secretsmanager:TagResource— 允許主參與者在建立的密碼上附加標籤。這是服務 Connect TLS 所必需的，因為 Amazon ECS 會代表客戶建立密碼，並將標籤附加至資源。這些標籤為客戶提供了一種更簡單的方法，以識別受管理的密碼並限制對這些機密的操作。
- secretsmanager:DescribeSecret— 允許主體描述密碼並擷取目前版本階段。Amazon ECS 需要進行 Amazon ECS 服務 Connect TLS 材料輪換。
- secretsmanager:UpdateSecret— 允許主體更新密碼。Amazon ECS 必須執行 Amazon ECS 服務 Connect TLS 材料輪換，並使用新材料更新秘密。
- secretsmanager:GetSecretValue— 允許主體取得秘密值。Amazon ECS 需要進行 Amazon ECS 服務 Connect TLS 材料輪換。
- secretsmanager:PutSecretValue— 允許主體放置秘密值。Amazon ECS 需要進行 Amazon ECS 服務 Connect TLS 材料輪換。

- secretsmanager:UpdateSecretVersionStage— 允許主體更新秘密版本階段。Amazon ECS 需要進行 Amazon ECS 服務 Connect TLS 材料輪換。
- acm-pca:IssueCertificate— 允許校長呼叫 IssueCertificate Amazon ECS 服務 Connect TLS. End entity certificate ECS 需要為客戶的上游服務生成證書。
- acm-pca:GetCertificate— 允許校長呼叫 GetCertificate Amazon ECS 服務 Connect TLS. End entity certificate
- acm-pca:GetCertificateAuthorityCertificate— 允許主體取得憑證授權單位憑證。Amazon ECS 服務 Connect TLS 需要此功能，以便客戶的下游服務可以信任上游終端實體憑證。
- acm-pca:DescribeCertificateAuthority— 允許主體取得有關憑證授權單位的詳細資料。Amazon ECS 服務 Connect TLS 必須重複使用簽署演算法等資訊，才能建立 CSR (憑證簽署要求)。

AmazonECSInfrastructureRolePolicyForServiceConnectTransportLayerSecurity 政策範例如下。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "CreateSecret",
            "Effect": "Allow",
            "Action": "secretsmanager:CreateSecret",
            "Resource": "arn:aws:secretsmanager:*:*:secret:ecs-sc!*",
            "Condition": {
                "ArnLike": {
                    "aws:RequestTag/AmazonECSCreated": [
                        "arn:aws:ecs:*:*:service/*/*",
                        "arn:aws:ecs:*:*:task-set/*/*"
                    ]
                },
                "StringEquals": {
                    "aws:RequestTag/AmazonECSManaged": "true",
                    "aws:ResourceAccount": "${aws:PrincipalAccount}"
                }
            }
        },
        {
            "Sid": "TagOnCreateSecret",
```

```
            "Effect": "Allow",
            "Action": "secretsmanager:TagResource",
            "Resource": "arn:aws:secretsmanager:*:*:secret:ecs-sc!*",
            "Condition": {
                "ArnLike": {
                     "aws:RequestTag/AmazonECSCreated": [
                        "arn:aws:ecs:*:*:service/*/*",
                        "arn:aws:ecs:*:*:task-set/*/*"
                    ]
                },
                "StringEquals": {
                    "aws:RequestTag/AmazonECSManaged": "true",
                    "aws:ResourceAccount": "${aws:PrincipalAccount}"
                }
            }
        },
        {
            "Sid": "RotateTLSCertificateSecret",
            "Effect": "Allow",
            "Action": [
                "secretsmanager:DescribeSecret",
                "secretsmanager:UpdateSecret",
                "secretsmanager:GetSecretValue",
                "secretsmanager:PutSecretValue",
                "secretsmanager:DeleteSecret",
                "secretsmanager:RotateSecret",
                "secretsmanager:UpdateSecretVersionStage"
            ],
            "Resource": "arn:aws:secretsmanager:*:*:secret:ecs-sc!*",
            "Condition": {
                "StringEquals": {
                    "secretsmanager:ResourceTag/aws:secretsmanager:owningService":
"ecs-sc",
                    "aws:ResourceAccount": "${aws:PrincipalAccount}"
                }
            }
        },
        {
            "Sid": "ManagePrivateCertificateAuthority",
            "Effect": "Allow",
            "Action": [
                "acm-pca:GetCertificate",
                "acm-pca:GetCertificateAuthorityCertificate",
                "acm-pca:DescribeCertificateAuthority"
```

```
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/AmazonECSManaged": "true"
                }
            }
        },
        {
            "Sid": "ManagePrivateCertificateAuthorityForIssuingEndEntityCertificate",
            "Effect": "Allow",
            "Action": [
                "acm-pca:IssueCertificate"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/AmazonECSManaged": "true",
                    "acm-pca:TemplateArn":"arn:aws:acm-pca:::template/
EndEntityCertificate/V1"
                }
            }
        }
    ]
}
```

AWSApplicationAutoscalingECSServicePolicy

您無法將 AWSApplicationAutoscalingECSServicePolicy 連接至您的 IAM 實體。此政策會連接到服務連結角色，允許 Application Auto Scaling 代表您執行動作。如需詳細資訊，請參閱 Application Auto Scaling 的服務連結角色。

AWSCodeDeployRoleForECS

您不得將 AWSCodeDeployRoleForECS 連接到 IAM 實體。此原則附加至服務連結角色，可 CodeDeploy 代表您執行動作。如需詳細資訊，請參閱《使用指南》 CodeDeploy中的〈建立服務角色 AWS CodeDeploy〉。

AWSCodeDeployRoleForECSLimited

您不得將 AWSCodeDeployRoleForECSLimited 連接到 IAM 實體。此原則附加至服務連結角色，可 CodeDeploy 代表您執行動作。如需詳細資訊，請參閱《使用指南》 CodeDeploy中的〈建立服務角色 AWS CodeDeploy〉。

Amazon ECS 更新 AWS 受管政策

檢視有關 Amazon ECS AWS 受管政策更新的詳細資訊，因為此服務開始追蹤這些變更。如需有關此頁面變更的自動提醒，請訂閱 Amazon ECS 文件歷史記錄頁面上的 RSS 摘要。

變更	描述	日期
添加新的 AmazonECS 安 InfrastructureRolePolicyFo rServiceConnectTransportLay er全政策	添加了新的 AmazonECS InfrastructureRolePolicyFor ServiceConnectTransportLaye rSecurity 政策，提供管理訪問 AWS KMS，Secrets Manager AWS Private Certificate Authority，並使 Amazon ECS 服務 Connect TLS 功能正常工作。	2024年1月22日
添加新政策 AmazonECS 卷 InfrastructureRole PolicyFor	已新增AmazonECS InfrastructureRole PolicyForVolumes 原則。該政策授予 Amazon ECS 進行 AWS API 呼叫以管理與 Amazon ECS 工作負載相關聯的 Amazon EBS 磁碟區所需的許可。	2024年1月11日
將權限添加到 AmazonECS 政策 ServiceRole	受AmazonECSServiceRo lePolicy 管 IAM 政策已更新為新events許可和其他許可autoscali	2023 年 12 月 4 日

變更	描述	日期
	ng 和autoscaling-plans許可。	
將許可添加到亞馬遜 EC2 ContainerService EventsRole	受AmazonECSServiceRolePolicy 管 IAM 政策已更新，以允許存取 AWS Cloud Map DiscoverInstancesRevision API 作業。	2023 年 10 月 4 日
將權限添加到亞馬遜 Container Servicefor	AmazonEC2ContainerServiceforEC2Role 原則已修改以新增ecs:TagResource 權限，其中包括僅限於新建立的叢集和已註冊容器執行個體的權限的條件。	2023 年 3 月 6 日
新增許可至 the section called “亞馬遜 FullAccess”	AmazonECS_FullAccess 原則已修改以新增elasticloadbalancing:AddTags 權限，其中包括僅限新建立的負載平衡器、目標群組、規則和建立的接聽程式的權限的條件。此許可不允許將標籤新增至任何已建立的 Elastic Load Balancing 資源。	2023 年 1 月 4 日
Amazon ECS 開始追蹤變更	Amazon ECS 開始追蹤其 AWS 受管政策的變更。	2021 年 6 月 8 日

淘汰 Amazon 彈性容器服務的 AWS 受管 IAM 政策

下列 AWS 受管 IAM 政策已逐步淘汰。這些政策現在會由更新的政策取代。我們建議您更新使用者或角色，以使用更新的政策。

亚马逊 ContainerService FullAccess

> **Important**
>
> AmazonEC2ContainerServiceFullAccess 受管 IAM 政策已於 2021 年 1 月 29 日淘汰，以使用 iam:passRole 許可回應安全問題。此許可授予帳戶中的角色對所有資源的存取權，包括憑證。因為政策已逐步淘汰，您無法將政策連接至任何新的使用者或角色。已連接政策的所有使用者或角色都可以繼續使用它。但我們建議您更新使用者或角色，以使用 AmazonECS_FullAccess 受管政策。如需詳細資訊，請參閱 遷移至 AmazonECS_FullAccess 受管政策。

亞馬遜角ContainerService色

> **Important**
>
> AmazonEC2ContainerServiceRole 受管 IAM 政策已逐步淘汰。現在已由 Amazon ECS 服務連結角色取代。如需詳細資訊，請參閱 使用 Amazon ECS 的服務連結角色。

亚马逊 ContainerService AutoscaleRole

> **Important**
>
> AmazonEC2ContainerServiceAutoscaleRole 受管 IAM 政策已逐步淘汰。現在已由 Amazon ECS 的 Application Auto Scaling 服務連結角色取代。如需詳細資訊，請參閱《Application Auto Scaling 使用者指南》中的適用於 Application Auto Scaling 的服務連結角色。

遷移至 AmazonECS_FullAccess 受管政策

AmazonEC2ContainerServiceFullAccess 受管 IAM 政策已於 2021 年 1 月 29 日淘汰，以使用 iam:passRole 許可回應安全問題。此許可授予帳戶中的角色對所有資源的存取權，包括憑證。因為政策已逐步淘汰，您無法將政策連接至任何新的群組、使用者或角色。已連接政策的所有群組、使用者或角色都可以繼續使用它。但我們建議您更新群組、使用者或角色，以使用 AmazonECS_FullAccess 受管政策。

AmazonECS_FullAccess 政策授予的許可包含以系統管理員身分使用 ECS 所需的完整許可清單。如果您目前使用不在原則中的AmazonEC2ContainerServiceFullAccess原則所授與的權限，您可以將它們新增至內嵌原則陳述式。AmazonECS_FullAccess如需詳細資訊，請參閱 AWS Amazon 彈性容器服務的受管政策。

使用下列步驟來判斷您是否有任何群組、使用者或角色目前正在使用 AmazonEC2ContainerServiceFullAccess 受管 IAM 政策。然後，更新它們以分開先前的政策並連接 AmazonECS_FullAccess 政策。

若要更新群組、使用者或角色以使用 AmazonECS FullAccess 政策 ()AWS Management Console

1. 前往 https://console.aws.amazon.com/iam/ 開啟 IAM 主控台。
2. 在導覽窗格中，選擇 Policies (政策)，然後搜尋並選取 AmazonEC2ContainerServiceFullAccess 政策。
3. 選擇 Policy usage (政策使用) 索引標籤，它顯示目前正在使用此政策的所有 IAM 角色。
4. 針對目前使用該AmazonEC2ContainerServiceFullAccess政策的每個 IAM 角色，選取角色並使用下列步驟卸離逐步淘汰的政策並附加AmazonECS_FullAccess政策。

    a. 在權限選項卡上，選擇 AmazonEC2 ContainerService FullAccess 政策旁邊的 X。

    b. 選擇新增許可。

    c. 選擇 [直接附加現有原則]，搜尋並選取 Amazonecs_ FullAccess 政策，然後選擇 [下一步：檢閱]。

    d. 檢閱變更，然後選擇 Add permissions (新增許可)。

    e. 對使用 AmazonEC2ContainerServiceFullAccess 政策的每個群組、使用者或角色重複這些步驟。

更新群組、使用者或角色以使用 **AmazonECS_FullAccess** 政策 (AWS CLI)

1. 使用此命generate-service-last-accessed-details令可產生報告，其中包含有關上次使用淘汰原則時間的詳細資訊。

```
aws iam generate-service-last-accessed-details \
     --arn arn:aws:iam::aws:policy/AmazonEC2ContainerServiceFullAccess
```

    輸出範例：

```
{
```

```
    "JobId": "32bb1fb0-1ee0-b08e-3626-ae83EXAMPLE"
}
```

2. 使用上一個輸出的工作 ID 搭配命get-service-last-accessed-details令來擷取服務的上次存取報告。此報告顯示上次使用淘汰政策的 IAM 實體的 Amazon 資源名稱 (ARN)。

```
aws iam get-service-last-accessed-details \
      --job-id 32bb1fb0-1ee0-b08e-3626-ae83EXAMPLE
```

3. 請使用下列其中一個命令，將 AmazonEC2ContainerServiceFullAccess 政策從群組、使用者或角色中分開。

   - detach-group-policy
   - detach-role-policy
   - detach-user-policy

4. 請使用下列其中一個命令，將 AmazonECS_FullAccess 政策連接至群組、使用者或角色。

   - attach-group-policy
   - attach-role-policy
   - attach-user-policy

使用 Amazon ECS 的服務連結角色

Amazon 彈性容器服務使用 AWS Identity and Access Management (IAM) 服務連結角色。服務連結角色是直接連結至 Amazon ECS 的一種特殊 IAM 角色類型。服務連結角色由 Amazon ECS 預先定義，內含該服務代您呼叫其他 AWS 服務所需的所有許可。

服務連結角色可讓設定 Amazon ECS 更為簡單，因為您不必手動新增必要的許可。Amazon ECS 定義其服務連結角色的許可，除非另有定義，否則僅有 Amazon ECS 可以擔任其角色。定義的許可包括信任政策和許可政策，且該許可政策無法附加至其他 IAM 實體。

如需關於支援服務連結角色的其他服務資訊，請參閱《可搭配 IAM 運作的AWS 服務》，尋找服務連結角色欄中顯示為是的服務。選擇具有連結的是，以檢視該服務的服務連結角色文件。

Amazon ECS 的服務連結角色許可

Amazon ECS 使用名為的服務連結角色。AWSServiceRoleForECS

服 AWSServiceRoleForECS 務連結角色會信任下列服務擔任該角色：

- ecs.amazonaws.com

名為 AmazonECS 的角色許可政策ServiceRolePolicy 允許 Amazon ECS 在指定的資源上完成下列動作：

- 動作：使用適用於 Amazon ECS 任務的 awsvpc 網路模式時，Amazon ECS 會管理與任務關聯的彈性網路介面的生命週期。這也包括 Amazon ECS 新增至彈性網路介面的標籤。
- 動作：將負載平衡器與 Amazon ECS 服務搭配使用時，Amazon ECS 會使用負載平衡器來管理資源的註冊和取消註冊。
- 動作：使用 Amazon ECS 服務探索時，Amazon ECS 會管理所需的路由 53 和 AWS Cloud Map 資源，以便服務探索運作。
- 動作：使用 Amazon ECS 服務自動擴展時，Amazon ECS 會管理必要的自動擴展資源。
- 動作：Amazon ECS 會建立和管理 CloudWatch 警示和日誌串流，以協助監控您的 Amazon ECS 資源。
- 動作：使用 Amazon ECS Exec 時，Amazon ECS 會管理必要的許可，以便對您的任務啟動 Amazon ECS Exec 工作階段。
- 動作：使用 Amazon ECS Service Connect 時，Amazon ECS 會管理必要的 AWS Cloud Map 資源，以便使用此功能。
- 動作：使用 Amazon ECS 容量提供者時，Amazon ECS 會管理必要的許可，以便修改 Auto Scaling 群組及其 Amazon EC2 執行個體。

您必須設定許可，IAM 實體 (如使用者、群組或角色) 才可建立、編輯或刪除服務連結角色。如需詳細資訊，請參閱 IAM 使用者指南中的服務連結角色許可。

建立 Amazon ECS 的服務連結角色

在大多數情況下，您不需要手動建立一個服務連結角色。當您在 AWS Management Console、或 AWS API 中建立叢集或更新服務時 AWS CLI，Amazon ECS 會為您建立服務連結角色。如果您在建立叢集後看不到AWSServiceRoleForECS角色，請執行下列步驟來修正此問題：

- 驗證和設定許可，以允許 Amazon ECS 代表您建立、編輯或刪除服務連結角色。如需詳細資訊，請參閱 IAM 使用者指南中的服務連結角色許可。
- 重試叢集建立作業，或手動建立服務連結角色。

您可以使用 IAM 主控台建立AWSServiceRoleForECS服務連結角色。在 AWS CLI 或 AWS API 中，使用ecs.amazonaws.com服務名稱建立服務連結角色。如需詳細資訊，請參閱《IAM 使用者指南》中的「建立服務連結角色」。

> **Important**
>
> 此服務連結角色可以顯示在您的帳戶，如果您於其他服務中完成一項動作時，可以使用支援此角色的功能。

若您刪除此服務連結角色，之後需要再次建立，您可以在帳戶中使用相同程序重新建立角色。在建立叢集，或者建立或更新服務時，Amazon ECS 會再次為您建立服務連結角色。

如果您刪除這個服務連結角色，則可使用相同的 IAM 程序再次建立該角色。

編輯 Amazon ECS 的服務連結角色

Amazon ECS 不允許您編輯 AWSServiceRoleForECS 服務連結角色。因為有各種實體可能會參考服務連結角色，所以您無法在建立角色之後變更角色名稱。然而，您可使用 IAM 來編輯角色描述。如需詳細資訊，請參閱 IAM 使用者指南中的編輯服務連結角色。

刪除 Amazon ECS 的服務連結角色

若您不再使用需要服務連結角色的功能或服務，我們建議您刪除該角色。如此一來，您就沒有未主動監控或維護的未使用實體。然而，在手動刪除服務連結角色之前，您必須先清除資源。

> **Note**
>
> 若 Amazon ECS 服務在您試圖刪除資源時正在使用該角色，刪除可能會失敗。若此情況發生，請等待數分鐘後並再次嘗試操作。

檢查服務連結角色是否有作用中工作階段

1. 前往 https://console.aws.amazon.com/iam/ 開啟 IAM 主控台。
2. 在導覽窗格中，選擇 [角色] 並選擇 AWSServiceRoleForECS 名稱 (而非核取方塊)。
3. 在 Summary (摘要) 頁面上，選擇 Access Advisor (存取 Advisor)，然後檢閱服務連結角色的近期活動。

> **Note**
>
> 如果您不確定 Amazon ECS 是否正在使用該 AWSServiceRoleForECS 角色，可以嘗試刪除該角色。如果服務正在使用該角色，則刪除會失敗，而您可以檢視正在使用該角色的區域。如果服務正在使用該角色，您必須先等到工作階段結束，才能刪除該角色。您無法撤銷服務連結角色的工作階段。

若要移除 AWSServiceRoleForECS 服務連結角色所使用的 Amazon ECS 資源

您必須先刪除所有 AWS 區域中的所有 Amazon ECS 叢集，才能刪除 AWSServiceRoleForECS 角色。

1. 將所有區域中所有 Amazon ECS 服務的所需計數都縮減到 0，然後刪除該服務。如需更多詳細資訊，請參閱 使用主控台更新 Amazon ECS 服務 和 使用主控台刪除 Amazon ECS 服務。
2. 強制將所有容器執行個體從所有區域中的所有叢集取消註冊。如需詳細資訊，請參閱 取消註冊 Amazon ECS 容器執行個體。
3. 刪除所有區域中的所有 Amazon ECS 叢集。如需詳細資訊，請參閱 刪除 Amazon ECS 叢集。

使用 IAM 手動刪除服務連結角色

使用 IAM 主控台或 AWS API 刪除 AWSServiceRoleForECS服務連結角色。 AWS CLI如需詳細資訊，請參閱《IAM 使用者指南》中的刪除服務連結角色。

Amazon ECS 服務連結角色的支援區域

Amazon ECS 在所有提供服務的區域中支援使用服務連結角色。如需詳細資訊，請參閱 AWS 區域與端點。

Amazon ECS 的 IAM 角色

IAM 角色是您可以在帳戶中建立的另一種 IAM 身分，具有特定的許可。在 Amazon ECS 中，您可以建立角色，將許可授與 Amazon ECS 資源 (例如容器或服務)。

Amazon ECS 所需的角色取決於任務定義啟動類型和您使用的功能。使用下表判斷 Amazon ECS 所需的 IAM 角色。

角色	定義	需要時	其他資訊
任務執行角色	此角色可讓 Amazon ECS 代表您使用其他 AWS 服務。	您的任務託管在外部實例上 AWS Fargate 或在外部實例上，並且： • 從 Amazon ECR 私有存儲庫中提取容器映像。 • 從執行任務的帳戶不同的帳戶中的 Amazon ECR 私有儲存庫提取容器映像。 • 使用記錄驅動程式將容器 CloudWatch 記錄檔傳送至記`awslogs`錄檔。 您的任務託管在任一個 AWS Fargate 或 Amazon EC2 執行個體上，並且： • 使用私人登錄驗證。 • 使用執行階段監視。 • 工作定義會使用 Secret Manager 密碼或 AWS Systems Manager 管理員參數存放區參數來參照敏感資料。	Amazon ECS 任務執行 IAM 角色

角色	定義	需要時	其他資訊
任務角色	此角色可讓您的應用程式程式碼 (在容器上) 使用其他 AWS 服務。	您的應用程式會存取其他 AWS 服務，例如 Amazon S3。	Amazon ECS 任務 IAM 角色
容器執行個體角色	此角色可讓您的 EC2 執行個體或外部執行個體向叢集註冊。	您的任務託管在 Amazon EC2 執行個體或外部執行個體上。	Amazon ECS 容器執行個體 IAM 角色
Amazon ECS Anywhere 角色	此角色可讓您的外部執行個體存取 AWS API。	您的任務託管在外部實例上。	Amazon ECS Anywhere IAM 角色
Amazon ECS 角色 CodeDeploy	此角色允許 CodeDeploy 對您的服務進行更新。	您可以使用 CodeDeploy 藍/綠部署類型來部署服務。	Amazon ECS CodeDeploy IAM 角色
Amazon ECS 角色 EventBridge	此角色允許 EventBridge 對您的服務進行更新。	您可以使用 EventBridge 規則和目標來排程工作。	Amazon ECS EventBridge IAM 角色

角色	定義	需要時	其他資訊
Amazon ECS 基礎設施角色	此角色可讓 Amazon ECS 管理叢集中的基礎設施資源。	• 您想要將 Amazon EBS 磁碟區連接到您的 Fargate 或 EC2 啟動類型的 Amazon ECS 任務。基礎設施角色可讓 Amazon ECS 為您的任務管理 Amazon EBS 磁碟區。 • 您想要使用傳輸層安全性 (TLS) 來加密 Amazon ECS 服務 Connect 服務之間的流量。	Amazon ECS 基礎設施 IAM 角色

Amazon ECS 中 IAM 角色的最佳實務

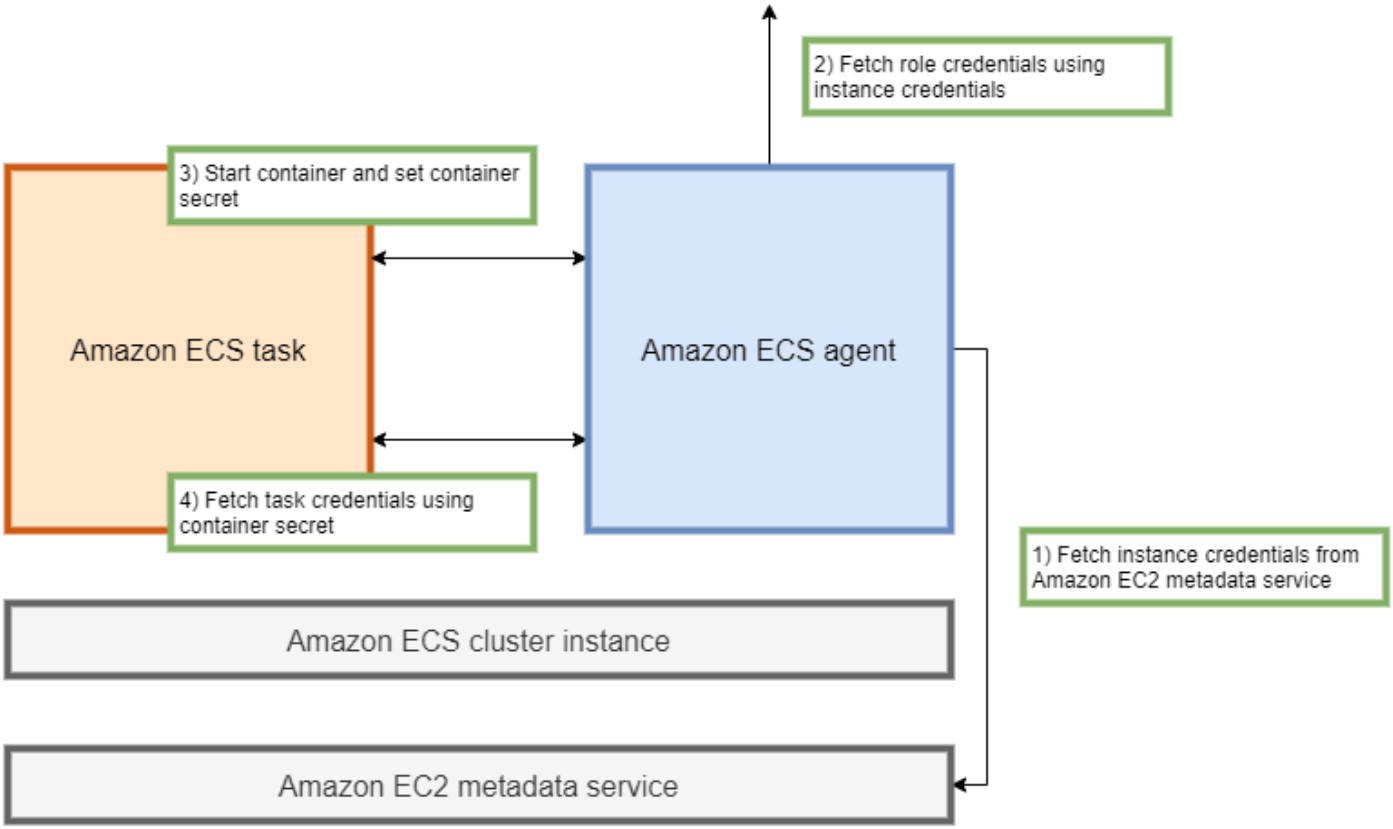
建議您指派工作角色。其角色可以與執行 Amazon EC2 執行個體所使用的角色有所區別。為每項任務指派一個角色的做法符合最低權限存取原則，並且允許對動作和資源進行更精細的控制。

為任務指派 IAM 角色時，您必須使用下列信任政策，以便每個任務都可以承擔與 EC2 執行個體所使用角色不同的 IAM 角色。如此一來，您的任務就不會繼承 EC2 執行個體的角色。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "Service": "ecs-tasks.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
```

```
}
```

當您將任務角色新增至任務定義時，Amazon ECS 容器代理程式會自動為該任務建立具有唯一憑證 ID 的字符 (例如 12345678-90ab-cdef-1234-567890abcdef)。然後，此字符和角色憑證會新增至代理程式的內部快取。代理程式會以憑證 ID 的 URI 填入容器中的環境變數 AWS_CONTAINER_CREDENTIALS_RELATIVE_URI (例如 /v2/credentials/12345678-90ab-cdef-1234-567890abcdef)。



您可以透過將環境變數附加至 Amazon ECS 容器代理程式的 IP 地址，然後在產生的字串上執行 curl 命令，從容器內手動擷取暫時性角色憑證。

```
curl 169.254.170.2$AWS_CONTAINER_CREDENTIALS_RELATIVE_URI
```

預期的輸出如下：

```
{
 "RoleArn": "arn:aws:iam::123456789012:role/SSMTaskRole-SSMFargateTaskIAMRole-
DASWWSF2WGD6",
 "AccessKeyId": "AKIAIOSFODNN7EXAMPLE",
```

```
 "SecretAccessKey": "wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY",
 "Token": "IQoJb3JpZ2luX2VjEEM/Example==",
 "Expiration": "2021-01-16T00:51:53Z"
}
```

較新版本的 AWS SDK 會在進行 AWS API 呼叫時自動從`AWS_CONTAINER_CREDENTIALS_RELATIVE_URI`環境變數擷取這些認證。

輸出包括一個存取金鑰配對，其中包含秘密存取金鑰 ID 和您的應用程式用來存取 AWS 資源的秘密金鑰。它還包括一個令牌， AWS 用於驗證憑據是否有效。依預設，指派給使用任務角色之任務的憑證有效期為六小時。之後，Amazon ECS 容器代理程式會自動輪換。

任務執行角色

任務執行角色用於授與 Amazon ECS 容器代理程式權限，以代表您呼叫特定 AWS API 動作。例如，當您使用時 AWS Fargate，Fargate 需要一個 IAM 角色，允許其從 Amazon ECR 提取映像並將日誌寫入 CloudWatch 日誌。當任務引用存儲在中的密碼（例如圖像提取密鑰）時 AWS Secrets Manager，也需要 IAM 角色。

> **Note**
>
> 如果您以經身份驗證的使用者提取映像，則不太可能受到 Docker Hub 提取速率限制所發生的變更影響。如需詳細資訊，請參閱容器執行個體的私有登錄檔身分驗證。
> 透過使用 Amazon ECR 和 Amazon ECR Public，您可以避免 Docker 施加的限制。如果您從 Amazon ECR 提取映像，這也有助於縮短網路提取時間，並減少流量離開 VPC 時的資料傳輸變更。

> **Important**
>
> 當您使用 Fargate 時，您必須使用 `repositoryCredentials` 對私有映像登錄檔進行驗證。無法為 Fargate 上託管的任務設定 Amazon ECS 容器代理程式環境變數 `ECS_ENGINE_AUTH_TYPE` 或 `ECS_ENGINE_AUTH_DATA` 或修改 `ecs.config` 檔案。如需詳細資訊，請參閱任務的私有登錄檔身分驗證。

容器執行個體角色

Amazon ECS 容器代理程式是在 Amazon ECS 叢集的每個 Amazon EC2 執行個體上執行的容器。其使用作業系統上可用的 `init` 命令在 Amazon ECS 外部進行初始化。因此，無法透過

任務角色授予許可。反之，必須將許可指派給代理程式執行的 Amazon EC2 執行個體。範例 AmazonEC2ContainerServiceforEC2Role 政策中的動作清單必須授予 ecsInstanceRole。如果您不這麼做，執行個體無法加入叢集。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ec2:DescribeTags",
                "ecs:CreateCluster",
                "ecs:DeregisterContainerInstance",
                "ecs:DiscoverPollEndpoint",
                "ecs:Poll",
                "ecs:RegisterContainerInstance",
                "ecs:StartTelemetrySession",
                "ecs:UpdateContainerInstancesState",
                "ecs:Submit*",
                "ecr:GetAuthorizationToken",
                "ecr:BatchCheckLayerAvailability",
                "ecr:GetDownloadUrlForLayer",
                "ecr:BatchGetImage",
                "logs:CreateLogStream",
                "logs:PutLogEvents"
            ],
            "Resource": "*"
        }
    ]
}
```

在此政策中，ecr和 logs API 動作允許執行個體上執行的容器從 Amazon ECR 提取映像並將日誌寫入 Amazon CloudWatch。ecs 動作可讓代理程式註冊和取消註冊執行個體，以及與 Amazon ECS 控制平面進行通訊。其中，ecs:CreateCluster 動作是選用項目。

服務連結角色

您可以使用 Amazon ECS 的服務連結角色來授予 Amazon ECS 服務代表您呼叫其他服務 API 的許可。Amazon ECS 需要許可，才能使用目標群組建立和刪除網路介面、註冊和取消註冊目標。其也需要必要的許可，才能建立和刪除擴展政策。透過任務連結角色授予許可。在您第一次使用服務時代表您建立此角色。

> **Note**
>
> 如果您不小心刪除服務連結角色，您可以重新建立角色。如需指示，請參閱建立服務連結角色。

角色建議

建議您在設定任務 IAM 角色和政策時執行下列動作。

封鎖 Amazon EC2 中繼資料的存取

當您在 Amazon EC2 執行個體上執行任務時，強烈建議您封鎖 Amazon EC2 中繼資料的存取，以防止容器繼承指派給這些執行個體的角色。如果您的應用程式必須呼叫 AWS API 動作，請改用 IAM 角色執行任務。

若要防止以橋接模式執行的任務存取 Amazon EC2 中繼資料，請執行以下命令或更新執行個體的使用者資料。如需更新執行個體使用者資料的詳細指示，請參閱此 AWS 支援文章。如需有關任務定義橋接模式的詳細資訊，請參閱任務定義網路模式。

```
sudo yum install -y iptables-services; sudo iptables --insert FORWARD 1 --in-interface
 docker+ --destination 192.0.2.0/32 --jump DROP
```

如需在重新開機後保留此變更，請執行下列 Amazon Machine Image (AMI) 專用的命令：

- Amazon Linux 2

  ```
  sudo iptables-save | sudo tee /etc/sysconfig/iptables && sudo systemctl enable --now
   iptables
  ```

- Amazon Linux

  ```
  sudo service iptables save
  ```

對於使用 `awsvpc` 網路模式的任務，請在 `/etc/ecs/ecs.config` 檔案中將環境變數 `ECS_AWSVPC_BLOCK_IMDS` 設定為 `true`。

您應該在 `ecs-agent config` 檔案中將 `ECS_ENABLE_TASK_IAM_ROLE_NETWORK_HOST` 變數設定為 `false`，以防止在 `host` 網路中執行的容器存取 Amazon EC2 中繼資料。

使用網**awsvpc**路模式

使用網路 awsvpc 網路模式來限制不同任務之間的流量，或是您的任務與 Amazon VPC 中執行的其他服務之間的流量。此新增其他的安全層。awsvpc 網路模式可為 Amazon EC2 上執行的任務提供任務層級的網路隔離。這是預設模式 AWS Fargate。這是唯一可以用來指派安全性群組給工作的網路模式。

使用 IAM Access Advisor 精簡角色

建議您移除任何從未使用或已有一段時間未使用的動作。這樣可以防止不必要的存取發生。若要這麼做，請檢閱 IAM Access Advisor 產生的結果，然後移除從未使用過或最近未使用過的動作。您可以按照下列步驟執行此操作。

執行下列命令來產生報告，其中顯示參考政策的上次存取資訊：

```
aws iam generate-service-last-accessed-details --arn arn:aws:iam::123456789012:policy/
ExamplePolicy1
```

使用輸出中的 JobId 來執行以下命令。執行這項操作之後，您可以檢視報告的結果。

```
aws iam get-service-last-accessed-details --job-id 98a765b4-3cde-2101-2345-example678f9
```

如需詳細資訊，請參閱 IAM Access Advisor。

監控 AWS CloudTrail 可疑活動

您可以監視 AWS CloudTrail 任何可疑活動。大多數 AWS API 呼叫會記錄 AWS CloudTrail 為事件。它們會由 AWS CloudTrail Insights 進行分析，而且您會收到與 write API 呼叫相關聯的任何可疑行為的警示。這可能包括呼叫量的激增。這些警示包括異常活動發生的時間和為 API 貢獻的頂端身份 ARN 等資訊。

您可以查看事件的 userIdentity 屬性，來識別 AWS CloudTrail 中具有 IAM 角色的任務所執行的動作。在下列範例中，arn 包含擔任角色的名稱 s3-write-go-bucket-role,後面接著任務名稱 7e9894e088ad416eb5cab92afExample。

```
"userIdentity": {
    "type": "AssumedRole",
    "principalId": "AROA36C6WWEJ2YEXAMPLE:7e9894e088ad416eb5cab92afExample",
    "arn": "arn:aws:sts::123456789012:assumed-role/s3-write-go-bucket-
role/7e9894e088ad416eb5cab92afExample",
    ...
```

```
}
```

> **Note**
>
> 當擔任角色的任務在 Amazon EC2 容器執行個體上執行時，Amazon ECS 容器代理程式會將請求記錄到位於 `/var/log/ecs/audit.log.YYYY-MM-DD-HH` 格式地址的代理程式稽核日誌中。如需詳細資訊，請參閱任務 IAM 角色日誌和記錄追蹤的 Insights 事件。

Amazon ECS 任務執行 IAM 角色

任務執行角色會授予 Amazon ECS 容器和 Fargate 代理程式許可，以代表您進行 AWS API 呼叫。視任務需求而定，任務執行 IAM 角色是必要的項目。您可擁有多個任務執行角色，以用於與帳戶相關聯的不同用途和服務。有關應用程式需要執行的 IAM 許可，請參閱 Amazon ECS 任務 IAM 角色。

以下是任務執行 IAM 角色的常用案例：

- 您的任務託管在外部實例上 AWS Fargate或在外部實例上，並且：
  - 從 Amazon ECR 私有存儲庫中提取容器映像。
  - 從執行任務的帳戶不同的帳戶中的 Amazon ECR 私有儲存庫提取容器映像。
  - 使用記錄驅動程式將容器 CloudWatch 記錄檔傳送至記`awslogs`錄檔。如需詳細資訊，請參閱 將 Amazon ECS 日誌傳送到 CloudWatch 。
- 您的任務託管在任一個 AWS Fargate 或 Amazon EC2 執行個體上，並且：
  - 使用私人登錄驗證。如需詳細資訊，請參閱 私人登錄驗證權限。
  - 使用執行階段監視。
  - 工作定義會使用 Secret Manager 密碼或 AWS Systems Manager 管理員參數存放區參數來參照敏感資料。如需詳細資訊，請參閱 Secrets Manager 或 Systems Manager 權限。

> **Note**
>
> Amazon ECS 容器代理程式 1.16.0 版和更新版本支援任務執行角色。

Amazon ECS 提供名為 AmazonECS 的受管政策，TaskExecutionRolePolicy 其中包含上述常見使用案例所需的許可。如需詳細資訊，請參閱《AWS 受管政策參考指南》TaskExecutionRolePolicy中的 AmazonECs。針對特殊使用案例，可能需要將內嵌政策新增至您的工作執行角色

Amazon ECS 主控台會建立任務執行角色。您可以手動附加任務的受管 IAM 政策，以允許 Amazon ECS 為 future 的功能和增強功能新增許可。您可以使用 IAM 主控台搜尋來搜尋`ecsTaskExecutionRole`並查看您的帳戶是否已具有工作執行角色。如需詳細資訊，請參閱 IAM 使用者指南中的 IAM 主控台搜尋。

如果您以經過身份驗證的用戶身份提取圖像，則不太可能受到 Docker Hub 提取速率限制發生的更改的影響。如需詳細資訊，請參閱容器執行個體的私有登錄檔身分驗證。

透過使用 Amazon ECR 和 Amazon ECR Public，您可以避免 Docker 施加的限制。如果您從 Amazon ECR 提取映像，這也有助於縮短網路提取時間，並減少流量離開 VPC 時的資料傳輸變更。

當您使用 Fargate 時，您必須使用 `repositoryCredentials` 對私有映像登錄檔進行驗證。無法為 Fargate 上託管的任務設定 Amazon ECS 容器代理程式環境變數 `ECS_ENGINE_AUTH_TYPE` 或 `ECS_ENGINE_AUTH_DATA` 或修改 `ecs.config` 檔案。如需詳細資訊，請參閱任務的私有登錄檔身分驗證。

建立任務執行 角色

如果您的帳戶還沒有工作執行角色，請使用下列步驟建立角色。

AWS Management Console

建立彈性容器服務 (IAM 主控台) 的服務角色

1. 登入 AWS Management Console 並開啟身分與存取權管理主控台，網址為 https://console.aws.amazon.com/iam/。
2. 在 IAM 主控台的導覽窗格中，選擇角色，然後選擇建立角色。
3. 對於 Trusted entity type (信任的實體類型)，請選擇 AWS 服務。
4. 對於服務或使用案例，請選擇彈性容器服務，然後選擇彈性容器服務工作使用案例。
5. 選擇下一步。
6. 在「新增權限」區段中，搜尋 AmazonECs TaskExecution RolePolicy，然後選取原則。
7. 選擇下一步。
8. 在「角色名稱」中，輸入 ecs TaskExecution 角色。
9. 檢閱角色，然後選擇 Create role (建立角色)。

AWS CLI

以您自己的資訊取代所有`#####`。

1. 建立名為 ecs-tasks-trust-policy.json 的檔案，其中包含用於 IAM 角色的信任政策。檔案應包含以下內容：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "Service": "ecs-tasks.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

2. 使用在上一個步驟中建立的信任政策，建立名為 ecsTaskExecutionRole 的 IAM 角色。

```
aws iam create-role \
      --role-name ecsTaskExecutionRole \
      --assume-role-policy-document file://ecs-tasks-trust-policy.json
```

3. 將 AWS 受管理的AmazonECSTaskExecutionRolePolicy原則附加至ecsTaskExecutionRole角色。

```
aws iam attach-role-policy \
      --role-name ecsTaskExecutionRole \
      --policy-arn arn:aws:iam::aws:policy/service-role/
AmazonECSTaskExecutionRolePolicy
```

建立角色後，請為下列功能的角色新增其他權限。

功能	額外許可
使用 Secrets Manager 認證存取您的容器映像檔私人儲存庫	私人登錄驗證權限
使用系統管理器或 Secrets Manager 管理器傳遞敏感數據	Secrets Manager 或 Systems Manager 權限

功能	額外許可
讓 Fargate 任務透過界面端點提取 Amazon ECR 映像	透過界面端點許可提取 Amazon ECR 映像的 Fargate 端任務
在 Amazon S3 存儲桶中託管配置文件	Amazon S3 檔案儲存許可

私人登錄驗證權限

若要將存取提供給您建立的秘密，請將以下許可字作為內嵌政策，新增到任務執行角色。如需詳細資訊，請參閱新增和移除 IAM 政策。

- `secretsmanager:GetSecretValue`
- `kms:Decrypt` - 只有在您的金鑰使用自訂 KMS 金鑰而非預設金鑰時，才需要此項目。您的自訂金鑰的 Amazon Resource Name (ARN) 必須新增為資源。

下列為新增許可的內嵌政策範例。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "kms:Decrypt",
                "secretsmanager:GetSecretValue"
            ],
            "Resource": [
                "arn:aws:secretsmanager:<region>:<aws_account_id>:secret:secret_name",
                "arn:aws:kms:<region>:<aws_account_id>:key/key_id"
            ]
        }
    ]
}
```

Secrets Manager 或 Systems Manager 權限

允許容器代理程式提取必要 AWS Systems Manager 或 Secrets Manager 資源的權限。如需詳細資訊，請參閱 將敏感資料傳遞至 Amazon ECS 容器。

使用 Secrets Manager

若要允許存取您建立的 Secrets Manager 秘密，請將以下許可新增到任務執行角色。如需有關如何管理許可的相關資訊，請參閱《IAM 使用者指南》中的新增和移除 IAM 身分許可。

- `secretsmanager:GetSecretValue` – 如果參考 Secrets Manager 秘密，則此項目為必要。新增從 Secrets Manager 擷取密碼的許可。

下列政策範例新增必要許可。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "secretsmanager:GetSecretValue"
      ],
      "Resource": [
        "arn:aws:secretsmanager:region:aws_account_id:secret:secret_name"
      ]
    }
  ]
}
```

使用 Systems Manager

> **Important**
>
> 對於使用 EC2 啟動類型的任務，您必須使用 ECS 代理程式組態變數 `ECS_ENABLE_AWSLOGS_EXECUTIONROLE_OVERRIDE=true`，才能使用此功能。您可以在建立容器執行個體期間將其新增至 `./etc/ecs/ecs.config` 檔案，也可以將其新增至現有的執行個體，然後重新啟動 ECS 代理程式。如需詳細資訊，請參閱 Amazon ECS 容器代理程式組態。

若要允許存取您建立的 Systems Manager Parameter Store 參數，請將以下許可當作政策，手動新增到任務執行角色。如需有關如何管理許可的相關資訊，請參閱《IAM 使用者指南》中的新增和移除 IAM 身分許可。

- `ssm:GetParameters` – 如果參考任務定義中的 Systems Manager Parameter Store 參數，則此項目為必要。新增擷取 Systems Manager 參數的許可。
- `secretsmanager:GetSecretValue` – 如果直接參考 Secrets Manager 秘密，或者 Systems Manager Parameter Store 參數參考任務定義中的 Secrets Manager 秘密，則此項目為必要。新增從 Secrets Manager 擷取密碼的許可。
- `kms:Decrypt` – 只有在秘密使用客戶受管金鑰而非預設金鑰時，此項目為必要。您的自訂金鑰的 ARN 應該新增為資源。新增解密客戶受管金鑰的許可。

下列政策範例新增必要許可：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ssm:GetParameters",
        "secretsmanager:GetSecretValue",
        "kms:Decrypt"
      ],
      "Resource": [
        "arn:aws:ssm:region:aws_account_id:parameter/parameter_name",
        "arn:aws:secretsmanager:region:aws_account_id:secret:secret_name",
        "arn:aws:kms:region:aws_account_id:key/key_id"
      ]
    }
  ]
}
```

透過界面端點許可提取 Amazon ECR 映像的 Fargate 端任務

當 Amazon ECR 設定為使用介面 VPC 端點時，只要您啟動使用 Fargate 啟動類型的任務時 (此類型可從 Amazon ECR 中提取映像)，您就可以限制任務存取特定的 VPC 或 VPC 端點。建立任務執行角色，讓任務使用利用 IAM 條件金鑰的角色，即可完成此操作。

使用以下 IAM 全域條件金鑰，限制存取特定 VPC 或 VPC 端點。如需詳細資訊，請參閱 AWS 全域條件內容金鑰。

- `aws:SourceVpc` - 限制存取特定 VPC。
- `aws:SourceVpce` - 限制存取特定 VPC 端點。

以下任務執行角色政策提供新增條件金鑰的範例：

> **Important**
>
> `ecr:GetAuthorizationToken`API 動作無法套用`aws:sourceVpc`或`aws:sourceVpce`條件金鑰，因為 GetAuthorizationToken API 呼叫會透過 AWS Fargate 擁有的 elastic network interface，而不是工作的 elastic network interface。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ecr:GetAuthorizationToken",
                "logs:CreateLogStream",
                "logs:PutLogEvents"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ecr:BatchCheckLayerAvailability",
                "ecr:GetDownloadUrlForLayer",
                "ecr:BatchGetImage"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:sourceVpce": "vpce-xxxxxx",
                    "aws:sourceVpc": "vpc-xxxxx"
                }
            }
        }
    ]
}
```

Amazon S3 檔案儲存許可

當您指定在 Amazon S3 中託管的組態檔時，您的任務執行角色必須包含組態檔的`s3:GetObject`許`s3:GetBucketLocation`可，以及該檔案所在 Amazon S3 儲存貯體上的許可。如需詳細資訊，請參閱《Amazon Simple Storage Service 使用者指南》中的在政策中指定許可。

下列範例政策新增從 Amazon S3 擷取檔案所需的許可。指定 Amazon S3 儲存貯體的名稱和組態檔案名稱。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject"
      ],
      "Resource": [
        "arn:aws:s3:::examplebucket/folder_name/config_file_name"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetBucketLocation"
      ],
      "Resource": [
        "arn:aws:s3:::examplebucket"
      ]
    }
  ]
}
```

Amazon ECS 任務 IAM 角色

您的 Amazon ECS 任務可以擁有與其關聯的 IAM 角色。在 IAM 角色中授予的許可是由任務中執行的容器承繼。此角色可讓您的應用程式程式碼 (在容器上) 使用其他 AWS 服務。當您的應用程式存取其他 AWS 服務 (例如 Amazon S3) 時，需要任務角色。如需 Amazon ECS 提取容器映像和執行任務所需的 IAM 許可，請參閱 Amazon ECS 任務執行 IAM 角色。

以下是使用任務角色的好處：

- 登入資料隔離：容器只擷取所屬任務定義中定義的 IAM 角色登入資料，容器從未能存取屬於其他任務之其他容器的登入資料。
- 授權：未經授權的容器無法存取為其他任務定義的 IAM 角色登入資料。
- 稽核：存取和事件記錄可透過 CloudTrail 以確保回溯性稽核。工作認證的`taskArn`內容會附加至工作階段，因此 CloudTrail 記錄會顯示哪個工作正在使用哪個角色。

> **Note**
>
> 當您為任務指定 IAM 角色時，該任務容器中的 AWS CLI 或其他 SDK 會專門使用任務角色提供的 AWS 登入資料，而且它們不再繼承其執行所在 Amazon EC2 或外部執行個體的任何 IAM 許可。

建立任務 IAM 角色

建立要使用的任務的 IAM 政策時，該政策必須包含您希望任務中容器承擔的許可。您可以使用現有的 AWS 受管原則，也可以從頭開始建立符合您特定需求的自訂原則。如需詳細資訊，請參閱《IAM 使用者指南》中的建立 IAM 政策。

> **Important**
>
> 針對 Amazon ECS 任務 (適用於所有啟動類型)，建議在任務中使用 IAM 政策和角色。這些認證允許您的任務發出 AWS API 請求，而無`sts:AssumeRole`需調用承擔與任務相關聯的相同角色。如果任務需要自己擔任的角色，則必須建立信任政策，明確允許該角色擔任它自己。如需詳細資訊，請參閱《IAM 使用者指南》中的修改角色信任政策。

在建立 IAM 政策後，您可以建立 IAM 角色，其中包含在 Amazon ECS 任務定義中參考的政策。您可以在 IAM 主控台中使用 Elastic Container Service Task (Elastic Container Service 任務) 使用案例來建立角色。然後，您可以將特定的 IAM 政策附加到角色，該角色為任務中的容器提供您所需的許可。以下程序說明如何執行此操作。

如果您有多個任務定義或服務需要 IAM 許可，您應該考慮為每個特定任務定義或服務建立具有操作任務所需最低許可的角色，以便將您為每項任務提供的存取降至最低。

如需有關您區域的服務端點的資訊，請參閱Amazon Web Services 一般參考 指南中的服務端點。

IAM 任務角色必須具有信任政策，該政策會指定 ecs-tasks.amazonaws.com 服務。sts:AssumeRole 許可允許您的任務承繼與 Amazon EC2 執行個體使用角色不同的 IAM 角色。如此一來，您的任務就不會沿用與 Amazon EC2 執行個體相關聯的角色。信任政策範例如下。取代區域識別碼，並指定啟動工作時所使用的 AWS 帳號。

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Principal":{
            "Service":[
               "ecs-tasks.amazonaws.com"
            ]
         },
         "Action":"sts:AssumeRole",
         "Condition":{
            "ArnLike":{
            "aws:SourceArn":"arn:aws:ecs:us-west-2:111122223333:*"
            },
            "StringEquals":{
               "aws:SourceAccount":"111122223333"
            }
         }
      }
   ]
}
```

Important

建立任務 IAM 角色時，建議您在信任關係aws:SourceAccount或與角色相關聯的 IAM 政策中使用或aws:SourceArn條件金鑰，進一步限制許可範圍，以避免混淆的副安全性問題。使用 aws:SourceArn 條件索引鍵來指定目前不受支援的特定叢集時，您應使用萬用字元來指定所有叢集。若要進一步了解混淆的副問題以及如何保護您的 AWS 帳戶，請參閱 IAM 使用者指南中的混淆副問題。

下列程序說明如何使用範例政策建立政策以從 Amazon S3 擷取物件。以您自己的值取代所有#####。

AWS Management Console

若要使用 JSON 政策編輯器來建立政策

1. 登入 AWS Management Console 並開啟身分與存取權管理主控台，網址為 https://console.aws.amazon.com/iam/。
2. 在左側的導覽窗格中，選擇 Policies (政策)。

   如果這是您第一次選擇 Policies (政策)，將會顯示 Welcome to Managed Policies (歡迎使用受管政策) 頁面。選擇 Get Started (開始使用)。
3. 在頁面頂端，選擇 Create policy (建立政策)。
4. 在政策編輯器中，選擇 JSON 選項。
5. 輸入下列 JSON 政策文件：

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "s3:GetObject"
         ],
         "Resource":[
            "arn:aws:s3:::my-task-secrets-bucket/*"
         ],
         "Condition":{
            "ArnLike":{
            "aws:SourceArn":"arn:aws:ecs:region:123456789012:*"
            },
            "StringEquals":{
               "aws:SourceAccount":"123456789012"
            }
         }
      }
   ]
}
```

6. 選擇下一步。

> **Note**
>
> 您可以隨時切換視覺化與 JSON 編輯器選項。不過，如果您進行變更或在視覺化編輯器中選擇下一步，IAM 就可能會調整您的政策結構，以便針對視覺化編輯器進行最佳化。如需詳細資訊，請參閱 IAM 使用者指南中的調整政策結構。

7. 在檢視與建立頁面上，為您正在建立的政策輸入政策名稱與描述 (選用)。檢視此政策中定義的許可，來查看您的政策所授予的許可。
8. 選擇 Create policy (建立政策) 儲存您的新政策。

AWS CLI

以您自己的值取代所有*#####*。

1. 建立稱為 `s3-policy.json` 的檔案，其中具有以下內容。

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "s3:GetObject"
         ],
         "Resource":[
            "arn:aws:s3:::my-task-secrets-bucket/*"
         ],
         "Condition":{
            "ArnLike":{
            "aws:SourceArn":"arn:aws:ecs:region:123456789012:*"
            },
            "StringEquals":{
               "aws:SourceAccount":"123456789012"
            }
         }
      }
   ]
}
```

2. 使用下列命令，使用 JSON 政策文件檔案建立 IAM 政策。

```
aws iam create-policy \
      --policy-name taskRolePolicy \
      --policy-document file://s3-policy.json
```

下列程序說明如何透過附加您建立的 IAM 政策來建立任務 IAM 角色。

AWS Management Console

建立彈性容器服務 (IAM 主控台) 的服務角色

1. 登入 AWS Management Console 並開啟身分與存取權管理主控台，網址為 https://console.aws.amazon.com/iam/。
2. 在 IAM 主控台的導覽窗格中，選擇角色，然後選擇建立角色。
3. 對於 Trusted entity type (信任的實體類型)，請選擇 AWS 服務。
4. 對於服務或使用案例，請選擇彈性容器服務，然後選擇彈性容器服務工作使用案例。
5. 選擇下一步。
6. 在 [新增權限] 中，搜尋並選擇您建立的原則。
7. 選擇下一步。
8. 針對 Role name (角色名稱)，輸入您的角色名稱。在此範例中，輸入 `AmazonECSTaskS3BucketRole` 來命名角色。
9. 檢閱角色，然後選擇 Create role (建立角色)。

AWS CLI

以您自己的值取代所有*#####*。

1. 建立名為的檔案`ecs-tasks-trust-policy.json`，其中包含要用於任務 IAM 角色的信任政策。該文件應包含以下內容。取代區域識別碼，並指定啟動工作時所使用的 AWS 帳號。

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Principal":{
            "Service":[
```

```
                "ecs-tasks.amazonaws.com"
            ]
         },
         "Action":"sts:AssumeRole",
         "Condition":{
            "ArnLike":{
            "aws:SourceArn":"arn:aws:ecs:us-west-2:111122223333:*"
            },
            "StringEquals":{
               "aws:SourceAccount":"111122223333"
            }
         }
      }
   ]
}
```

2. 使用在上一個步驟中建立的信任政策，建立名為 ecsTaskRole 的 IAM 角色。

```
aws iam create-role \
      --role-name ecsTaskRole \
      --assume-role-policy-document file://ecs-tasks-trust-policy.json
```

3. 使用下列命令擷取您建立的 IAM 政策的 ARN。以您建立RolePolicy的策略名稱取代#作。

```
aws iam list-policies --scope Local --query 'Policies[?
PolicyName==`taskRolePolicy`].Arn'
```

4. 將您建立的 IAM 政策附加到該ecsTaskRole角色。以您所policy-arn建立之原則的 ARN 取代。

```
aws iam attach-role-policy \
      --role-name ecsTaskRole \
      --policy-arn arn:aws:iam:111122223333:aws:policy/taskRolePolicy
```

建立角色後，請為下列功能的角色新增其他權限。

功能	額外許可
使用 ECS 執行	ECS 執行權限
使用 EC2 執行個體 (視窗和 Linux)	Amazon EC2 執行個體的其他

功能	額外許可
使用外部實例	外部執行個體的其他
使用執行個體	Amazon EC2 窗口實例附加配置

ECS 執行權限

ECS Exec 功能需要任務 IAM 角色，才能授與容器在受管 SSM 代理程式 (execute-command代理程式) 和 SSM 服務之間通訊所需的許可。您應將下列許可新增至任務 IAM 角色，並在任務定義中包含任務 IAM 角色。如需詳細資訊，請參閱 IAM 使用者指南中的新增和移除 IAM 政策。

針對您的任務 IAM 角色使用下列政策來新增所需的 SSM 許可。

```
{
   "Version": "2012-10-17",
   "Statement": [
       {
       "Effect": "Allow",
       "Action": [
            "ssmmessages:CreateControlChannel",
            "ssmmessages:CreateDataChannel",
            "ssmmessages:OpenControlChannel",
            "ssmmessages:OpenDataChannel"
       ],
      "Resource": "*"
      }
   ]
}
```

Amazon EC2 執行個體的其他

我們建議您將容器執行個體角色中的許可限制為 AmazonEC2ContainerServiceforEC2Role 受管 IAM 政策所使用的最小許可清單。

您的 Amazon EC2 執行個體需要至少版本1.11.0的容器代理程式才能使用任務角色；不過，我們建議您使用最新的容器代理程式版本。如需檢查代理程式版本及更新至最新版本的資訊，請參閱「更新 Amazon ECS 容器代理程式」。如果您使用 Amazon ECS 優化 AMI，則您1.11.0-1的執行個體至少需要套件ecs-init。如果執行個體使用最新的 Amazon ECS 最佳化 AMI，則它們會包含所需的容器代理程式和 ecs-init。如需詳細資訊，請參閱 Amazon ECS 優化的 Linux AMI。

如果您沒有為容器執行個體使用 Amazon ECS 優化 AMI，請將該`--net=host`選項新增至啟動代理程式的docker run命令，並為所需組態新增下列代理程式組態變數 (如需詳細資訊，請參閱Amazon ECS 容器代理程式組態)：

`ECS_ENABLE_TASK_IAM_ROLE=true`

為具有 `bridge` 和 `default` 網路模式之容器的任務使用 IAM 角色。

`ECS_ENABLE_TASK_IAM_ROLE_NETWORK_HOST=true`

為具有 `host` 網路模式之容器的任務使用 IAM 角色。僅代理 1.12.0 版和更新版本支援此變數。

如需執行命令範例，請參閱「手動更新 Amazon ECS 容器代理程式 (適用於非 Amazon ECS 最佳化 AMI)」。您還需要在容器執行個體上設定下列網路命令，以便工作中的容器可以擷取其 AWS 認證：

```
sudo sysctl -w net.ipv4.conf.all.route_localnet=1
sudo iptables -t nat -A PREROUTING -p tcp -d 169.254.170.2 --dport 80 -j DNAT --to-
destination 127.0.0.1:51679
sudo iptables -t nat -A OUTPUT -d 169.254.170.2 -p tcp -m tcp --dport 80 -j REDIRECT --
to-ports 51679
```

您必須將這些 iptables 規則儲存在您的容器執行個體上，才能避免在重新開機時將其刪除。您可以使用 iptables-save 和 iptables-restore 命令來儲存 iptables 規則以及在啟動時將其還原。如需詳細資訊，請參閱特定的作業系統文件。

針對任務中採用 `awsvpc` 網路模式的容器，如欲避免其存取提供給 Amazon EC2 執行個體設定檔的憑證資訊 (同時仍然允許任務角色所提供的許可)，請在代理程式組態檔案中將 `ECS_AWSVPC_BLOCK_IMDS` 代理程式組態變數設定為 `true`，並重新啟動代理程式。如需詳細資訊，請參閱 Amazon ECS 容器代理程式組態。

至於任務中採用 `bridge` 網路模式的容器，則可在 Amazon EC2 執行個體上執行下列 iptables 命令，藉此避免其存取提供給 Amazon EC2 執行個體設定檔的憑證資訊 (同時仍然允許任務角色所提供的許可)。此命令不會影響任務中採用 `host` 或 `awsvpc` 網路模式的容器。如需詳細資訊，請參閱 網路模式。

- ```
  sudo yum install -y iptables-services; sudo iptables --insert DOCKER-USER 1 --in-
  interface docker+ --destination 169.254.169.254/32 --jump DROP
  ```
  ```

您必須將此 iptables 規則儲存在您的 Amazon EC2 執行個體上，才能避免在重新開機時遭刪除。使用 Amazon ECS 最佳化 AMI 時，您可以使用下列命令。至於其他作業系統，請參閱該作業系統的文件。

```
sudo iptables-save | sudo tee /etc/sysconfig/iptables && sudo systemctl enable --now
 iptables
```

外部執行個體的其他

您的外部執行個體需要至少版本`1.11.0`的容器代理程式才能使用任務 IAM 角色；不過，我們建議您使用最新的容器代理程式版本。如需檢查代理程式版本及更新至最新版本的資訊，請參閱「更新 Amazon ECS 容器代理程式」。如果使用的是 Amazon ECS 最佳化 AMI，您的執行個體至少需要 `ecs-init` 套件的 `1.11.0-1` 版。如果執行個體使用最新的 Amazon ECS 最佳化 AMI，則它們會包含所需的容器代理程式和 `ecs-init`。如需詳細資訊，請參閱 Amazon ECS 優化的 Linux AMI。

如果您沒有為容器執行個體使用 Amazon ECS 優化 AMI，請將該`--net=host`選項新增至啟動代理程式的docker run命令，並為所需組態新增下列代理程式組態變數 (如需詳細資訊，請參閱Amazon ECS 容器代理程式組態)：

`ECS_ENABLE_TASK_IAM_ROLE=true`

為具有 `bridge` 和 `default` 網路模式之容器的任務使用 IAM 角色。

`ECS_ENABLE_TASK_IAM_ROLE_NETWORK_HOST=true`

為具有 `host` 網路模式之容器的任務使用 IAM 角色。僅代理 1.12.0 版和更新版本支援此變數。

如需執行命令範例，請參閱「手動更新 Amazon ECS 容器代理程式 (適用於非 Amazon ECS 最佳化 AMI)」。您還需要在容器執行個體上設定下列網路命令，以便工作中的容器可以擷取其 AWS 認證：

```
sudo sysctl -w net.ipv4.conf.all.route_localnet=1
sudo iptables -t nat -A PREROUTING -p tcp -d 169.254.170.2 --dport 80 -j DNAT --to-
destination 127.0.0.1:51679
sudo iptables -t nat -A OUTPUT -d 169.254.170.2 -p tcp -m tcp --dport 80 -j REDIRECT --
to-ports 51679
```

您必須將這些 iptables 規則儲存在您的容器執行個體上，才能避免在重新開機時將其刪除。您可以使用 iptables-save 和 iptables-restore 命令來儲存 iptables 規則以及在啟動時將其還原。如需詳細資訊，請參閱特定的作業系統文件。

Amazon EC2 窗口實例附加配置

> **Important**
>
> 這僅適用於 EC2 上使用任務角色的 Windows 容器。

具有 Windows 功能的任務角色需要 EC2 上的其他組態。

- 當您啟動容器執行個體時，您必須在容器執行個體使用者資料指令碼中設定 - `EnableTaskIAMRole` 選項。`EnableTaskIAMRole` 會開啟任務的任務 IAM 角色功能。例如：

```
<powershell>
Import-Module ECSTools
Initialize-ECSAgent -Cluster 'windows' -EnableTaskIAMRole
</powershell>
```

- 您必須使用「Amazon ECS 容器引導腳本」中提供的聯網命令引導您的容器。
- 您必須為您的任務建立 IAM 角色和政策。如需詳細資訊，請參閱 建立任務 IAM 角色。
- 任務登入資料供應者的 IAM 角色在容器執行個體上使用連接埠 80。因此，如果您在容器執行個體上設定任務 IAM 角色，您的容器無法在任何連接埠映射中為主機連接埠使用連接埠 80。若要在連接埠 80 公開您的容器，我們建議您為其設定使用負載平衡的服務。您可以在負載平衡器上使用連接埠 80。這樣做可以將流量路由到容器執行個體上的另一個主機連接埠。如需詳細資訊，請參閱 使用負載平衡來分配 Amazon ECS 服務流量。
- 如果 Windows 執行個體已重新啟動，您必須刪除代理界面並再次初始化 Amazon ECS 容器代理程式，以重新啟動登入資料代理。

Amazon ECS 容器引導腳本

容器必須使用所需的聯網命令加以引導，才能夠存取容器執行個體的登入資料代理，取得登入資料。以下程式碼範例指令碼應該在您的容器啟動時於其上執行。

> **Note**
>
> 當您在 Windows 上使用 `awsvpc` 網路模式時，您不需要執行此指令碼。

如果您執行包含 Powershell 的 Windows 容器，請使用下列指令碼：

```
# Copyright Amazon.com Inc. or its affiliates. All Rights Reserved.
#
# Licensed under the Apache License, Version 2.0 (the "License"). You may
# not use this file except in compliance with the License. A copy of the
# License is located at
#
# http://aws.amazon.com/apache2.0/
#
# or in the "license" file accompanying this file. This file is distributed
# on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either
# express or implied. See the License for the specific language governing
# permissions and limitations under the License.

$gateway = (Get-NetRoute | Where { $_.DestinationPrefix -eq '0.0.0.0/0' } | Sort-Object
 RouteMetric | Select NextHop).NextHop
$ifIndex = (Get-NetAdapter -InterfaceDescription "Hyper-V Virtual Ethernet*" | Sort-
Object | Select ifIndex).ifIndex
New-NetRoute -DestinationPrefix 169.254.170.2/32 -InterfaceIndex $ifIndex -NextHop
 $gateway -PolicyStore ActiveStore # credentials API
New-NetRoute -DestinationPrefix 169.254.169.254/32 -InterfaceIndex $ifIndex -NextHop
 $gateway -PolicyStore ActiveStore # metadata API
```

如果您執行只有命令 shell 的 Windows 容器，請使用下列指令碼：

```
# Copyright Amazon.com Inc. or its affiliates. All Rights Reserved.
#
# Licensed under the Apache License, Version 2.0 (the "License"). You may
# not use this file except in compliance with the License. A copy of the
# License is located at
#
# http://aws.amazon.com/apache2.0/
#
# or in the "license" file accompanying this file. This file is distributed
# on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either
# express or implied. See the License for the specific language governing
# permissions and limitations under the License.

for /f "tokens=1" %i  in ('netsh interface ipv4 show interfaces ^| findstr /x /r
 ".*vEthernet.*"') do set interface=%i
for /f "tokens=3" %i  in ('netsh interface ipv4 show addresses %interface% ^| findstr /
x /r ".*Default.Gateway.*"') do set gateway=%i
netsh interface ipv4 add route prefix=169.254.170.2/32 interface="%interface%"
 nexthop="%gateway%" store=active # credentials API
```

```
netsh interface ipv4 add route prefix=169.254.169.254/32 interface="%interface%"
 nexthop="%gateway%" store=active # metadata API
```

Amazon ECS 容器執行個體 IAM 角色

Amazon ECS 容器執行個體 (包括 Amazon EC2 和外部執行個體) 會執行 Amazon ECS 容器代理程式，並要求 IAM 角色，才能讓服務知道該代理程式為您所有。在您啟動容器執行個體並將它們註冊到叢集之前，您必須先為要使用的容器執行個體建立 IAM 角色。角色是在您用來登入主控台或執行 AWS CLI 命令的帳戶中建立的。

> **Important**
>
> 如果您要向叢集註冊外部執行個體，則您使用的 IAM 角色也需要 Systems Manager 許可。如需詳細資訊，請參閱 Amazon ECS Anywhere IAM 角色。

Amazon ECS 提供 AmazonEC2ContainerServiceforEC2Role 受管 IAM 政策，其中包含使用完整 Amazon ECS 功能集所需的許可。此受管政策可連接至 IAM 角色，並與您的容器執行個體相關聯。或者，您可以在建立要使用的自訂政策時，使用受管政策作為指南。容器執行個體角色提供 Amazon ECS 容器代理程式和 Docker 精靈所需的許可，以代表您呼叫 AWS API。如需受管政策的詳細資訊，請參閱 亚马逊角ContainerServicefor色。

Amazon ECS 支援使用受支援的 Amazon EC2 執行個體類型，以更高的 ENI 密度來啟動容器執行個體。使用此功能時，建議您建立 2 個容器執行個體角色。為一個角色啟用awsvpcTrunking帳戶設定，並將該角色用於需要 ENI 中繼的工作。如需有關awsvpcTrunking帳戶設定的資訊，請參閱透過帳戶設定存取 Amazon ECS 功能。

建立容器執行個體角色

> **Important**
>
> 如果您要向叢集註冊外部執行個體，請參閱 Amazon ECS Anywhere IAM 角色。

您可手動建立角色並連接容器執行個體的受管 IAM 政策，讓 Amazon ECS 為日後推出的功能和改進新增許可。如有需要，請使用下列程序附加受管 IAM 政策。

AWS Management Console

建立彈性容器服務 (IAM 主控台) 的服務角色

1. 登入 AWS Management Console 並開啟身分與存取權管理主控台，網址為 https://console.aws.amazon.com/iam/。
2. 在 IAM 主控台的導覽窗格中，選擇角色，然後選擇建立角色。
3. 對於 Trusted entity type (信任的實體類型)，請選擇 AWS 服務。
4. 對於服務或使用案例，請選擇彈性容器服務，然後選擇彈性容器服務的 EC2 角色使用案例。
5. 選擇下一步。
6. 在許可政策部分中，確認已選取亞馬遜 ContainerServicefor EC2Role 政策。

> **Important**
>
> AmazonEC2 ContainerServicefor EC2Role 受管政策應附加至容器執行個體 IAM 角色，否則您將收到使用建立叢集的錯誤訊息。 AWS Management Console

7. 選擇下一步。
8. 在「角色名稱」中，輸入 ecs InstanceRole
9. 檢閱角色，然後選擇 Create role (建立角色)。

AWS CLI

以您自己的值取代所有#####。

1. 建立稱為 `instance-role-trust-policy.json` 的檔案，其中具有以下內容。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": { "Service": "ec2.amazonaws.com"},
      "Action": "sts:AssumeRole"
    }
  ]
}
```

2. 使用下列命令，透過信任政策文件建立執行個體 IAM 角色。

```
aws iam create-role \
    --role-name ecsInstanceRole \
    --assume-role-policy-document file://instance-role-trust-policy.json
```

3. 使用 create-instance-profile 命令建立名為 ecsInstanceRole-profile 的執行個體設定檔。

```
aws iam create-instance-profile --instance-profile-name ecsInstanceRole-profile
```

回應範例

```
{
    "InstanceProfile": {
        "InstanceProfileId": "AIPAJTLBPJLEGREXAMPLE",
        "Roles": [],
        "CreateDate": "2022-04-12T23:53:34.093Z",
        "InstanceProfileName": "ecsInstanceRole-profile",
        "Path": "/",
        "Arn": "arn:aws:iam::123456789012:instance-profile/ecsInstanceRole-
profile"
    }
}
```

4. 將 *ecsInstanceRole* 角色新增至 *ecsInstanceRole-profile* 執行個體設定檔。

```
aws iam add-role-to-instance-profile \
    --instance-profile-name ecsInstanceRole-profile \
    --role-name ecsInstanceRole
```

5. 使用下列命令將AmazonEC2ContainerServiceRoleForEC2Role受管理的原則附加至角色。

```
aws iam attach-role-policy \
    --policy-arn arn:aws:iam::aws:policy/service-role/
AmazonEC2ContainerServiceforEC2Role \
    --role-name ecsInstanceRole
```

建立角色後，請為下列功能的角色新增其他權限。

功能	額外許可
Amazon ECR 具有容器映像	Amazon ECR 許可
讓 CloudWatch 日誌監視容器實例	監視容器執行個體權
在 Amazon S3 存儲桶中託管配置文件	Amazon S3 只讀訪問

Amazon ECR 許可

您與容器執行個體搭配使用的 Amazon ECS 容器執行個體角色必須具有下列 Amazon ECR 的 IAM 政策許可。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ecr:BatchCheckLayerAvailability",
                "ecr:BatchGetImage",
                "ecr:GetDownloadUrlForLayer",
                "ecr:GetAuthorizationToken"
            ],
            "Resource": "*"
        }
    ]
}
```

如果您的容器執行個體使用 AmazonEC2ContainerServiceforEC2Role 受管政策，則您的角色具有適當的許可。若要檢查您的角色是否支援 Amazon ECR，請參閱 Amazon Elastic Container Service 開發人員指南中的 Amazon ECS 容器執行個體 IAM 角色。

Amazon S3 只讀訪問

在 Amazon S3 中的私有儲存貯體內存放組態資訊，並對容器執行個體 IAM 角色授予唯讀存取權，這是一種允許容器執行個體在啟動時進行設定的安全且便利方法。您可以將ecs.config檔案副本存放在私有儲存貯體中，使用 Amazon EC2 使用者資料安裝， AWS CLI 然後將組態資訊複製到執行個體啟動/etc/ecs/ecs.config時。

如需建立 ecs.config 檔案、將其存放在 Amazon S3 以及使用此組態啟動執行個體的詳細資訊，請參閱 在 Amazon S3 中存儲亞馬遜 ECS 容器實例配置。

您可以使用下列 AWS CLI 命令允許 Amazon S3 對您的容器執行個體角色進行唯讀存取。以您建立 InstanceRole的角色名稱取代 *ecs*。

```
aws iam attach-role-policy \
      --role-name ecsInstanceRole \
      --policy-arn arn:aws::iam::aws:policy/AmazonS3ReadOnlyAccess
```

您也可以使用 IAM 主控台將 Amazon S3 唯讀存取權 (AmazonS3ReadOnlyAccess) 新增至您的角色。如需詳細資訊，請參閱《AWS Identity and Access Management 使用手冊》中的〈修改角色權限原則 (主控台)〉。

監視容器執行個體權

您必須先建立 IAM 政策，容器執行個體才能將 CloudWatch 日誌資料傳送至記錄檔，以允許容器執行個體使用 CloudWatch 記錄 API，然後您必須將該政策附加至ecsInstanceRole。

AWS Management Console

若要使用 JSON 政策編輯器來建立政策

1. 登入 AWS Management Console 並開啟身分與存取權管理主控台，網址為 https://console.aws.amazon.com/iam/。
2. 在左側的導覽窗格中，選擇 Policies (政策)。

   如果這是您第一次選擇 Policies (政策)，將會顯示 Welcome to Managed Policies (歡迎使用受管政策) 頁面。選擇 Get Started (開始使用)。
3. 在頁面頂端，選擇 Create policy (建立政策)。
4. 在政策編輯器中，選擇 JSON 選項。
5. 輸入下列 JSON 政策文件：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
```

```
                "logs:CreateLogGroup",
                "logs:CreateLogStream",
                "logs:PutLogEvents",
                "logs:DescribeLogStreams"
            ],
            "Resource": ["arn:aws:logs:*:*:*"]
        }
    ]
}
```

6. 選擇下一步。

> **Note**
>
> 您可以隨時切換視覺化與 JSON 編輯器選項。不過，如果您進行變更或在視覺化編輯器中選擇下一步，IAM 就可能會調整您的政策結構，以便針對視覺化編輯器進行最佳化。如需詳細資訊，請參閱 IAM 使用者指南中的調整政策結構。

7. 在檢視與建立頁面上，為您正在建立的政策輸入政策名稱與描述 (選用)。檢視此政策中定義的許可，來查看您的政策所授予的許可。
8. 選擇 Create policy (建立政策) 儲存您的新政策。

建立原則之後，請將原則附加至容器執行個體角色。如需如何將原則附加至角色的詳細資訊，請參閱《AWS Identity and Access Management 使用手冊》中的〈修改角色權限原則 (主控台)〉。

AWS CLI

1. 建立稱為 `instance-cw-logs.json` 的檔案，其中具有以下內容。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogGroup",
                "logs:CreateLogStream",
                "logs:PutLogEvents",
                "logs:DescribeLogStreams"
            ],
            "Resource": ["arn:aws:logs:*:*:*"]
        }
```

```
    ]
}
```

2. 使用下列命令，使用 JSON 政策文件檔案建立 IAM 政策。

```
aws iam create-policy \
      --policy-name cwlogspolicy \
      --policy-document file://instance-cw-logs.json
```

3. 使用下列命令擷取您建立的 IAM 政策的 ARN。將 *cwlogspolicy* 取代為您建立的策略名稱。

```
aws iam list-policies --scope Local --query 'Policies[?
PolicyName==`cwlogsppolicy`].Arn'
```

4. 使用下列命令，使用政策 ARN 將政策附加至容器執行個體 IAM 角色。

```
aws iam attach-role-policy \
      --role-name ecsInstanceRole \
      --policy-arn arn:aws:iam:111122223333:aws:policy/cwlogspolicy
```

Amazon ECS Anywhere IAM 角色

當您向叢集註冊內部部署伺服器或虛擬機器 (VM) 時，伺服器或虛擬機器需要 IAM 角色才能與 AWS API 通訊。您只需為每個 AWS 帳戶建立一次此 IAM 角色。不過，此 IAM 角色必須與您向叢集註冊的每個伺服器或虛擬機器產生關聯。此角色為 ECSAnywhereRole。您可以手動建立此角色。或者，當您在 AWS Management Console中註冊外部執行個體時，Amazon ECS 可以代表您建立角色。您可以使用 IAM 主控台搜尋來搜尋ecsAnywhereRole並查看您的帳戶是否已具備該角色。如需詳細資訊，請參閱 IAM 使用者指南中的 IAM 主控台搜尋。

AWS 提供兩項受管身分與存取權管理政策，可在建立 ECS 任何地方 IAM 角色AmazonSSMManagedInstanceCore和AmazonEC2ContainerServiceforEC2Role政策時使用。AmazonEC2ContainerServiceforEC2Role 政策包含的許可提供的存取權很可能比您需要的更多。因此，視您的特定使用案例而定，建議您建立自訂政策，僅在該政策中新增所需許可。如需詳細資訊，請參閱Amazon ECS 容器執行個體 IAM 角色。

任務執行 IAM 角色會授予 Amazon ECS 容器代理程式許可，以代表您進行 AWS API 呼叫。使用任務執行 IAM 角色時，必須在您的任務定義中指定它。如需詳細資訊，請參閱 Amazon ECS 任務執行 IAM 角色。

如果以下任何條件適用，則需要任務執行角色：

- 您正在使用日誌驅動程序將容器 CloudWatch 日誌發送到日誌。`awslogs`
- 您的任務定義會指定託管在 Amazon ECR 私有儲存庫中的容器映像。但是，如果與外部執行個體相關聯的`ECSAnywhereRole`角色也包含從 Amazon ECR 提取映像所需的許可，則您的任務執行角色不需要包含它們。

創建 Amazon ECS Anywhere 角色

以您自己的資訊取代所有`#####`。

1. 建立以下列信任原則命名`ssm-trust-policy.json`的本機檔案。

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Principal": {"Service": [
      "ssm.amazonaws.com"
    ]},
    "Action": "sts:AssumeRole"
  }
}
```

2. 使用下列 AWS CLI 命令建立角色並附加信任原則。

```
aws iam create-role --role-name ecsAnywhereRole --assume-role-policy-document
 file://ssm-trust-policy.json
```

3. 使用下列命令附加 AWS 受管理的策略。

```
aws iam attach-role-policy --role-name ecsAnywhereRole --policy-arn
 arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore
aws iam attach-role-policy --role-name ecsAnywhereRole --policy-arn
 arn:aws:iam::aws:policy/service-role/AmazonEC2ContainerServiceforEC2Role
```

您也可以使用 IAM 自訂信任政策工作流程來建立角色。如需詳細資訊，請參閱《IAM 使用者指南》中的使用自訂信任政策 (主控台) 建立角色。

Amazon ECS 基礎設施 IAM 角色

Amazon ECS 基礎設施 IAM 角色可讓 Amazon ECS 代表您管理叢集中的基礎設施資源，並在下列情況下使用：

- 您想要將 Amazon EBS 磁碟區連接到您的 Fargate 或 EC2 啟動類型的 Amazon ECS 任務。基礎設施角色可讓 Amazon ECS 為您的任務管理 Amazon EBS 磁碟區。
- 您想要使用傳輸層安全性 (TLS) 來加密 Amazon ECS 服務 Connect 服務之間的流量。

當 Amazon ECS 假定此角色代表您採取行動時，事件將會顯示在中 AWS CloudTrail。如果 Amazon ECS 使用該角色來管理連接到您的任務的 Amazon EBS 磁碟區，則 CloudTrail記錄`roleSessionNamefolefole`將會是。`ECSTaskVolumesForEBS`如果該角色用於加密 Amazon ECS 服務 Connect 服務之間的流量，則 CloudTrail 日誌`roleSessionName`將`ECSServiceConnectForTLS`為。您可以透過篩選 [使用者名稱]，使用此名稱來搜尋 CloudTrail 主控台中的事件。

Amazon ECS 提供受管政策，其中包含磁碟區附件和 TLS 所需的許可。如需詳細資訊，請參閱《管理策略參考指南》中的 AmazonECS InfrastructureRole PolicyFor 磁碟區和 AmazonECs InfrastructureRole PolicyFor ServiceConnect TransportLayer 安全性。AWS

建立 Amazon ECS 基礎設施角色

以您自己的資訊取代所有*#####*。

1. 建立名為 `ecs-infrastructure-trust-policy.json` 的檔案，其中包含用於 IAM 角色的信任政策。檔案應包含以下內容：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowAccessToECSForInfrastructureManagement",
      "Effect": "Allow",
      "Principal": {
        "Service": "ecs.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

2. 使用下列 AWS CLI 命令，建立使用您在上一個步驟中建立的信任原則命名ecsInfrastructureRole的角色。

```
aws iam create-role \
      --role-name ecsInfrastructureRole \
      --assume-role-policy-document file://ecs-infrastructure-trust-policy.json
```

3. 根據您的使用案例，將受 AWS 管理AmazonECSInfrastructureRolePolicyForVolumes或AmazonECSInfrastructureRolePoli略附加到ecsInfrastructureRole角色。

```
aws iam attach-role-policy \
      --role-name ecsInfrastructureRole \
      --policy-arn arn:aws:iam::aws:policy/service-role/
AmazonECSInfrastructureRolePolicyForVolumes
```

```
aws iam attach-role-policy \
      --role-name ecsInfrastructureRole \
      --policy-arn arn:aws:iam::aws:policy/service-role/
AmazonECSInfrastructureRolePolicyForServiceConnectTransportLayerSecurity
```

您也可以使用 IAM 主控台的自訂信任政策工作流程來建立角色。如需詳細資訊，請參閱《IAM 使用者指南》中的使用自訂信任政策 (主控台) 建立角色。

Important

如果 Amazon ECS 正在使用 ECS 基礎設施角色來管理連接到任務的 Amazon EBS 磁碟區，請在停止使用 Amazon EBS 磁碟區的任務之前，先確保以下事項。

- 不會刪除角色。
- 不會修改角色的信任政策以移除 Amazon ECS 存取權 (ecs.amazonaws.com)。
- AmazonECSInfrastructureRolePolicyForVolumes不會移除受管理的原則。如果您必須修改角色的權限，請至少保留 ec2:DetachVolumeec2:DeleteVolume，並ec2:DescribeVolumes用於刪除磁碟區。

在停止具有連接 Amazon EBS 磁碟區的任務之前刪除或修改角色，將導致任務卡住，DEPROVISIONING且相關聯的 Amazon EBS 磁碟區無法刪除。Amazon ECS 會定期自動

重試以停止工作並刪除磁碟區，直到恢復必要的許可為止。您可以使用 DescribeTasksAPI 來檢視工作的磁碟區附件狀態和相關的狀態原因。

建立檔案之後，您必須授與使用者權限，才能將角色傳遞給 Amazon ECS。

將基礎設施角色傳遞給 Amazon ECS 的權限

若要使用 ECS 基礎設施 IAM 角色，您必須授與使用者權限，才能將角色傳遞給 Amazon ECS。將下列`iam:PassRole`權限附加給您的使用者。以您建立InfrastructureRole的基礎結構角色名稱取代*`ecs`*。

```
{
    "Version": "2012-10-17",
    "Statement": [

        {
            "Action": "iam:PassRole",
            "Effect": "Allow",
            "Resource": ["arn:aws:iam::*:role/ecsInfrastructureRole"],
            "Condition": {
                "StringEquals": {"iam:PassedToService": "ecs.amazonaws.com"}
            }
        }
    ]
}
```

有關`iam:Passrole`和更新使用者許可的詳細資訊，請參閱《使用者指南》中的〈授予使用者將角色傳遞給 AWS 服務的權限〉和〈變更 IAM 使用AWS Identity and Access Management者的許可〉。

Amazon ECS CodeDeploy IAM 角色

在您可以搭配 Amazon ECS 使用 CodeDeploy 藍色/綠色部署類型之前，該 CodeDeploy 服務需要許可以代表您更新 Amazon ECS 服務。這些許可由 CodeDeploy IAM 角色 (`ecsCodeDeployRole`) 提供。

> **Note**
>
> 使用者也需要使用權限 CodeDeploy；這些權限在中有所說明所需的 IAM 許可。

提供兩個受管政策。如需詳細資訊，請參閱《AWS 受管理策略參考指南》中的下列其中一項：

- AWSCodeDeployRoleForECS- CodeDeploy 授予使用關聯動作更新任何資源的權限。
- AWSCodeDeployRoleForECSLimited-提供 CodeDeploy 更多有限的權限。

建立角 CodeDeploy 色

您可以使用下列程序為 Amazon ECS 建立 CodeDeploy 角色

AWS Management Console

若要為 CodeDeploy (IAM 主控台) 建立服務角色

1. 登入 AWS Management Console 並開啟身分與存取權管理主控台，網址為 https://console.aws.amazon.com/iam/。
2. 在 IAM 主控台的導覽窗格中，選擇角色，然後選擇建立角色。
3. 對於 Trusted entity type (信任的實體類型)，請選擇 AWS 服務。
4. 對於服務或使用案例，請選擇 CodeDeploy，然後選擇 CodeDeploy -ECS 使用案例。
5. 選擇下一步。
6. 在 [附加權限原則] 區段中，確定已選取AWSCodeDeployRoleForECS原則。
7. 選擇下一步。
8. 在「角色名稱」中，輸入 ecs CodeDeploy 角色。
9. 檢閱角色，然後選擇 Create role (建立角色)。

AWS CLI

以您自己的資訊取代所有#####。

1. 建立名為的檔案`codedeploy-trust-policy.json`，其中包含要用於 CodeDeploy IAM 角色的信任政策。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "",
            "Effect": "Allow",
            "Principal": {
```

```
                "Service": ["codedeploy.amazonaws.com"]
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

2. 使用在上一個步驟中建立的信任政策，建立名為 ecsCodedeployRole 的 IAM 角色。

```
aws iam create-role \
      --role-name ecsCodedeployRole \
      --assume-role-policy-document file://codedeploy-trust-policy.json
```

3. 將AWSCodeDeployRoleForECS或受AWSCodeDeployRoleForECSLimited管理的原則附加至ecsTaskRole角色。

```
aws iam attach-role-policy \
      --role-name ecsCodedeployRole \
      --policy-arn arn:aws::iam::aws:policy/AWSCodeDeployRoleForECS
```

```
aws iam attach-role-policy \
      --role-name ecsCodedeployRole \
      --policy-arn arn:aws::iam::aws:policy/AWSCodeDeployRoleForECSLimited
```

當服務中的任務需要任務 ececution 角色時，您必須將每個任務執行角色或任務角色覆寫的iam:PassRole權限新增至作為原則的 CodeDeploy 角色。

工作執行角色權限

當服務中的任務需要任務 ececution 角色時，您必須將每個任務執行角色或任務角色覆寫的iam:PassRole權限新增至作為原則的 CodeDeploy 角色。如需詳細資訊，請參閱 Amazon ECS 任務執行 IAM 角色 及 Amazon ECS 任務 IAM 角色。然後，您將該策略附加到 CodeDeploy 角色

建立 政策

AWS Management Console

若要使用 JSON 政策編輯器來建立政策

1. 登入 AWS Management Console 並開啟身分與存取權管理主控台，網址為 https://console.aws.amazon.com/iam/。

2. 在左側的導覽窗格中，選擇 Policies (政策)。

   如果這是您第一次選擇 Policies (政策)，將會顯示 Welcome to Managed Policies (歡迎使用受管政策) 頁面。選擇 Get Started (開始使用)。

3. 在頁面頂端，選擇 Create policy (建立政策)。
4. 在政策編輯器中，選擇 JSON 選項。
5. 輸入下列 JSON 政策文件：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": ["arn:aws:iam::<aws_account_id>:role/
<ecsCodeDeployRole>"]
        }
    ]
}
```

6. 選擇下一步。

> **Note**
>
> 您可以隨時切換視覺化與 JSON 編輯器選項。不過，如果您進行變更或在視覺化編輯器中選擇下一步，IAM 就可能會調整您的政策結構，以便針對視覺化編輯器進行最佳化。如需詳細資訊，請參閱 IAM 使用者指南中的調整政策結構。

7. 在檢視與建立頁面上，為您正在建立的政策輸入政策名稱與描述 (選用)。檢視此政策中定義的許可，來查看您的政策所授予的許可。
8. 選擇 Create policy (建立政策) 儲存您的新政策。

建立原則之後，請將原則附加至 CodeDeploy 角色。如需如何將原則附加至角色的詳細資訊，請參閱《AWS Identity and Access Management 使用手冊》中的〈修改角色權限原則 (主控台)〉。

AWS CLI

以您自己的資訊取代所有*#####*。

1. 建立稱為 `blue-green-iam-passrole.json` 的檔案，其中具有以下內容。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": ["arn:aws:iam::<aws_account_id>:role/
<ecsCodeDeployRole>"]
        }
    ]
}
```

2. 使用下列命令，使用 JSON 政策文件檔案建立 IAM 政策。

```
aws iam create-policy \
      --policy-name cdTaskExecutionPolicy \
      --policy-document file://blue-green-iam-passrole.json
```

3. 使用下列命令擷取您建立的 IAM 政策的 ARN。

```
aws iam list-policies --scope Local --query 'Policies[?
PolicyName==`cdTaskExecutionPolicy`].Arn'
```

4. 使用下列命令將政策附加至 CodeDeploy IAM 角色。

```
aws iam attach-role-policy \
      --role-name ecsCodedeployRole \
      --policy-arn arn:aws:iam:111122223333:aws:policy/cdTaskExecutionPolicy
```

Amazon ECS EventBridge IAM 角色

在您可以將 Amazon ECS 排程任務與 EventBridge 規則和目標搭配使用之前，該 EventBridge 服務需要允許以您的名義執行 Amazon ECS 任務。這些許可由 EventBridge IAM 角色 (`ecsEventsRole`) 提供。

`AmazonEC2ContainerServiceEventsRole` 政策顯示如下。

```
{
    "Version": "2012-10-17",
    "Statement": [
```

```
        {
            "Effect": "Allow",
            "Action": ["ecs:RunTask"],
            "Resource": ["*"]
        },
        {
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": ["*"],
            "Condition": {
                "StringLike": {"iam:PassedToService": "ecs-tasks.amazonaws.com"}
            }
        },
        {
            "Effect": "Allow",
            "Action": "ecs:TagResource",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "ecs:CreateAction": ["RunTask"]
                }
            }
        }
    ]
}
```

如果排定的工作需要使用任務執行角色、任務角色或任務角色覆寫，則您必須將每個任務執行角色、任務角色或任務角色覆寫的`iam:PassRole`權限新增至 EventBridge IAM 角色。如需任務執行角色的詳細資訊，請參閱「Amazon ECS 任務執行 IAM 角色」。

> **Note**
>
> 指定您任務執行角色或任務角色覆寫的完整 ARN。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "iam:PassRole",
```

```
            "Resource": ["arn:aws:iam::<aws_account_id>:role/
<ecsTaskExecutionRole_or_TaskRole_name>"]
        }
    ]
}
```

您可以選擇在設定排程的工作時，讓該 EventBridge 角色為您 AWS Management Console 建立角色。如需詳細資訊，請參閱 使用 Amazon EventBridge 排程器來安排 Amazon ECS 任務 。

建立角 EventBridge 色

以您自己的資訊取代所有*#####*。

1. 建立名為 `eventbridge-trust-policy.json` 的檔案，其中包含用於 IAM 角色的信任政策。檔案應包含以下內容：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "Service": "events.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

2. 使用下列命令建立 IAM 角色，並使`ecsEventsRole`用您在上一個步驟中建立的信任政策命名。

```
aws iam create-role \
      --role-name ecsEventsRole \
      --assume-role-policy-document file://eventbridge-policy.json
```

3. 使用以下命令`AmazonEC2ContainerServiceEventsRole`將 AWS 受管理的`ecsEventsRole`角色附加到角色。

```
aws iam attach-role-policy \
      --role-name ecsEventsRole \
```

```
    --policy-arn arn:aws:iam::aws:policy/service-role/
AmazonEC2ContainerServiceEventsRole
```

您也可以使用 IAM 主控台的自訂信任政策工作流程 (https://console.aws.amazon.com/iam/) 建立角色。如需詳細資訊，請參閱《IAM 使用者指南》中的使用自訂信任政策 (主控台) 建立角色。

將政策連接至 **ecsEventsRole** 角色

您可以使用下列程序將任務執行角色的許可新增至 EventBridge IAM 角色。

AWS Management Console

若要使用 JSON 政策編輯器來建立政策

1. 登入 AWS Management Console 並開啟身分與存取權管理主控台，網址為 https://console.aws.amazon.com/iam/。
2. 在左側的導覽窗格中，選擇 Policies (政策)。

   如果這是您第一次選擇 Policies (政策)，將會顯示 Welcome to Managed Policies (歡迎使用受管政策) 頁面。選擇 Get Started (開始使用)。
3. 在頁面頂端，選擇 Create policy (建立政策)。
4. 在政策編輯器中，選擇 JSON 選項。
5. 輸入下列 JSON 政策文件：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": ["arn:aws:iam::<aws_account_id>:role/
<ecsTaskExecutionRole_or_TaskRole_name>"]
        }
    ]
}
```

6. 選擇下一步。

> **Note**
>
> 您可以隨時切換視覺化與 JSON 編輯器選項。不過，如果您進行變更或在視覺化編輯器中選擇下一步，IAM 就可能會調整您的政策結構，以便針對視覺化編輯器進行最佳化。如需詳細資訊，請參閱 IAM 使用者指南中的調整政策結構。

7. 在檢視與建立頁面上，為您正在建立的政策輸入政策名稱與描述 (選用)。檢視此政策中定義的許可，來查看您的政策所授予的許可。
8. 選擇 Create policy (建立政策) 儲存您的新政策。

建立原則之後，請將原則附加至 EventBridge 角色。如需如何將原則附加至角色的詳細資訊，請參閱《AWS Identity and Access Management 使用手冊》中的〈修改角色權限原則 (主控台)〉。

AWS CLI

以您自己的資訊取代所有*#####*。

1. 建立稱為 `ev-iam-passrole.json` 的檔案，其中具有以下內容。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": ["arn:aws:iam::<aws_account_id>:role/
<ecsTaskExecutionRole_or_TaskRole_name>"]
        }
    ]
}
```

2. 使用下列 AWS CLI 命令，使用 JSON 政策文件檔案建立 IAM 政策。

```
aws iam create-policy \
      --policy-name eventsTaskExecutionPolicy \
      --policy-document file://ev-iam-passrole.json
```

3. 使用下列命令擷取您建立的 IAM 政策的 ARN。

```
aws iam list-policies --scope Local --query 'Policies[?
PolicyName==`eventsTaskExecutionPolicy`].Arn'
```

4. 使用下列命令，透過使用政策 ARN 將政策附加至 EventBridge IAM 角色。

```
aws iam attach-role-policy \
      --role-name ecsEventsRole \
      --policy-arn arn:aws:iam:111122223333:aws:policy/eventsTaskExecutionPolicy
```

Amazon ECS 主控台必要的許可

遵循授予最低權限的最佳實務，您可以使用 AmazonECS_FullAccess 受管政策作為範本，以建立您自己的自訂政策。如此一來，您就可以根據特定需求，從受管政策中取消或新增許可。如需詳細資訊，請參閱 許可詳細資訊。

Amazon ECS 主控台由 AWS CloudFormation 支援，在下列情況下需要額外的 IAM 許可：

- 建立叢集
- 建立服務
- 建立容量提供者

您可以為其他許可建立政策，然後將其連接至用於存取主控台的 IAM 角色。如需詳細資訊，請參閱《IAM 使用者指南》中的建立 IAM 政策。

建立叢集所需的權限

在主控台中建立叢集時，您需要其他權限來授與您管理 AWS CloudFormation 堆疊的權限。

以下是所需的其他許可：

- `cloudformation` - 允許委託人建立和管理 AWS CloudFormation 堆疊。在使用 AWS Management Console 建立 Amazon ECS 叢集以及後續管理這些叢集時，這為必要項。

下列政策包含必要的 AWS CloudFormation 許可，並將動作限制在 Amazon ECS 主控台中建立的資源。

```
{
```

```
  "Statement": [
      {
          "Effect": "Allow",
          "Action": [
                "cloudformation:CreateStack",
                "cloudformation:DeleteStack",
                "cloudformation:DescribeStack*",
                "cloudformation:UpdateStack"
             ],
            "Resource": [
                "arn:*:cloudformation:*:*:stack/Infra-ECS-Cluster-*"
            ]
      }
   ]
}
```

如果您尚未建立 Amazon ECS 容器執行個體角色 (ecsInstanceRole)，並且正建立使用 Amazon EC2 執行個體的叢集，則主控台將代表您建立角色。

此外，如果您使用 Auto Scaling 群組，則需要額外的權限，以便主控台可以在使用叢集 auto 調整規模功能時，將標籤新增至自動調整資源規模群組。

以下是所需的其他許可：

- autoscaling – 允許主控台標記 Amazon EC2 Auto Scaling 群組。使用叢集自動擴展功能管理 Amazon EC2 Auto Scaling 群組時，這為必要項。標籤是 ECS 受管標籤，主控台會自動新增至群組，表示已在主控台中建立。
- iam - 允許委託人列出 IAM 角色及其連接的政策。委託人還可列出 Amazon EC2 執行個體可用的執行個體設定檔。

以下政策包含必要的 IAM 許可，並將動作限制為 ecsInstanceRole 角色。

Auto Scaling 許可不受限制。

```
{
  "Statement": [
      {
          "Effect": "Allow",
            "Action": [
              "iam:AttachRolePolicy",
              "iam:CreateRole",
```

```
                "iam:CreateInstanceProfile",
                "iam:AddRoleToInstanceProfile",
                "iam:ListInstanceProfilesForRole",
                "iam:GetRole"
            ],
            "Resource": "arn:aws:iam::*:role/ecsInstanceRole"
        },
        {
            "Effect": "Allow",
            "Action": "autoscaling:CreateOrUpdateTags",
            "Resource": "*"
        }
    ]
}
```

建立容量提供者所需的權限

在主控台中建立服務時，您需要其他權限來授與您管理 AWS CloudFormation 堆疊的權限。以下是所需的其他許可：

- `cloudformation` - 允許委託人建立和管理 AWS CloudFormation 堆疊。在使用 AWS Management Console 建立 Amazon ECS 容量提供者和後續管理這些容量提供者時，這為必要項。

下列政策包含必要的許可，並將動作限制為在 Amazon ECS 主控台中建立的資源。

```
{
  "Statement": [
      {
          "Effect": "Allow",
          "Action": [
                "cloudformation:CreateStack",
                "cloudformation:DeleteStack",
                "cloudformation:DescribeStack*",
                "cloudformation:UpdateStack"
             ],
            "Resource": [
                "arn:*:cloudformation:*:*:stack/Infra-ECS-CapacityProvider-*"
            ]
      }
   ]
}
```

建立服務所需的權限

在主控台中建立服務時，您需要其他權限來授與您管理 AWS CloudFormation 堆疊的權限。以下是所需的其他許可：

- `cloudformation` - 允許委託人建立和管理 AWS CloudFormation 堆疊。在使用 AWS Management Console 建立 Amazon ECS 服務以及後續管理這些服務時，這為必要項。

下列政策包含必要的許可，並將動作限制為在 Amazon ECS 主控台中建立的資源。

```
{
  "Statement": [
      {
          "Effect": "Allow",
          "Action": [
                "cloudformation:CreateStack",
                "cloudformation:DeleteStack",
                "cloudformation:DescribeStack*",
                "cloudformation:UpdateStack"
             ],
            "Resource": [
                "arn:*:cloudformation:*:*:stack/ECS-Console-V2-Service-*"
            ]
      }
   ]
}
```

建立 IAM 角色的許可

下列動作需要其他許可才能完成操作：

- 註冊外部執行個體 - 如需詳細資訊，請參閱 Amazon ECS Anywhere IAM 角色
- 註冊任務定義 - 如需詳細資訊，請參閱 Amazon ECS 任務執行 IAM 角色
- 建立用於排程工作的 EventBridge 規則-如需詳細資訊，請參閱 Amazon ECS EventBridge IAM 角色

您可以先在 IAM 中建立角色來新增這些許可，然後在 Amazon ECS 主控台中使用這些許可。如果您不建立角色，Amazon ECS 主控台會代表您建立角色。

將外部執行個體註冊到叢集所需的權限

當您將外部執行個體註冊至叢集並想要建立新的外部執行個體 (escExternalInstanceRole) 角色時，您需要其他許可。

以下是所需的其他許可：

- iam - 允許委託人建立並列出 IAM 角色及其連接的政策。
- ssm – 允許主體使用 Systems Manager 註冊外部執行個體。

> **Note**
>
> 若要選擇現有 escExternalInstanceRole，您必須擁有 iam:GetRole 和 iam:PassRole 許可。

下列政策包含必要的許可，並將動作限制為 escExternalInstanceRole 角色。

```
{
  "Statement": [
      {
          "Effect": "Allow",
            "Action": [
              "iam:AttachRolePolicy",
              "iam:CreateRole",
              "iam:CreateInstanceProfile",
              "iam:AddRoleToInstanceProfile",
              "iam:ListInstanceProfilesForRole",
              "iam:GetRole"
            ],
            "Resource": "arn:aws:iam::*:role/escExternalInstanceRole"
        },
        {
            "Effect": "Allow",
            "Action": ["iam:PassRole","ssm:CreateActivation"],
            "Resource": "arn:aws:iam::*:role/escExternalInstanceRole"
        }
    ]
}
```

註冊任務定義所需的權限

當您註冊任務定義並想要建立新的任務執行 (ecsTaskExecutionRole) 角色時，您需要其他許可。

以下是所需的其他許可：

- iam - 允許委託人建立並列出 IAM 角色及其連接的政策。

> **Note**
>
> 若要選擇現有 ecsTaskExecutionRole，您必須擁有 iam:GetRole 許可。

下列政策包含必要的許可，並將動作限制為 ecsTaskExecutionRole 角色。

```
{
  "Statement": [
      {
          "Effect": "Allow",
            "Action": [
              "iam:AttachRolePolicy",
              "iam:CreateRole",
              "iam:GetRole"
            ],
            "Resource": "arn:aws:iam::*:role/ecsTaskExecutionRole"
        }
    ]
}
```

為排程工作建立 EventBridge 規則所需的權限

當您排程工作且想要建立新的 E CloudWatch vents role (ecsEventsRole) 角色時，您需要其他權限。

以下是所需的其他許可：

- iam - 允許委託人建立並列出 IAM 角色及其附加政策，並允許 Amazon ECS 將角色傳遞給其他服務以擔任該角色。

> **Note**
>
> 若要選擇現有 `ecsEventsRole`，您必須擁有 `iam:GetRole` 和 `iam:PassRole` 許可。

下列政策包含必要的許可，並將動作限制為 `ecsEventsRole` 角色。

```
{
  "Statement": [
      {
          "Effect": "Allow",
            "Action": [
              "iam:AttachRolePolicy",
              "iam:CreateRole",
              "iam:GetRole",
              "iam: PassRole"
            ],
            "Resource": "arn:aws:iam::*:role/ecsEventsRole"
        }
    ]
}
```

Amazon ECS 服務 auto 擴展所需的 IAM 許可

服務 Auto Scaling 可透過 Amazon ECS 和 Application Auto Scaling API 的組合實現。 CloudWatch服務是透過 Amazon ECS 建立和更新、使用建立警示 CloudWatch，並使用應用程式自動擴展建立擴展政策。

除了用於建立和更新服務的標準 IAM 許可外，還需要下列許可才能與 Service Auto Scaling 設定互動，如下列範例政策所示。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "application-autoscaling:*",
                "ecs:DescribeServices",
                "ecs:UpdateService",
                "cloudwatch:DescribeAlarms",
                "cloudwatch:PutMetricAlarm",
```

```
                "cloudwatch:DeleteAlarms",
                "cloudwatch:DescribeAlarmHistory",
                "cloudwatch:DescribeAlarmsForMetric",
                "cloudwatch:GetMetricStatistics",
                "cloudwatch:ListMetrics",
                "cloudwatch:DisableAlarmActions",
                "cloudwatch:EnableAlarmActions",
                "iam:CreateServiceLinkedRole",
                "sns:CreateTopic",
                "sns:Subscribe",
                "sns:Get*",
                "sns:List*"
            ],
            "Resource": ["*"]
        }
    ]
}
```

建立 Amazon ECS 服務範例 和 更新 Amazon ECS 服務示例 IAM 政策範例示範在 AWS Management Console中使用 Service Auto Scaling 所需的許可。

應用程式 Auto Scaling 服務還需要許可才能描述 Amazon ECS 服務和 CloudWatch 警示，以及代表您修改服務所需計數的許可。這些sns:許可適用於在超過閾值時 CloudWatch 傳送至 Amazon SNS 主題的通知。如果您為 Amazon ECS 服務使用自動擴展，則會建立其名為 AWSServiceRoleForApplicationAutoScaling_ECSService 的服務連結角色。此服務連結角色會授與 Application Auto Scaling 許可以描述政策的警示、監控服務目前執行的任務計數，以及修改服務需要的計數。Application Auto Scaling 的原始受管 Amazon ECS 角色為 ecsAutoscaleRole，但已不再需要。服務連結的角色預設為 Application Auto Scaling。如需詳細資訊，請參閱《Application Auto Scaling 使用者指南》中的適用於 Application Auto Scaling 的服務連結角色。

如果您在 Amazon ECS 可用 CloudWatch指標之前建立 Amazon ECS 容器執行個體角色，則可能需要新增許可。ecs:StartTelemetrySession如需詳細資訊，請參閱 考量事項。

在建立時授予標記資源的許可

以下的標籤建立 Amazon ECS API 動作允許您在建立資源時指定標籤。如果在資源建立動作中指定了標籤，請 AWS 執行其他授權以驗證是否已指派正確的權限來建立標籤。

- CreateCapacityProvider
- CreateCluster
- CreateService

- CreateTaskSet
- RegisterContainerInstance
- RegisterTaskDefinition
- RunTask
- StartTask

您可以使用資源標籤來實作以屬性為基礎的控制 (ABAC)。如需詳細資訊，請參閱 the section called "使用資源標籤控制對 Amazon ECS 資源的存取" 及 標記資源。

若要允許在建立時標記，請建立或修改政策，以同時包含使用建立資源之動作的許可，例如 ecs:CreateCluster 或 ecs:RunTask 和 ecs:TagResource 動作。

下列範例示範一項原則，可讓使用者在叢集建立期間建立叢集及新增標籤。使用者沒有標記現有資源的權限 (他們不能直接呼叫 ecs:TagResource 動作)。

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
         "ecs:CreateCluster"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
         "ecs:TagResource"
      ],
      "Resource": "*",
      "Condition": {
         "StringEquals": {
                  "ecs:CreateAction": [
                      "CreateCluster",
                      "CreateCapacityProvider",
                      "CreateService",
                      "CreateTaskSet",
                      "RegisterContainerInstance",
                      "RegisterTaskDefinition",
                      "RunTask",
```

```
                    "StartTask"
                ]
            }
        }
    }
  ]
}
```

只有在資源建立動作中套用了標籤時，才評估 ecs:TagResource 動作。因此，在沒有標記條件的情況下，若請求中未指定標籤，則具備資源建立許可的使用者不需要使用 ecs:TagResource 動作的許可。然而，若該使用者試圖建立具有標籤的資源卻未具備使用 ecs:TagResource 動作的許可，則該請求會失敗。

Amazon ECS 控制對特定標籤的訪問

您可以在 IAM 政策的 Condition 元素中使用其他條件，來控制可套用至資源的標籤金鑰索和值。

下列條件金鑰機可與前一節中的範例搭配使用：

- aws:RequestTag：表示請求中必須存在特定標籤金鑰或標籤金鑰與值。請求內亦可指定其他標籤。
  - 搭配 StringEquals 條件運算子使用，以強制結合特定標籤金鑰與值，例如強制執行標籤 cost-center=cc123：

    ```
    "StringEquals": { "aws:RequestTag/cost-center": "cc123" }
    ```

  - 搭配 StringLike 條件運算子使用，以在請求中強制執行特定標籤金鑰，例如強制執行標籤金鑰 purpose：

    ```
    "StringLike": { "aws:RequestTag/purpose": "*" }
    ```

- aws:TagKeys：強制執行請求中使用的標籤金鑰。
  - 搭配 ForAllValues 修飾詞使用，若請求內提供特定標籤金鑰，將強制加以執行 (若請求內指定標籤，則僅允許特定標籤金鑰，不允許其他標籤)。例如，允許標籤金鑰 environment 或 cost-center：

    ```
    "ForAllValues:StringEquals": { "aws:TagKeys": ["environment","cost-center"] }
    ```

  - 搭配 ForAnyValue 修飾詞使用，以強制要求請求內至少具有一個指定的標籤金鑰。例如，請求內必須出現至少下列標籤金鑰 environment 或 webserver 之一：

```
"ForAnyValue:StringEquals": { "aws:TagKeys": ["environment","webserver"] }
```

這些條件金鑰可套用於支援標記的資源建立動作，以及 ecs:TagResource 動作。若要了解 Amazon ECS API 動作是否支援標記，請參閱 Amazon ECS 的動作、資源和條件金鑰。

若要強制使用者在建立資源時指定標籤，您必須在資源建立動作內，搭配 aws:RequestTag 修飾詞使用 aws:TagKeys 條件金鑰或 ForAnyValue 條件金鑰。若使用者未針對資源建立動作指定標籤，則不會評估 ecs:TagResource 動作。

以條件而言，條件金鑰不區分大小寫，而條件值會區分大小寫。因此，欲強制標籤鍵區分大小寫，請使用 aws:TagKeys 條件索引鍵，其中標籤鍵指定為條件值。

如需多值條件的詳細資訊，請參閱IAM 使用者指南中的建立可測試多個金鑰值的條件。

使用資源標籤控制對 Amazon ECS 資源的存取

當您建立授予使用者使用 Amazon ECS 資源之許可的 IAM 政策時，您可以在政策的 Condition 元素中包含標籤資訊，以根據標籤控制存取。這稱為以屬性為基礎的存取控制 (ABAC)。ABAC 對於使用者可以修改、使用或刪除哪些資源提供了更佳的控制。如需詳細資訊，請參閱 AWS的 ABAC 是什麼？

例如：您可以建立一個政策，允許使用者刪除叢集，但如果叢集具有標籤 environment=production，則拒絕該動作。若要這樣做，您可以使用 aws:ResourceTag 條件金鑰，根據連接至資源的標籤允許或拒絕存取資源。

```
"StringEquals": { "aws:ResourceTag/environment": "production" }
```

若要了解 Amazon ECS API 動作是否支援使用 aws:ResourceTag 條件金鑰控制存取，請參閱 Amazon ECS 的動作、資源和條件金鑰。由於 Describe 動作不支援資源層級許可，您必須在不同的陳述式中指定它們，無需條件。

如需 IAM 政策的範例，請參閱Amazon ECS 示例政策 。

如果您允許或拒絕使用者根據標籤存取資源，請務必考慮明確拒絕使用者將這些標籤新增至相同資源或從中移除的能力。否則，使用者可能透過修改標籤來避開您的限制，並取得資源的存取。

Amazon ECS 示例政策

您可以使用 IAM 政策來授予使用者許可，以檢視和使用 Amazon ECS 主控台中的特定資源。您可以使用上一節中的範例原則；不過，這些原則是針對使用 AWS CLI 或 AWS SDK 發出的要求而設計的。

範例：允許使用者根據標籤刪除 Amazon ECS 叢集

當標籤具有「目的/測試」的金鑰/值對時，下列政策允許使用者刪除叢集。

```
{
  "Version": "2012-10-17",
  "Statement": [
       {
      "Action": [
        "ecs:DeleteCluster"
      ],
      "Effect": "Allow",
      "Resource": "arn:aws:ecs:region:account-id:cluster/*",
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/Purpose": "Testing"
        }
      }
    }
  ]
}
```

對 Amazon Elastic Container Service 身分和存取進行故障診斷

請使用以下資訊來協助您診斷和修正使用 Amazon ECS 和 IAM 時可能遇到的常見問題。

主題

- 我未獲授權，不得在 Amazon ECS 中執行動作
- 我沒有授權執行 iam：PassRole
- 我想允許我以外的人訪 AWS 帳戶 問我的 Amazon ECS 資源
- 其他疑難排解資源

我未獲授權，不得在 Amazon ECS 中執行動作

如果您收到錯誤，告知您未獲授權執行動作，您的政策必須更新，允許您執行動作。

下列範例錯誤會在mateojackson IAM 使用者嘗試使用主控台檢視一個虛構 *my-example-widget* 資源的詳細資訊，但卻無虛構 ecs:*GetWidget* 許可時發生。

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 ecs:GetWidget on resource: my-example-widget
```

在此情況下，必須更新 mateojackson 使用者的政策，允許使用 ecs:*GetWidget* 動作存取 *my-example-widget* 資源。

如果您需要協助，請聯絡您的 AWS 管理員。您的管理員提供您的簽署憑證。

我沒有授權執行 iam：PassRole

如果您收到錯誤，告知您無權執行 iam:PassRole 動作，則必須更新您的政策，以允許您將角色傳遞至 Amazon ECS。

有些 AWS 服務 允許您將現有角色傳遞給該服務，而不是建立新的服務角色或服務連結角色。如需執行此作業，您必須擁有將角色傳遞至該服務的許可。

當名為 marymajor 的 IAM 使用者嘗試使用主控台在 Amazon ECS 中執行動作時，發生下列範例錯誤。但是，動作請求服務具備服務角色授予的許可。Mary 沒有將角色傳遞至該服務的許可。

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

在這種情況下，Mary 的政策必須更新，允許她執行 iam:PassRole 動作。

如果您需要協助，請聯絡您的 AWS 管理員。您的管理員提供您的簽署憑證。

我想允許我以外的人訪 AWS 帳戶 問我的 Amazon ECS 資源

您可以建立一個角色，讓其他帳戶中的使用者或您組織外部的人員存取您的資源。您可以指定要允許哪些信任物件取得該角色。針對支援基於資源的政策或存取控制清單 (ACL) 的服務，您可以使用那些政策來授予人員存取您的資源的許可。

如需進一步了解，請參閱以下內容：

- 若要了解 Amazon ECS 是否支援這些功能，請參閱 Amazon Elastic Container Service 如何與 IAM 搭配使用。
- 若要了解如何提供對您所擁有資源 AWS 帳戶 的存取權，請參閱 IAM 使用者指南中您擁有的另一 AWS 帳戶 個 IAM 使用者提供存取權限。
- 若要了解如何將資源存取權提供給第三方 AWS 帳戶，請參閱 IAM 使用者指南中的提供第三方 AWS 帳戶 擁有的存取權。

- 若要了解如何透過聯合身分提供存取權，請參閱 IAM 使用者指南中的將存取權提供給在外部進行身分驗證的使用者 (聯合身分)。
- 若要了解使用角色和資源型政策進行跨帳戶存取之間的差異，請參閱 IAM 使用者指南中的 IAM 角色與資源型政策的差異。

其他疑難排解資源

下列頁面提供錯誤碼的相關資訊：

- Amazon ECS 停止任務錯誤消息
- 檢視 Amazon ECS 服務事件訊息

Amazon ECS 的 IAM 最佳做法

您可以使用 AWS Identity and Access Management (IAM) 透過基於規則的政策來管理和控制對 AWS 服務和資源的存取，以進行身份驗證和授權。更具體地說，透過此服務，您可以使用套用至使用者、群組或角色的原則來控制對 AWS 資源的存取。在這三個使用者中，使用者是可以對您的資源進行存取的帳戶。而且，IAM 角色是一組可以由經驗證身份擔任操作的許可，該身份與 IAM 以外的特定身份沒有關聯。如需詳細資訊，請參閱 Amazon ECS 存取管理概觀：許可和政策。

遵循最低權限存取的政策

建立範圍為允許使用者執行其指定工作的政策。例如，如果開發人員需要定期停止任務，請建立僅允許該特定動作的政策。下列範例僅允許使用者停止屬於具特定 Amazon Resource Name (ARN) 叢集上特定 `task_family` 的任務。條件中參考 ARN 也是使用資源層級許可的範例。您可以使用資源層級許可來指定要套用動作的資源。

> **Note**
>
> 政策中參考 ARN 時，請使用新的較長 ARN 格式。如需詳細資訊，請參閱《Amazon Elastic Container Service 開發人員指南》中的 Amazon Resource Name (ARN) 和 ID。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
```

```
      "Action": [
        "ecs:StopTask"
      ],
      "Condition": {
        "ArnEquals": {
          "ecs:cluster": "arn:aws:ecs:region:account_id:cluster/cluster_name"
        }
      },
      "Resource": [
        "arn:aws:ecs:region:account_id:task-definition/task_family:*"
      ]
    }
  ]
}
```

讓叢集資源做為管理邊界

範圍太窄的政策可能會造成角色擴散，並增加系統管理額外負荷。不要建立範圍僅限於特定任務或服務的角色，而是建立範圍為叢集的角色，並使用叢集做為主要管理界限。

建立自動化管道以將最終使用者與 API 隔離

您可以透過建立自動封裝應用程式並將應用程式部署到 Amazon ECS 叢集的管道來限制使用者可以使用的動作。這可以有效地將建立、更新和刪除任務的工作委派給管道。如需詳細資訊，請參閱AWS CodePipeline 使用者指南 CodePipeline中的教學課程：Amazon ECS 標準部署。

針對新增的安全層使用政策條件

當您需要新增的安全層時，請在政策中新增條件。如果您正在執行特權操作或需要限制可針對特定資源執行的一組動作時，這種做法非常有用。刪除叢集時，下列範例政策需要多重要素驗證。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ecs:DeleteCluster"
      ],
      "Condition": {
        "Bool": {
          "aws:MultiFactorAuthPresent": "true"
```

```
        }
      },
    "Resource": ["*"]
    }
  ]
}
```

套用至服務的標籤會傳播至屬於該服務一部分的所有任務。因此，您可以使用特定標籤建立範圍為 Amazon ECS 資源的角色。在下列政策中，IAM 主體使用 `Department` 的 tag-key (標籤金鑰) 和 `Accounting` 的 tag-value (標籤值) 啟動和停止所有任務。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ecs:StartTask",
                "ecs:StopTask",
                "ecs:RunTask"
            ],
            "Resource": "arn:aws:ecs:*",
            "Condition": {
                "StringEquals": {"ecs:ResourceTag/Department": "Accounting"}
            }
        }
    ]
}
```

定期稽核對 API 的存取

使用者可能會變更角色。變更角色之後，先前授予他們的許可可能不再適用。確保您稽核有權存取 Amazon ECS API 的人員，以及該存取權是否仍然被授權。考慮將 IAM 與使用者生命週期管理解決方案整合，該解決方案會在使用者離開組織時自動撤銷存取。如需詳細資訊，請參閱 Amazon Web Services 一般參考 中的 Amazon ECS 安全性稽核指南。

在 Amazon Elastic Container Service 中記錄和監控

監控是維護 Amazon 彈性容器服務和您的 AWS 解決方案的可靠性、可用性和效能的重要組成部分。您應該從 AWS 解決方案的所有部分收集監視資料，以便在發生多點失敗時更輕鬆地偵錯。 AWS 提供數種工具來監控 Amazon ECS 資源並回應潛在事件：

Amazon CloudWatch 警報

監看指定時段內的單一指標，並根據與多個時段內給定之閾值相對的指標值來執行一或多個動作。動作是傳送至亞馬遜簡單通知服務 (Amazon SNS) 主題或 Amazon EC2 Auto Scaling 政策的通知。 CloudWatch 警示不會僅因為處於特定狀態而叫用動作；狀態必須已變更並維持指定數目的期間。如需詳細資訊，請參閱 使用監控 Amazon ECS CloudWatch 。

對於具有使用 Fargate 啟動類型之任務的服務，您可以根據 CloudWatch 指標 (例如 CPU 和記憶體使用率) 使用 CloudWatch警示來擴展和擴充服務中的任務。如需詳細資訊，請參閱 自動擴展您的 Amazon ECS 服務。

對於具有使用 EC2 啟動類型的任務或服務的叢集，您可以使用 CloudWatch警示根據 CloudWatch 指標 (例如叢集記憶體保留) 擴展和擴展容器執行個體。

Amazon CloudWatch 日誌

透過在任務定義中指定 `awslogs` 日誌驅動程式，可監控、存放及存取 Amazon ECS 任務中容器的日誌檔案。如需詳細資訊，請參閱使用 awslogs 驅動程式。

您也可以從 Amazon ECS 容器執行個體監控、存放及存取作業系統和 Amazon ECS 容器代理程式日誌檔案。這種存取日誌的方法可用於使用 EC2 啟動類型的容器。

Amazon CloudWatch 活動

比對事件並將其路由至一或多個目標函數或串流，以進行變更、擷取狀態資訊，以及採取修正動作。有關更多信息，請參閱本指南使用以下方式自動回應 Amazon ECS 錯誤 EventBridge中的以及什麼是 Amazon CloudWatch 活動？ 在 Amazon CloudWatch 活動用戶指南。

AWS CloudTrail 日誌

CloudTrail 提供 Amazon ECS 中使用者、角色或 AWS 服務所採取的動作記錄。使用收集的資訊 CloudTrail，您可以判斷向 Amazon ECS 發出的請求、提出請求的 IP 地址、提出請求的人員、提出請求的時間以及其他詳細資訊。如需詳細資訊，請參閱 使用記錄 Amazon ECS API 呼叫 AWS CloudTrail。

AWS Trusted Advisor

Trusted Advisor 利用為數十萬名 AWS 客戶提供服務所學到的最佳實踐。 Trusted Advisor 檢查您的 AWS 環境，然後在存在機會時提出建議，以節省資金、改善系統可用性和效能，或協助縮小安全性漏洞。所有 AWS 客戶都可以使用五張 Trusted Advisor 支票。擁有商業或企業支援方案的客戶可以檢視所有 Trusted Advisor 檢查。

如需詳細資訊，請參閱《AWS Support 使用者指南》中的 AWS Trusted Advisor。

AWS Compute Optimizer

AWS Compute Optimizer 是一項服務，用於分析 AWS 資源的組態和使用率指標。這會報告您的資源是否已為最佳化，並產生最佳化建議，以降低成本並改善工作負載的效能。

如需詳細資訊，請參閱 AWS Compute Optimizer 針對 Amazon ECS 的建議。

監控 Amazon ECS 的另一個重要部分是手動監控 CloudWatch 警示未涵蓋的項目。 CloudWatch Trusted Advisor、和其他 AWS 主控台儀表板可提供您 AWS 環境狀態的 at-a-glance 檢視。我們建議您也檢查您容器執行個體上的日誌檔，以及您任務中的容器。

Amazon Elastic Container Service 的合規驗證

若要瞭解 AWS 服務 是否屬於特定規範遵循方案的範圍內，請參閱AWS 服務 遵循規範計劃方案中的，並選擇您感興趣的合規方案。如需一般資訊，請參閱AWS 規範計劃AWS。

您可以使用下載第三方稽核報告 AWS Artifact。如需詳細資訊，請參閱下載中的報告中的 AWS Artifact。

您在使用時的合規責任取決 AWS 服務 於資料的敏感性、公司的合規目標以及適用的法律和法規。AWS 提供下列資源以協助遵循法規：

- 安全性與合規性快速入門指南 — 這些部署指南討論架構考量，並提供部署以安全性和合規性 AWS 為重點的基準環境的步驟。
- 在 Amazon Web Services 上架構 HIPAA 安全性與合規性 — 本白皮書說明公司如何使用建立符合 HIPAA 資格的應 AWS 用程式。

> **Note**
>
> 並非所有人 AWS 服務 都符合 HIPAA 資格。如需詳細資訊，請參閱 HIPAA 資格服務參照。

- AWS 合規資源AWS — 此工作簿和指南集合可能適用於您的產業和所在地。
- AWS 客戶合規指南 — 透過合規的角度瞭解共同的責任模式。這份指南總結了在多個架構 (包括美國國家標準技術研究所 (NIST)、支付卡產業安全標準委員會 (PCI) 和國際標準化組織 (ISO)) 中，保 AWS 服務 護指引並對應至安全控制的最佳實務。
- 使用AWS Config 開發人員指南中的規則評估資源 — 此 AWS Config 服務會評估您的資源組態符合內部實務、產業準則和法規的程度。

- AWS Security Hub— 這 AWS 服務 提供了內部安全狀態的全面視圖 AWS。Security Hub 使用安全控制，可評估您的 AWS 資源並檢查您的法規遵循是否符合安全業界標準和最佳實務。如需支援的服務和控制清單，請參閱 Security Hub controls reference。
- Amazon GuardDuty — 透過監控環境中的 AWS 帳戶可疑和惡意活動，藉此 AWS 服務 偵測您的工作負載、容器和資料的潛在威脅。 GuardDuty 可協助您因應各種合規性需求，例如 PCI DSS，滿足特定合規性架構所規定的入侵偵測需求。
- AWS Audit Manager— 這 AWS 服務 有助於您持續稽核您的 AWS 使用情況，以簡化您管理風險的方式，以及遵守法規和業界標準的方式。

Amazon ECS 的合規性和安全最佳實務

您使用 Amazon ECS 時的合規責任取決於資料的敏感度、您公司的合規目標及適用的法律和法規。

AWS 提供下列資源以協助遵循法規：

- 安全性與合規性快速入門指南：這些部署指南討論架構考量，並提供在上部署以安全性和法規遵循為重點的基準環境的步驟。 AWS
- 建構 HIPAA 安全性與合規性白皮書：本白皮書說明公司如何使用建立符合 HIPAA 標準的應用 AWS 程式。
- 合規計劃範圍內的AWS 服務：此清單包含特定合規計劃範圍內的 AWS 服務。如需詳細資訊，請參閱 AWS 合規計劃。

支付卡產業資料安全標準 (PCI DSS)

在遵守 PCI DSS 時，請務必了解環境中持卡人資料 (CHD) 的完整流程。CHD 流程決定 PCI DSS 的適用性、定義持卡人資料環境 (CDE) 的界限和元件，以及 PCI DSS 評估的範圍。精確判斷 PCI DSS 範圍是定義安全狀態以及最終成功評估的關鍵。客戶必須具備範圍判定程序，以確保其完整性並偵測範圍的變更或偏離。

容器化應用程式的暫時性本質可在稽核組態時提供其他的複雜性。因此，客戶需要保持所有容器組態參數的感知，以確保在容器生命週期的所有階段都能滿足合規要求。

如需有關 Amazon ECS 上實現 PCI DSS 合規的其他資訊，請參閱下列白皮書。

- Amazon ECS 上 PCI DSS 合規性的架構
- AWS上 PCI DSS 範圍設定和分段的架構

HIPAA (美國健康保險流通與責任法案)

將 Amazon ECS 與處理受保護醫療資訊 (PHI) 的工作負載搭配使用，無需額外設定。Amazon ECS 充當協同運作服務，可協調在 Amazon EC2 上啟動容器的情況。其不會與正協同運作的工作負載中的資料一起運作，或對資料進行操作。符合 HIPAA 法規和 AWS 商業夥伴增補合約，在使用 Amazon ECS 啟動的容器存取時，PHI 應該在傳輸過程中和靜態加密。

每個 AWS 儲存選項均提供多種靜態加密機制，例如 Amazon S3、Amazon EBS 和. AWS KMS您可以部署覆蓋網路 (例如 VMS3 或 Weave Net)，以確保在容器之間傳輸的 PHI 完全加密，或提供備援加密層。還應啟用完整的日誌記錄，並將所有容器日誌導向到 Amazon CloudWatch。若要使用基礎架構安全性的最佳做法來設計您的 AWS 環境，請參閱安全性支柱架構良 AWS 好的架構中的基礎結構保護。

AWS Security Hub

用 AWS Security Hub 於監控與安全最佳實務相關的 Amazon ECS 使用情況。Security Hub 會使用控制來評估資源組態和安全標準，協助您遵守各種合規架構。如需有關使用 Security Hub 評估 Amazon ECS 資源的詳細資訊，請參閱《AWS Security Hub 使用者指南》中的 Amazon ECS 控制項。

Amazon GuardDuty 與 Amazon ECS 運行時監控

Amazon GuardDuty 是一種威脅偵測服務，可協助保護您的帳戶、容器、工作負載和 AWS 環境中的資料。使用機器學習 (ML) 模型，以及異常和威脅偵測功能， GuardDuty持續監控不同的記錄檔來源和執行階段活動，以識別環境中潛在的安全風險和惡意活動並排定優先順序。

使用中的執行階段監控 GuardDuty 來識別惡意或未經授權的行為 執行階段監控透過持續監控 AWS 日誌和聯網活動來識別惡意或未經授權的行為，保護 Fargate 和 EC2 上執行的工作負載。執行階段監控使用輕量型、完全受控的 GuardDuty 安全代理程式來分析主機上的行為，例如檔案存取、程序執行和網路連線。這涵蓋的問題包括權限提升、使用暴露的登入資料，或與惡意 IP 地址、網域通訊，以及 Amazon EC2 執行個體和容器工作負載上存在惡意軟體的情況。如需詳細資訊，請參閱GuardDuty 使用指南中的GuardDuty 執行階段監視。

合規性建議

您應該儘早讓合規計劃擁有者參與您的企業，並使用 AWS 共同的責任模型來識別合規控制擁有權，以便在相關合規計劃中取得成功。

AWS Fargate 美國聯邦資訊處理標準 (FIPS-140)

美國聯邦資訊處理標準 (FIPS)。FIPS-140 是美國和加拿大政府標準，指定密碼模組的安全要求以保護敏感資訊。FIPS-140 定義了一組經驗證的密碼函數，可用於加密傳輸中的資料和靜態資料。

當您啟用 FIPS-140 合規時，您可以在 Fargate 上以符合 FIPS-140 的方式執行工作負載。如需 FIPS-140 合規的詳細資訊，請參閱聯邦資訊處理標準 (FIPS) 140-2。

AWS Fargate FIPS-140 考量事項

Fargate 上使用 FIPS-140 合規時，請考量下列事項：

- FIPS-140 合規僅適用於 AWS GovCloud (US) 區域。
- FIPS-140 合規預設為關閉。您必須將其開啟。
- 您的任務必須使用下列組態才能符合 FIPS-140 規範：
  - `operatingSystemFamily` 必須是 `LINUX`。
  - `cpuArchitecture` 必須是 `X86_64`。
  - Fargate 平台版本必須是 `1.4.0` 以上的版本。

在 Fargate 上使用 FIPS

使用下列程序以在 Fargate 上使用 FIPS-140 合規。

1. 開啟 FIPS-140 合規的功能。如需詳細資訊，請參閱 the section called “AWS Fargate 符合聯邦資訊處理標準 (FIPS-140) 規範”。
2. 您可以選擇性使用 ECS Exec 執行下列命令，以驗證叢集的 FIPS-140 合規狀態。

   使用您叢集的名稱取代 *my-cluster*。

   傳回值 "1" 表示您正在使用 FIPS。

   ```
   aws ecs execute-command --cluster cluster-name \
       --interactive \
       --command "cat /proc/sys/crypto/fips_enabled"
   ```

用 CloudTrail 於 Fargate FIPS-140 稽核

CloudTrail 當您建立 AWS 帳戶時，會在您的帳戶中開啟。當 Amazon ECS 中發生 API 和主控台活動時，該活動會與事件歷史記錄中的其他 AWS 服務 CloudTrail 事件一起記錄在事件中。您可以在帳戶中查看，搜索和下載最近的事 AWS 件。如需詳細資訊，請參閱檢視具有事 CloudTrail 件記錄的事件。

如需 AWS 帳戶中持續的事件記錄 (包括 Amazon ECS 的事件)，請建立 CloudTrail 用於將日誌檔傳送到 Amazon S3 儲存貯體的追蹤。根據預設，當您在主控台建立權杖時，權杖會套用到所有區域。追蹤記錄來自 AWS 分區中所有區域的事件，並將日誌檔傳送到您指定的 Amazon S3 儲存貯體。此外，您還可以設定其他 AWS 服務，以進一步分析 CloudTrail 記錄中收集的事件資料並採取行動。如需詳細資訊，請參閱 the section called “使用記錄 Amazon ECS API 呼叫 AWS CloudTrail”。

下列範例顯示示範 `PutAccountSettingDefault` API 動作的 CloudTrail 記錄項目：

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "AIDAIV5AJI5LXF5EXAMPLE",
         "arn": "arn:aws:iam::123456789012:user/jdoe",
        "accountId": "123456789012",
        "accessKeyId": "AKIAIPWIOFC3EXAMPLE",
    },
    "eventTime": "2023-03-01T21:45:18Z",
    "eventSource": "ecs.amazonaws.com",
    "eventName": "PutAccountSettingDefault",
    "awsRegion": "us-gov-east-1",
    "sourceIPAddress": "52.94.133.131",
    "userAgent": "aws-cli/2.9.8 Python/3.9.11 Windows/10 exe/AMD64 prompt/off command/
ecs.put-account-setting",
    "requestParameters": {
        "name": "fargateFIPSMode",
        "value": "enabled"
    },
    "responseElements": {
        "setting": {
            "name": "fargateFIPSMode",
            "value": "enabled",
            "principalArn": "arn:aws:iam::123456789012:user/jdoe"
        }
    },
    "requestID": "acdc731e-e506-447c-965d-f5f75EXAMPLE",
    "eventID": "6afced68-75cd-4d44-8076-0beEXAMPLE",
    "readOnly": false,
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "123456789012",
    "eventCategory": "Management",
    "tlsDetails": {
```

```
        "tlsVersion": "TLSv1.2",
        "cipherSuite": "ECDHE-RSA-AES128-GCM-SHA256",
        "clientProvidedHostHeader": "ecs-fips.us-gov-east-1.amazonaws.com"
    }
}
```

Amazon Elastic Container Service 的基礎設施安全性

作為受管服務，Amazon 彈性容器服務受到 AWS 全球網路安全的保護。有關 AWS 安全服務以及如何 AWS 保護基礎結構的詳細資訊，請參閱AWS 雲端安全 若要使用基礎架構安全性的最佳做法來設計您的 AWS 環境，請參閱安全性支柱架構良 AWS 好的架構中的基礎結構保護。

您可以使用 AWS 已發佈的 API 呼叫透過網路存取 Amazon ECS。使用者端必須支援下列專案：

- Transport Layer Security (TLS)。我們需要 TLS 1.2 並建議使用 TLS 1.3。
- 具備完美轉送私密(PFS)的密碼套件，例如 DHE (Ephemeral Diffie-Hellman)或 ECDHE (Elliptic Curve Ephemeral Diffie-Hellman)。現代系統(如 Java 7 和更新版本)大多會支援這些模式。

此外，請求必須使用存取金鑰 ID 和與 IAM 主體相關聯的私密存取金鑰來簽署。或者，您可以透過 AWS Security Token Service (AWS STS) 來產生暫時安全憑證來簽署請求。

您可以從任何網路位置呼叫這些 API 操作。Amazon ECS 支援資源型存取政策，這類政策可根據來源 IP 地址納入限制，因此請確認該政策已將網路位置的 IP 地址考慮在內。您也可以使用 Amazon ECS 政策，控制特定 Amazon Virtual Private Cloud 端點或特定 VPC 的存取權。實際上，這會將對指定 Amazon ECS 資源的網路存取從網路內的特定 VPC 隔離出來。 AWS 如需詳細資訊，請參閱 Amazon ECS 介面 VPC 端點 (AWS PrivateLink)。

Amazon ECS 介面 VPC 端點 (AWS PrivateLink)

透過設定 Amazon ECS 以使用介面 VPC 端點，可提升 VPC 的安全狀態。界面端點採用這種技術 AWS PrivateLink，可讓您使用私有 IP 地址私有存取 Amazon ECS API。 AWS PrivateLink 將 VPC 和 Amazon ECS 之間的所有網路流量限制在 Amazon 網路上。您不需要網際網路閘道、NAT 裝置或虛擬私有閘道。

如需 AWS PrivateLink 和 VPC 端點的詳細資訊，請參閱 Amazon VPC 使用者指南中的 VPC 端點。

考量事項

自 2023 年 12 月 23 日起推出之區域中端點的考量事項

在您設定 Amazon ECS 的介面 VPC 端點之前，請注意以下幾點考量：

- 您必須具有下列區域特定的 VPC 端點：
  - `com.amazonaws.region.ecs-agent`
  - `com.amazonaws.region.ecs-telemetry`
  - `com.amazonaws.region.ecs`

  例如，加拿大西部 (卡加利) (ca-west-1) 區域需要下列 VPC 端點：
  - `com.amazonaws.ca-west-1.ecs-agent`
  - `com.amazonaws.ca-west-1.ecs-telemetry`
  - `com.amazonaws.ca-west-1.ecs`
- 當您使用樣版在新區域中建立 AWS 資源，且樣版是從 2023 年 12 月 23 日之前引入的「區域」複製時，視複製來源區域而定，請執行下列其中一項作業。

  例如，複製來源區域為美國東部 (維吉尼亞北部) (us-east-1)。複製到地區是加拿大西部 (卡爾加里) (ca-west-1)。

組態	動作	
複製來源區域沒有任何 VPC 端點。	為新區域建立全部三個 VPC 端點 (例如，`com.amazonaws.ca-west-1.ecs-agent`)。	
複製來源區域包含區域特定的 VPC 端點。	a. 為新區域建立全部三個 VPC 端點 (例如，`com.amazonaws.ca-west-1.ecs-agent`)。 b. 刪除從區域複製的所有三個 VPC 端點 (例如)。`com.amazo`	

組態	動作	
	naws.us-east-1.ecs-agent	

Fargate 啟動類型的 Amazon ECS VPC 端點的考量事項

如果在部署 Fargate 工作ecr.dkr的相同 VPC ecr.api 中存在 VPC 的 VPC 端點，則會使用 VPC 端點。如果沒有 VPC 端點，它將使用 Fargate 介面。

在您設定 Amazon ECS 的介面 VPC 端點之前，請注意以下幾點考量：

- 使用 Fargate 啟動類型的任務不需要 Amazon ECS 的介面虛擬私人雲端節點，但您可能需要 Amazon ECR、Secrets Manager 或 Amazon CloudWatch 日誌的介面 VPC 端點，如以下幾點所述。
  - 若要讓任務從 Amazon ECR 中提取私有映像，請務必為 Amazon ECR 建立介面 VPC 端點。如需詳細資訊，請參閱 《Amazon Elastic Container Registry 使用者指南》中的介面 VPC 端點 (AWS PrivateLink)。

    如果 VPC 沒有網際網路閘道，則必須為 Amazon S3 建立閘道端點。如需詳細資訊，請參閱 《Amazon Elastic Container Registry 使用者指南》中的建立 Amazon S3 閘道端點。Amazon S3 的介面端點無法與 Amazon ECR 一起使用。

    > **Important**
    >
    > 如果您設定 Amazon ECR 以使用介面 VPC 端點，您就可以建立包含條件金鑰的任務執行角色，限制存取特定的 VPC 或 VPC 端點。如需詳細資訊，請參閱 透過界面端點許可提取 Amazon ECR 映像的 Fargate 端任務。

  - 若要讓任務從 Secrets Manager 中提取敏感資料，您必須建立 Secrets Manager 的介面 VPC 端點。如需詳細資訊，請參閱 《AWS Secrets Manager 使用者指南》中的搭配使用 Secrets Manager 與 VPC 端點。
  - 如果您的 VPC 沒有網際網路閘道，且您的工作使用記awslogs錄驅動程式將記錄資訊傳送至 CloudWatch 記錄檔，則必須為 CloudWatch 記錄檔建立介面 VPC 端點。如需詳細資訊，請參閱 Amazon CloudWatch 日誌使用指南中的將日 CloudWatch 誌與界面 VPC 端點搭配使用。
- VPC 端點目前不支援跨區域請求。請確保在計劃對 Amazon ECS 發出 API 呼叫的相同區域中建立端點。例如，假設您想在美國東部 (維吉尼亞北部) 執行任務。那麼，您必須在美國東部 (維吉尼亞北

部) 建立 Amazon ECS VPC 端點。在任何其他區域建立的 Amazon ECS VPC 端點無法在美國東部 (維吉尼亞北部) 執行任務。

- 透過 Amazon Route 53，VPC 端點僅支援 Amazon 提供的 DNS。如果您想要使用自己的 DNS，您可以使用條件式 DNS 轉送。如需詳細資訊，請參閱《Amazon VPC 使用者指南》中的 DHCP 選項集。
- 連接到 VPC 端點的安全群組必須允許從 VPC 的私有子網路，透過 TCP 連接埠 443 傳入的連線。
- Envoy 代理的 Service Connect (服務連接) 管理會使用 `com.amazonaws.region.ecs-agent` VPC 端點。當您未使用 VPC 端點時，Envoy 代理的 Service Connect (服務連接) 管理會使用該區域中的 `ecs-sc` 端點。如需每個區域中的 Amazon ECS 端點清單，請參閱 Amazon ECS 端點和配額。

EC2 啟動類型的 Amazon ECS VPC 端點的考量事項

在您設定 Amazon ECS 的介面 VPC 端點之前，請注意以下幾點考量：

- 使用 EC2 啟動類型的任務需要任務啟動所在的容器執行個體，才能執行 `1.25.1` 版或更新版本的 Amazon ECS 容器代理程式。如需詳細資訊，請參閱 Amazon ECS Linux 容器執行個體管理。
- 若要讓任務從 Secrets Manager 中提取敏感資料，您必須建立 Secrets Manager 的介面 VPC 端點。如需詳細資訊，請參閱 《AWS Secrets Manager 使用者指南》中的搭配使用 Secrets Manager 與 VPC 端點。
- 如果您的 VPC 沒有網際網路閘道，且您的工作使用記`awslogs`錄驅動程式將記錄資訊傳送至 CloudWatch 記錄檔，則必須為 CloudWatch 記錄檔建立介面 VPC 端點。如需詳細資訊，請參閱 Amazon CloudWatch 日誌使用指南中的將日 CloudWatch 誌與界面 VPC 端點搭配使用。
- VPC 端點目前不支援跨區域請求。請確保在計劃對 Amazon ECS 發出 API 呼叫的相同區域中建立端點。例如，假設您想在美國東部 (維吉尼亞北部) 執行任務。那麼，您必須在美國東部 (維吉尼亞北部) 建立 Amazon ECS VPC 端點。在任何其他區域建立的 Amazon ECS VPC 端點無法在美國東部 (維吉尼亞北部) 執行任務。
- 透過 Amazon Route 53，VPC 端點僅支援 Amazon 提供的 DNS。如果您想要使用自己的 DNS，您可以使用條件式 DNS 轉送。如需詳細資訊，請參閱《Amazon VPC 使用者指南》中的 DHCP 選項集。
- 連接到 VPC 端點的安全群組必須允許從 VPC 的私有子網路，透過 TCP 連接埠 443 傳入的連線。

建立 Amazon ECS 的 VPC 端點

若要為 Amazon ECS 服務建立 VPC 端點，請使用 《Amazon VPC 使用者指南》中的建立介面端點程序，建立下列端點。如果您的 VPC 中有現有的容器執行個體，您應該依照其列出順序建立端點。如果您計劃在建立 VPC 端點後才建立您的容器執行個體，則其順序並不重要。

- `com.amazonaws.region.ecs-agent`
- `com.amazonaws.region.ecs-telemetry`
- `com.amazonaws.region.ecs`

> **Note**
>
> *##* 表示 Amazon ECS 支援之 AWS 區域的區域識別符，例如 `us-east-2` 表示美國東部 (俄亥俄) 區域。

`ecs-agent`端點會使用 `ecs:poll` API，且`ecs-telemetry`端點會使用`ecs:poll`和 `ecs:StartTelemetrySession` API。

如果您擁有的現有任務是使用 EC2 啟動類型，在建立 VPC 端點後，每個容器執行個體需要挑選新的組態。若要這麼做，您必須擇一重新啟動每個容器執行個體或重新啟動每個容器執行個體上的 Amazon ECS 容器代理程式。若要重新開機容器代理程式，請執行下列動作：

若要重新啟動 Amazon ECS 容器代理程式

1. 透過 SSH 登入您的容器執行個體。
2. 停用容器代理程式。

```
sudo docker stop ecs-agent
```

3. 啟動容器代理程式。

```
sudo docker start ecs-agent
```

在建立 VPC 端點並重新啟動每個容器執行個體上的 Amazon ECS 容器代理程式後，所有新啟動的任務都會挑選新的組態。

為 Amazon ECS 建立 VPC 端點政策

您可以將端點政策連接至控制 Amazon ECS 存取權的 VPC 端點。此政策會指定下列資訊：

- 可執行動作的主體。
- 可執行的動作。
- 可供執行動作的資源。

如需詳細資訊，請參閱 Amazon VPC 使用者指南中的使用 VPC 端點控制對服務的存取。

範例：Amazon ECS 動作的 VPC 端點政策

以下是 Amazon ECS 端點政策的範例。在連接至端點後，此政策會授予存取許可，以允許建立並列出叢集。`CreateCluster` 和 `ListClusters` 動作不接受任何資源，因此資源定義已針對所有資源設為 *。

```
{
   "Statement":[
    {
      "Effect": "Allow",
      "Action": [
        "ecs:CreateCluster",
        "ecs:ListClusters"
      ],
      "Resource": [
        "*"
      ]
    }
  ]
}
```

Amazon ECS 任務和容器安全最佳實務

您應該將容器映像視為對抗攻擊的第一道防線。不安全、構造不良的映像可允許攻擊者逸出容器的邊界並取得主機的存取。您應該執行以下操作以減輕發生這種情況的風險。

建議您在設定任務和容器時進行下列操作。

建立最小或使用 distroless 映像

首先從容器映像中刪除所有無關的二進位檔案。如果您使用 Amazon ECR Public Gallery 中不熟悉的映像，請檢查映像以參考每個容器層的內容。您可以使用 Dive 之類的應用程式來執行此操作。

或者，您可以使用僅包含應用程式及其執行期相依項的 distroless 映像。此映像不包含套件管理工具或 Shell。Distroless 映像可改善「掃描器的訊號雜訊，並減少您所需要確定來源的負擔」。有關更多信息，請參閱有關無發行版的 GitHub 文檔。

Docker 具有一種機制，用於從稱為 scratch 的保留最小映像建立映像。如需詳細資訊，請參閱 Docker 文件中的使用 scratch 建立簡單的父映像。使用 Go 等語言，您可以建立一個靜態連結的二進位檔案並在 Dockerfile 中作為參考。以下範例展示如何完成這項操作。

```
############################
# STEP 1 build executable binary
############################
FROM golang:alpine AS builder
# Install git.
# Git is required for fetching the dependencies.
RUN apk update && apk add --no-cache git
WORKDIR $GOPATH/src/mypackage/myapp/
COPY . .
# Fetch dependencies.
# Using go get.
RUN go get -d -v
# Build the binary.
RUN go build -o /go/bin/hello
############################
# STEP 2 build a small image
############################
FROM scratch
# Copy our static executable.
COPY --from=builder /go/bin/hello /go/bin/hello
# Run the hello binary.
ENTRYPOINT ["/go/bin/hello"]
This creates a container image that consists of your application and nothing else,
 making it extremely secure.
```

上一個範例也是多階段建置的範例。從安全的角度來看，這些類型的建置很有吸引力，因為您可以使用這些建置來最小化推送至容器登錄檔的最終映像大小。沒有構建工具和其他無關二進位檔案的容器映像透過減少映像的受攻擊面來改善您的安全狀態。如需有關多階段建置的詳細資訊，請參閱建立多階段建置。

掃描您的映像是否有漏洞

容器映像檔與虛擬機器對應部分類似，容器映像檔可以包含具有漏洞的二進位檔案和應用程式程式庫，或隨著時間的推移而產生漏洞。防止漏洞攻擊的最佳方法是使用映像掃描器定期掃描映像。

可以透過推送或隨需掃描 (每 24 小時掃描一次) 儲存在 Amazon ECR 中的映像。Amazon ECR 基本掃描使用 Clair，這是一種開放原始碼映像掃描解決方案。Amazon ECR 增強了使用 Amazon Inspector 的掃描。掃描影像後，結果會記錄到 Amazon 中的 Amazon ECR 事件串流中。 EventBridge您也可以從 Amazon ECR 主控台或呼叫 DescribeImageScanFindingsAPI 來查看掃描結果。應刪除或重建具有 `HIGH` 或 `CRITICAL` 漏洞的映像。如果已部署的映像出現漏洞，則應盡快更換。

Docker Desktop Edge 2.3.6.0 以上版本可以掃描本機映像。掃描由 Snyk 提供支援，其為一種應用程式安全服務。當發現漏洞時，Snyk 會識別 Dockerfile 中具漏洞的圖層和相依項。也建議使用安全的替代方法，例如使用具有較少漏洞的更細微基礎映像，或將特定套件升級至較新版本。透過使用 Docker 掃描，開發人員可以在將映像推送至登錄檔之前解決潛在的安全問題。

- 使用 Amazon ECR 自動化映像合規，並 AWS Security Hub說明如何從 AWS Security Hub Amazon ECR 中顯示漏洞資訊，以及透過封鎖易受攻擊映像的存取來自動修補。

從映像移除特殊權限

存取權標記 `setuid` 並 `setgid` 允許執行具有擁有者許可或可執行檔群組的可執行檔。從映像中移除具有這些存取權的所有二進位檔案，因為這些二進位檔案可用來提升權限。考慮刪除所有可用於惡意目的的 Shell 和共用程序，如 `nc` 和 `curl` 。您可以使用以下命令找到具有 `setuid` 和 `setgid` 存取權的檔案。

```
find / -perm /6000 -type f -exec ls -ld {} \;
```

若要從這些檔案中刪除這些特殊權限，請將以下指令新增至容器映像中。

```
RUN find / -xdev -perm /6000 -type f -exec chmod a-s {} \; || true
```

建立一組精選映像

為組織中的不同應用程式堆疊建立一組經過審查的映像，而不是允許開發人員建立自己的映像。透過執行此動作，開發人員可以放棄學習如何編寫 Dockerfile 並專注於編寫程式碼。當變更合併至您的程式碼庫中時，CI/CD 管道可以自動編譯資產，然後將其儲存在成品儲存庫中。最後，將成品複製到適

當的映像中，然後再將其推送至 Docker 登錄檔 (例如 Amazon ECR)。至少您應該建立一組基礎映像，開發人員可以從中建立自己的 Dockerfile。您應避免從 Docker Hub 推送映像。您並非永遠知道映像中的內容，而前 1000 個映像中大約有五分之一存在漏洞。這些映像及其漏洞的清單可以在 https://vulnerablecontainers.org/ 找到。

掃描應用程式套件和程式庫是否有漏洞

現在，使用開放原始碼程式庫很普遍。如同使用作業系統和 OS 套件，這些程式庫可能存在漏洞。作為開發生命週期的一部分，這些程式庫應在發現嚴重漏洞時進行掃描和更新。

Docker Desktop 使用 Snyk 執行本機掃描。也可以用來尋找開放原始碼程式庫中的漏洞和潛在的授權問題。可以直接整合至開發人員工作流程中，讓您能夠減輕開放原始碼程式庫所帶來的風險。如需詳細資訊，請參閱下列主題：

- 開放原始碼應用程式安全工具包括用於偵測應用程式漏洞的工具清單。

執行靜態程式碼分析

建置容器映像之前，您應該先執行靜態程式碼分析。此分析會針對您的原始程式碼執行，並用來識別惡意行為 (例如故障注入) 可能利用的編碼錯誤和程式碼。SonarQube是靜態應用程式安全性測試 (SAST) 的熱門選項，可支援各種不同的程式設計語言。

以非根使用者身分執行容器

您應以非根使用者身分執行容器。在預設情況下，容器以 `root` 使用者身份執行，除非在 Dockerfile 中包含該 USER 指令。Docker 指派的預設 Linux 功能會限制可做為 `root` 執行的動作，但僅限於稍微限制。例如，做為 `root` 執行的容器仍不允許存取裝置。

作為 CI/CD 管道的一部分，您應該檢查 Docerfile 來查找 USER 指令，如果缺少該指令，則建置失敗。如需詳細資訊，請參閱下列主題：

- 碼頭文件-LINT 是一個開源工具，可 RedHat 用於檢查文件是否符合最佳實踐。
- Hadolint 是建置符合最佳實務 Docker 映像檔的另一種工具。

使用唯讀的根檔案系統

您應該使用唯讀的根檔案系統。容器的根檔案系統預設為可寫入。當您使用 RO (唯讀) 根檔案系統設定容器時，會強制您明確定義資料可以保存的位置。這會減少您的受攻擊面，因為除非特別授予許可，否則無法寫入容器的檔案系統。

> **Note**
>
> 擁有唯讀根檔案系統可能會導致某些作業系統套件發生問題，這些套件預期能夠寫入檔案系統。如果您打算使用唯讀的根檔案系統，請事先徹底測試。

使用 CPU 和記憶體限制來設定任務 (Amazon EC2)

您應該使用 CPU 和記憶體限制來設定任務，以將下列風險降至最低。任務的資源限制會設定任務內所有容器可保留的 CPU 和記憶體數量上限。如果未設定限制，則任務可以存取主機的 CPU 和記憶體。這可能會導致部署在共用主機上的任務會剝奪其他任務系統資源的問題。

> **Note**
>
> AWS Fargate 任務上的 Amazon ECS 會要求您指定 CPU 和記憶體限制，因為它會使用這些值進行計費。對於 Amazon ECS Fargate 而言，一項佔用所有系統資源的任務並不是問題，因為每個任務都在自己的專用預留執行個體上執行。如果您未指定記憶體限制，Amazon ECS 會為每個容器配置至少 4MB。同樣，如果沒有為任務設定 CPU 限制，Amazon ECS 容器代理程式至少會為其指派 2 個 CPU。

不可變標籤與 Amazon ECR 一起使用

使用 Amazon ECR，您可以且應使用具有不可變標籤的設定映像。這樣可以防止將已更改或更新版本的映像推送至具相同標籤的映像儲存庫。這樣可以防止攻擊者將破解版本的映像推送至具相同標籤的映像。透過使用不可變標籤，您可以有效地強制自己為每次變更推送具有不同標籤的新映像。

避免以特權方式執行容器 (Amazon EC2)

您應該避免以特權方式執行容器。對於背景，在主機上使用擴展權限執行作為 `privileged` 執行的容器。這表示容器會繼承在主機上指派給 `root` 的所有 Linux 功能。應嚴格限制或禁止其使用。我們建議將 Amazon ECS 容器代理程式環境變數 `ECS_DISABLE_PRIVILEGED` 設定為 `true`，如不需要 `privileged` 時，以防止容器在特定主機上作為 `privileged` 執行。或者，您可 AWS Lambda 以使用掃描任務定義以使用`privileged`參數。

> **Note**
>
> AWS Fargate上的Amazon ECS 不支援作為 `privileged` 執行的容器。

從容器中移除不必要的 Linux 功能

以下是指派給 Docker 容器的預設 Linux 功能清單。如需有關每項功能的詳細資訊，請參閱 Linux 功能概觀。

```
CAP_CHOWN, CAP_DAC_OVERRIDE, CAP_FOWNER, CAP_FSETID, CAP_KILL,
CAP_SETGID, CAP_SETUID, CAP_SETPCAP, CAP_NET_BIND_SERVICE,
CAP_NET_RAW, CAP_SYS_CHROOT, CAP_MKNOD, CAP_AUDIT_WRITE,
CAP_SETFCAP
```

如果容器不需要上面列出的所有 Docker 核心功能，請考慮將其從容器中刪除。如需每個 Docker 核心功能的詳細資訊，請參閱KernelCapabilities。透過執行以下操作，您可以了解正在使用哪些功能：

- 安裝作業系統套件 libcap-ng 並執行 `pscap` 公用程式，以列出每個程序正在使用的功能。
- 您也可以使用 capsh 來解讀程序正在使用哪些功能。

使用客戶自管金鑰 (CMK) 來加密推送至 Amazon ECR 的映像

您應該使用客戶自管金鑰 (CMK) 來加密推送至 Amazon ECR 的映像。推送至 Amazon ECR 的映像會使用 AWS Key Management Service (AWS KMS) 受管金鑰自動以靜態加密。如果您想使用自己的金鑰，Amazon ECR 現在支援使用客戶受管金鑰 (CMK) 進行 AWS KMS 加密。使用 CMK 啟用伺服器端加密之前，請檢閱靜態加密文件中列出的注意事項。

Amazon ECS 教學課程

以下教學課程說明如何使用 Amazon ECS 執行常見的任務。

您可以使用下列任何一個教學課程，在 Amazon ECS 上部署任務 AWS CLI

教學課程概觀	進一步了解
為 Fargate 啟動類型建立 Linux 工作。	使用以下方式為 Fargate 啟動類型建立 Amazon ECS Linux 任務 AWS CLI
建立 Fargate 啟動類型的 Windows 工作。	為 Fargate 啟動類型建立 Amazon ECS 視窗任務 AWS CLI
為 EC2 啟動類型建立 Linux 工作。	為 EC2 啟動類型建立 Amazon ECS 任務，並使用 AWS CLI

您可以使用下列任一教學課程，進一步了解監視和記錄。

教學課程概觀	進一步了解
設定簡單的 Lambda 函數，以偵聽任務事件並將其寫入 CloudWatch 日誌記錄串流。	設定 Amazon ECS 以監聽 CloudWatch 事件事件
設定 Amazon EventBridge 事件規則，該規則僅擷取任務因為其中一個基本容器已終止而停止執行的任務事件。	發送 Amazon 簡單通知服務提醒 Amazon ECS 任務停止事件
串連原本屬於一個內容但分割為多個記錄或記錄行的記錄訊息。	串連多行或堆疊追蹤 Amazon ECS 日誌訊息

教學課程概觀	進一步了解	
在 Amazon ECS 中執行的 Windows 執行個體上部署 Fluent Bit 容器，以將 Windows 任務產生的日誌串流到 Amazon 以進 CloudWatch 行集中記錄。	在 Amazon ECS 視窗容器上部署流暢的位元	

您可以使用下列任何教學課程，進一步了解如何在 Amazon ECS 上使用群組受管服務帳戶使用 Active Directory 身份驗證。

教學課程概觀	進一步了解	
在 EC2 上搭配 Linux 容器使用群組受管服務帳戶。	用gMSA於 Amazon ECS 上的 EC2 Linux 容器	
使用群組受管服務帳戶搭配 EC2 上的 Windows 容器。	了解如何將 GMA 用於適用於 Amazon ECS 的 EC2 視窗容器	
在 Fargate 上使用群組受管理服務帳戶搭配 Linux 容器。	用gMSA於 Far Linux gate 上的容器	
建立執行具有認證的 Windows 容器的工作，以存取具有無網域群組受管理服務帳戶的 Active Directory。	使用 Amazon ECS 視窗容器搭配無網域gMSA使用 AWS CLI	

使用以下方式為 Fargate 啟動類型建立 Amazon ECS Linux 任務 AWS CLI

以下步驟會協助您使用 AWS CLI在 Amazon ECS 中設定叢集、註冊任務定義、執行 Linux 任務，以及執行其他常見案例。請使用最新版本的 AWS CLI. 如需如何升級至最新版本的詳細資訊，請參閱安裝 AWS Command Line Interface。

主題

- 必要條件
- 步驟 1：建立叢集
- 步驟 2：註冊 Linux 任務定義
- 步驟 3：列出任務定義
- 步驟 4：建立服務
- 步驟 5：列出服務
- 步驟 6：描述執行中的服務
- 步驟 7：測試
- 步驟 8：清除

必要條件

本教學課程假設已完成下列先決條件。

- 已安裝並設定最新版本的。 AWS CLI 如需有關安裝或升級的詳細資訊 AWS CLI，請參閱安裝 AWS Command Line Interface.
- 已完成「設定以使用 Amazon ECS。」中的步驟。
- 您的 AWS 使用者具有 亞馬遜 FullAccess IAM 政策範例中指定的必要許可。
- 您已建立 VPC 和安全群組。本教學課程使用託管於 Amazon ECR Public 的容器映像，因此任務必須有網際網路連線。若要將網際網路的路由提供給任務，請使用下列其中一個選項。
  - 透過具有彈性 IP 地址的 NAT 閘道使用私有子網路。
  - 使用公有子網路，然後將公有 IP 地址指派給任務。

  如需詳細資訊，請參閱 the section called “建立 Virtual Private Cloud”。

  如需有關安全群組和規則的資訊，請參閱《Amazon Virtual Private Cloud 使用者指南》中的 VPC 的預設安全群組和規則範例。
- 如果您使用私有子網路遵循本教學課程，則可以使用 Amazon ECS Exec 直接與您的容器互動並測試部署。您必須建立任務 IAM 角色才能使用 ECS Exec。如需有關任務 IAM 角色和其他先決條件的詳細資訊，請參閱使用 Amazon ECS Exec 進行偵錯。
- (選用) AWS CloudShell 這項工具可為客戶提供命令列，而不需要建立自己的 EC2 執行個體。如需詳細資訊，請參閱什麼是 AWS CloudShell？ 在《AWS CloudShell 使用者指南》中。

步驟 1：建立叢集

您的帳戶預設會得到 default 叢集。

> **Note**
>
> 使用為您提供之 default 叢集的優點是，您不必在後續的命令中指定 --cluster *cluster_name* 選項。如果您建立的是自己的叢集，不是預設叢集，您必須在打算對該叢集使用的每個命令中指定 --cluster *cluster_name*。

使用下列命令以唯一的名稱建立您自己的叢集：

```
aws ecs create-cluster --cluster-name fargate-cluster
```

輸出：

```
{
    "cluster": {
        "status": "ACTIVE",
        "defaultCapacityProviderStrategy": [],
        "statistics": [],
        "capacityProviders": [],
        "tags": [],
        "clusterName": "fargate-cluster",
        "settings": [
            {
                "name": "containerInsights",
                "value": "disabled"
            }
        ],
        "registeredContainerInstancesCount": 0,
        "pendingTasksCount": 0,
        "runningTasksCount": 0,
        "activeServicesCount": 0,
        "clusterArn": "arn:aws:ecs:region:aws_account_id:cluster/fargate-cluster"
    }
}
```

步驟 2：註冊 Linux 任務定義

您必須先註冊任務定義，才能在您的 ECS 叢集中執行任務。任務定義是分在一組的容器清單。下列範例是使用託管於 Docker Hub 的 httpd 容器映像來建立 PHP Web 應用程式的簡單任務定義。如需可用之任務定義參數的詳細資訊，請參閱「Amazon ECS 任務定義」。在本教學課程中，僅當您在私有子網路中部署任務並希望測試部署時，才需要 taskRoleArn。如 必要條件 中所述，將 taskRoleArn 取代為您為使用 ECS Exec 建立的 IAM 任務角色。

```
{
       "family": "sample-fargate",
       "networkMode": "awsvpc",
       "taskRoleArn": "arn:aws:iam::aws_account_id:role/execCommandRole",
       "containerDefinitions": [
           {
               "name": "fargate-app",
               "image": "public.ecr.aws/docker/library/httpd:latest",
               "portMappings": [
                   {
                       "containerPort": 80,
                       "hostPort": 80,
                       "protocol": "tcp"
                   }
               ],
               "essential": true,
               "entryPoint": [
                   "sh",
                   "-c"
               ],
               "command": [
                   "/bin/sh -c \"echo '<html> <head> <title>Amazon ECS Sample
 App</title> <style>body {margin-top: 40px; background-color: #333;} </style> </
head><body> <div style=color:white;text-align:center> <h1>Amazon ECS Sample App</h1>
 <h2>Congratulations!</h2> <p>Your application is now running on a container in Amazon
 ECS.</p> </div></body></html>' >  /usr/local/apache2/htdocs/index.html && httpd-
foreground\""
               ]
           }
       ],
       "requiresCompatibilities": [
           "FARGATE"
       ],
       "cpu": "256",
       "memory": "512"
```

```
}
```

將任務定義 JSON 儲存為檔案，並使用 `--cli-input-json file://path_to_file.json` 選項進行傳遞。

將 JSON 檔案用於容器定義中：

```
aws ecs register-task-definition --cli-input-json file://$HOME/tasks/fargate-task.json
```

register-task-definition 命令會在完成註冊後傳回任務定義的描述。

步驟 3：列出任務定義

您可以使用 list-task-definitions 命令隨時列出您帳戶的任務定義。這個命令的輸出會顯示 `family` 和 `revision` 值，可在呼叫 run-task 或 start-task 時一起使用。

```
aws ecs list-task-definitions
```

輸出：

```
{
    "taskDefinitionArns": [
        "arn:aws:ecs:region:aws_account_id:task-definition/sample-fargate:1"
    ]
}
```

步驟 4：建立服務

為您的帳戶註冊任務之後，您就可以在您的叢集中建立已註冊任務的服務。在此範例中，您將透過在叢集中執行之 `sample-fargate:1` 任務定義的一個執行個體建立服務。該任務需要網際網路的路由，因此您可透過兩種方法達成。一種方法是，使用透過 NAT 閘道所設定的私有子網路，而該閘道在公有子網路中具有彈性 IP 地址。另一種方法是，使用公有子網路並將公有 IP 地址指派至任務。以下提供這兩種範例。

使用私有子網路的範例。需要 `enable-execute-command` 選項才能使用 Amazon ECS Exec。

```
aws ecs create-service --cluster fargate-cluster --service-name fargate-service --
task-definition sample-fargate:1 --desired-count 1 --launch-type "FARGATE" --network-
configuration "awsvpcConfiguration={subnets=[subnet-abcd1234],securityGroups=[sg-
abcd1234]}" --enable-execute-command
```

使用公有子網路的範例。

```
aws ecs create-service --cluster fargate-cluster --service-name fargate-service --
task-definition sample-fargate:1 --desired-count 1 --launch-type "FARGATE" --network-
configuration "awsvpcConfiguration={subnets=[subnet-abcd1234],securityGroups=[sg-
abcd1234],assignPublicIp=ENABLED}"
```

create-service 命令會在完成註冊後傳回任務定義的描述。

步驟 5：列出服務

列出您叢集的服務。您應該會看到您在先前一節所建立的服務。您可以記下這個命令傳回的服務名稱或完整的 ARN，稍後將用以描述服務。

```
aws ecs list-services --cluster fargate-cluster
```

輸出：

```
{
    "serviceArns": [
        "arn:aws:ecs:region:aws_account_id:service/fargate-cluster/fargate-service"
    ]
}
```

步驟 6：描述執行中的服務

使用之前擷取的服務名稱來描述服務，以取得任務的詳細資訊。

```
aws ecs describe-services --cluster fargate-cluster --services fargate-service
```

如果成功，這會傳回服務失敗和服務的描述。例如，在 `services` 區段中，您將可以找到部署的相關資訊，例如執行中或擱置中的任務狀態。您也可以找到任務定義、網路組態和時間戳記事件的相關資訊。在失敗區段中，您將可以找到與呼叫相關聯的失敗 (如果有的話) 的相關資訊。如需故障診斷，請參閱服務事件訊息。如需服務描述的詳細資訊，請參閱描述服務。

```
{
    "services": [
        {
            "networkConfiguration": {
                "awsvpcConfiguration": {
                    "subnets": [
```

```
                    "subnet-abcd1234"
                ],
                "securityGroups": [
                    "sg-abcd1234"
                ],
                "assignPublicIp": "ENABLED"
            }
        },
        "launchType": "FARGATE",
        "enableECSManagedTags": false,
        "loadBalancers": [],
        "deploymentController": {
            "type": "ECS"
        },
        "desiredCount": 1,
        "clusterArn": "arn:aws:ecs:region:aws_account_id:cluster/fargate-cluster",
        "serviceArn": "arn:aws:ecs:region:aws_account_id:service/fargate-service",
        "deploymentConfiguration": {
            "maximumPercent": 200,
            "minimumHealthyPercent": 100
        },
        "createdAt": 1692283199.771,
        "schedulingStrategy": "REPLICA",
        "placementConstraints": [],
        "deployments": [
            {
                "status": "PRIMARY",
                "networkConfiguration": {
                    "awsvpcConfiguration": {
                        "subnets": [
                            "subnet-abcd1234"
                        ],
                        "securityGroups": [
                            "sg-abcd1234"
                        ],
                        "assignPublicIp": "ENABLED"
                    }
                },
                "pendingCount": 0,
                "launchType": "FARGATE",
                "createdAt": 1692283199.771,
                "desiredCount": 1,
                "taskDefinition": "arn:aws:ecs:region:aws_account_id:task-
definition/sample-fargate:1",
```

```
                    "updatedAt": 1692283199.771,
                    "platformVersion": "1.4.0",
                    "id": "ecs-svc/9223370526043414679",
                    "runningCount": 0
                }
            ],
            "serviceName": "fargate-service",
            "events": [
                {
                    "message": "(service fargate-service) has started 2 tasks: (task
 53c0de40-ea3b-489f-a352-623bf1235f08) (task d0aec985-901b-488f-9fb4-61b991b332a3).",
                    "id": "92b8443e-67fb-4886-880c-07e73383ea83",
                    "createdAt": 1510811841.408
                },
                {
                    "message": "(service fargate-service) has started 2 tasks: (task
 b4911bee-7203-4113-99d4-e89ba457c626) (task cc5853e3-6e2d-4678-8312-74f8a7d76474).",
                    "id": "d85c6ec6-a693-43b3-904a-a997e1fc844d",
                    "createdAt": 1510811601.938
                },
                {
                    "message": "(service fargate-service) has started 2 tasks: (task
 cba86182-52bf-42d7-9df8-b744699e6cfc) (task f4c1ad74-a5c6-4620-90cf-2aff118df5fc).",
                    "id": "095703e1-0ca3-4379-a7c8-c0f1b8b95ace",
                    "createdAt": 1510811364.691
                }
            ],
            "runningCount": 0,
            "status": "ACTIVE",
            "serviceRegistries": [],
            "pendingCount": 0,
            "createdBy": "arn:aws:iam::aws_account_id:user/user_name",
            "platformVersion": "LATEST",
            "placementStrategy": [],
            "propagateTags": "NONE",
            "roleArn": "arn:aws:iam::aws_account_id:role/aws-service-role/
ecs.amazonaws.com/AWSServiceRoleForECS",
            "taskDefinition": "arn:aws:ecs:region:aws_account_id:task-definition/
sample-fargate:1"
        }
    ],
    "failures": []
}
```

步驟 7：測試

使用公有子網路部署的測試任務

描述服務中的任務，以便取得任務的彈性網路介面 (ENI)。

首先，要取得任務 ARN。

```
aws ecs list-tasks --cluster fargate-cluster --service fargate-service
```

輸出包含任務 ARN。

```
{
    "taskArns": [
        "arn:aws:ecs:us-east-1:123456789012:task/fargate-service/EXAMPLE
    ]
}
```

描述任務並找到 ENI ID。將任務 ARN 用於 tasks 參數。

```
aws ecs describe-tasks --cluster fargate-cluster --tasks arn:aws:ecs:us-
east-1:123456789012:task/service/EXAMPLE
```

連接資訊將在輸出中列出。

```
{
    "tasks": [
        {
            "attachments": [
                {
                    "id": "d9e7735a-16aa-4128-bc7a-b2d5115029e9",
                    "type": "ElasticNetworkInterface",
                    "status": "ATTACHED",
                    "details": [
                        {
                            "name": "subnetId",
                            "value": "subnetabcd1234"
                        },
                        {
                            "name": "networkInterfaceId",
                            "value": "eni-0fa40520aeEXAMPLE"
                        },
```

```
                    ]
                }
…
}
```

描述 ENI 以獲得公有 IP 地址。

```
aws ec2 describe-network-interfaces --network-interface-id  eni-0fa40520aeEXAMPLE
```

公有 IP 地址會在輸出中。

```
{
    "NetworkInterfaces": [
        {
            "Association": {
                "IpOwnerId": "amazon",
                "PublicDnsName": "ec2-34-229-42-222.compute-1.amazonaws.com",
                "PublicIp": "198.51.100.2"
            },
…
}
```

在 Web 瀏覽器中輸入公有 IP 地址，然後您應該會看到顯示 Amazon ECS 範例應用程式的網頁。

使用私有子網路部署的測試任務

描述任務並找出 managedAgents，以驗證 ExecuteCommandAgent 是否正在執行。記下 privateIPv4Address 以供稍後使用。

```
aws ecs describe-tasks --cluster fargate-cluster --tasks arn:aws:ecs:us-
east-1:123456789012:task/fargate-service/EXAMPLE
```

受管代理程式資訊將在輸出中列出。

```
{
     "tasks": [
        {
            "attachments": [
                {
                    "id": "d9e7735a-16aa-4128-bc7a-b2d5115029e9",
                    "type": "ElasticNetworkInterface",
                    "status": "ATTACHED",
```

```
                    "details": [
                        {
                            "name": "subnetId",
                            "value": "subnetabcd1234"
                        },
                        {
                            "name": "networkInterfaceId",
                            "value": "eni-0fa40520aeEXAMPLE"
                        },
                        {
                            "name": "privateIPv4Address",
                            "value": "10.0.143.156"
                        }
                    ]
                }
            ],
    ...
    "containers": [
        {
        ...
       "managedAgents": [
                        {
                            "lastStartedAt": "2023-08-01T16:10:13.002000+00:00",
                            "name": "ExecuteCommandAgent",
                            "lastStatus": "RUNNING"
                        }
                ],
        ...
    }
```

驗證 ExecuteCommandAgent 正在執行之後，您可以執行下列命令，以在任務中的容器上執行互動式 Shell。

```
aws ecs execute-command --cluster fargate-cluster \
    --task  arn:aws:ecs:us-east-1:123456789012:task/fargate-service/EXAMPLE  \
    --container  fargate-app \
    --interactive \
    --command "/bin/sh"
```

互動式 Shell 執行後，執行以下命令來安裝 cURL。

```
apt update
```

```
apt install curl
```

安裝 cURL 後，使用您先前獲得的私有 IP 地址執行以下命令。

```
 curl 10.0.143.156
```

您應該會看到相當於 Amazon ECS 範例應用程式網頁的 HTML。

```
<html>
    <head>
     <title>Amazon ECS Sample App</title>
     <style>body {margin-top: 40px; background-color: #333;} </style>
    </head>
      <body>
      <div style=color:white;text-align:center>
      <h1>Amazon ECS Sample App</h1>
      <h2>Congratulations!</h2> <p>Your application is now running on a container in
 Amazon ECS.</p>
      </div>
      </body>
</html>
```

步驟 8：清除

完成此教學課程時，建議您清除相關聯的資源，以免未使用的資源產生費用。

刪除服務。

```
aws ecs delete-service --cluster fargate-cluster --service fargate-service --force
```

刪除叢集。

```
aws ecs delete-cluster --cluster fargate-cluster
```

為 Fargate 啟動類型建立 Amazon ECS 視窗任務 AWS CLI

以下步驟會協助您使用 AWS CLI在 Amazon ECS 中設定叢集、註冊任務定義、執行 Windows 任務，以及執行其他常見案例。請務必使用最新版本的 AWS CLI。如需如何升級至最新版本的詳細資訊，請參閱安裝 AWS Command Line Interface。

主題

- 必要條件
- 步驟 1：建立叢集
- 步驟 2：註冊 Windows 任務定義
- 步驟 3：列出任務定義
- 步驟 4：建立服務
- 步驟 5：列出服務
- 步驟 6：描述執行中的服務
- 步驟 7：清除

必要條件

本教學課程假設已完成下列先決條件。

- 已安裝並設定最新版本的。 AWS CLI 如需有關安裝或升級的詳細資訊 AWS CLI，請參閱安裝 AWS Command Line Interface.
- 已完成「設定以使用 Amazon ECS。」中的步驟。
- 您的 AWS 使用者具有 亞馬遜 FullAccess IAM 政策範例中指定的必要許可。
- 您已建立 VPC 和安全群組。本教學課程使用託管於 Docker Hub 的容器映像，因此任務必須有網際網路連線。若要將網際網路的路由提供給任務，請使用下列其中一個選項。
  - 透過具有彈性 IP 地址的 NAT 閘道使用私有子網路。
  - 使用公有子網路，然後將公有 IP 地址指派給任務。

  如需詳細資訊，請參閱 the section called “建立 Virtual Private Cloud”。

  如需有關安全群組和規則的資訊，請參閱《Amazon Virtual Private Cloud 使用者指南》中的 VPC 的預設安全群組和規則範例。
- (選用) AWS CloudShell 這項工具可為客戶提供命令列，而不需要建立自己的 EC2 執行個體。如需詳細資訊，請參閱什麼是 AWS CloudShell？ 在《AWS CloudShell 使用者指南》中。

步驟 1：建立叢集

您的帳戶預設會得到 `default` 叢集。

> **Note**
>
> 使用為您提供之 `default` 叢集的優點是，您不必在後續的命令中指定 `--cluster` *`cluster_name`* 選項。如果您建立的是自己的叢集，不是預設叢集，您必須在打算對該叢集使用的每個命令中指定 `--cluster` *`cluster_name`*。

使用下列命令以唯一的名稱建立您自己的叢集：

```
aws ecs create-cluster --cluster-name fargate-cluster
```

輸出：

```
{
    "cluster": {
        "status": "ACTIVE",
        "statistics": [],
        "clusterName": "fargate-cluster",
        "registeredContainerInstancesCount": 0,
        "pendingTasksCount": 0,
        "runningTasksCount": 0,
        "activeServicesCount": 0,
        "clusterArn": "arn:aws:ecs:region:aws_account_id:cluster/fargate-cluster"
    }
}
```

步驟 2：註冊 Windows 任務定義

您必須先註冊任務定義，才能在您的 Amazon ECS 叢集中執行 Windows 任務。任務定義是分在一組的容器清單。以下範例是建立 Web 應用程式的簡單任務定義。如需可用之任務定義參數的詳細資訊，請參閱「Amazon ECS 任務定義」。

```
{
    "containerDefinitions": [
        {
            "command": ["New-Item -Path C:\\inetpub\\wwwroot\\index.html -Type file
 -Value '<html> <head> <title>Amazon ECS Sample App</title> <style>body {margin-top:
 40px; background-color: #333;} </style> </head><body> <div style=color:white;text-
align:center> <h1>Amazon ECS Sample App</h1> <h2>Congratulations!</h2> <p>Your
 application is now running on a container in Amazon ECS.</p>'; C:\\ServiceMonitor.exe
 w3svc"],
```

```
            "entryPoint": [
                "powershell",
                "-Command"
            ],
            "essential": true,
            "cpu": 2048,
            "memory": 4096,
            "image": "mcr.microsoft.com/windows/servercore/iis:windowsservercore-
ltsc2019",
            "name": "sample_windows_app",
            "portMappings": [
                {
                    "hostPort": 80,
                    "containerPort": 80,
                    "protocol": "tcp"
                }
            ]
        }
    ],
    "memory": "4096",
    "cpu": "2048",
    "networkMode": "awsvpc",
    "family": "windows-simple-iis-2019-core",
    "executionRoleArn": "arn:aws:iam::012345678910:role/ecsTaskExecutionRole",
    "runtimePlatform": {"operatingSystemFamily": "WINDOWS_SERVER_2019_CORE"},
    "requiresCompatibilities": ["FARGATE"]
}
```

上述範例 JSON 可以透過兩種方式傳遞給：您可以將工作定義 JSON 儲存為檔案，並使用`--cli-input-json file://path_to_file.json`選項來傳遞。 AWS CLI

將 JSON 檔案用於容器定義中：

```
aws ecs register-task-definition --cli-input-json file://$HOME/tasks/fargate-task.json
```

register-task-definition 命令會在完成註冊後傳回任務定義的描述。

步驟 3：列出任務定義

您可以使用 list-task-definitions 命令隨時列出您帳戶的任務定義。這個命令的輸出會顯示 `family` 和 `revision` 值，可在呼叫 run-task 或 start-task 時一起使用。

```
aws ecs list-task-definitions
```

輸出：

```
{
    "taskDefinitionArns": [
        "arn:aws:ecs:region:aws_account_id:task-definition/sample-fargate-windows:1"
    ]
}
```

步驟 4：建立服務

為您的帳戶註冊任務之後，您就可以在您的叢集中建立已註冊任務的服務。在此範例中，您將透過在叢集中執行之 sample-fargate:1 任務定義的一個執行個體建立服務。該任務需要網際網路的路由，因此您可透過兩種方法達成。一種方法是，使用透過 NAT 閘道所設定的私有子網路，而該閘道在公有子網路中具有彈性 IP 地址。另一種方法是，使用公有子網路並將公有 IP 地址指派至任務。以下提供這兩種範例。

使用私有子網路的範例。

```
aws ecs create-service --cluster fargate-cluster --service-name fargate-service
 --task-definition sample-fargate-windows:1 --desired-count 1 --launch-type
 "FARGATE" --network-configuration "awsvpcConfiguration={subnets=[subnet-
abcd1234],securityGroups=[sg-abcd1234]}"
```

使用公有子網路的範例。

```
aws ecs create-service --cluster fargate-cluster --service-name fargate-service
 --task-definition sample-fargate-windows:1 --desired-count 1 --launch-type
 "FARGATE" --network-configuration "awsvpcConfiguration={subnets=[subnet-
abcd1234],securityGroups=[sg-abcd1234],assignPublicIp=ENABLED}"
```

create-service 命令會在完成註冊後傳回任務定義的描述。

步驟 5：列出服務

列出您叢集的服務。您應該會看到您在先前一節所建立的服務。您可以記下這個命令傳回的服務名稱或完整的 ARN，稍後將用以描述服務。

```
aws ecs list-services --cluster fargate-cluster
```

輸出：

```
{
    "serviceArns": [
        "arn:aws:ecs:region:aws_account_id:service/fargate-service"
    ]
}
```

步驟 6：描述執行中的服務

使用之前擷取的服務名稱來描述服務，以取得任務的詳細資訊。

```
aws ecs describe-services --cluster fargate-cluster --services fargate-service
```

如果成功，這會傳回服務失敗和服務的描述。例如，在服務區段中，您將可以找到部署的相關資訊，例如執行中或擱置中的任務狀態。您也可以找到任務定義、網路組態和時間戳記事件的相關資訊。在失敗區段中，您將可以找到與呼叫相關聯的失敗 (如果有的話) 的相關資訊。如需故障診斷，請參閱服務事件訊息。如需服務描述的詳細資訊，請參閱描述服務。

```
{
    "services": [
        {
            "status": "ACTIVE",
            "taskDefinition": "arn:aws:ecs:region:aws_account_id:task-definition/
sample-fargate-windows:1",
            "pendingCount": 2,
            "launchType": "FARGATE",
            "loadBalancers": [],
            "roleArn": "arn:aws:iam::aws_account_id:role/aws-service-role/
ecs.amazonaws.com/AWSServiceRoleForECS",
            "placementConstraints": [],
            "createdAt": 1510811361.128,
            "desiredCount": 2,
            "networkConfiguration": {
                "awsvpcConfiguration": {
                    "subnets": [
                        "subnet-abcd1234"
                    ],
                    "securityGroups": [
                        "sg-abcd1234"
                    ],
                    "assignPublicIp": "DISABLED"
```

```
                }
            },
            "platformVersion": "LATEST",
            "serviceName": "fargate-service",
            "clusterArn": "arn:aws:ecs:region:aws_account_id:cluster/fargate-cluster",
            "serviceArn": "arn:aws:ecs:region:aws_account_id:service/fargate-service",
            "deploymentConfiguration": {
                "maximumPercent": 200,
                "minimumHealthyPercent": 100
            },
            "deployments": [
                {
                    "status": "PRIMARY",
                    "networkConfiguration": {
                        "awsvpcConfiguration": {
                            "subnets": [
                                "subnet-abcd1234"
                            ],
                            "securityGroups": [
                                "sg-abcd1234"
                            ],
                            "assignPublicIp": "DISABLED"
                        }
                    },
                    "pendingCount": 2,
                    "launchType": "FARGATE",
                    "createdAt": 1510811361.128,
                    "desiredCount": 2,
                    "taskDefinition": "arn:aws:ecs:region:aws_account_id:task-
definition/sample-fargate-windows:1",
                    "updatedAt": 1510811361.128,
                    "platformVersion": "0.0.1",
                    "id": "ecs-svc/9223370526043414679",
                    "runningCount": 0
                }
            ],
            "events": [
                {
                    "message": "(service fargate-service) has started 2 tasks: (task
 53c0de40-ea3b-489f-a352-623bf1235f08) (task d0aec985-901b-488f-9fb4-61b991b332a3).",
                    "id": "92b8443e-67fb-4886-880c-07e73383ea83",
                    "createdAt": 1510811841.408
                },
                {
```

```
                    "message": "(service fargate-service) has started 2 tasks: (task
 b4911bee-7203-4113-99d4-e89ba457c626) (task cc5853e3-6e2d-4678-8312-74f8a7d76474).",
                    "id": "d85c6ec6-a693-43b3-904a-a997e1fc844d",
                    "createdAt": 1510811601.938
                },
                {
                    "message": "(service fargate-service) has started 2 tasks: (task
 cba86182-52bf-42d7-9df8-b744699e6cfc) (task f4c1ad74-a5c6-4620-90cf-2aff118df5fc).",
                    "id": "095703e1-0ca3-4379-a7c8-c0f1b8b95ace",
                    "createdAt": 1510811364.691
                }
            ],
            "runningCount": 0,
            "placementStrategy": []
        }
    ],
    "failures": []
}
```

步驟 7：清除

完成此教學課程時，建議您清除相關聯的資源，以免未使用的資源產生費用。

刪除服務。

```
aws ecs delete-service --cluster fargate-cluster --service fargate-service --force
```

刪除叢集。

```
aws ecs delete-cluster --cluster fargate-cluster
```

為 EC2 啟動類型建立 Amazon ECS 任務，並使用 AWS CLI

以下步驟會協助您使用 AWS CLI在 Amazon ECS 中設定叢集、註冊任務定義、執行任務，以及執行其他常見案例。請使用最新版本的 AWS CLI. 如需如何升級至最新版本的詳細資訊，請參閱安裝 AWS Command Line Interface。

主題

- 必要條件

- 步驟 1：建立叢集
- 步驟 2：使用 Amazon ECS AMI 啟動執行個體
- 步驟 3：列出容器執行個體
- 步驟 4：描述您的容器執行個體
- 步驟 5：註冊任務定義
- 步驟 6：列出任務定義
- 步驟 7：執行任務
- 步驟 8：列出任務
- 步驟 9：描述執行中的任務

必要條件

本教學課程假設已完成下列先決條件：

- 已安裝並設定最新版本的。 AWS CLI 如需有關安裝或升級的詳細資訊 AWS CLI，請參閱安裝 AWS Command Line Interface.
- 已完成「設定以使用 Amazon ECS。」中的步驟。
- 您的 AWS 使用者具有 亞馬遜 FullAccess IAM 政策範例中指定的必要許可。
- 您已建立 VPC 和安全群組。如需詳細資訊，請參閱 the section called “建立 Virtual Private Cloud”。
- (選用) AWS CloudShell 這項工具可為客戶提供命令列，而不需要建立自己的 EC2 執行個體。如需詳細資訊，請參閱什麼是 AWS CloudShell？ 在《AWS CloudShell 使用者指南》中。

步驟 1：建立叢集

當您啟動第一個容器執行個體時，您的帳戶預設會得到 `default` 叢集。

> **Note**
>
> 使用為您提供之 `default` 叢集的優點是，您不必在後續的命令中指定 `--cluster` *`cluster_name`* 選項。如果您建立的是自己的叢集，不是預設叢集，您必須在打算對該叢集使用的每個命令中指定 `--cluster` *`cluster_name`*。

使用下列命令以唯一的名稱建立您自己的叢集：

```
aws ecs create-cluster --cluster-name MyCluster
```

輸出：

```
{
    "cluster": {
        "clusterName": "MyCluster",
        "status": "ACTIVE",
        "clusterArn": "arn:aws:ecs:region:aws_account_id:cluster/MyCluster"
    }
}
```

步驟 2：使用 Amazon ECS AMI 啟動執行個體

您的叢集中必須要有 Amazon ECS 容器執行個體，才能在其中執行任務。如果您的叢集中沒有任何容器執行個體，請參閱 啟動 Amazon ECS Linux 容器執行個體 以取得詳細資訊。

步驟 3：列出容器執行個體

在您啟動容器執行個體後，Amazon ECS 代理會於幾分鐘內在您的預設叢集註冊執行個體。您可以執行下列命令列出叢集中的容器執行個體：

```
aws ecs list-container-instances --cluster default
```

輸出：

```
{
    "containerInstanceArns": [
        "arn:aws:ecs:us-east-1:aws_account_id:container-instance/container_instance_ID"
    ]
}
```

步驟 4：描述您的容器執行個體

在您取得容器執行個體的 ARN 或 ID 之後，您可以使用 describe-container-instances 命令取得執行個體的實用資訊，例如剩餘的和已註冊的 CPU 與記憶體資源。

```
aws ecs describe-container-instances --cluster default --container-
instances container_instance_ID
```

輸出：


```
{
    "failures": [],
    "containerInstances": [
        {
            "status": "ACTIVE",
            "registeredResources": [
                {
                    "integerValue": 1024,
                    "longValue": 0,
                    "type": "INTEGER",
                    "name": "CPU",
                    "doubleValue": 0.0
                },
                {
                    "integerValue": 995,
                    "longValue": 0,
                    "type": "INTEGER",
                    "name": "MEMORY",
                    "doubleValue": 0.0
                },
                {
                    "name": "PORTS",
                    "longValue": 0,
                    "doubleValue": 0.0,
                    "stringSetValue": [
                        "22",
                        "2376",
                        "2375",
                        "51678"
                    ],
                    "type": "STRINGSET",
                    "integerValue": 0
                },
                {
                    "name": "PORTS_UDP",
                    "longValue": 0,
                    "doubleValue": 0.0,
                    "stringSetValue": [],
                    "type": "STRINGSET",
                    "integerValue": 0
                }
            ],
```

```
            "ec2InstanceId": "instance_id",
            "agentConnected": true,
            "containerInstanceArn": "arn:aws:ecs:us-west-2:aws_account_id:container-
instance/container_instance_ID",
            "pendingTasksCount": 0,
            "remainingResources": [
                {
                    "integerValue": 1024,
                    "longValue": 0,
                    "type": "INTEGER",
                    "name": "CPU",
                    "doubleValue": 0.0
                },
                {
                    "integerValue": 995,
                    "longValue": 0,
                    "type": "INTEGER",
                    "name": "MEMORY",
                    "doubleValue": 0.0
                },
                {
                    "name": "PORTS",
                    "longValue": 0,
                    "doubleValue": 0.0,
                    "stringSetValue": [
                        "22",
                        "2376",
                        "2375",
                        "51678"
                    ],
                    "type": "STRINGSET",
                    "integerValue": 0
                },
                {
                    "name": "PORTS_UDP",
                    "longValue": 0,
                    "doubleValue": 0.0,
                    "stringSetValue": [],
                    "type": "STRINGSET",
                    "integerValue": 0
                }
            ],
            "runningTasksCount": 0,
            "attributes": [
```

```
                {
                    "name": "com.amazonaws.ecs.capability.privileged-container"
                },
                {
                    "name": "com.amazonaws.ecs.capability.docker-remote-api.1.17"
                },
                {
                    "name": "com.amazonaws.ecs.capability.docker-remote-api.1.18"
                },
                {
                    "name": "com.amazonaws.ecs.capability.docker-remote-api.1.19"
                },
                {
                    "name": "com.amazonaws.ecs.capability.logging-driver.json-file"
                },
                {
                    "name": "com.amazonaws.ecs.capability.logging-driver.syslog"
                }
            ],
            "versionInfo": {
                "agentVersion": "1.5.0",
                "agentHash": "b197edd",
                "dockerVersion": "DockerVersion: 1.7.1"
            }
        }
    ]
}
```

您也可以找到 Amazon EC2 執行個體 ID，用來在 Amazon EC2 主控台中監控執行個體，或搭配 aws ec2 describe-instances --instance-id ***instance_id*** 命令使用。

步驟 5：註冊任務定義

您必須先註冊任務定義，才能在您的 ECS 叢集中執行任務。任務定義是分在一組的容器清單。以下範例是一種簡單的任務定義，使用 Docker Hub 的 busybox 映像，只睡眠 360 秒。如需可用之任務定義參數的詳細資訊，請參閱「Amazon ECS 任務定義」。

```
{
    "containerDefinitions": [
        {
            "name": "sleep",
            "image": "busybox",
```

```
            "cpu": 10,
            "command": [
                "sleep",
                "360"
            ],
            "memory": 10,
            "essential": true
        }
    ],
    "family": "sleep360"
}
```

上述範例 JSON 可以透過兩種方式傳遞給：您可以將工作定義 JSON 儲存為檔案，並使用`--cli-input-json file://path_to_file.json`選項來傳遞。 AWS CLI 或者，您可以逸出 JSON 中的引號，然後在命令行上傳遞 JSON 容器定義，如下範例所示。如果您選擇用命令列傳送容器定義，您的命令需要使用額外的 `--family` 參數，以保留多個彼此相關聯的任務定義版本。

將 JSON 檔案用於容器定義中：

```
aws ecs register-task-definition --cli-input-json file://$HOME/tasks/sleep360.json
```

將 JSON 字串用於容器定義中：

```
aws ecs register-task-definition --family sleep360 --container-definitions "[{\"name
\":\"sleep\",\"image\":\"busybox\",\"cpu\":10,\"command\":[\"sleep\",\"360\"],\"memory
\":10,\"essential\":true}]"
```

register-task-definition 會在完成註冊後，傳回任務定義的描述。

```
{
    "taskDefinition": {
        "volumes": [],
        "taskDefinitionArn": "arn:aws:ec2:us-east-1:aws_account_id:task-definition/
sleep360:1",
        "containerDefinitions": [
            {
                "environment": [],
                "name": "sleep",
                "mountPoints": [],
                "image": "busybox",
                "cpu": 10,
```

```
                "portMappings": [],
                "command": [
                    "sleep",
                    "360"
                ],
                "memory": 10,
                "essential": true,
                "volumesFrom": []
            }
        ],
        "family": "sleep360",
        "revision": 1
    }
}
```

步驟 6：列出任務定義

您可以使用 list-task-definitions 命令隨時列出您帳戶的任務定義。這個命令的輸出會顯示 `family` 和 `revision` 值，可在呼叫 run-task 或 start-task 時一起使用。

```
aws ecs list-task-definitions
```

輸出：

```
{
    "taskDefinitionArns": [
        "arn:aws:ec2:us-east-1:aws_account_id:task-definition/sleep300:1",
        "arn:aws:ec2:us-east-1:aws_account_id:task-definition/sleep300:2",
        "arn:aws:ec2:us-east-1:aws_account_id:task-definition/sleep360:1",
        "arn:aws:ec2:us-east-1:aws_account_id:task-definition/wordpress:3",
        "arn:aws:ec2:us-east-1:aws_account_id:task-definition/wordpress:4",
        "arn:aws:ec2:us-east-1:aws_account_id:task-definition/wordpress:5",
        "arn:aws:ec2:us-east-1:aws_account_id:task-definition/wordpress:6"
    ]
}
```

步驟 7：執行任務

在您註冊帳戶任務，並啟動已向叢集註冊的容器執行個體之後，您就可以在叢集中執行註冊的任務。在此範例中，您會在您的預設叢集中放置 `sleep360:1` 任務定義的單一執行個體。

```
aws ecs run-task --cluster default --task-definition sleep360:1 --count 1
```

輸出：

```
{
    "tasks": [
        {
            "taskArn": "arn:aws:ecs:us-east-1:aws_account_id:task/task_ID",
            "overrides": {
                "containerOverrides": [
                    {
                        "name": "sleep"
                    }
                ]
            },
            "lastStatus": "PENDING",
            "containerInstanceArn": "arn:aws:ecs:us-east-1:aws_account_id:container-
instance/container_instance_ID",
            "clusterArn": "arn:aws:ecs:us-east-1:aws_account_id:cluster/default",
            "desiredStatus": "RUNNING",
            "taskDefinitionArn": "arn:aws:ecs:us-east-1:aws_account_id:task-definition/
sleep360:1",
            "containers": [
                {
                    "containerArn": "arn:aws:ecs:us-
east-1:aws_account_id:container/container_ID",
                    "taskArn": "arn:aws:ecs:us-east-1:aws_account_id:task/task_ID",
                    "lastStatus": "PENDING",
                    "name": "sleep"
                }
            ]
        }
    ]
}
```

步驟 8：列出任務

列出您叢集的任務。您應該會看到您在先前一節所執行的任務。您可以記下這個命令傳回的任務 ID 或完整的 ARN，稍後用以描述任務。

```
aws ecs list-tasks --cluster default
```

輸出：

```
{
    "taskArns": [
        "arn:aws:ecs:us-east-1:aws_account_id:task/task_ID"
    ]
}
```

步驟 9：描述執行中的任務

使用之前擷取的任務 ID 來描述任務，以取得任務的詳細資訊。

```
aws ecs describe-tasks --cluster default --task task_ID
```

輸出：

```
{
    "failures": [],
    "tasks": [
        {
            "taskArn": "arn:aws:ecs:us-east-1:aws_account_id:task/task_ID",
            "overrides": {
                "containerOverrides": [
                    {
                        "name": "sleep"
                    }
                ]
            },
            "lastStatus": "RUNNING",
            "containerInstanceArn": "arn:aws:ecs:us-east-1:aws_account_id:container-
instance/container_instance_ID",
            "clusterArn": "arn:aws:ecs:us-east-1:aws_account_id:cluster/default",
            "desiredStatus": "RUNNING",
            "taskDefinitionArn": "arn:aws:ecs:us-east-1:aws_account_id:task-definition/
sleep360:1",
            "containers": [
                {
                    "containerArn": "arn:aws:ecs:us-
east-1:aws_account_id:container/container_ID",
                    "taskArn": "arn:aws:ecs:us-east-1:aws_account_id:task/task_ID",
                    "lastStatus": "RUNNING",
                    "name": "sleep",
```

```
                    "networkBindings": []
                }
            ]
        }
    ]
}
```

設定 Amazon ECS 以監聽 CloudWatch 事件事件

了解如何設定簡單的 Lambda 函數，以偵聽任務事件並將其寫入 CloudWatch 日誌記錄串流。

先決條件：設定測試叢集

如果您沒有可供擷取事件的執行中叢集，請遵循「the section called “建立 Fargate 啟動類型的叢集”」中的步驟建立叢集。在此教學課程的最後，您在此叢集上執行任務，以測試您的 Lambda 函數設定是否正確。

步驟 1：建立 Lambda 函數

在此程序中，您要建立一個簡單的 Lambda 函數，做為 Amazon ECS 事件資料流訊息的目標。

1. 請在以下位置開啟 AWS Lambda 主控台。 https://console.aws.amazon.com/lambda/
2. 選擇 Create function (建立函數)。
3. 在 Author from scratch (從頭開始撰寫) 畫面上，執行下列操作：
   a. 對於 Name (名稱)，輸入值。
   b. 針對 Runtime (執行時間)，選擇您的 Python 版本，例如：Python 3.9。
   c. 針對 Role (角色)，選擇 Create a new role with basic Lambda permissions (建立具備基本 Lambda 許可的新角色)。
4. 選擇建立函數。
5. 在 Function code (函數程式碼) 區段中，編輯範本程式碼以符合下列範例：

```
import json

def lambda_handler(event, context):
    if event["source"] != "aws.ecs":
       raise ValueError("Function only supports input from events with a source
 type of: aws.ecs")
```

```
    print('Here is the event:')
    print(json.dumps(event))
```

這是一個簡單的 Python 3.9 函數，此函數列印 Amazon ECS 所傳送的事件。如果一切都設定正確，在本教學課程結束時，您會看到事件詳細資料會出現在與此 Lambda 函數相關聯的 CloudWatch 記錄日誌串流中。

6. 選擇儲存。

步驟 2：註冊事件規則

接下來，您可以建立 CloudWatch 事件事件規則，以擷取來自 Amazon ECS 叢集的任務事件。此規則會擷取來自所定義的帳戶內所有叢集的所有事件。任務訊息本身包含事件來源的資訊 (包含其所在的叢集)，從而能以程式設計方式篩選和排序事件。

> **Note**
>
> 當您使用建立事件規則時，主控台會自動新增必要的 IAM 許可，以授予 CloudWatch 事件權限以呼叫 Lambda 函數。 AWS Management Console 如果您要使用建立事件規則 AWS CLI，則需要明確授與此權限。如需詳細資訊，請參閱 Amazon CloudWatch 事件使用者指南中的事件和事件模式。

將事件路由至 Lambda 函數

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導覽窗格上，選擇 Events (事件)、Rules (規則)、Create rule (建立規則)。
3. 針對 Event Source (事件來源)，選擇 ECS 做為事件來源。根據預設，此規則套用至所有 Amazon ECS 群組的 Amazon ECS 事件。或者，您可以選取特定事件或特定 Amazon ECS 群組。
4. 對於 Targets (目標)，選擇 Add target (新增目標)，並對於 Target type (目標類型)，選擇 Lambda function (Lambda 函數)，然後選取您的 Lambda 函數。
5. 選擇設定詳細資訊。
6. 針對 Rule definition (規則定義)，輸入您規則的名稱和描述，然後選擇 Create rule (建立規則)。

步驟 3：建立任務定義

建立任務定義。

1. 開啟主控台，網址為 https://console.aws.amazon.com/ecs/v2。
2. 在導覽窗格中，選擇 Task Definitions (任務定義)。
3. 選擇 Create new Task Definition (建立新任務定義)，以及 Create new revision with JSON (使用 JSON 建立新修訂版)。
4. 複製下列任務定義範例並貼到方塊中，然後選擇 Save (儲存)。

```
{
    "containerDefinitions": [
        {
            "entryPoint": [
                "sh",
                "-c"
            ],
            "portMappings": [
                {
                    "hostPort": 80,
                    "protocol": "tcp",
                    "containerPort": 80
                }
            ],
            "command": [
                "/bin/sh -c \"echo '<html> <head> <title>Amazon ECS Sample
 App</title> <style>body {margin-top: 40px; background-color: #333;} </style> </
head><body> <div style=color:white;text-align:center> <h1>Amazon ECS Sample App</
h1> <h2>Congratulations!</h2> <p>Your application is now running on a container in
 Amazon ECS.</p> </div></body></html>' >  /usr/local/apache2/htdocs/index.html &&
 httpd-foreground\""
            ],
            "cpu": 10,
            "memory": 300,
            "image": "httpd:2.4",
            "name": "simple-app"
        }
    ],
    "family": "console-sample-app-static"
}
```

5. 選擇建立。

步驟 4：測試您的規則

最後，您會建立一個 CloudWatch 事件事件規則，以擷取來自 Amazon ECS 叢集的任務事件。此規則會擷取來自所定義的帳戶內所有叢集的所有事件。任務訊息本身包含事件來源的資訊 (包含其所在的叢集)，從而能以程式設計方式篩選和排序事件。

測試規則

1. 開啟主控台，網址為 https://console.aws.amazon.com/ecs/v2。
2. 選擇 Task definitions (任務定義)。
3. 選擇 console-sample-app-static，然後選擇 Deploy (部署)、Run new task (執行新任務)。
4. 在 Cluster (叢集) 欄位中選擇預設，然後選擇 Deploy (部署)。
5. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
6. 在導覽窗格上，選擇 Logs (日誌)，然後選取 Lambda 函數的日誌群組 (例如，/aws/lambda/*my-function*)。
7. 選取日誌串流，以檢視事件資料。

發送 Amazon 簡單通知服務提醒 Amazon ECS 任務停止事件

設定 Amazon EventBridge 事件規則，該規則僅擷取任務因為其中一個基本容器已終止而停止執行的任務事件。事件只會將具有特定 `stoppedReason` 屬性的任務事件傳送至指定的 Amazon SNS 主題。

先決條件：設定測試叢集

如果您沒有可供擷取事件的執行中叢集，請遵循使用 AWS Fargate上的 Linux 容器來開始使用主控台中的步驟建立叢集。在本教學結束時，您會在此叢集上執行任務，以測試您是否已正確設定 Amazon SNS 主題和 EventBridge 規則。

先決條件：設定 Amazon SNS 的許可

要允許發布 EventBridge 到 Amazon SNS 主題，請使用 aws SNS get-topic-attributes 和 aws SNS set-topic-attributes 命令。

如需有關如何建立許可的詳細資訊，請參閱《Amazon Simple Notification Service 開發人員指南》中的 Amazon SNS 許可。

新增下列許可：

```
{
  "Sid": "PublishEventsToMyTopic",
  "Effect": "Allow",
  "Principal": {
     "Service": "events.amazonaws.com"
  },
  "Action": "sns: Publish",
  "Resource": "arn:aws:sns:region:account-id:TaskStoppedAlert",
}
```

步驟 1：建立並訂閱 Amazon SNS 主題

在此教學課程中，您會設定 Amazon SNS 主題，做為新事件規則的事件目標。

如需有關如何建立和訂閱 Amazon SNS 主題的資訊，請參閱《Amazon Simple Notification Service 開發人員指南》中的 Amazon SNS 入門，並使用下表來決定要選取的選項。

選項	Value
Type	標準
名稱	TaskStopped警示
通訊協定	電子郵件
端點	您目前能存取的電子郵件地址

步驟 2：註冊事件規則

接下來，您會登錄事件規則，只擷取具有停止容器之任務的任務停止事件。

有關如何建立和訂閱 Amazon SNS 主題的詳細資訊，請參閱 Amazon 使用 EventBridge 者指南 EventBridge中的在 Amazon 中建立規則，並使用下表決定要選取的選項。

選項	Value
規則類型	具有事件模式的規則

選項	Value
事件來源	AWS 活動或 EventBridge 合作夥伴活動
事件模式	自訂模式 (JSON 編輯器)

事件模式	{
	"source":[
	"aws.ecs"
	],
	"detail-type":[
	"ECS Task State Change"
	],
	"detail":{
	"lastStatus":[
	"STOPPED"
	],
	"stoppedReason":[
	"Essential container in task exited"
	]
	}
	}

Target type (目標類型)	AWS 服務
目標	SNS 主題
主題	TaskStoppedAlert (您在步驟 1 中建立的主題)

步驟 3：測試您的規則

藉由執行在啟動後立即結束的任務，確認規則是否正常運作。如果您的事件規則設定正確，您應該會在幾分鐘內收到一封包含事件文字的電子郵件訊息。如果您有可以滿足規則需求的現有任務定義，請使用它來執行任務。如果沒有，以下步驟將引導您註冊 Fargate 任務定義並執行它。

1. 開啟主控台，網址為 https://console.aws.amazon.com/ecs/v2。
2. 在導覽窗格中，選擇 Task Definitions (任務定義)。
3. 選擇 Create new task definitio (建立新任務定義)、Create new task definition with JSON (使用 JSON 建立新的任務定義)。
4. 在 JSON 編輯工具方塊中，編輯您的 JSON 檔案，將下列內容複製至編輯器。

```
{
   "containerDefinitions":[
      {
         "command":[
            "sh",
            "-c",
            "sleep 5"
         ],
         "essential":true,
         "image":"amazonlinux:2",
         "name":"test-sleep"
      }
   ],
   "cpu":"256",
   "executionRoleArn":"arn:aws:iam::012345678910:role/ecsTaskExecutionRole",
   "family":"fargate-task-definition",
   "memory":"512",
   "networkMode":"awsvpc",
   "requiresCompatibilities":[
      "FARGATE"
   ]
}
```

5. 選擇建立。

從主控台執行任務

1. 開啟主控台，網址為 https://console.aws.amazon.com/ecs/v2。
2. 在叢集頁面，選擇您要在先決條件中建立的叢集。
3. 在 Tasks (任務) 標籤中，選擇 Run new task (執行新任務)。
4. 針對 Application type (應用程式類型)，選擇 Task (任務)。
5. 在任務定義中，選擇 fargate-task-definition。
6. 針對 Desired tasks (所需任務)，輸入要啟動的任務數目。

7. 選擇建立。

串連多行或堆疊追蹤 Amazon ECS 日誌訊息

從流利位版本 2.22.0 開始，包 AWS 括多行過濾器。多行篩選條件有助於串連最初屬於一筆內容但分割到多筆記錄或日誌明細行的日誌訊息。如需有關多行篩選條件的詳細資訊，請參閱《Fluent Bit 說明文件》。

分割日誌訊息的常見範例有：

- 堆疊追蹤。
- 在多行上列印日誌的應用程式。
- 因為日誌訊息長於指定的執行時間緩衝大小上限而分割的日誌訊息。您可以依照下列範例 GitHub：FireLens 串連部分/分割容器記錄檔中的範例，連接由容器執行階段分割的記錄訊息。

所需的 IAM 許可

您具有容器代理程式必要的 IAM 許可，可以從 Amazon ECR 提取容器映像，以及將日誌路由傳送到 CloudWatch 日誌的容器。

對於這些許可，您必須具有下列角色：

- 任務 IAM 角色。
- 任務執行 IAM 角色。

若要使用 JSON 政策編輯器來建立政策

1. 登入 AWS Management Console 並開啟身分與存取權管理主控台，網址為 https://console.aws.amazon.com/iam/。
2. 在左側的導覽窗格中，選擇 Policies (政策)。

   如果這是您第一次選擇 Policies (政策)，將會顯示 Welcome to Managed Policies (歡迎使用受管政策) 頁面。選擇 Get Started (開始使用)。
3. 在頁面頂端，選擇 Create policy (建立政策)。
4. 在政策編輯器中，選擇 JSON 選項。
5. 輸入下列 JSON 政策文件：

```
{
 "Version": "2012-10-17",
 "Statement": [{
  "Effect": "Allow",
  "Action": [
   "logs:CreateLogStream",
   "logs:CreateLogGroup",
   "logs:PutLogEvents"
  ],
  "Resource": "*"
 }]
}
```

6. 選擇下一步。

> **Note**
>
> 您可以隨時切換視覺化與 JSON 編輯器選項。不過，如果您進行變更或在視覺化編輯器中選擇下一步，IAM 就可能會調整您的政策結構，以便針對視覺化編輯器進行最佳化。如需詳細資訊，請參閱 IAM 使用者指南中的調整政策結構。

7. 在檢視與建立頁面上，為您正在建立的政策輸入政策名稱與描述 (選用)。檢視此政策中定義的許可，來查看您的政策所授予的許可。
8. 選擇 Create policy (建立政策) 儲存您的新政策。

確定何時使用多行日誌設定

以下是您在「記錄」主控台中看到的範例 CloudWatch 記錄片段，其中包含預設記錄設定。您可以查看以 `log` 開頭的行，以確定是否需要多行篩選條件。當上下文相同時，您可以使用多行日誌設置，在此示例中，上下文是「com.myproject.model。 MyProject」.

```
2022-09-20T15:47:56:595-05-00                              {"container_id":
 "82ba37cada1d44d389b03e78caf74faa-EXAMPLE", "container_name": "example-
app", "source=": "stdout", "log": ": "     at com.myproject.modele.
(MyProject.badMethod.java:22)",
    {
      "container_id":  "82ba37cada1d44d389b03e78caf74faa-EXAMPLE",
      "container_name: ": "example-app",
      "source": "stdout",
```

```
      "log": ": "     at com.myproject.model.MyProject.badMethod(MyProject.java:22)",
      "ecs_cluster": "default",
      "ecs_task_arn": "arn:aws:region:123456789012:task/default/
b23c940d29ed4714971cba72cEXAMPLE",
      "ecs_task_definition": "firelense-example-multiline:3"
     }
```

```
2022-09-20T15:47:56:595-05-00                              {"container_id":
 "82ba37cada1d44d389b03e78caf74faa-EXAMPLE", "container_name": "example-app", "stdout",
 "log": ": "     at com.myproject.modele.(MyProject.oneMoreMethod.java:18)",
    {
      "container_id":  "82ba37cada1d44d389b03e78caf74faa-EXAMPLE",
      "container_name: ": "example-app",
      "source": "stdout",
      "log": ": "     at
 com.myproject.model.MyProject.oneMoreMethod(MyProject.java:18)",
      "ecs_cluster": "default",
      "ecs_task_arn": "arn:aws:region:123456789012:task/default/
b23c940d29ed4714971cba72cEXAMPLE,
      "ecs_task_definition": "firelense-example-multiline:3"
     }
```

使用多行日誌設定後，輸出看起來類似下列範例。

```
2022-09-20T15:47:56:595-05-00                              {"container_id":
 "82ba37cada1d44d389b03e78caf74faa-EXAMPLE", "container_name": "example-app",
 "stdout",...
    {
      "container_id":  "82ba37cada1d44d389b03e78caf74faa-EXAMPLE",
      "container_name: ": "example-app",
      "source": "stdout",
      "log:    "September 20, 2022 06:41:48 Exception in thread \"main\"
 java.lang.RuntimeException: Something has gone wrong, aborting!\n
    at com.myproject.module.MyProject.badMethod(MyProject.java:22)\n    at
    at com.myproject.model.MyProject.oneMoreMethod(MyProject.java:18)
 com.myproject.module.MyProject.main(MyProject.java:6)",
      "ecs_cluster": "default",
      "ecs_task_arn": "arn:aws:region:123456789012:task/default/
b23c940d29ed4714971cba72cEXAMPLE",
      "ecs_task_definition": "firelense-example-multiline:2"
     }
```

剖析並串連選項

若要剖析日誌並串連因新行而分割的行數，您可以使用這兩個選項中的任一選項。

- 使用您自己的剖析器檔案 (包含要剖析的規則) 並串連屬於相同訊息的明細行。
- 使用 Fluent Bit 內建剖析器。如需 Fluent Bit 內建剖析器所支援的語言清單，請參閱 Fluent Bit 說明文件。

下列教學會引導您逐步演練每個使用案例的步驟。這些步驟向您展示瞭如何連接多線並將日誌發送到 Amazon。 CloudWatch您可以為您的日誌指定其他目的地。

範例：使用您建立的剖析器

在本範例中，您會完成下列步驟：

1. 建置並上傳 Fluent Bit 容器的映像。
2. 建置並上傳可執行、失敗並產生多行堆疊追蹤之試用版多行應用程式的映像。
3. 建立任務定義並執行任務。
4. 檢視日誌以確認跨多行的訊息是否顯示為串連。

建置並上傳 Fluent Bit 容器的映像

此映像將包含您指定常規運算式的剖析器檔案和參照剖析器檔案的組態文件。

1. 建立名稱為 `FluentBitDockerImage` 的資料夾。
2. 在資料夾內，建立剖析器檔案 (包含要剖析日誌的規則) 並串連屬於相同訊息的明細行。

   a. 將以下內容貼到該剖析器檔案：

```
[MULTILINE_PARSER]
    name          multiline-regex-test
    type          regex
    flush_timeout 1000
    #
    # Regex rules for multiline parsing
    # ---------------------------------
    #
```

```
    # configuration hints:
    #
    #  - first state always has the name: start_state
    #  - every field in the rule must be inside double quotes
    #
    # rules |   state name  | regex pattern                  | next state
    # ------|---------------|--------------------------------------------
    rule      "start_state"   "/(Dec \d+ \d+\:\d+\:\d+)(.*)/"  "cont"
    rule      "cont"          "/^\s+at.*/"                     "cont"
```

當您自訂 Regex 運算式模式時，我們建議您使用常規運算式編輯器來測試運算式。

b. 儲存檔案為 `parsers_multiline.conf`。

3. 在 `FluentBitDockerImage` 資料夾內，建立參照了您在前一個步驟中建立的剖析器檔案之自訂組態檔案。

   如需有關自訂組態檔案的詳細資訊，請參閱《Amazon Elastic Container Service 開發人員指南》中的指定自訂組態檔案

   a. 將以下內容貼到該檔案：

```
[SERVICE]
    flush                 1
    log_level             info
    parsers_file          /parsers_multiline.conf

[FILTER]
    name                  multiline
    match                 *
    multiline.key_content log
    multiline.parser      multiline-regex-test
```

> **Note**
>
> 您必須使用剖析器的絕對路徑。

   b. 儲存檔案為 `extra.conf`。

4. 在 `FluentBitDockerImage` 資料夾內，建立具有 Fluent Bit 映像與剖析器和您建立的組態檔案之 Dockerfile。

   a. 將以下內容貼到該檔案：

```
FROM public.ecr.aws/aws-observability/aws-for-fluent-bit:latest

ADD parsers_multiline.conf /parsers_multiline.conf
ADD extra.conf /extra.conf
```

b. 儲存檔案為 Dockerfile。

5. 使用 Dockerfile，建置內含剖析器與自訂組態檔案的自訂 Fluent Bit 映像。

> **Note**
>
> 您可以將解析器文件和配置文件放置在 Docker 映像中的任何位置/fluent-bit/etc/fluent-bit.conf，除非由 FireLens此文件路徑使用。

a. 建置映像：`docker build -t fluent-bit-multiline-image .`

其中：`fluent-bit-multiline-image` 是此範例中映像的名稱。

b. 驗證映像已正確建立：`docker images –filter reference=fluent-bit-multiline-image`

如果成功，輸出會顯示映像和 `latest` 標籤。

6. 將自訂 Fluent Bit 映像上傳至 Amazon Elastic 容器登錄檔。

a. 建立 Amazon ECR 儲存庫，以便存放映像：`aws ecr create-repository --repository-name fluent-bit-multiline-repo --region us-east-1`

其中：`fluent-bit-multiline-repo` 是此範例中儲存庫的名稱，`us-east-1` 是此範例中的區域。

輸出為您提供了新儲存庫的詳細資訊。

b. 使用來自先前輸出的 `repositoryUri` 值，標記您的映像：`docker tag` *`fluent-bit-multiline-image repositoryUri`*

範例：`docker tag fluent-bit-multiline-image xxxxxxxxxxxx.dkr.ecr.us-east-1.amazonaws.com/fluent-bit-multiline-repo`

c. 執行 Docker 影像以確認是否能正確執行：`docker images –filter reference=`*`repositoryUri`*

在輸出中，存放庫名稱會從變更 fluent-bit-multiline-repo 為repositoryUri。

d. 透過執行 aws ecr get-login-password 命令並指定您要驗證的登錄 ID 來驗證 Amazon ECR：aws ecr get-login-password | docker login --username AWS --password-stdin *registry ID*.dkr.ecr.*region*.amazonaws.com

   範例：ecr get-login-password | docker login --username AWS --password-stdin xxxxxxxxxxxx.dkr.ecr.us-east-1.amazonaws.com

   畫面上會顯示成功登入訊息。

e. 將映像推送至 Amazon ECR：docker push *registry ID*.dkr.ecr.*region*.amazonaws.com/*repository name*

   範例：docker push xxxxxxxxxxxx.dkr.ecr.us-east-1.amazonaws.com/fluent-bit-multiline-repo

建置並上傳試用版多行應用程式的映像

此映像將包含執行應用程式的 Python 指令碼檔案和樣本日誌檔案。

當您執行任務時，應用程式會模擬執行，然後失敗並建立堆疊追蹤。

1. 建立名為 multiline-app 的資料夾：mkdir multiline-app
2. 建立 Python 指令碼檔案。
   a. 在 multiline-app 資料夾中，建立檔案，並將它命名為 main.py。
   b. 將以下內容貼到該檔案：

```
import os
import time
file1 = open('/test.log', 'r')
Lines = file1.readlines()

count = 0

for i in range(10):
    print("app running normally...")
    time.sleep(1)

# Strips the newline character
```

```
for line in Lines:
    count += 1
    print(line.rstrip())
print(count)
print("app terminated.")
```

c. 儲存 main.py 檔案。

3. 建立範例日誌檔案。

a. 在 multiline-app 資料夾中，建立檔案，並將它命名為 test.log。

b. 將以下內容貼到該檔案：

```
single line...
Dec 14 06:41:08 Exception in thread "main" java.lang.RuntimeException:
 Something has gone wrong, aborting!
    at com.myproject.module.MyProject.badMethod(MyProject.java:22)
    at com.myproject.module.MyProject.oneMoreMethod(MyProject.java:18)
    at com.myproject.module.MyProject.anotherMethod(MyProject.java:14)
    at com.myproject.module.MyProject.someMethod(MyProject.java:10)
    at com.myproject.module.MyProject.main(MyProject.java:6)
another line...
```

c. 儲存 test.log 檔案。

4. 在 multiline-app 資料夾內，建立 Dockerfile。

a. 將以下內容貼到該檔案：

```
FROM public.ecr.aws/amazonlinux/amazonlinux:latest
ADD test.log /test.log

RUN yum upgrade -y && yum install -y python3

WORKDIR /usr/local/bin

COPY main.py .

CMD ["python3", "main.py"]
```

b. 儲存 Dockerfile 檔案。

5. 使用 Dockerfile，建置映像。

a. 建置映像：`docker build -t multiline-app-image .`

   其中：`multiline-app-image` 是此範例中映像的名稱。

b. 驗證映像已正確建立：`docker images –filter reference=multiline-app-image`

   如果成功，輸出會顯示映像和 `latest` 標籤。

6. 將映像上傳至 Amazon Elastic 容器登錄檔。

a. 建立 Amazon ECR 儲存庫，以便存放映像：`aws ecr create-repository --repository-name multiline-app-repo --region us-east-1`

   其中：`multiline-app-repo` 是此範例中儲存庫的名稱，`us-east-1` 是此範例中的區域。

   輸出為您提供了新儲存庫的詳細資訊。請記下 `repositoryUri` 值，因為您將在後續步驟用到它。

b. 使用來自先前輸出的 `repositoryUri` 值，標記您的映像：`docker tag *multiline-app-image repositoryUri*`

   範例：`docker tag multiline-app-image xxxxxxxxxxxx.dkr.ecr.us-east-1.amazonaws.com/multiline-app-repo`

c. 執行 Docker 影像以確認是否能正確執行：`docker images –filter reference=*repositoryUri*`

   在輸出中，儲存庫名稱會從 `multiline-app-repo` 變更為 `repositoryUri` 值。

d. 將映像推送至 Amazon ECR：`docker push *aws_account_id*.dkr.ecr.*region*.amazonaws.com/*repository name*`

   範例：`docker push *xxxxxxxxxxxx*.dkr.ecr.*us-east-1*.amazonaws.com/multiline-app-repo`

建立任務定義並執行任務

1. 建立檔案名稱為 `multiline-task-definition.json` 的任務定義檔案。
2. 將以下內容貼到該 `multiline-task-definition.json` 檔案：

```
{
    "family": "firelens-example-multiline",
    "taskRoleArn": "task role ARN,
```

```
    "executionRoleArn": "execution role ARN",
    "containerDefinitions": [
        {
            "essential": true,
            "image": "aws_account_id.dkr.ecr.us-east-1.amazonaws.com/fluent-bit-
multiline-image:latest",
            "name": "log_router",
            "firelensConfiguration": {
                "type": "fluentbit",
                "options": {
                    "config-file-type": "file",
                    "config-file-value": "/extra.conf"
                }
            },
            "memoryReservation": 50
        },
        {
            "essential": true,
            "image": "aws_account_id.dkr.ecr.us-east-1.amazonaws.com/multiline-app-
image:latest",
            "name": "app",
            "logConfiguration": {
                "logDriver": "awsfirelens",
                "options": {
                    "Name": "cloudwatch_logs",
                    "region": "us-east-1",
                    "log_group_name": "multiline-test/application",
                    "auto_create_group": "true",
                    "log_stream_prefix": "multiline-"
                }
            },
            "memoryReservation": 100
        }
    ],
    "requiresCompatibilities": ["FARGATE"],
    "networkMode": "awsvpc",
    "cpu": "256",
    "memory": "512"
}
```

取代下列在 multiline-task-definition.json 任務定義中的項目：

a. *task role ARN*

若要尋找任務角色 ARN，請前往 IAM 主控台。選擇 Roles (角色) 並尋找您建立的 `ecs-task-role-for-firelens` 任務角色。選擇角色並複製顯示於 Summary (摘要) 區段的 ARN。

b. *`execution role ARN`*

若要尋找執行角色 ARN，請前往 IAM 主控台。選擇 Roles (角色) 並尋找 `ecsTaskExecutionRole` 角色。選擇角色並複製顯示於 Summary (摘要) 區段的 ARN。

c. *`aws_account_id`*

若要尋找您的 `aws_account_id`，請登入 AWS Management Console。選擇右上角的使用者名稱並複製您的帳戶 ID。

d. *`us-east-1`*

如有必要，請取代該區域。

3. 註冊任務定義檔案：`aws ecs register-task-definition --cli-input-json file://multiline-task-definition.json --region region`
4. 開啟主控台，網址為 https://console.aws.amazon.com/ecs/v2。
5. 在導覽窗格中，選擇 Task Definitions (任務定義)，然後選擇 `firelens-example-multiline` 系列，因為我們已在上述任務定義的第一行中將任務定義註冊到此系列。
6. 選擇最新版本。
7. 選擇部署、執行任務。
8. 在執行任務頁面上，針對叢集選擇叢集，然後在聯網下，針對子網路選擇任務的可用子網路。
9. 選擇建立。

驗證 Amazon CloudWatch 中的多行日誌消息是連接的

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導覽窗格中，展開 Logs (日誌) 並選擇 Log groups (日誌群組)。
3. 選擇 `multiline-test/applicatio` 日誌群組。
4. 選擇日誌。檢視訊息。系統會串連與剖析器檔案中的規則相符之明細行，並顯示為單則訊息。

下列日誌程式碼片段顯示了串連到單一 Java 堆疊追蹤事件的明細行：

```
{
    "container_id": "xxxxxx",
```

```
    "container_name": "app",
    "source": "stdout",
    "log": "Dec 14 06:41:08 Exception in thread \"main\"
 java.lang.RuntimeException: Something has gone wrong, aborting!\n
 at com.myproject.module.MyProject.badMethod(MyProject.java:22)\n    at
 com.myproject.module.MyProject.oneMoreMethod(MyProject.java:18)\n
 at com.myproject.module.MyProject.anotherMethod(MyProject.java:14)\n
 at com.myproject.module.MyProject.someMethod(MyProject.java:10)\n    at
 com.myproject.module.MyProject.main(MyProject.java:6)",
    "ecs_cluster": "default",
    "ecs_task_arn": "arn:aws:ecs:us-east-1:xxxxxxxxxxxx:task/default/xxxxxx",
    "ecs_task_definition": "firelens-example-multiline:2"
}
```

下列日誌程式碼片段顯示了相同訊息僅以單筆明細行顯示的方式 (如果您執行了未串連多行日誌訊息的 Amazon ECS 容器)。

```
{
    "log": "Dec 14 06:41:08 Exception in thread \"main\"
 java.lang.RuntimeException: Something has gone wrong, aborting!",
    "container_id": "xxxxxx-xxxxxx",
    "container_name": "app",
    "source": "stdout",
    "ecs_cluster": "default",
    "ecs_task_arn": "arn:aws:ecs:us-east-1:xxxxxxxxxxxx:task/default/xxxxxx",
    "ecs_task_definition": "firelens-example-multiline:3"
}
```

範例：使用 Fluent Bit 內建剖析器

在本範例中，您會完成下列步驟：

1. 建置並上傳 Fluent Bit 容器的映像。
2. 建置並上傳可執行、失敗並產生多行堆疊追蹤之試用版多行應用程式的映像。
3. 建立任務定義並執行任務。
4. 檢視日誌以確認跨多行的訊息是否顯示為串連。

建置並上傳 Fluent Bit 容器的映像

此映像將包含參照了 Fluent Bit 剖析器的組態檔案。

1. 建立名稱為 FluentBitDockerImage 的資料夾。
2. 在 FluentBitDockerImage 資料夾內，建立參照了 Fluent Bit 內建剖析器檔案的自訂組態檔案。

   如需有關自訂組態檔案的詳細資訊，請參閱《Amazon Elastic Container Service 開發人員指南》中的指定自訂組態檔案

   a. 將以下內容貼到該檔案：

   ```
   [FILTER]
       name                  multiline
       match                 *
       multiline.key_content log
       multiline.parser      go
   ```

   b. 儲存檔案為 extra.conf。
3. 在 FluentBitDockerImage 資料夾內，建立具有 Fluent Bit 映像與剖析器和您建立的組態檔案之 Dockerfile。

   a. 將以下內容貼到該檔案：

   ```
   FROM public.ecr.aws/aws-observability/aws-for-fluent-bit:latest
   ADD extra.conf /extra.conf
   ```

   b. 儲存檔案為 Dockerfile。
4. 使用 Dockerfile，建置內含自訂組態檔案的自訂 Fluent Bit 映像。

   > **Note**
   >
   > 您可以將配置檔案放置在 Docker 映像中的任何位置/fluent-bit/etc/fluent-bit.conf，除非此檔案路徑被 FireLens使用。

   a. 建置映像：docker build -t fluent-bit-multiline-image .

   其中：fluent-bit-multiline-image 是此範例中映像的名稱。

   b. 驗證映像已正確建立：docker images –filter reference=fluent-bit-multiline-image

如果成功，輸出會顯示映像和 latest 標籤。

5. 將自訂 Fluent Bit 映像上傳至 Amazon Elastic 容器登錄檔。

   a. 建立 Amazon ECR 儲存庫，以便存放映像：aws ecr create-repository --repository-name fluent-bit-multiline-repo --region us-east-1

      其中：fluent-bit-multiline-repo 是此範例中儲存庫的名稱，us-east-1 是此範例中的區域。

      輸出為您提供了新儲存庫的詳細資訊。

   b. 使用來自先前輸出的 repositoryUri 值，標記您的映像：docker tag *fluent-bit-multiline-image repositoryUri*

      範例：docker tag fluent-bit-multiline-image xxxxxxxxxxxx.dkr.ecr.us-east-1.amazonaws.com/fluent-bit-multiline-repo

   c. 執行 Docker 影像以確認是否能正確執行：docker images –filter reference=*repositoryUri*

      在輸出中，存放庫名稱會從變更 fluent-bit-multiline-repo 為repositoryUri。

   d. 透過執行 aws ecr get-login-password 命令並指定您要驗證的登錄 ID 來驗證 Amazon ECR：aws ecr get-login-password | docker login --username AWS --password-stdin *registry ID*.dkr.ecr.*region*.amazonaws.com

      範例：ecr get-login-password | docker login --username AWS --password-stdin xxxxxxxxxxxx.dkr.ecr.us-east-1.amazonaws.com

      畫面上會顯示成功登入訊息。

   e. 將映像推送至 Amazon ECR：docker push *registry ID*.dkr.ecr.*region*.amazonaws.com/*repository name*

      範例：docker push xxxxxxxxxxxx.dkr.ecr.us-east-1.amazonaws.com/fluent-bit-multiline-repo

建置並上傳試用版多行應用程式的映像

此映像將包含執行應用程式的 Python 指令碼檔案和樣本日誌檔案。

1. 建立名為 multiline-app 的資料夾：mkdir multiline-app

2. 建立 Python 指令碼檔案。

   a. 在 multiline-app 資料夾中，建立檔案，並將它命名為 main.py。

   b. 將以下內容貼到該檔案：

```
import os
import time
file1 = open('/test.log', 'r')
Lines = file1.readlines()

count = 0

for i in range(10):
    print("app running normally...")
    time.sleep(1)

# Strips the newline character
for line in Lines:
    count += 1
    print(line.rstrip())
print(count)
print("app terminated.")
```

   c. 儲存 main.py 檔案。

3. 建立範例日誌檔案。

   a. 在 multiline-app 資料夾中，建立檔案，並將它命名為 test.log。

   b. 將以下內容貼到該檔案：

```
panic: my panic

goroutine 4 [running]:
panic(0x45cb40, 0x47ad70)
  /usr/local/go/src/runtime/panic.go:542 +0x46c fp=0xc42003f7b8 sp=0xc42003f710
 pc=0x422f7c
main.main.func1(0xc420024120)
  foo.go:6 +0x39 fp=0xc42003f7d8 sp=0xc42003f7b8 pc=0x451339
runtime.goexit()
  /usr/local/go/src/runtime/asm_amd64.s:2337 +0x1 fp=0xc42003f7e0
 sp=0xc42003f7d8 pc=0x44b4d1
created by main.main
```

```
  foo.go:5 +0x58

goroutine 1 [chan receive]:
runtime.gopark(0x4739b8, 0xc420024178, 0x46fcd7, 0xc, 0xc420028e17, 0x3)
  /usr/local/go/src/runtime/proc.go:280 +0x12c fp=0xc420053e30 sp=0xc420053e00
 pc=0x42503c
runtime.goparkunlock(0xc420024178, 0x46fcd7, 0xc, 0x1000f010040c217, 0x3)
  /usr/local/go/src/runtime/proc.go:286 +0x5e fp=0xc420053e70 sp=0xc420053e30
 pc=0x42512e
runtime.chanrecv(0xc420024120, 0x0, 0xc420053f01, 0x4512d8)
  /usr/local/go/src/runtime/chan.go:506 +0x304 fp=0xc420053f20 sp=0xc420053e70
 pc=0x4046b4
runtime.chanrecv1(0xc420024120, 0x0)
  /usr/local/go/src/runtime/chan.go:388 +0x2b fp=0xc420053f50 sp=0xc420053f20
 pc=0x40439b
main.main()
  foo.go:9 +0x6f fp=0xc420053f80 sp=0xc420053f50 pc=0x4512ef
runtime.main()
  /usr/local/go/src/runtime/proc.go:185 +0x20d fp=0xc420053fe0 sp=0xc420053f80
 pc=0x424bad
runtime.goexit()
  /usr/local/go/src/runtime/asm_amd64.s:2337 +0x1 fp=0xc420053fe8
 sp=0xc420053fe0 pc=0x44b4d1

goroutine 2 [force gc (idle)]:
runtime.gopark(0x4739b8, 0x4ad720, 0x47001e, 0xf, 0x14, 0x1)
  /usr/local/go/src/runtime/proc.go:280 +0x12c fp=0xc42003e768 sp=0xc42003e738
 pc=0x42503c
runtime.goparkunlock(0x4ad720, 0x47001e, 0xf, 0xc420000114, 0x1)
  /usr/local/go/src/runtime/proc.go:286 +0x5e fp=0xc42003e7a8 sp=0xc42003e768
 pc=0x42512e
runtime.forcegchelper()
  /usr/local/go/src/runtime/proc.go:238 +0xcc fp=0xc42003e7e0 sp=0xc42003e7a8
 pc=0x424e5c
runtime.goexit()
  /usr/local/go/src/runtime/asm_amd64.s:2337 +0x1 fp=0xc42003e7e8
 sp=0xc42003e7e0 pc=0x44b4d1
created by runtime.init.4
  /usr/local/go/src/runtime/proc.go:227 +0x35

goroutine 3 [GC sweep wait]:
runtime.gopark(0x4739b8, 0x4ad7e0, 0x46fdd2, 0xd, 0x419914, 0x1)
  /usr/local/go/src/runtime/proc.go:280 +0x12c fp=0xc42003ef60 sp=0xc42003ef30
 pc=0x42503c
```

```
runtime.goparkunlock(0x4ad7e0, 0x46fdd2, 0xd, 0x14, 0x1)
  /usr/local/go/src/runtime/proc.go:286 +0x5e fp=0xc42003efa0 sp=0xc42003ef60
 pc=0x42512e
runtime.bgsweep(0xc42001e150)
  /usr/local/go/src/runtime/mgcsweep.go:52 +0xa3 fp=0xc42003efd8
 sp=0xc42003efa0 pc=0x419973
runtime.goexit()
  /usr/local/go/src/runtime/asm_amd64.s:2337 +0x1 fp=0xc42003efe0
 sp=0xc42003efd8 pc=0x44b4d1
created by runtime.gcenable
  /usr/local/go/src/runtime/mgc.go:216 +0x58
one more line, no multiline
```

c. 儲存 `test.log` 檔案。

4. 在 `multiline-app` 資料夾內，建立 Dockerfile。

a. 將以下內容貼到該檔案：

```
FROM public.ecr.aws/amazonlinux/amazonlinux:latest
ADD test.log /test.log

RUN yum upgrade -y && yum install -y python3

WORKDIR /usr/local/bin

COPY main.py .

CMD ["python3", "main.py"]
```

b. 儲存 `Dockerfile` 檔案。

5. 使用 Dockerfile，建置映像。

a. 建置映像：`docker build -t multiline-app-image .`

其中：`multiline-app-image` 是此範例中映像的名稱。

b. 驗證映像已正確建立：`docker images –filter reference=multiline-app-image`

如果成功，輸出會顯示映像和 `latest` 標籤。

6. 將映像上傳至 Amazon Elastic 容器登錄檔。

a. 建立 Amazon ECR 儲存庫，以便存放映像：`aws ecr create-repository --repository-name multiline-app-repo --region us-east-1`

其中：multiline-app-repo 是此範例中儲存庫的名稱，us-east-1 是此範例中的區域。

輸出為您提供了新儲存庫的詳細資訊。請記下 repositoryUri 值，因為您將在後續步驟用到它。

b. 使用來自先前輸出的 repositoryUri 值，標記您的映像：docker tag *multiline-app-image repositoryUri*

範例：docker tag multiline-app-image xxxxxxxxxxxx.dkr.ecr.us-east-1.amazonaws.com/multiline-app-repo

c. 執行 Docker 影像以確認是否能正確執行：docker images –filter reference=*repositoryUri*

在輸出中，儲存庫名稱會從 multiline-app-repo 變更為 repositoryUri 值。

d. 將映像推送至 Amazon ECR：docker push *aws_account_id*.dkr.ecr.*region*.amazonaws.com/*repository name*

範例：docker push *xxxxxxxxxxxx*.dkr.ecr.*us-east-1*.amazonaws.com/multiline-app-repo

建立任務定義並執行任務

1. 建立檔案名稱為 multiline-task-definition.json 的任務定義檔案。
2. 將以下內容貼到該 multiline-task-definition.json 檔案：

```
{
    "family": "firelens-example-multiline",
    "taskRoleArn": "task role ARN",
    "executionRoleArn": "execution role ARN",
    "containerDefinitions": [
        {
            "essential": true,
            "image": "aws_account_id.dkr.ecr.us-east-1.amazonaws.com/fluent-bit-multiline-image:latest",
            "name": "log_router",
            "firelensConfiguration": {
                "type": "fluentbit",
                "options": {
                    "config-file-type": "file",
                    "config-file-value": "/extra.conf"
```

```
                }
            },
            "memoryReservation": 50
        },
        {
            "essential": true,
            "image": "aws_account_id.dkr.ecr.us-east-1.amazonaws.com/multiline-app-
image:latest",
            "name": "app",
            "logConfiguration": {
                "logDriver": "awsfirelens",
                "options": {
                    "Name": "cloudwatch_logs",
                    "region": "us-east-1",
                    "log_group_name": "multiline-test/application",
                    "auto_create_group": "true",
                    "log_stream_prefix": "multiline-"
                }
            },
            "memoryReservation": 100
        }
    ],
    "requiresCompatibilities": ["FARGATE"],
    "networkMode": "awsvpc",
    "cpu": "256",
    "memory": "512"
}
```

取代下列在 multiline-task-definition.json 任務定義中的項目：

a. *task role ARN*

   若要尋找任務角色 ARN，請前往 IAM 主控台。選擇 Roles (角色) 並尋找您建立的 ecs-task-role-for-firelens 任務角色。選擇角色並複製顯示於 Summary (摘要) 區段的 ARN。

b. *execution role ARN*

   若要尋找執行角色 ARN，請前往 IAM 主控台。選擇 Roles (角色) 並尋找 ecsTaskExecutionRole 角色。選擇角色並複製顯示於 Summary (摘要) 區段的 ARN。

c. *aws_account_id*

若要尋找您的 aws_account_id，請登入 AWS Management Console。選擇右上角的使用者名稱並複製您的帳戶 ID。

d. *us-east-1*

如有必要，請取代該區域。

3. 註冊任務定義檔案：aws ecs register-task-definition --cli-input-json file://multiline-task-definition.json --region us-east-1
4. 開啟主控台，網址為 https://console.aws.amazon.com/ecs/v2。
5. 在導覽窗格中，選擇 Task Definitions (任務定義)，然後選擇 firelens-example-multiline 系列，因為我們已在上述任務定義的第一行中將任務定義註冊到此系列。
6. 選擇最新版本。
7. 選擇部署、執行任務。
8. 在執行任務頁面上，針對叢集選擇叢集，然後在聯網下，針對子網路選擇任務的可用子網路。
9. 選擇建立。

驗證 Amazon CloudWatch 中的多行日誌消息是連接的

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導覽窗格中，展開 Logs (日誌) 並選擇 Log groups (日誌群組)。
3. 選擇 multiline-test/applicatio 日誌群組。
4. 選擇日誌並檢視訊息。系統會串連與剖析器檔案中的規則相符之明細行，並顯示為單則訊息。

下列日誌程式碼片段顯示了串連到單一事件的 Go 堆疊追蹤：

```
{
    "log": "panic: my panic\n\ngoroutine 4 [running]:\npanic(0x45cb40,
 0x47ad70)\n  /usr/local/go/src/runtime/panic.go:542 +0x46c fp=0xc42003f7b8
 sp=0xc42003f710 pc=0x422f7c\nmain.main.func1(0xc420024120)\n  foo.go:6
 +0x39 fp=0xc42003f7d8 sp=0xc42003f7b8 pc=0x451339\nruntime.goexit()\n  /usr/
local/go/src/runtime/asm_amd64.s:2337 +0x1 fp=0xc42003f7e0 sp=0xc42003f7d8
 pc=0x44b4d1\ncreated by main.main\n  foo.go:5 +0x58\n\ngoroutine 1 [chan receive]:
\nruntime.gopark(0x4739b8, 0xc420024178, 0x46fcd7, 0xc, 0xc420028e17, 0x3)\n  /usr/
local/go/src/runtime/proc.go:280 +0x12c fp=0xc420053e30 sp=0xc420053e00 pc=0x42503c
\nruntime.goparkunlock(0xc420024178, 0x46fcd7, 0xc, 0x1000f010040c217, 0x3)\n
  /usr/local/go/src/runtime/proc.go:286 +0x5e fp=0xc420053e70 sp=0xc420053e30
 pc=0x42512e\nruntime.chanrecv(0xc420024120, 0x0, 0xc420053f01, 0x4512d8)\n
```

```
  /usr/local/go/src/runtime/chan.go:506 +0x304 fp=0xc420053f20 sp=0xc420053e70
 pc=0x4046b4\nruntime.chanrecv1(0xc420024120, 0x0)\n  /usr/local/go/src/runtime/
chan.go:388 +0x2b fp=0xc420053f50 sp=0xc420053f20 pc=0x40439b\nmain.main()\n
 foo.go:9 +0x6f fp=0xc420053f80 sp=0xc420053f50 pc=0x4512ef\nruntime.main()\n
  /usr/local/go/src/runtime/proc.go:185 +0x20d fp=0xc420053fe0 sp=0xc420053f80
 pc=0x424bad\nruntime.goexit()\n  /usr/local/go/src/runtime/asm_amd64.s:2337
 +0x1 fp=0xc420053fe8 sp=0xc420053fe0 pc=0x44b4d1\n\ngoroutine 2 [force gc
 (idle)]:\nruntime.gopark(0x4739b8, 0x4ad720, 0x47001e, 0xf, 0x14, 0x1)\n  /
usr/local/go/src/runtime/proc.go:280 +0x12c fp=0xc42003e768 sp=0xc42003e738
 pc=0x42503c\nruntime.goparkunlock(0x4ad720, 0x47001e, 0xf, 0xc420000114, 0x1)\n
  /usr/local/go/src/runtime/proc.go:286 +0x5e fp=0xc42003e7a8 sp=0xc42003e768
 pc=0x42512e\nruntime.forcegchelper()\n  /usr/local/go/src/runtime/proc.go:238
 +0xcc fp=0xc42003e7e0 sp=0xc42003e7a8 pc=0x424e5c\nruntime.goexit()\n  /usr/
local/go/src/runtime/asm_amd64.s:2337 +0x1 fp=0xc42003e7e8 sp=0xc42003e7e0
 pc=0x44b4d1\ncreated by runtime.init.4\n  /usr/local/go/src/runtime/proc.go:227
 +0x35\n\ngoroutine 3 [GC sweep wait]:\nruntime.gopark(0x4739b8, 0x4ad7e0,
 0x46fdd2, 0xd, 0x419914, 0x1)\n  /usr/local/go/src/runtime/proc.go:280 +0x12c
 fp=0xc42003ef60 sp=0xc42003ef30 pc=0x42503c\nruntime.goparkunlock(0x4ad7e0,
 0x46fdd2, 0xd, 0x14, 0x1)\n  /usr/local/go/src/runtime/proc.go:286 +0x5e
 fp=0xc42003efa0 sp=0xc42003ef60 pc=0x42512e\nruntime.bgsweep(0xc42001e150)\n
  /usr/local/go/src/runtime/mgcsweep.go:52 +0xa3 fp=0xc42003efd8 sp=0xc42003efa0
 pc=0x419973\nruntime.goexit()\n  /usr/local/go/src/runtime/asm_amd64.s:2337 +0x1
 fp=0xc42003efe0 sp=0xc42003efd8 pc=0x44b4d1\ncreated by runtime.gcenable\n  /usr/
local/go/src/runtime/mgc.go:216 +0x58",
    "container_id": "xxxxxx-xxxxxx",
    "container_name": "app",
    "source": "stdout",
    "ecs_cluster": "default",
    "ecs_task_arn": "arn:aws:ecs:us-east-1:xxxxxxxxxxxx:task/default/xxxxxx",
    "ecs_task_definition": "firelens-example-multiline:2"
}
```

下列日誌程式碼片段顯示了相同事件顯示的方式 (如果您執行了未串連多行日誌訊息的 ECS 容器)。日誌欄位包含單筆明細行。

```
{
    "log": "panic: my panic",
    "container_id": "xxxxxx-xxxxxx",
    "container_name": "app",
    "source": "stdout",
    "ecs_cluster": "default",
    "ecs_task_arn": "arn:aws:ecs:us-east-1:xxxxxxxxxxxx:task/default/xxxxxx",
```

```
    "ecs_task_definition": "firelens-example-multiline:3"
```

> **Note**
>
> 如果您的日誌轉到日誌檔案而不是標準輸出，我們建議您在結尾輸入外掛程式 (而非篩選條件) 中指定 `multiline.parser` 和 `multiline.key_content` 組態參數。

在 Amazon ECS 視窗容器上部署流暢的位元

Fluent Bit 是一種快速靈活的日誌處理器和路由器，支援各種作業系統。它可用於將日誌路由到各種 AWS 目的地，例如 Amazon CloudWatch 日誌，Firehose Amazon S3 和 Amazon OpenSearch 服務。Fluent Bit 支援常見的合作夥伴解決方案，例如 Datadog、Splunk 和自訂 HTTP 伺服器。如需有關 Fluent Bit 的詳細資訊，請參閱 Fluent Bit 網站。

AWS for Fluent Bit 映像在大多數區域的 Amazon ECR Public Gallery 和 Amazon ECR 儲存庫的 Amazon ECR 上可用以提供高可用性。有關更多信息，請參閱 GitHub 網站 aws-for-fluent-bit 上的。

本教學將逐步引導您如何在 Amazon ECS 中執行的 Windows 執行個體上部署 Fluent Bit 容器，以將 Windows 任務產生的日誌串流到 Amazon 以 CloudWatch 進行集中記錄。

本教學課程使用下列作法：

- Fluent Bit 可作為具有常駐程式排程策略的服務執行。此策略可確保 Fluent Bit 的單一執行個體一律在叢集中的容器執行個體上執行。
  - 使用轉寄輸入外掛程式接聽連接埠 24224。
  - 向主機公開連接埠 24224，以便 Docker 執行時間可使用此公開連接埠將日誌傳送到 Fluent Bit。
  - 具有允許 Fluent Bit 將日誌記錄傳送到指定目的地的組態。
- 使用 Fluentd 日誌記錄驅動程式啟動所有其他 Amazon ECS 任務容器。如需詳細資訊，請參閱 Docker 文件網站上的 Fluentd logging driver (Fluentd 日誌記錄驅動程式)。
  - Docker 連線到主機命名空間內 localhost 上的 TCP 通訊端 24224。
  - Amazon ECS 代理程式會將標籤新增至容器，其中包括叢集名稱、任務定義系列名稱、任務定義修訂版編號、任務 ARN 和容器名稱。使用 Fluentd Docker 日誌記錄驅動程式的標籤選項將相同的資訊新增到日誌記錄中。如需詳細資訊，請參閱 Docker 文件網站上的 labels、labels-regex、env 和 env-regex。

- 由於 Fluentd 日誌記錄驅動程式的 `async` 選項設定為 `true`，當 Fluent Bit 容器重新啟動時，Docker 會緩衝日誌，直到 Fluent Bit 容器重新啟動為止。您可以透過設定 fluentd-buffer-limit 選項來增加緩衝區限制。如需詳細資訊，請參閱 Docker 文件網站上的 fluentd-buffer-limit。

工作流程如下所示：

- Fluent Bit 容器會啟動並接聽公開至主機的連接埠 24224。
- Fluent Bit 會使用任務定義中指定的任務 IAM 角色憑證。
- 在同一個執行個體上啟動的其他任務，會使用 Fluentd Docker 日誌記錄驅動程式連線到連接埠 24224 上的 Fluent Bit 容器。
- 當應用程式容器生成日誌時，Docker 執行時間會標記這些記錄，新增標籤中指定的其他中繼資料，然後在主機命名空間的連接埠 24224 上轉寄這些中繼資料。
- Fluent Bit 會接收連接埠 24224 上的日誌記錄，因為其公開到主機命名空間。
- Fluent Bit 會執行其內部處理，並依照指定的方式路由日誌。

本教學課程使用預設的 CloudWatch Fluent Bit 組態，可執行下列作業：

- 為每個叢集和任務定義系列建立新的日誌群組。
- 每當啟動新任務時，為上述產生的日誌群組中的每個任務容器建立新的日誌串流。每個串流將標記容器所屬的任務 ID。
- 在每個日誌項目中新增其他中繼資料，包括叢集名稱、任務 ARN、任務容器名稱、任務定義系列以及任務定義修訂版編號。

  例如，如果您task_1使用 container_1 and container_2 and and t ask_2container_3，則下列是 CloudWatch 記錄資料流：

  - `/aws/ecs/windows.ecs_task_1`

    `task-out.`*`TASK_ID`*`.container_1`

    `task-out.`*`TASK_ID`*`.container_2`
  - `/aws/ecs/windows.ecs_task_2`

    `task-out.`*`TASK_ID`*`.container_3`

步驟

- 必要條件
- 步驟 1：建立 IAM 存取角色
- 步驟 2：建立 Amazon ECS Windows 容器執行個體
- 步驟 3：設定 Fluent Bit
- 步驟 4：註冊將日誌路由到的 Windows 流利位任務定義 CloudWatch
- 步驟 5：使用常駐程式排程策略以 Amazon ECS 服務形式執行 ecs-windows-fluent-bit 任務定義
- 步驟 6：註冊會產生日誌的 Windows 任務定義
- 步驟 7：執行 windows-app-task 任務定義
- 步驟 8：驗證日誌 CloudWatch
- 步驟 9：清除

必要條件

本教學課程假設已完成下列先決條件：

- 已安裝並設定最新版本的。 AWS CLI 如需詳細資訊，請參閱安裝 AWS Command Line Interface。
- `aws-for-fluent-bit` 容器映像可用於下列 Windows 作業系統：
  - Windows Server 2019 Core
  - Windows Server 2019 Full
  - Windows Server 2022 Core
  - Windows Server 2022 Full
- 已完成「設定以使用 Amazon ECS。」中的步驟。
- 您有一個叢集。在本教程中，群集名稱是 FluentBit-cluster。
- 您有一個帶有公共子網路的 VPC，EC2 執行個體將在其中啟動。您可以使用預設 VPC。您也可以使用允許 Amazon CloudWatch 端點到達子網路的私有子網路。如需 Amazon CloudWatch 端點的詳細資訊，請參閱 AWS 一般參考. CloudWatch 如需使用 Amazon VPC 精靈建立 VPC 的相關資訊，請參閱the section called “建立 Virtual Private Cloud”。

步驟 1：建立 IAM 存取角色

建立 Amazon ECS IAM 角色。

1. 建立名為「ecs」的 Amazon ECS 容器執行個體角色。InstanceRole如需詳細資訊，請參閱 Amazon ECS 容器執行個體 IAM 角色。
2. 為命名為 `fluentTaskRole` 的 Fluent Bit 任務建立 IAM 角色。如需詳細資訊，請參閱 the section called “任務 IAM 角色”。

   在此 IAM 角色中授予的 IAM 許可是由任務容器承繼。為了允許 Fluent Bit 傳送日誌 CloudWatch，您需要將以下許可附加到任務 IAM 角色。

   ```
   {
       "Version": "2012-10-17",
       "Statement": [
       {
           "Effect": "Allow",
           "Action": [
               "logs:CreateLogStream",
               "logs:CreateLogGroup",
               "logs:DescribeLogStreams",
               "logs:PutLogEvents"
           ],
           "Resource": "*"
       }
       ]
   }
   ```

3. 將政策連接到角色。

   a. 將上述內容儲存在名為 `fluent-bit-policy.json` 的檔案中。

   b. 執行下列命令，將內嵌政策附加到 `fluentTaskRole` IAM 角色。

   ```
   aws iam put-role-policy --role-name fluentTaskRole --policy-name
    fluentTaskPolicy --policy-document file://fluent-bit-policy.json
   ```

步驟 2：建立 Amazon ECS Windows 容器執行個體

建立 Amazon ECS Windows 容器執行個體。

建立 Amazon ECS 執行個體

1. 使用 `aws ssm get-parameters` 命令擷取託管 VPC 的區域 AMI ID。如需詳細資訊，請參閱擷取 Amazon ECS 最佳化 AMI 中繼資料。

2. 使用 Amazon EC2 主控台來啟動執行個體。

   a. 前往 https://console.aws.amazon.com/ec2/ 開啟 Amazon EC2 主控台。

   b. 從導覽列中選取要使用的「區域」。

   c. 在 EC2 儀表板中，選擇 Launch instance (啟動執行個體)。

   d. 在 Name (名稱) 輸入唯一的名稱。

   e. 在 Application and OS Images (Amazon Machine Image) (應用程式和作業系統映像 (Amazon Machine Image)) 欄位中，選擇您在步驟 1 中擷取的 AMI。

   f. 對於 Instance type (執行個體類型)，選擇 `t3.xlarge`。

   g. 在 Key pair (login) (金鑰對 (登入)) 欄位中，選擇一個金鑰對。

   h. 在 Network settings (網路設定) 下的 Security group (安全群組) 欄位中，選擇一個現有的安全群組，或建立一個新的安全群組。

   i. 在 Network settings (網路設定) 下的 Auto-assign Public IP (自動指派公有 IP) 欄位中，選取 Enable (啟用)。

   j. 在 [進階詳細資料] 下方，針對 IAM 執行個體設定檔選擇 ecs InstanceRole。

   k. 利用下列使用者資料設定您的 Amazon ECS 容器執行個體。在 Advanced details (進階詳細資料) 下的 User data (使用者資料) 欄位中，貼入下列指令碼，以您的叢集名稱取代 *`cluster_name`*。

   ```
   <powershell>
   Import-Module ECSTools
   Initialize-ECSAgent -Cluster cluster-name -EnableTaskENI -EnableTaskIAMRole -
   LoggingDrivers '["awslogs","fluentd"]'
   </powershell>
   ```

   l. 準備就緒後，請選取 acknowledgment (確認) 欄位，再選擇 Launch Instances (啟動執行個體)。

   m. 會有確認頁面讓您知道您的執行個體正在啟動。選擇 View Instances (檢視執行個體) 關閉確認頁面並返回主控台。

步驟 3：設定 Fluent Bit

您可以使用提供的下列預設組態 AWS 來快速開始使用：

- Amazon 基 CloudWatch於流利位官方手冊中 Amazon CloudWatch 的流利位插件。

或者，您也可以使用由提供的其他預設組態 AWS。如需詳細資訊，請參閱 Github 網站上有關 aws-for-fluent-bit 的 Overriding the entrypoint for the Windows image (覆寫 Windows 映像的進入點)。

預設的 Amazon CloudWatch 流利位元組態如下所示。

取代下列變數：

- ##與您要發送 Amazon CloudWatch 日誌的區域。

```
[SERVICE]
    Flush               5
    Log_Level           info
    Daemon              off

[INPUT]
    Name                forward
    Listen              0.0.0.0
    Port                24224
    Buffer_Chunk_Size   1M
    Buffer_Max_Size     6M
    Tag_Prefix          ecs.

# Amazon ECS agent adds the following log keys as labels to the docker container.
# We would use fluentd logging driver to add these to log record while sending it to
 Fluent Bit.
[FILTER]
    Name                modify
    Match               ecs.*
    Rename              com.amazonaws.ecs.cluster ecs_cluster
    Rename              com.amazonaws.ecs.container-name ecs_container_name
    Rename              com.amazonaws.ecs.task-arn ecs_task_arn
    Rename              com.amazonaws.ecs.task-definition-family
 ecs_task_definition_family
    Rename              com.amazonaws.ecs.task-definition-version
 ecs_task_definition_version

[FILTER]
    Name                rewrite_tag
    Match               ecs.*
    Rule                $ecs_task_arn ^([a-z-:0-9]+)/([a-zA-Z0-9-_]+)/([a-z0-9]+)$
 out.$3.$ecs_container_name false
```

```
    Emitter_Name        re_emitted

[OUTPUT]
    Name                cloudwatch_logs
    Match               out.*
    region              region
    log_group_name      fallback-group
    log_group_template  /aws/ecs/$ecs_cluster.$ecs_task_definition_family
    log_stream_prefix   task-
    auto_create_group   On
```

每個進入 Fluent Bit 的日誌都有一個您指定的標籤，或者在您不提供標籤時自動生成。這些標籤可用來將不同的日誌路由到不同的目的地。如需詳細資訊，請參閱《Fluent Bit 官方手冊》中的標籤。

上述 Fluent Bit 組態具有下列屬性：

- 轉寄輸入外掛程式會接聽 TCP 連接埠 24224 上的傳入流量。
- 在該連接埠上收到的每個日誌項目都有一個標籤，轉寄輸入外掛程式會修改該標籤，以在記錄前面加上 `ecs.` 字串。
- Fluent Bit 內部管道會路由日誌項目，以使用符合規則運算式修改篩選條件。此篩選條件將日誌記錄 JSON 中的金鑰替換為 Fluent Bit 可以使用的格式。
- 然後，rewrite_tag 篩選條件會使用修改後的日誌項目。此篩選條件會將日誌記錄的標籤變更為格式輸出 *TASK_ID*.*CONTAINER_NAME*。
- 新標籤將路由到輸出 cloudwatch_logs 外掛程式，該外掛程式會使用輸出外掛程式的和`log_stream_prefix`選項建立記錄群組`log_group_template`和資料流，如前所述。CloudWatch 如需其他資訊，請參閱《Fluent Bit 官方手冊》中的組態參數。

步驟 4：註冊將日誌路由到的 Windows 流利位任務定義 CloudWatch

註冊將記錄路由到的 Windows 流利位元工作定義 CloudWatch。

> **Note**
>
> 此任務定義會將 Fluent Bit 容器連接埠 24224 公開至主機連接埠 24224。確認此連接埠未在 EC2 執行個體安全群組中開啟，防止從外部存取。

註冊任務定義

1. 使用下列內容建立名為 fluent-bit.json 的檔案。

   取代下列變數：

   - 將 *task-iam-role* 替代為任務 IAM 角色的 Amazon Resource Name (ARN)
   - 將 *region* 替代為任務執行所在的區域

```
{
  "family": "ecs-windows-fluent-bit",
  "taskRoleArn": "task-iam-role",
  "containerDefinitions": [
    {
      "name": "fluent-bit",
      "image": "public.ecr.aws/aws-observability/aws-for-fluent-
bit:windowsservercore-latest",
      "cpu": 512,
      "portMappings": [
        {
          "hostPort": 24224,
          "containerPort": 24224,
          "protocol": "tcp"
        }
      ],
      "entryPoint": [
        "Powershell",
        "-Command"
      ],
      "command": [
        "C:\\entrypoint.ps1 -ConfigFile C:\\ecs_windows_forward_daemon\
\cloudwatch.conf"
      ],
      "environment": [
        {
          "name": "AWS_REGION",
          "value": "region"
        }
      ],
      "memory": 512,
      "essential": true,
      "logConfiguration": {
```

```
        "logDriver": "awslogs",
        "options": {
          "awslogs-group": "/ecs/fluent-bit-logs",
          "awslogs-region": "region",
          "awslogs-stream-prefix": "flb",
          "awslogs-create-group": "true"
        }
      }
    }
  ],
  "memory": "512",
  "cpu": "512"
}
```

2. 執行下列命令來註冊任務定義。

```
aws ecs register-task-definition --cli-input-json file://fluent-bit.json --
region region
```

您可以執行 `list-task-definitions` 命令列出您帳戶的任務定義。輸出顯示可與 `run-task` 或 `start-task` 搭配使用的系列和修訂版值。

步驟 5：使用常駐程式排程策略以 Amazon ECS 服務形式執行 **ecs-windows-fluent-bit** 任務定義

註冊帳戶的任務定義後，您就可以在叢集中執行任務。在本教學課程中，您會在 `FluentBit-cluster` 叢集中執行 `ecs-windows-fluent-bit:1` 任務定義的一個執行個體。在使用常駐程式排程策略的服務中執行任務，用以確保 Fluent Bit 的單一執行個體一律在每個容器執行個體上執行。

執行任務

1. 執行下列命令，以服務形式啟動 `ecs-windows-fluent-bit:1` 任務定義 (在上一個步驟中註冊)。

> **Note**
>
> 此任務定義使用 `awslogs` 日誌記錄驅動程式，您的容器執行個體需要具有必要的許可。

取代下列變數：

- 將 *region* 替代為服務執行所在的區域

```
aws ecs create-service \
    --cluster FluentBit-cluster \
    --service-name FluentBitForwardDaemonService \
    --task-definition ecs-windows-fluent-bit:1 \
    --launch-type EC2 \
    --scheduling-strategy DAEMON \
    --region region
```

2. 執行下列命令以列出您的任務。

   取代下列變數：

   - 將 *region* 替代為服務任務執行所在的區域

```
aws ecs list-tasks --cluster FluentBit-cluster --region region
```

步驟 6：註冊會產生日誌的 Windows 任務定義

註冊會產生日誌的任務定義。此任務定義會部署 Windows 容器映像，該映像會每秒向 `stdout` 寫入一個遞增數字。

任務定義會使用連線到 Fluent Bit 外掛程式接聽連接埠 24224 的 Fluentd 日誌記錄驅動程式。Amazon ECS 代理程式會為每個 Amazon ECS 容器新增標籤，包括叢集名稱、任務 ARN、任務定義系列名稱、任務定義修訂版編號和任務容器名稱。這些鍵/值對標籤會傳遞至 Fluent Bit。

> **Note**
>
> 此任務會使用 `default` 網路模式。不過，您也可以在任務中使用 `awsvpc` 網路模式。

註冊任務定義

1. 使用下列內容建立名為 `windows-app-task.json` 的檔案。

```
{
```

```
  "family": "windows-app-task",
  "containerDefinitions": [
    {
      "name": "sample-container",
      "image": "mcr.microsoft.com/windows/servercore:ltsc2019",
      "cpu": 512,
      "memory": 512,
      "essential": true,
      "entryPoint": [
        "Powershell",
        "-Command"
      ],
      "command": [
        "$count=1;while(1) { Write-Host $count; sleep 1; $count=$count+1;}"
      ],
      "logConfiguration": {
        "logDriver": "fluentd",
        "options": {
          "fluentd-address": "localhost:24224",
          "tag": "{{ index .ContainerLabels \"com.amazonaws.ecs.task-definition-
family\" }}",
          "fluentd-async": "true",
          "labels": "com.amazonaws.ecs.cluster,com.amazonaws.ecs.container-
name,com.amazonaws.ecs.task-arn,com.amazonaws.ecs.task-definition-
family,com.amazonaws.ecs.task-definition-version"
        }
      }
    }
  ],
  "memory": "512",
  "cpu": "512"
}
```

2. 執行下列命令來註冊任務定義。

   取代下列變數：

   - 將 *region* 替代為任務執行所在的區域

```
aws ecs register-task-definition --cli-input-json file://windows-app-task.json --
region region
```

您可以執行 list-task-definitions 命令列出您帳戶的任務定義。輸出顯示可與 run-task 或 start-task 搭配使用的系列和修訂版值。

步驟 7：執行 windows-app-task 任務定義

註冊 windows-app-task 任務定義之後，請在 FluentBit-cluster 叢集中執行該定義。

執行任務

1. 執行您在上一個步驟中註冊的 windows-app-task:1 任務定義。

   取代下列變數：

   - 將 *region* 替代為任務執行所在的區域

   ```
   aws ecs run-task --cluster FluentBit-cluster --task-definition windows-app-task:1
    --count 2 --region region
   ```

2. 執行下列命令以列出您的任務。

   ```
   aws ecs list-tasks --cluster FluentBit-cluster
   ```

步驟 8：驗證日誌 CloudWatch

若要驗證您的 Fluent Bit 設定，請在 CloudWatch 主控台中檢查下列記錄群組：

- /ecs/fluent-bit-logs – 這是日誌群組，對應在容器執行個體上執行的 Fluent Bit 常駐程式容器。
- /aws/ecs/FluentBit-cluster.windows-app-task - 這是日誌群組，對應在 FluentBit-cluster 叢集內為 windows-app-task 任務定義系列啟動的所有任務。

  task-out.*FIRST_TASK_ID*.sample-container – 此日誌串流包含 sample-container 任務容器中由任務的第一個執行個體產生的所有日誌。

  task-out.*SECOND_TASK_ID*.sample-container – 此日誌串流包含 sample-container 任務容器中由任務的第二個執行個體產生的所有日誌。

task-out.*TASK_ID*.sample-container 日誌串流的欄位與下列類似：

```
{
    "source": "stdout",
    "ecs_task_arn": "arn:aws:ecs:region:0123456789012:task/FluentBit-
cluster/13EXAMPLE",
    "container_name": "/ecs-windows-app-task-1-sample-container-cEXAMPLE",
    "ecs_cluster": "FluentBit-cluster",
    "ecs_container_name": "sample-container",
    "ecs_task_definition_version": "1",
    "container_id": "61f5e6EXAMPLE",
    "log": "10",
    "ecs_task_definition_family": "windows-app-task"
}
```

驗證 Fluent Bit 設定

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導覽窗格中，選擇 Log groups (日誌群組)。請確定您位於將 Fluent Bit 部署到容器的區域。

   在的記錄群組清單中 AWS 區域，您應該會看到下列內容：

   - /ecs/fluent-bit-logs
   - /aws/ecs/FluentBit-cluster.windows-app-task

   如果您看到這些日誌群組，表示 Fluent Bit 設定已通過驗證。

步驟 9：清除

完成此教學課程時，清除與其相關的資源，以免未使用的資源產生費用。

清除教學課程資源

1. 停止 windows-simple-task 任務和 ecs-fluent-bit 任務。如需詳細資訊，請參閱 the section called “停止工作”。
2. 執行下列命令來刪除 /ecs/fluent-bit-logs 日誌群組。如需有關刪除日誌群組的詳細資訊，請參閱《AWS Command Line Interface 參考》中的 delete-log-group。

   ```
   aws logs delete-log-group --log-group-name /ecs/fluent-bit-logs
   ```

```
aws logs delete-log-group --log-group-name /aws/ecs/FluentBit-cluster.windows-app-task
```

3. 執行下列命令以終止執行個體。

```
aws ec2 terminate-instances --instance-ids instance-id
```

4. 執行下列命令以刪除 IAM 角色。

```
aws iam delete-role --role-name ecsInstanceRole
aws iam delete-role --role-name fluentTaskRole
```

5. 執行下列命令以刪除 Amazon ECS 叢集。

```
aws ecs delete-cluster --cluster FluentBit-cluster
```

用gMSA於 Amazon ECS 上的 EC2 Linux 容器

Amazon ECS 透過一種稱為群組受管服務帳戶 (gMSA) 的特殊服務帳戶，支援 EC2 上 Linux 容器的作用中目錄身份驗證。

.NET Core 應用程式等以 Linux 為基礎的網路應用程式，可使用 Active Directory 來促進使用者和服務之間的身分驗證和授權管理。您可以透過設計與 Active Directory 整合並在加入網域的伺服器上執行的應用程式來使用此功能。但是，由於 Linux 容器無法加入網域，因此您需要設定 Linux 容器來執行 gMSA。

使用 gMSA 執行的 Linux 容器依賴在容器主機 Amazon EC2 執行個體上執行的 `credentials-fetcher` 常駐程式。也就是說，常駐程式會從 Active Directory 網域控制站擷取 gMSA 憑證，然後將這些憑證傳輸到容器執行個體。如需有關服務帳戶的詳細資訊，請參閱 Microsoft Learn 網站上的針對 Windows 容器建立 gMSAs。

考量事項

針對 Linux 容器使用 gMSA 前，請考慮以下事項：

- 如果您的容器在 EC2 上執行，則可以針對 Windows 容器和 Linux 容器使用 gMSA。如需有關如何在 Fargate 上使gMSA用 Linux 容器的資訊，請參閱用gMSA於 Far Linux gate 上的容器。

- 您可能需要加入網域的 Windows 電腦才能完成先決條件。例如，您可能需要加入網域的 Windows 電腦，才能在使用 PowerShell 的 Active Directory 中建立 gMSA。「RSAT作用中導演」PowerShell 工具僅適用於Windows。如需詳細資訊，請參閱安裝 Active Directory 管理工具。
- 您可以選擇無網域 gMSA 或將每個執行個體加入單一網域。使用無網域 gMSA 時，容器執行個體不會加入網域，執行個體上的其他應用程式無法使用憑證來存取網域，而加入不同網域的任務則可在相同的執行個體上執行。

  然後，選擇 CredSpec 的資料儲存，也可以選擇無網域 gMSA 的 Active Directory 使用者憑證的資料儲存。

  Amazon ECS 使用 Active Directory 憑證規格檔案 (CredSpec)。此檔案包含用於將 gMSA 帳戶內容傳播至容器的 gMSA 中繼資料。您會產生 CredSpec 檔案，然後將其儲存在下表中特定於容器執行個體的作業系統的 CredSpec 儲存選項之一。若要使用無網域方法，CredSpec 檔案中的選擇性區段可以在下表中的 domainless user credentials 儲存選項之一中指定憑證 (特定於容器執行個體的作業系統)。

  gMSA 資料儲存選項 (依作業系統)

儲存位置	Linux	Windows
Amazon Simple Storage Service	CredSpec	CredSpec
AWS Secrets Manager	無網域使用者憑證	無網域使用者憑證
Amazon EC2 Systems Manager 參數存放區	CredSpec	CredSpec，無網域使用者憑證
本機檔案	N/A	CredSpec

必要條件

在搭配 Amazon ECS 使用 Linux 容器功能適用的 gMSA 之前，請先完成下列各項：

- 您可以使用您希望容器存取的資源來設定 Active Directory 網域。Amazon ECS 支援下列設定：
  - 一個 AWS Directory Service 活動目錄。 AWS Directory Service 是託 AWS 管在 Amazon EC2 上的受管活動目錄。如需詳細資訊，請參閱《 AWS 管理指南》中的受管理 Microsoft AD 入門。AWS Directory Service

  - 內部部署 Active Directory。您必須確認 Amazon ECS Linux 容器執行個體可以加入網域。如需詳細資訊，請參閱 AWS Direct Connect。
- 您在 Active Directory 中擁有現有的 gMSA 帳戶。如需詳細資訊，請參閱 用gMSA於 Amazon ECS 上的 EC2 Linux 容器。
- 您已在 Amazon ECS Linux 容器執行個體上安裝並執行 `credentials-fetcher` 常駐程式。您也將一組初始的憑證新增至 `credentials-fetcher` 常駐程式，以便透過 Active Directory 進行身分驗證。

  > **Note**
  >
  > `credentials-fetcher` 常駐程式僅適用於 Amazon Linux 2023 和 Fedora 37 及更新版本。該常駐程式不適用於 Amazon Linux 2。如需詳細資訊，請參閱上的 aw/ 憑證取得程式。 GitHub

- 您可以設定 `credentials-fetcher` 常駐程式的憑證，以便使用 Active Directory 進行身分驗證。憑證必須是具有 gMSA 帳戶存取權的 Active Directory 安全群組的成員。決定是否要將執行個體加入網域，或是要使用無網域 gMSA。 中有多個選項。
- 您已新增必要的 IAM 許可 所需的許可取決於您為初始憑證選擇的方法以及儲存憑證規格的方法：
  - 如果您使用無網域作gMSA為初始登入資料，則工作執行角色需要的 IAM 許可。 AWS Secrets Manager
  - 如果您將憑證規格存放在 SSM 參數存放區中，則任務執行角色需要 Amazon EC2 Systems Manager 參數存放區的 IAM 許可。
  - 如果您將憑證規格存放在 Amazon S3 中，則任務執行角色需要 Amazon Simple Storage Service 的 IAM 許可。

在 Amazon ECS 上設定能夠使用 gMSA 功能的 Linux 容器

準備基礎設施

下列步驟是執行一次的考量和設定。完成這些步驟後，您可以自動建立容器執行個體以重複使用此組態。

決定如何提供初始憑證，並在可重複使用的 EC2 啟動範本中設定 EC2 使用者資料以安裝 `credentials-fetcher` 常駐程式。

1. 決定是否要將執行個體加入網域，或是要使用無網域 gMSA。
   - 將 EC2 執行個體加入 Active Directory 網域
     - 透過使用者資料加入執行個體

       在 EC2 啟動範本中新增將 Active Directory 網域加入 EC2 使用者資料的步驟。多個 Amazon EC2 Auto Scaling 群組可以使用相同的啟動範本。

       您可以在 Fedora 文件中使用這些步驟加入 Active Directory 或 FreeIPA 網域。
   - 建立無網域 gMSA 的 Active Directory 使用者

     `credentials-fetcher` 常駐程式具有稱為無網域 gMSA 的功能。此功能需要網域，但 EC2 執行個體不需要加入網域。使用無網域 gMSA 時，容器執行個體不會加入網域，執行個體上的其他應用程式無法使用憑證來存取網域，而加入不同網域的任務則可在相同的執行個體上執行。相反地，您會在 CredSpec 檔案中提供 AWS Secrets Manager 祕密的名稱。祕密必須包含使用者名稱、密碼和要登入的網域。

     此功能受到支援，可與 Linux 和 Windows 容器搭配使用。

     此功能與 gMSA support for non-domain-joined container hosts 功能類似。如需有關 Windows 功能的詳細資訊，請參閱 Microsoft Learn 網站上的 gMSA 架構和改進。

     a. 在 Active Directory 網域中建立使用者。Active Directory 中的使用者必須具有存取您在任務中使用之 gMSA 服務帳戶的權限。

     b. 在中建立密碼之後 AWS Secrets Manager，在使用中目錄中建立使用者。如需詳細資訊，請參閱建立 AWS Secrets Manager 密碼。

     c. 將使用者的使用者名稱、密碼和網域分別輸入到名為 `username`、`password` 和 `domainName` 的 JSON 鍵值組中。

     ```
     {"username":"username","password":"passw0rd", "domainName":"example.com"}
     ```

     d. 將組態新增至服務帳戶的 CredSpec 檔案。其他的 `HostAccountConfig` 包含 Secrets Manager 祕密的 Amazon Resource Name (ARN)。

     在 Windows 上，`PluginGUID` 必須符合下列範例程式碼片段中的 GUID。在 Linux 上，會忽略 `PluginGUID`。將 `MySecret` 替換為含有祕密的 Amazon Resource Name (ARN) 的範例。

     ```
         "ActiveDirectoryConfig": {
     ```

```
        "HostAccountConfig": {
            "PortableCcgVersion": "1",
            "PluginGUID": "{859E1386-BDB4-49E8-85C7-3070B13920E1}",
            "PluginInput": {
                "CredentialArn": "arn:aws:secretsmanager:aws-
region:111122223333:secret:MySecret"
            }
        }
```

e. 無網域 gMSA 功能需要任務執行角色中的其他許可。請遵循步驟 (選用) 無網域 gMSA 祕密。

2. 設定執行個體並安裝 **credentials-fetcher** 常駐程式

您可以在 EC2 啟動範本中使用使用者資料指令碼來安裝 credentials-fetcher 常駐程式。以下範例示範兩種類型的使用者資料、cloud-config YAML 或 bash 指令碼。這些範例適用於 Amazon Linux 2023 (AL2023)。將 MyCluster 替換為您想要這些執行個體加入的 Amazon ECS 叢集的名稱。

- **cloud-config** YAML

```
Content-Type: text/cloud-config
package_reboot_if_required: true
packages:
  # prerequisites
  - dotnet
  - realmd
  - oddjob
  - oddjob-mkhomedir
  - sssd
  - adcli
  - krb5-workstation
  - samba-common-tools
  # https://github.com/aws/credentials-fetcher gMSA credentials management for
 containers
  - credentials-fetcher
write_files:
# configure the ECS Agent to join your cluster.
# replace MyCluster with the name of your cluster.
- path: /etc/ecs/ecs.config
  owner: root:root
  permissions: '0644'
  content: |
    ECS_CLUSTER=MyCluster
```

```
    ECS_GMSA_SUPPORTED=true
runcmd:
# start the credentials-fetcher daemon and if it succeeded, make it start after
 every reboot
- "systemctl start credentials-fetcher"
- "systemctl is-active credentials-fetch && systemctl enable credentials-
fetcher"
```

- bash 指令碼

  如果您比較習慣 bash 指令碼，並且有多個要寫入 /etc/ecs/ecs.config 的變數，請使用以下 heredoc 格式。此格式會將開頭為 cat 和 EOF 之行間的所有項目寫入組態檔案。

```
#!/usr/bin/env bash
set -euxo pipefail

# prerequisites
timeout 30 dnf install -y dotnet realmd oddjob oddjob-mkhomedir sssd adcli
 krb5-workstation samba-common-tools
# install https://github.com/aws/credentials-fetcher gMSA credentials
 management for containers
timeout 30 dnf install -y credentials-fetcher

# start credentials-fetcher
systemctl start credentials-fetcher
systemctl is-active credentials-fetch && systemctl enable credentials-fetcher

cat <<'EOF' >> /etc/ecs/ecs.config
ECS_CLUSTER=MyCluster
ECS_GMSA_SUPPORTED=true
EOF
```

選用組態變數適用於 credentials-fetcher 常駐程式，您可以在 /etc/ecs/ecs.config 中設定該變數。我們建議您在 YAML 區塊或與先前範例類似的 heredoc 內的使用者資料中設定變數。這樣做可防止多次編輯檔案時可能會發生的部分組態問題。如需 ECS 代理程式組態的詳細資訊，請參閱上的 Amazon ECS 容器代理程式。 GitHub

- 或者，如果您變更 credentials-fetcher 常駐程式組態，將通訊端移至其他位置，您也可以使用變數 CREDENTIALS_FETCHER_HOST。

設定許可和祕密

針對每個應用程式和每個任務定義執行下列步驟一次。我們建議您使用授予最低權限的最佳實務，並縮減政策中使用的許可。這樣一來，每個任務都只能讀取所需的祕密。

1. (選用) 無網域 gMSA 祕密

   如果您使用執行個體未加入網域的無網域方法，請依照此步驟執行。

   您必須將以下許可作為內嵌政策新增至任務執行 IAM 角色。這樣做可讓 `credentials-fetcher` 常駐程式存取 Secrets Manager 的祕密。將 `MySecret` 替換為在 `Resource` 清單中含有祕密的 Amazon Resource Name (ARN) 的範例。

   ```
   {
       "Version": "2012-10-17",
       "Statement": [
           {
               "Effect": "Allow",
               "Action": [
                   "secretsmanager:GetSecretValue"
               ],
               "Resource": [
                   "arn:aws:ssm:aws-region:111122223333:secret:MySecret"
               ]
           }
       ]
   }
   ```

   > **Note**
   >
   > 如果您使用自己的 KMS 金鑰來加密密碼，則必須將必要的權限新增至此角色，並將此角色新增至金 AWS KMS 鑰原則。

2. 決定您要使用 SSM 參數存放區或 S3 來存放 CredSpec

   Amazon ECS 支援以下方法，在任務定義的 `credentialSpecs` 欄位中參考檔案路徑。

   如果您將執行個體加入單一網域，請在字串中在 ARN 開頭使用 `credentialspec:` 作為字首。如果您使用無網域 gMSA，請使用 `credentialspecdomainless:`。

   如需 CredSpec 的詳細資訊，請參閱 憑證規格檔案。

- Amazon S3 儲存貯體

  將憑證規格新增至 Amazon S3 儲存貯體。然後，在任務定義的 `credentialSpecs` 欄位中參考 Amazon S3 儲存貯體的 Amazon Resource Name (ARN)。

```
{
    "family": "",
    "executionRoleArn": "",
    "containerDefinitions": [
        {
            "name": "",
            ...
            "credentialSpecs": [
                "credentialspecdomainless:arn:aws:s3:::${BucketName}/
${ObjectName}"
            ],
            ...
        }
    ],
    ...
}
```

  若要讓您的任務能夠存取 S3 儲存貯體，請新增下列許可做為 Amazon ECS 任務執行 IAM 角色的內嵌政策。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor",
            "Effect": "Allow",
            "Action": [
                "s3:Get*",
                "s3:List*"
            ],
            "Resource": [
                "arn:aws:s3:::DOC-EXAMPLE-BUCKET",
                "arn:aws:s3:::DOC-EXAMPLE-BUCKET/{object}"
            ]
        }
    ]
```

```
}
```

- SSM 參數存放區參數

  將憑證規格新增至 SSM 參數存放區參數。然後，在任務定義的 `credentialSpecs` 欄位中參考 SSM 參數存放區參數的 Amazon Resource Name (ARN)。

```
{
    "family": "",
    "executionRoleArn": "",
    "containerDefinitions": [
        {
            "name": "",
            ...
            "credentialSpecs": [
                "credentialspecdomainless:arn:aws:ssm:aws-
region:111122223333:parameter/parameter_name"
            ],
            ...
        }
    ],
    ...
}
```

  若要讓您的任務能夠存取 SSM 參數存放區參數，請新增下列許可做為 Amazon ECS 任務執行 IAM 角色的內嵌政策。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ssm:GetParameters"
            ],
            "Resource": [
                "arn:aws:ssm:aws-region:111122223333:parameter/parameter_name"
            ]
        }
    ]
}
```

憑證規格檔案

Amazon ECS 使用 Active Directory 憑證規格檔案 (CredSpec)。此檔案包含用於將 gMSA 帳戶內容傳播至容器 Linux 的 gMSA 中繼資料。您會產生 CredSpec，並在任務定義的 `credentialSpecs` 欄位中參考該檔案。CredSpec 檔案不包含任何祕密。

以下是範例 CredSpec 檔案。

```
{
    "CmsPlugins": [
        "ActiveDirectory"
    ],
    "DomainJoinConfig": {
        "Sid": "S-1-5-21-2554468230-2647958158-2204241789",
        "MachineAccountName": "WebApp01",
        "Guid": "8665abd4-e947-4dd0-9a51-f8254943c90b",
        "DnsTreeName": "example.com",
        "DnsName": "example.com",
        "NetBiosName": "example"
    },
    "ActiveDirectoryConfig": {
        "GroupManagedServiceAccounts": [
            {
                "Name": "WebApp01",
                "Scope": "example.com"
            }
        ],
        "HostAccountConfig": {
            "PortableCcgVersion": "1",
            "PluginGUID": "{859E1386-BDB4-49E8-85C7-3070B13920E1}",
            "PluginInput": {
                "CredentialArn": "arn:aws:secretsmanager:aws-
region:111122223333:secret:MySecret"
            }
        }
    }
}
```

建立CredSpec

您可以在加入網域的 Windows 電腦上使用 CredSpec PowerShell 模組來建立 CredSpec。請依照 Microsoft Learn 網站上建立憑證規格中的步驟進行。

用gMSA於 Far Linux gate 上的容器

Amazon ECS 透過一種稱為群組受管服務帳戶 (gMSA) 的特殊服務帳戶，支援 Fargate 上 Linux 容器的作用中目錄身份驗證。

.NET Core 應用程式等以 Linux 為基礎的網路應用程式，可使用 Active Directory 來促進使用者和服務之間的身分驗證和授權管理。您可以透過設計與 Active Directory 整合並在加入網域的伺服器上執行的應用程式來使用此功能。但是，由於 Linux 容器無法加入網域，因此您需要設定 Linux 容器來執行 gMSA。

考量事項

在 Fargate 上使用gMSALinux容器之前，請考慮以下事項：

- 您必須執行平台版本 1.4 或更新版本。
- 您可能需要加入網域的 Windows 電腦才能完成先決條件。例如，您可能需要加入網域的 Windows 電腦，才能在使用 PowerShell 的 Active Directory 中建立 gMSA。「RSAT作用中導演」PowerShell 工具僅適用於Windows。如需詳細資訊，請參閱安裝 Active Directory 管理工具。
- 您必須使用無網域gMSA。

  Amazon ECS 使用 Active Directory 憑證規格檔案 (CredSpec)。此檔案包含用於將 gMSA 帳戶內容傳播至容器的 gMSA 中繼資料。您可以產生CredSpec檔案，然後將其存放在 Amazon S3 儲存貯體中。
- 一個工作只能支援一個作用中目錄。

必要條件

在搭配 Amazon ECS 使用 Linux 容器功能適用的 gMSA 之前，請先完成下列各項：

- 您可以使用您希望容器存取的資源來設定 Active Directory 網域。Amazon ECS 支援下列設定：
  - 一個 AWS Directory Service 活動目錄。 AWS Directory Service 是託 AWS 管在 Amazon EC2 上的受管活動目錄。如需詳細資訊，請參閱《 AWS 管理指南》中的受管理 Microsoft AD 入門。AWS Directory Service
  - 內部部署 Active Directory。您必須確認 Amazon ECS Linux 容器執行個體可以加入網域。如需詳細資訊，請參閱 AWS Direct Connect。
- 您在 Active Directory 中有一個現有gMSA帳戶，以及具有存取gMSA服務帳戶權限的使用者。如需詳細資訊，請參閱 建立無網域 gMSA 的 Active Directory 使用者。

- 你有一個 Amazon S3 存儲桶。如需詳細資訊，請參閱 Amazon S3 使用者指南中的建立儲存貯體。

在 Amazon ECS 上設定能夠使用 gMSA 功能的 Linux 容器

準備基礎設施

下列步驟是執行一次的考量和設定。

- 建立無網域 gMSA 的 Active Directory 使用者

  當您使用無網域時gMSA，容器不會加入網域。在容器上執行的其他應用程式無法使用認證來存取網域。使用不同網域的工作可以在同一個容器上執行。您可以在CredSpec檔案中 AWS Secrets Manager 提供密碼的名稱。祕密必須包含使用者名稱、密碼和要登入的網域。

  此功能與 gMSA support for non-domain-joined container hosts 功能類似。如需有關 Windows 功能的詳細資訊，請參閱 Microsoft Learn 網站上的 gMSA 架構和改進。

  a. 在您的作用中目錄網域中設定使用者。Active Directory 中的使用者必須具有存取您在工作中使用的gMSA服務帳戶的權限。

  b. 您擁有可以解析 Active Directory 網域的 VPC 和子網路。使用指向作用中目錄服務名稱的網域名稱設定 DHCP 選項的 VPC。如需如何為 VPC 設定 DHCP 選項的詳細資訊，請參閱 Amazon Virtual Private Cloud 使用者指南中的使用 DHCP 選項集。

  c. 在中建立密碼 AWS Secrets Manager。

  d. 建立認證規格檔案。

設定許可和祕密

針對每個應用程式和每個工作定義執行下列步驟一次。我們建議您使用授予最低權限的最佳實務，並縮減政策中使用的許可。這樣一來，每個任務都只能讀取所需的祕密。

1. 在 Active Directory 網域中建立使用者。Active Directory 中的使用者必須具有存取您在任務中使用之 gMSA 服務帳戶的權限。
2. 建立使用中目錄使用者之後，請在中建立密碼 AWS Secrets Manager。如需詳細資訊，請參閱建立 AWS Secrets Manager 祕密。
3. 將使用者的使用者名稱、密碼和網域分別輸入到名為 `username`、`password` 和 `domainName` 的 JSON 鍵值組中。

```
{"username":"username","password":"passw0rd", "domainName":"example.com"}
```

4. 您必須將以下許可作為內嵌政策新增至任務執行 IAM 角色。這樣做可讓 `credentials-fetcher` 常駐程式存取 Secrets Manager 的祕密。將 `MySecret` 替換為在 `Resource` 清單中含有祕密的 Amazon Resource Name (ARN) 的範例。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "secretsmanager:GetSecretValue"
      ],
      "Resource": [
        "arn:aws:secretsmanager:aws-region:111122223333:secret:MySecret"
      ]
    }
  ]
}
```

> **Note**
>
> 如果您使用自己的 KMS 金鑰來加密密碼，則必須將必要的權限新增至此角色，並將此角色新增至金 AWS KMS 鑰原則。

5. 將憑證規格新增至 Amazon S3 儲存貯體。然後，在任務定義的 `credentialSpecs` 欄位中參考 Amazon S3 儲存貯體的 Amazon Resource Name (ARN)。

```
{
    "family": "",
    "executionRoleArn": "",
    "containerDefinitions": [
        {
            "name": "",
            ...
            "credentialSpecs": [
                "credentialspecdomainless:arn:aws:s3:::${BucketName}/${ObjectName}"
            ],
            ...
```

```
        }
    ],
    ...
}
```

若要讓您的任務能夠存取 S3 儲存貯體，請新增下列許可做為 Amazon ECS 任務執行 IAM 角色的內嵌政策。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:ListObject"
            ],
            "Resource": [
                "arn:aws:s3:::{bucket_name}",
                "arn:aws:s3:::{bucket_name}/{object}"
            ]
        }
    ]
}
```

憑證規格檔案

Amazon ECS 使用 Active Directory 憑證規格檔案 (CredSpec)。此檔案包含用於將 gMSA 帳戶內容傳播至容器 Linux 的 gMSA 中繼資料。您會產生 CredSpec，並在任務定義的 `credentialSpecs` 欄位中參考該檔案。CredSpec 檔案不包含任何祕密。

以下是範例 CredSpec 檔案。

```
{
    "CmsPlugins": [
        "ActiveDirectory"
    ],
    "DomainJoinConfig": {
        "Sid": "S-1-5-21-2554468230-2647958158-2204241789",
        "MachineAccountName": "WebApp01",
```

```
        "Guid": "8665abd4-e947-4dd0-9a51-f8254943c90b",
        "DnsTreeName": "example.com",
        "DnsName": "example.com",
        "NetBiosName": "example"
    },
    "ActiveDirectoryConfig": {
        "GroupManagedServiceAccounts": [
            {
                "Name": "WebApp01",
                "Scope": "example.com"
            }
        ],
        "HostAccountConfig": {
            "PortableCcgVersion": "1",
            "PluginGUID": "{859E1386-BDB4-49E8-85C7-3070B13920E1}",
            "PluginInput": {
                "CredentialArn": "arn:aws:secretsmanager:aws-
region:111122223333:secret:MySecret"
            }
        }
    }
}
```

創建一個CredSpec並將其上傳到 Amazon S3

您可以在加入網域的 Windows 電腦上使用 CredSpec PowerShell 模組來建立 CredSpec。請依照 Microsoft Learn 網站上建立憑證規格中的步驟進行。

建立登入資料規格檔案後，將其上傳到 Amazon S3 儲存貯體。將 CredSpec 檔案複製至正在執行 AWS CLI 命令的電腦或環境中。

執行下列 AWS CLI 命令以上傳CredSpec至 Amazon S3。將 `MyBucket` 取代為您的 Amazon S3 儲存貯體的名稱。您可以將檔案作為物件儲存在任何儲存貯體和位置，但必須在附加至任務執行角色的政策中允許存取該儲存貯體和位置。

對於 PowerShell，請使用下列命令：

```
$ Write-S3Object -BucketName "MyBucket" -Key "ecs-domainless-gmsa-credspec" -File
 "gmsa-cred-spec.json"
```

下列 AWS CLI 命令會使用與相容 shell 所sh使用的反斜線接續字元。

```
$ aws s3 cp gmsa-cred-spec.json \
```

```
s3://MyBucket/ecs-domainless-gmsa-credspec
```

使用 Amazon ECS 視窗容器搭配無網域gMSA使用 AWS CLI

以下教學課程示範如何建立一個執行 Windows 容器的 Amazon ECS 任務，此容器具有透過 AWS CLI 存取 Active Directory 的憑證。使用無網域 gMSA 時，容器執行個體不會加入網域，執行個體上的其他應用程式無法使用憑證來存取網域，而加入不同網域的任務則可在相同的執行個體上執行。

主題

- 必要條件
- 步驟 1：在 Active Directory Domain Services (AD DS) 上建立和設定 gMSA 帳戶
- 步驟 2：將憑證上傳至 Secrets Manager
- 步驟 3：修改您的 CredSpec JSON 以包含無網域 gMSA 資訊
- 步驟 4：將 CredSpec 上傳至 Amazon S3
- 步驟 5：(選用) 建立 Amazon ECS 叢集
- 步驟 6：針對容器執行個體建立 IAM 角色
- 步驟 7：建立自訂任務執行角色
- 步驟 8：為 Amazon ECS Exec 建立任務角色
- 步驟 9：註冊使用無網域 gMSA 的任務定義
- 步驟 10：將 Windows 容器執行個體註冊至叢集
- 步驟 11：驗證容器執行個體
- 步驟 12：執行 Windows 任務
- 步驟 13：驗證容器具有 gMSA 憑證
- 步驟 14：清除
- 偵錯 Windows 容器的 Amazon ECS 無網域 gMSA

必要條件

本教學課程假設已完成下列先決條件：

- 已完成「設定以使用 Amazon ECS。」中的步驟。
- 您的 AWS 使用者具有 亞馬遜 FullAccess IAM 政策範例中指定的必要許可。

- 已安裝並設定最新版本的。 AWS CLI 如需有關安裝或升級的詳細資訊 AWS CLI，請參閱安裝 AWS Command Line Interface.
- 您可以使用您希望容器存取的資源來設定 Active Directory 網域。Amazon ECS 支援下列設定：
  - 一個 AWS Directory Service 活動目錄。 AWS Directory Service 是託 AWS 管在 Amazon EC2 上的受管活動目錄。如需詳細資訊，請參閱《 AWS 管理指南》中的受管理 Microsoft AD 入門。AWS Directory Service
  - 內部部署 Active Directory。您必須確認 Amazon ECS Linux 容器執行個體可以加入網域。如需詳細資訊，請參閱 AWS Direct Connect。
- 您擁有可以解析 Active Directory 網域的 VPC 和子網路。
- 您可以選擇無網域 gMSA 或將每個執行個體加入單一網域。使用無網域 gMSA 時，容器執行個體不會加入網域，執行個體上的其他應用程式無法使用憑證來存取網域，而加入不同網域的任務則可在相同的執行個體上執行。

  然後，選擇 CredSpec 的資料儲存，也可以選擇無網域 gMSA 的 Active Directory 使用者憑證的資料儲存。

  Amazon ECS 使用 Active Directory 憑證規格檔案 (CredSpec)。此檔案包含用於將 gMSA 帳戶內容傳播至容器的 gMSA 中繼資料。您會產生 CredSpec 檔案，然後將其儲存在下表中特定於容器執行個體的作業系統的 CredSpec 儲存選項之一。若要使用無網域方法，CredSpec 檔案中的選擇性區段可以在下表中的 domainless user credentials 儲存選項之一中指定憑證 (特定於容器執行個體的作業系統)。

  gMSA 資料儲存選項 (依作業系統)

儲存位置	Linux	Windows
Amazon Simple Storage Service	CredSpec	CredSpec
AWS Secrets Manager	無網域使用者憑證	無網域使用者憑證
Amazon EC2 Systems Manager 參數存放區	CredSpec	CredSpec，無網域使用者憑證
本機檔案	N/A	CredSpec

- (選用) AWS CloudShell 這項工具可為客戶提供命令列，而不需要建立自己的 EC2 執行個體。如需詳細資訊，請參閱什麼是 AWS CloudShell？ 在《AWS CloudShell 使用者指南》中。

步驟 1：在 Active Directory Domain Services (AD DS) 上建立和設定 gMSA 帳戶

在 Active Directory 網域上建立並設定 gMSA 帳戶。

1. 產生金鑰分佈服務根金鑰

   > **Note**
   >
   > 如果您正在使用 AWS Directory Service，則可以跳過此步驟。

   KDS 根金鑰和 gMSA 許可是使用您 AWS 受管的 Microsoft AD 來設定。

   如果您尚未在網域中建立 gMSA 服務帳戶，則必須先產生金鑰分佈服務 (KDS) 根金鑰。KDS 負責建立、輪換和向授權主機發佈 gMSA 密碼。當 `ccg.exe` 需要擷取 gMSA 憑證時，將會聯絡 KDS 以擷取目前密碼。

   若要檢查是否已建立 KDS 根金鑰，請使用模組在網域控制站上 PowerShell以網域管理員權限執行下列指令程式。`ActiveDirectory` PowerShell 如需有關此模組的詳細資訊，請參閱 Microsoft 學習網站上的ActiveDirectory 模組。

   ```
   PS C:\> Get-KdsRootKey
   ```

   如果命令傳回金鑰 ID，您可以略過此步驟的其餘部分。否則，執行下列命令以建立 KDS 根金鑰：

   ```
   PS C:\> Add-KdsRootKey -EffectiveImmediately
   ```

   儘管命令的引數 `EffectiveImmediately` 意味著金鑰會立即生效，但您需要等待 10 小時才能複製 KDS 根金鑰並可在所有網域控制器上使用。

2. 建立 gMSA 帳戶

   若要建立gMSA帳戶並允許擷`ccg.exe`取gMSA密碼，請從具有網域存取權的 Windows Server 或用戶端執行下列 PowerShell 命令。將 `ExampleAccount` 替換為您想要的 gMSA 帳戶名稱。

   a.
   ```
   PS C:\> Install-WindowsFeature RSAT-AD-PowerShell
   ```

b.
```
PS C:\> New-ADGroup -Name "ExampleAccount Authorized Hosts" -SamAccountName
 "ExampleAccountHosts" -GroupScope DomainLocal
```

c.
```
PS C:\> New-ADServiceAccount -Name "ExampleAccount" -DnsHostName
 "contoso" -ServicePrincipalNames "host/ExampleAccount", "host/contoso" -
PrincipalsAllowedToRetrieveManagedPassword "ExampleAccountHosts"
```

d. 建立一個具有不會過期的永久密碼的使用者。這些認證會儲存在中， AWS Secrets Manager 並由每個工作用來加入網域。

```
PS C:\> New-ADUser -Name "ExampleAccount" -AccountPassword (ConvertTo-
SecureString -AsPlainText "Test123" -Force) -Enabled 1 -PasswordNeverExpires 1
```

e.
```
PS C:\> Add-ADGroupMember -Identity "ExampleAccountHosts" -Members
 "ExampleAccount"
```

f. 安裝在活動目錄中創建CredSpec對象的 PowerShell 模塊並輸出 CredSpec JSON。

```
PS C:\> Install-PackageProvider -Name NuGet -Force
```

```
PS C:\> Install-Module CredentialSpec
```

g.
```
PS C:\> New-CredentialSpec -AccountName ExampleAccount
```

3. 將上一個命令的 JSON 輸出複製到名為 `gmsa-cred-spec.json` 的檔案中。這是 CredSpec 檔案。其在步驟 3 步驟 3：修改您的 CredSpec JSON 以包含無網域 gMSA 資訊 中使用。

步驟 2：將憑證上傳至 Secrets Manager

將 Active Directory 憑證複製至安全的憑證儲存系統中，以便每個任務都能擷取此憑證。這是無網域 gMSA 方法。使用無網域 gMSA 時，容器執行個體不會加入網域，執行個體上的其他應用程式無法使用憑證來存取網域，而加入不同網域的任務則可在相同的執行個體上執行。

此步驟會使用 AWS CLI. 您可以在預設 Shell (即 `bash`) 的 AWS CloudShell 中執行這些命令。

- 執行下列 AWS CLI 命令，並取代使用者名稱、密碼和網域名稱，以符合您的環境。保留密碼的 ARN 以在下一個步驟中使用，步驟 3：修改您的 CredSpec JSON 以包含無網域 gMSA 資訊

以下命令使用由 sh 和相容 shell 所使用的反斜線接續字元。這個命令不相容於 PowerShell. 您必須修改指令才能與其搭配使用 PowerShell。

```
$ aws secretsmanager create-secret \
--name gmsa-plugin-input \
--description "Amazon ECS - gMSA Portable Identity." \
--secret-string "{\"username\":\"ExampleAccount\",\"password\":\"Test123\",
\"domainName\":\"contoso.com\"}"
```

步驟 3：修改您的 CredSpec JSON 以包含無網域 gMSA 資訊

在將 CredSpec 上傳至其中一個儲存選項之前，請使用上一個步驟中 Secrets Manager 中密碼的 ARN 將資訊新增至 CredSpec。如需詳細資訊，請參閱 Microsoft Learn 網站上的 non-domain-joined 容器主機使用案例的其他認證規格組態。

1. 將下列資訊新增至 `ActiveDirectoryConfig` 內的 CredSpec 檔案。將 ARN 替換為上一個步驟 Secrets Manager 中的密碼。

   請注意，`PluginGUID` 值必須與下列範例程式碼片段中的 GUID 相符，且為必要值。

   ```
   "HostAccountConfig": {
         "PortableCcgVersion": "1",
         "PluginGUID": "{859E1386-BDB4-49E8-85C7-3070B13920E1}",
         "PluginInput": "{\"credentialArn\": \"arn:aws:secretsmanager:aws-
   region:111122223333:secret:gmsa-plugin-input\"}"
       }
   ```

   您也可以透過使用以下格式的 ARN，在 SSM 參數存放區中使用祕密：`\"arn:aws:ssm:aws-region:111122223333:parameter/gmsa-plugin-input\"`。
2. 修改 CredSpec 檔案之後，檔案應類似於以下範例：

   ```
   {
     "CmsPlugins": [
       "ActiveDirectory"
     ],
     "DomainJoinConfig": {
       "Sid": "S-1-5-21-4066351383-705263209-1606769140",
       "MachineAccountName": "ExampleAccount",
       "Guid": "ac822f13-583e-49f7-aa7b-284f9a8c97b6",
   ```

```
    "DnsTreeName": "contoso",
    "DnsName": "contoso",
    "NetBiosName": "contoso"
  },
  "ActiveDirectoryConfig": {
    "GroupManagedServiceAccounts": [
      {
        "Name": "ExampleAccount",
        "Scope": "contoso"
      },
      {
        "Name": "ExampleAccount",
        "Scope": "contoso"
      }
    ],
    "HostAccountConfig": {
      "PortableCcgVersion": "1",
      "PluginGUID": "{859E1386-BDB4-49E8-85C7-3070B13920E1}",
      "PluginInput": "{\"credentialArn\": \"arn:aws:secretsmanager:aws-
region:111122223333:secret:gmsa-plugin-input\"}"
    }
  }
}
```

步驟 4：將 CredSpec 上傳至 Amazon S3

此步驟會使用 AWS CLI. 您可以在預設 Shell (即 bash) 的 AWS CloudShell 中執行這些命令。

1. 將 CredSpec 檔案複製至正在執行 AWS CLI 命令的電腦或環境中。
2. 執行下列 AWS CLI 命令以上傳CredSpec至 Amazon S3。將 MyBucket 取代為您的 Amazon S3 儲存貯體的名稱。您可以將檔案作為物件儲存在任何儲存貯體和位置，但必須在附加至任務執行角色的政策中允許存取該儲存貯體和位置。

   以下命令使用由 sh 和相容 shell 所使用的反斜線接續字元。這個命令不相容於 PowerShell. 您必須修改指令才能與其搭配使用 PowerShell。

```
$ aws s3 cp gmsa-cred-spec.json \
s3://MyBucket/ecs-domainless-gmsa-credspec
```

步驟 5：(選用) 建立 Amazon ECS 叢集

預設情況下，您的帳戶會有一個名為 `default` 的 Amazon ECS 叢集。依預設會在 AWS CLI、SDK 和 AWS CloudFormation中使用此叢集。您可以使用其他叢集來分組和組織任務和基礎結構，並為某些組態指派預設值。

您可以從 AWS Management Console、 AWS CLI、SDK 或 AWS CloudFormation建立叢集。叢集中的設定和組態不會影響 gMSA。

此步驟會使用 AWS CLI. 您可以在預設 Shell (即 `bash`) 的 AWS CloudShell 中執行這些命令。

```
$ aws ecs create-cluster --cluster-name windows-domainless-gmsa-cluster
```

> **Important**
>
> 如果您選擇建立自己的叢集，您必須在打算對該叢集使用的每個命令中指定 `--cluster clusterName`。

步驟 6：針對容器執行個體建立 IAM 角色

容器執行個體是在 ECS 任務中執行容器的主機電腦，例如 Amazon EC2 執行個體。每個容器執行個體都會註冊至 Amazon ECS 叢集。在您啟動 Amazon EC2 執行個體並將其註冊到叢集之前，您必須先為要使用的容器執行個體建立 IAM 角色。

若要建立容器執行個體角色，請參閱 Amazon ECS 容器執行個體 IAM 角色。預設值 `ecsInstanceRole` 具有足夠的許可來完成此教學課程。

步驟 7：建立自訂任務執行角色

Amazon ECS 可針對啟動每個任務所需的許可使用不同的 IAM 角色，而不是容器執行個體角色。此角色是任務執行角色。我們建議您建立僅具有 ECS 執行任務所需許可的任務執行角色，也稱為最低權限許可。如需有關最低權限原則的詳細資訊，請參閱《AWS Well-Architected 的架構》中的 SEC03-BP02 授予最低權限存取權。

1. 若要建立任務執行角色，請參閱 建立任務執行 角色。預設許可允許容器執行個體從 Amazon 彈性容器登錄中提取容器映像，以`stdout`及`stderr`從您的應用程式提取容器映像，以記錄到 Amazon CloudWatch 日誌。

由於角色需要本教學課程的自訂許可，因此您可以為該角色指定與 ecsTaskExecutionRole 不同的名稱。本教學課程將在後續步驟中使用 ecsTaskExecutionRole。

2. 建立自訂原則 (僅存在於此角色的內嵌政策或可重複使用的政策)，以新增下列許可。將第一個陳述式中 Resource 的 ARN 替換為 Amazon S3 儲存貯體和位置，並將第二個 Resource 替換為 Secrets Manager 中密碼的 ARN。

   如果您使用自訂金鑰在 Secrets Manager 中加密密碼，則也必須允許金鑰使用 kms:Decrypt。

   如果您使用 SSM 參數存放區而不是 Secrets Manager，您必須允許參數使用 ssm:GetParameter，而不是 secretsmanager:GetSecretValue。

```
{
"Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject"
      ],
      "Resource": "arn:aws:s3:::MyBucket/ecs-domainless-gmsa-credspec/gmsa-cred-spec.json"
    },
    {
      "Effect": "Allow",
      "Action": [
        "secretsmanager:GetSecretValue"
      ],
      "Resource": "arn:aws:secretsmanager:aws-region:111122223333:secret:gmsa-plugin-input"
    }
  ]
}
```

步驟 8：為 Amazon ECS Exec 建立任務角色

本教學課程使用 Amazon ECS Exec，透過在執行中的任務內執行命令來驗證功能。若要使用 ECS Exec，服務或任務必須開啟 ECS Exec，且任務角色 (但不是任務執行角色) 必須具有 ssmmessages 許可。如需了解所需的 IAM 政策，請參閱ECS 執行權限。

此步驟會使用 AWS CLI. 您可以在預設 Shell (即 bash) 的 AWS CloudShell 中執行這些命令。

若要使用建立任務角色 AWS CLI，請遵循下列步驟。

1. 建立稱為 ecs-tasks-trust-policy.json 的檔案，其中具有以下內容：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "ecs-tasks.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

2. 建立 IAM 角色。您可以替換名稱 ecs-exec-demo-task-role，但保留以下步驟的名稱。

   以下命令使用由 sh 和相容 shell 所使用的反斜線接續字元。這個命令不相容於 PowerShell. 您必須修改指令才能與其搭配使用 PowerShell。

```
$ aws iam create-role --role-name ecs-exec-demo-task-role \
--assume-role-policy-document file://ecs-tasks-trust-policy.json
```

   您可以刪除檔案 ecs-tasks-trust-policy.json。

3. 建立稱為 ecs-exec-demo-task-role-policy.json 的檔案，其中具有以下內容：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ssmmessages:CreateControlChannel",
                "ssmmessages:CreateDataChannel",
                "ssmmessages:OpenControlChannel",
                "ssmmessages:OpenDataChannel"
            ],
            "Resource": "*"
        }
    ]
```

```
}
```

4. 建立 IAM 政策並將其附加至上一個步驟中的角色。

   以下命令使用由 sh 和相容 shell 所使用的反斜線接續字元。這個命令不相容於 PowerShell. 您必須修改指令才能與其搭配使用 PowerShell。

```
$ aws iam put-role-policy \
    --role-name ecs-exec-demo-task-role \
    --policy-name ecs-exec-demo-task-role-policy \
    --policy-document file://ecs-exec-demo-task-role-policy.json
```

   您可以刪除檔案 ecs-exec-demo-task-role-policy.json。

步驟 9：註冊使用無網域 gMSA 的任務定義

此步驟會使用 AWS CLI. 您可以在預設 Shell (即 bash) 的 AWS CloudShell 中執行這些命令。

1. 建立稱為 windows-gmsa-domainless-task-def.json 的檔案，其中具有以下內容：

```
{
  "family": "windows-gmsa-domainless-task",
  "containerDefinitions": [
    {
      "name": "windows_sample_app",
      "image": "mcr.microsoft.com/windows/servercore/iis",
      "cpu": 1024,
      "memory": 1024,
      "essential": true,
      "credentialSpecs": [
                "credentialspecdomainless:arn:aws:s3:::ecs-domainless-gmsa-
credspec/gmsa-cred-spec.json"
      ],
      "entryPoint": [
        "powershell",
        "-Command"
      ],
      "command": [
        "New-Item -Path C:\\inetpub\\wwwroot\\index.html -ItemType file -Value
 '<html> <head> <title>Amazon ECS Sample App</title> <style>body {margin-top:
 40px; background-color: #333;} </style> </head><body> <div style=color:white;text-
```

```
align:center> <h1>Amazon ECS Sample App</h1> <h2>Congratulations!</h2> <p>Your
 application is now running on a container in Amazon ECS.</p>' -Force ; C:\
\ServiceMonitor.exe w3svc"
      ],
      "portMappings": [
        {
          "protocol": "tcp",
          "containerPort": 80,
          "hostPort": 8080
        }
      ]
    }
  ],
  "taskRoleArn": "arn:aws:iam::111122223333:role/ecs-exec-demo-task-role",
  "executionRoleArn": "arn:aws:iam::111122223333:role/ecsTaskExecutionRole"
}
```

2. 執行下列命令來註冊任務定義：

   以下命令使用由 sh 和相容 shell 所使用的反斜線接續字元。這個命令不相容於 PowerShell. 您必須修改指令才能與其搭配使用 PowerShell。

```
$ aws ecs register-task-definition \
--cli-input-json file://windows-gmsa-domainless-task-def.json
```

步驟 10：將 Windows 容器執行個體註冊至叢集

啟動 Amazon EC2 Windows 執行個體並執行 ECS 容器代理程式，以將其註冊為叢集中的容器執行個體。ECS 會在已註冊至叢集 (任務在其中啟動) 的容器執行個體上執行任務。

1. 若要啟動針對 Amazon ECS 設定的亞馬遜 EC2 視窗執行個體 AWS Management Console，請參閱啟動 Amazon ECS Windows 容器執行個體。在使用者資料的步驟停止。
2. 若為 gMSA，使用者資料必須先設定環境變數 `ECS_GMSA_SUPPORTED`，才能啟動 ECS 容器代理程式。

   對於 ECS Exec，代理程式必須以引數 `-EnableTaskIAMRole` 開頭。

   若要防止任務到達 EC2 IMDS Web 服務以擷取角色憑證，以保護執行個體 IAM 角色，請新增引數 `-AwsvpcBlockIMDS`。這只適用於使用 `awsvpc` 網路模式的任務。

```
<powershell>
[Environment]::SetEnvironmentVariable("ECS_GMSA_SUPPORTED", $TRUE, "Machine")
Import-Module ECSTools
Initialize-ECSAgent -Cluster windows-domainless-gmsa-cluster -EnableTaskIAMRole -
AwsvpcBlockIMDS
</powershell>
```

3. 檢閱 Summary (摘要) 面板中執行個體組態的摘要，並在準備就緒時選擇 Launch instance (啟動執行個體)。

步驟 11：驗證容器執行個體

您可以使用 AWS Management Console驗證叢集中是否存在容器執行個體。但是，gMSA 需要指示為屬性的其他功能。這些屬性在中不可見 AWS Management Console，因此本教學課程會使用 AWS CLI.

此步驟會使用 AWS CLI. 您可以在預設 Shell (即 bash) 的 AWS CloudShell 中執行這些命令。

1. 列出叢集中的容器執行個體。容器執行個體的 ID 與 EC2 執行個體的 ID 不同。

```
$ aws ecs list-container-instances
```

輸出：

```
{
    "containerInstanceArns": [
        "arn:aws:ecs:aws-region:111122223333:container-
instance/default/MyContainerInstanceID"
    ]
}
```

例如，526bd5d0ced448a788768334e79010fd 是有效的容器執行個體 ID。

2. 使用上一個步驟中的容器執行個體 ID 來取得容器執行個體的詳細資訊。使用 ID 取代 MyContainerInstanceID。

以下命令使用由 sh 和相容 shell 所使用的反斜線接續字元。這個命令不相容於 PowerShell. 您必須修改指令才能與其搭配使用 PowerShell。

```
$ aws ecs describe-container-instances \
     ----container-instances MyContainerInstanceID
```

請注意，輸出內容非常長。

3. 驗證 attributes 清單中是否存在一個索引鍵為 name 且值為 ecs.capability.gmsa-domainless 的物件。下列為物件的範例。

   輸出：

```
{
    "name": "ecs.capability.gmsa-domainless"
}
```

步驟 12：執行 Windows 任務

執行 Amazon ECS 任務。如果叢集中只有 1 個容器執行個體，您可以使用 run-task。如果有許多不同的容器執行個體，則相較於將置放條件限制新增至任務定義，以控制要執行此任務的容器執行個體類型，使用 start-task 並指定要在其上執行任務的容器執行個體 ID 可能會更為容易。

此步驟會使用 AWS CLI. 您可以在預設 Shell (即 bash) 的 AWS CloudShell 中執行這些命令。

1. 以下命令使用由 sh 和相容 shell 所使用的反斜線接續字元。這個命令不相容於 PowerShell. 您必須修改指令才能與其搭配使用 PowerShell。

```
aws ecs run-task --task-definition windows-gmsa-domainless-task \
    --enable-execute-command --cluster windows-domainless-gmsa-cluster
```

   請注意，任務 ID 是由命令傳回。

2. 執行下列命令，以驗證任務已開始。這個命令會等待，直到任務開始時才傳回 Shell 提示字元。使用前一步驟中的任務 ID，取代 MyTaskID。

```
$ aws ecs wait tasks-running --task MyTaskID
```

步驟 13：驗證容器具有 gMSA 憑證

驗證任務中的容器具有 Kerberos 權杖。gMSA

此步驟會使用 AWS CLI. 您可以在預設 Shell (即 bash) 的 AWS CloudShell 中執行這些命令。

1. 以下命令使用由 sh 和相容 shell 所使用的反斜線接續字元。這個命令不相容於 PowerShell. 您必須修改指令才能與其搭配使用 PowerShell。

```
$ aws ecs execute-command \
--task MyTaskID \
--container windows_sample_app \
--interactive \
--command powershell.exe
```

   輸出將是一個 PowerShell 提示。

2. 在容器內的 PowerShell 終端中運行以下命令。

```
PS C:\> klist get ExampleAccount$
```

   在輸出中，請注意 Principal 是您先前所建立的項目。

步驟 14：清除

完成此教學課程時，建議您清除相關聯的資源，以免未使用的資源產生費用。

此步驟會使用 AWS CLI. 您可以在預設 Shell (即 bash) 的 AWS CloudShell 中執行這些命令。

1. 停止任務。使用步驟 12 中的任務 ID 步驟 12：執行 Windows 任務 取代 MyTaskID。

```
$ aws ecs stop-task --task MyTaskID
```

2. 終止 Amazon EC2 執行個體。之後，叢集中的容器執行個體將會在一小時後自動刪除。

   您可以使用 Amazon EC2 主控台來尋找和終止執行個體。或者，您可以執行下列命令。若要執行命令，請在步驟 1 步驟 11：驗證容器執行個體 的 aws ecs describe-container-instances 命令輸出中尋找 EC2 執行個體識別碼。i-10a64379 是 EC2 執行個體 ID 的範例。

```
$ aws ec2 terminate-instances --instance-ids MyInstanceID
```

3. 刪除 Amazon S3 中的 CredSpec 檔案。將 `MyBucket` 取代為您的 Amazon S3 儲存貯體的名稱。

```
$ aws s3api delete-object --bucket MyBucket --key ecs-domainless-gmsa-credspec/
gmsa-cred-spec.json
```

4. 從 Secrets Manager 中刪除密碼。如果您改用 SSM 參數存放區，請刪除參數。

   以下命令使用由 sh 和相容 shell 所使用的反斜線接續字元。這個命令不相容於 PowerShell. 您必須修改指令才能與其搭配使用 PowerShell。

```
$ aws secretsmanager delete-secret --secret-id gmsa-plugin-input \
    --force-delete-without-recovery
```

5. 取消註冊並刪除任務定義。透過取消註冊任務定義，您可以將其標記為非作用中狀態，以便無法用於啟動新任務。然後，您可以刪除任務定義。

   a. 透過指定版本取消註冊任務定義。ECS 會自動建立任務定義的版本，編號從 1 開始。您引用的版本與容器映像上的標籤格式相同，例如 `:1`。

```
$ aws ecs deregister-task-definition --task-definition windows-gmsa-domainless-
task:1
```

   b. 刪除任務定義。

```
$ aws ecs delete-task-definitions --task-definition windows-gmsa-domainless-
task:1
```

6. (選用) 如果您已建立叢集，請刪除 ECS 叢集。

```
$ aws ecs delete-cluster --cluster windows-domainless-gmsa-cluster
```

偵錯 Windows 容器的 Amazon ECS 無網域 gMSA

Amazon ECS 任務狀態

ECS 會嘗試僅啟動一次任務。任何有問題的任務均會停止，並設定為狀態 `STOPPED`。任務有兩種常見的問題類型。首先是無法啟動的任務。其次是應用程式已在容器之一內停止的任務。在中

AWS Management Console，查看任務的「已停止原因」欄位，瞭解工作停止的原因。在 AWS CLI中，描述任務並查看 `stoppedReason`。如需和中的步驟 AWS Management Console AWS CLI，請參閱查看 Amazon ECS 停止的任務錯誤。

Windows 事件

容器中 gMSA 的 Windows 事件記錄在 `Microsoft-Windows-Containers-CCG` 日誌中，您可以在 `Logs\Microsoft\Windows\Containers-CCG\Admin`「應用程式和服務」區段的「事件檢視器」中找到。如需詳細的偵錯提示，請參閱 Microsoft Learn 網站上的 Windows 容器 gMSA 故障排除。

ECS 代理程式 gMSA 外掛程式

Windows 容器執行個體上 ECS 代理程式的 gMSA 外掛程式記錄位於下列目錄 `C:/ProgramData/Amazon/gmsa-plugin/` 中。查看此日誌，查看是否已從儲存位置 (例如 Secrets Manager) 下載無網域使用者憑證，以及是否正確讀取憑證格式。

了解如何將 GMA 用於適用於 Amazon ECS 的 EC2 視窗容器

Amazon ECS 透過名為群組受管服務帳戶 (gMSA) 的特殊類型的服務帳戶，支援對 Windows 容器的 Active Directory 身分驗證。

.NET 應用程式之類的以 Windows 為基礎的網路應用程式通常會使用 Active Directory 來促進使用者和服務之間的身分驗證和授權管理。開發人員通常會針對此目的設計其應用程式，以與 Active Directory 整合，並在已加入網域的伺服器上執行。由於以 Windows 為基礎的容器無法加入網域，因此您必須將 Windows 容器設定為與 gMSA 搭配執行。

使用 gMSA 執行的 Windows 容器依賴其主機 Amazon EC2 執行個體，以從 Active Directory 網域控制站擷取 gMSA 登入資料，並將這些資料提供給容器執行個體。如需詳細資訊，請參閱 為 Windows 容器建立 gMSA。

> **Note**
>
> Fargate 上的 Windows 容器不支援此功能。

主題

- 考量事項
- 必要條件
- 在 Amazon ECS 上設定 Windows 容器的 gMSA

考量事項

針對 Windows 容器使用 gMSA 時，應考慮下列事項：

- 針對您的容器執行個體使用 Amazon ECS 最佳化 Windows Server 2016 Full AMI 時，容器主機名稱必須與登入資料規格檔案中定義的 gMSA 帳戶名稱相同。若要指定容器的主機名稱，請使用 `hostname` 容器定義參數。如需詳細資訊，請參閱 Network settings (網路設定)。
- 您可以選擇無網域 gMSA 或將每個執行個體加入單一網域。使用無網域 gMSA 時，容器執行個體不會加入網域，執行個體上的其他應用程式無法使用憑證來存取網域，而加入不同網域的任務則可在相同的執行個體上執行。

  然後，選擇 CredSpec 的資料儲存，也可以選擇無網域 gMSA 的 Active Directory 使用者憑證的資料儲存。

  Amazon ECS 使用 Active Directory 憑證規格檔案 (CredSpec)。此檔案包含用於將 gMSA 帳戶內容傳播至容器的 gMSA 中繼資料。您會產生 CredSpec 檔案，然後將其儲存在下表中特定於容器執行個體的作業系統的 CredSpec 儲存選項之一。若要使用無網域方法，CredSpec 檔案中的選擇性區段可以在下表中的 domainless user credentials 儲存選項之一中指定憑證 (特定於容器執行個體的作業系統)。

  gMSA 資料儲存選項 (依作業系統)

儲存位置	Linux	Windows
Amazon Simple Storage Service	CredSpec	CredSpec
AWS Secrets Manager	無網域使用者憑證	無網域使用者憑證
Amazon EC2 Systems Manager 參數存放區	CredSpec	CredSpec，無網域使用者憑證
本機檔案	N/A	CredSpec

必要條件

在搭配 Amazon ECS 使用 Windows 容器功能適用的 gMSA 之前，請先完成下列各項：

- 您可以使用您希望容器存取的資源來設定 Active Directory 網域。Amazon ECS 支援下列設定：

  - 一個 AWS Directory Service 活動目錄。 AWS Directory Service 是託 AWS 管在 Amazon EC2 上的受管活動目錄。如需詳細資訊，請參閱《 AWS 管理指南》中的受管理 Microsoft AD 入門。AWS Directory Service
  - 內部部署 Active Directory。您必須確認 Amazon ECS Linux 容器執行個體可以加入網域。如需詳細資訊，請參閱 AWS Direct Connect。
- 您在 Active Directory 中擁有現有的 gMSA 帳戶。如需詳細資訊，請參閱 為 Windows 容器建立 gMSA。
- 您已選擇使用無網域 gMSA 或託管 Amazon ECS 任務的 Amazon ECS Windows 容器執行個體必須是加入 Active Directory 的網域，並且是具有 gMSA 帳戶存取權的 Active Directory 安全群組成員。

  使用無網域 gMSA 時，容器執行個體不會加入網域，執行個體上的其他應用程式無法使用憑證來存取網域，而加入不同網域的任務則可在相同的執行個體上執行。
- 您已新增必要的 IAM 許可 所需的許可取決於您為初始憑證選擇的方法以及儲存憑證規格的方法：
  - 如果您使用無網域作gMSA為初始登入資料，則 Amazon EC2 執行個體角色需要的 IAM 許可。AWS Secrets Manager
  - 如果您將憑證規格存放在 SSM 參數存放區中，則任務執行角色需要 Amazon EC2 Systems Manager 參數存放區的 IAM 許可。
  - 如果您將憑證規格存放在 Amazon S3 中，則任務執行角色需要 Amazon Simple Storage Service 的 IAM 許可。

在 Amazon ECS 上設定 Windows 容器的 gMSA

若要在 Amazon ECS 上設定 Windows 容器的 gMSA，您可以遵循包含設定先決條件 使用 Amazon ECS 視窗容器搭配無網域gMSA使用 AWS CLI 的完整教學課程。

以下章節詳細介紹了 CredSpec 組態。

主題

- 範例 CredSpec
- 無網域 gMSA 設定
- 在任務定義中參考登入資料規格檔案

範例 CredSpec

Amazon ECS 使用憑證規格檔案，其中包含將 gMSA 帳戶內容傳播到 Windows 容器所用的 gMSA 中繼資料。您可以產生登入資料規格檔案，並在任務定義的 `credentialSpec` 欄位中參考該檔案。登入資料規格檔案不包含任何秘密。

以下是範例登入資料規格檔案：

```
{
    "CmsPlugins": [
        "ActiveDirectory"
    ],
    "DomainJoinConfig": {
        "Sid": "S-1-5-21-2554468230-2647958158-2204241789",
        "MachineAccountName": "WebApp01",
        "Guid": "8665abd4-e947-4dd0-9a51-f8254943c90b",
        "DnsTreeName": "contoso.com",
        "DnsName": "contoso.com",
        "NetBiosName": "contoso"
    },
    "ActiveDirectoryConfig": {
        "GroupManagedServiceAccounts": [
            {
                "Name": "WebApp01",
                "Scope": "contoso.com"
            }
        ]
    }
}
```

無網域 gMSA 設定

我們建議使用無網域 gMSA，而不是將容器執行個體加入至單一網域。使用無網域 gMSA 時，容器執行個體不會加入網域，執行個體上的其他應用程式無法使用憑證來存取網域，而加入不同網域的任務則可在相同的執行個體上執行。

1. 在將 CredSpec 上傳至其中一個儲存選項之前，請使用 Secrets Manager 或 SSM 參數存放區中祕密的 ARN 將資訊新增 CredSpec。如需詳細資訊，請參閱 Microsoft Learn 網站上的 non-domain-joined容器主機使用案例的其他認證規格組態。

   無網域 gMSA 憑證格式

以下是您 Active Directory 的無網域 gMSA 憑證的 JSON 格式。將憑證儲存在 Secrets Manager 或 SSM 參數存放區中。

```
{
    "username":"WebApp01",
    "password":"Test123!",
    "domainName":"contoso.com"
}
```

2. 將下列資訊新增至 ActiveDirectoryConfig 內的 CredSpec 檔案。將 ARN 替代為 Secrets Manager 或 SSM 參數存放區中的祕密。

   請注意，PluginGUID 值必須與下列範例程式碼片段中的 GUID 相符，且為必要值。

```
    "HostAccountConfig": {
        "PortableCcgVersion": "1",
        "PluginGUID": "{859E1386-BDB4-49E8-85C7-3070B13920E1}",
        "PluginInput": "{\"credentialArn\": \"arn:aws:secretsmanager:aws-
region:111122223333:secret:gmsa-plugin-input\"}"
    }
```

   您也可以透過使用以下格式的 ARN，在 SSM 參數存放區中使用祕密：\"arn:aws:ssm:*aws-region*:111122223333:parameter/*gmsa-plugin-input*\"。

3. 修改 CredSpec 檔案之後，檔案應類似於以下範例：

```
{
  "CmsPlugins": [
    "ActiveDirectory"
  ],
  "DomainJoinConfig": {
    "Sid": "S-1-5-21-4066351383-705263209-1606769140",
    "MachineAccountName": "WebApp01",
    "Guid": "ac822f13-583e-49f7-aa7b-284f9a8c97b6",
    "DnsTreeName": "contoso",
    "DnsName": "contoso",
    "NetBiosName": "contoso"
  },
  "ActiveDirectoryConfig": {
    "GroupManagedServiceAccounts": [
      {
```

```
        "Name": "WebApp01",
        "Scope": "contoso"
      },
      {
        "Name": "WebApp01",
        "Scope": "contoso"
      }
    ],
    "HostAccountConfig": {
      "PortableCcgVersion": "1",
      "PluginGUID": "{859E1386-BDB4-49E8-85C7-3070B13920E1}",
      "PluginInput": "{\"credentialArn\": \"arn:aws:secretsmanager:aws-
region:111122223333:secret:gmsa-plugin-input\"}"
    }
  }
}
```

在任務定義中參考登入資料規格檔案

Amazon ECS 支援以下方法，在任務定義的 `credentialSpecs` 欄位中參考檔案路徑。根據您是將容器執行個體加入單一網域，還是分別使用無網域 gMSA，您都可以針對這些選項提供 `credentialspec:` 或 `domainlesscredentialspec:`。

Amazon S3 儲存貯體

將憑證規格新增到 Amazon S3 儲存貯體，然後在任務定義的 `credentialSpecs` 欄位中參考 Amazon S3 儲存貯體的 Amazon Resource Name (ARN)。

```
{
    "family": "",
    "executionRoleArn": "",
    "containerDefinitions": [
        {
            "name": "",
            ...
            "credentialSpecs": [
                "credentialspecdomainless:arn:aws:s3:::${BucketName}/${ObjectName}"
            ],
            ...
        }
    ],
    ...
```

```
}
```

您也必須新增下列許可做為 Amazon ECS 任務執行 IAM 角色的內嵌政策，讓任務能夠存取 Amazon S3 儲存貯體。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor",
            "Effect": "Allow",
            "Action": [
                "s3:Get*",
                "s3:List*"
            ],
            "Resource": [
                "arn:aws:s3:::{bucket_name}",
                "arn:aws:s3:::{bucket_name}/{object}"
            ]
        }
    ]
}
```

SSM 參數存放區參數

將憑證規格新增至 SSM 參數存放區參數，然後在任務定義的 `credentialSpecs` 欄位中參考 SSM 參數存放區參數的 Amazon Resource Name (ARN)。

```
{
    "family": "",
    "executionRoleArn": "",
    "containerDefinitions": [
        {
            "name": "",
            ...
            "credentialSpecs": [

 "credentialspecdomainless:arn:aws:ssm:region:111122223333:parameter/parameter_name"
            ],
            ...
        }
    ],
```

```
    ...
}
```

您也必須新增下列許可做為 Amazon ECS 任務執行 IAM 角色的內嵌政策，讓任務能夠存取 SSM 參數存放區參數。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ssm:GetParameters"
            ],
            "Resource": [
                "arn:aws:ssm:region:111122223333:parameter/parameter_name"
            ]
        }
    ]
}
```

本機檔案

使用本機檔案中的登入資料規格詳細資訊，在任務定義的 `credentialSpecs` 欄位中參考檔案路徑。參照的檔案路徑必須相對於 `C:\ProgramData\Docker\CredentialSpecs` 目錄，並使用反斜線 (「\」) 作為檔案路徑分隔符號。

```
{
    "family": "",
    "executionRoleArn": "",
    "containerDefinitions": [
        {
            "name": "",
            ...
            "credentialSpecs": [
                "credentialspec:file://CredentialSpecDir\CredentialSpecFile.json"
            ],
            ...
        }
    ],
    ...
}
```

使用 EC2 Image Builder 建置自訂的 Amazon ECS 最佳化 AMI

AWS 建議您使用 Amazon ECS 優化 AMI，因為它們已預先設定執行容器工作負載的需求和建議。有時您可能需要自定義 AMI 以添加其他軟件。您可以使用 EC2 Image Builder 來建立、管理和部署伺服器映像。您保留帳戶中建立的自訂影像的所有權。您可以使用 EC2 Image Builder 管道為映像自動更新和系統修補，或使用獨立命令使用定義的組態資源建立映像。

您為您的圖像創建一個配方。方案包括父映像檔以及任何其他元件。您還可以創建一個分發自定義 AMI 的管道。

您為您的圖像創建一個配方。Image Builder 映像配方是一份文件，用於定義基本影像，以及套用至基本影像的元件，以產生輸出 AMI 影像所需組態的文件。您還可以創建一個分發自定義 AMI 的管道。如需詳細資訊，請參閱 EC2 Image Builder 使用者指南中的 EC2 Image Builder 的運作方式。

我們建議您在 EC2 Image Builder 中使用下列其中一個 Amazon ECS 最佳化 AMI 作為您的「父映像」：

- Linux
  - Amazon ECS 優化 AL2023
  - Amazon ECS 最佳化 Amazon Linux 2023 (arm64) AMI
  - Amazon ECS 優化 Amazon Linux 2 內核 5 AMI
  - Amazon EC 優化的 Amazon Linux 2 x86 AMI
- Windows
  - Amazon ECS 優化視窗 2022 年完整版 x86
  - Amazon ECS 優化視窗 2022 核心
  - Amazon ECS 優化的視窗 2019 完整版 x86
  - Amazon ECS 優化視窗 2019 核心
  - Amazon ECS 優化視窗 2016 年全 x86

我們也建議您選擇「使用最新的可用操作系統版本」。管道會針對父映像使用語意版本化，這有助於偵測自動排程工作中的相依性更新。如需詳細資訊，請參閱 EC2 Image Builder 使用者指南中的語意版本控制。

AWS 使用安全修補程式和新的容器代理程式版本定期更新 Amazon ECS 最佳化 AMI 映像。在映像配方中使用 AMI ID 作為父映像時，您需要定期檢查父映像的更新。如果有更新，您必須使用更新的 AMI 建立配方的新版本。這樣可以確保您的自訂映像包含最新版本的父映像。如需如何建立工作流程以使用

新建立的 AMI 自動更新 Amazon ECS 叢集中的 EC2 執行個體的詳細資訊，請參閱如何建立 AMI 強化管道並自動更新 ECS 執行個體叢集。

您也可以指定透過受管 EC2 Image Builder 管道發佈的父映像的 Amazon 資源名稱 (ARN)。Amazon 定期透過受管管管道發佈 Amazon ECS 最佳化 AMI 映像。這些圖像是可公開訪問的。您必須擁有正確的權限才能存取影像。當您在映像產生器方案中使用映像 ARN 而非 AMI 時，管線會在每次執行時自動使用父映像的最新版本。這種方法消除了為每個更新手動創建新的配方版本的需要。

將映像 ARN 與基礎結構即程式碼 (IaC) 搭配使用

您可以使用 EC2 Image Builder 主控台或基礎設施即程式碼 (例如 AWS CloudFormation) 或 AWS SDK 來設定方案。當您在方案中指定父映像時，可以指定 EC2 AMI ID、Image Builder 映像 ARN、 AWS Marketplace 產品 ID 或容器映像。 AWS 公開發佈 Amazon ECS 最佳化 AMI 的 AMI ID 和映 Image Builder 影像 ARN。以下是圖像的 ARN 格式：

```
arn:${Partition}:imagebuilder:${Region}:${Account}:image/${ImageName}/${ImageVersion}
```

具ImageVersion有以下格式。將##、#要和####取代為最新值。

```
<major>.<minor>.<patch>
```

您可以patch將minor和取代為特定值major，或者您可以使用映像的無版本 ARN，以便您的管道保持 up-to-date最新版本的父映像。無版本 ARN 使用萬用字元格式 'x.x.x' 來表示映像版本。此方法可讓 Image Builder 服務自動解析為最新版本的映像檔。使用無版本 ARN 可確保您的參考始終指向可用的最新映像，從而簡化部署中維護最新映像的程序。當您使用主控台建立方案時，EC2 Image Builder 會自動識別父映像的 ARN。當您使用 IaC 來建立配方時，您必須識別 ARN 並選取所需的映像版本，或使用無版本 arn 來指示最新的可用映像。我們建議您建立自動化指令碼，以篩選並僅顯示符合您條件的影像。下列 Python 指令碼顯示如何擷取 Amazon ECS 最佳化 AMI 的清單。

該腳本接受兩個可選參數：owner和platform，默認值分別為「Amazon」和「Windows」。擁有者引數的有效值為：SelfShared、Amazon、和ThirdParty。平台引數的有效值為Windows和Linux。例如，如果執行指令碼並將owner引數設定為Amazon且設定為Linux，則指令碼會產生由 Amazon 發佈的映像清單，其中包括 Amazon ECS 優化映像。platform

```
import boto3
import argparse

def list_images(owner, platform):
    # Create a Boto3 session
    session = boto3.Session()
```

```
    # Create an EC2 Image Builder client
    client = session.client('imagebuilder')

    # Define the initial request parameters
    request_params = {
        'owner': owner,
        'filters': [
            {
                'name': 'platform',
                'values': [platform]
            }
        ]
    }

    # Initialize the results list
    all_images = []

    # Get the initial response with the first page of results
    response = client.list_images(**request_params)

    # Extract images from the response
    all_images.extend(response['imageVersionList'])

    # While 'nextToken' is present, continue paginating
    while 'nextToken' in response and response['nextToken']:
        # Update the token for the next request
        request_params['nextToken'] = response['nextToken']

        # Get the next page of results
        response = client.list_images(**request_params)

        # Extract images from the response
        all_images.extend(response['imageVersionList'])

    return all_images

def main():
    # Initialize the parser
    parser = argparse.ArgumentParser(description="List AWS images based on owner and
 platform")

    # Add the parameters/arguments
```

```
    parser.add_argument("--owner", default="Amazon", help="The owner of the images.
 Default is 'Amazon'.")
    parser.add_argument("--platform", default="Windows", help="The platform type of the
 images. Default is 'Windows'.")

    # Parse the arguments
    args = parser.parse_args()

    # Retrieve all images based on the provided owner and platform
    images = list_images(args.owner, args.platform)

    # Print the details of the images
    for image in images:
        print(f"Name: {image['name']}, Version: {image['version']}, ARN:
 {image['arn']}")

if __name__ == "__main__":
    main()
```

搭配使用圖像 ARN AWS CloudFormation

Image Builder 映像方案是一種藍圖，用於指定父映像以及達成輸出映像預定組態所需的元件。您使用的`AWS::ImageBuilder::ImageRecipe`資源。將`ParentImage`值設置為圖像 ARN。使用所需映像的無版本 ARN，以確保管線始終使用最新版本的映像。例如 `arn:aws:imagebuilder:us-east-1:aws:image/amazon-linux-2023-ecs-optimized-x86/x.x.x`。下列`AWS::ImageBuilder::ImageRecipe`資源定義使用 Amazon 受管映像檔 ARN：

```
ECSRecipe:
    Type: AWS::ImageBuilder::ImageRecipe
    Properties:
      Name: MyRecipe
      Version: '1.0.0'
      Components:
        - ComponentArn: [<The component arns of the image recipe>]
      ParentImage: "arn:aws:imagebuilder:us-east-1:aws:image/amazon-linux-2023-ecs-
optimized-x86/x.x.x"
```

若要取得有關資`AWS::ImageBuilder::ImageRecipe`源的更多資訊，請參閱AWS CloudFormation使用指南中的〈〉

您可以通過設置AWS::ImageBuilder::ImagePipeline資源的Schedule屬性自動創建管道中的新映像。排程包含開始條件和 cron 運算式。如需詳細資訊，請參閱《AWS CloudFormation 使用者指南》中的 AWS::ImageBuilder::ImagePipeline。

下列範AWS::ImageBuilder::ImagePipeline例讓管道在每天上午 10:00 協調世界時 (UTC) 執行組建。設定下列Schedule值：

- 將 PipelineExecutionStartCondition 設定為 EXPRESSION_MATCH_AND_DEPENDENCY_UPDATES_AVAILABLE。只有在相關資源（如父映像或組件）在其語義版本中使用通配符 'x' 時，構建才會啟動。這樣可以確保構建包含這些資源的最新更新。
- 設定 ScheduleExpression 為 cron 運算式(0 10 * * ? *)。

```
ECSPipeline:
    Type: AWS::ImageBuilder::ImagePipeline
    Properties:
      Name: my-pipeline
      ImageRecipeArn: <arn of the recipe you created in previous step>
      InfrastructureConfigurationArn: <ARN of the infrastructure configuration
 associated with this image pipeline>
      Schedule:
        PipelineExecutionStartCondition:
 EXPRESSION_MATCH_AND_DEPENDENCY_UPDATES_AVAILABLE
        ScheduleExpression: 'cron(0 10 * * ? *)'
```

將圖像 ARN 與地形一起使用

在 Terraform 中指定管線的父影像和排程的方法會與中的圖像對齊。 AWS CloudFormation您使用的aws_imagebuilder_image_recipe資源。將parent_image值設置為圖像 ARN。使用所需映像的無版本 ARN，以確保管線始終使用最新版本的映像。如需詳細資訊，請參閱aws_imagebuilder_image_recipe地形文件中的。

在的排程組態區塊中aws_imagebuilder_image_pipeline resource，將schedule_expression引數值設定為您選擇的 cron 運算式，以指定管線執行的頻率，並將設定pipeline_execution_start_condition為EXPRESSION_MATCH_AND_DEPENDENCY_UPDATES_AVA
如需詳細資訊，請參閱aws_imagebuilder_image_pipeline地形文件中的。

在 Amazon ECS 上使用 AWS Deep Learning Containers

AWS Deep Learning Containers 提供一組 Docker 映像，用於訓練和服務模型，以 TensorFlow及 Amazon ECS 上的 Apache MXNet (孵化)。Deep Learning Containers 可以通過 TensorFlow NVIDIA CUDA（適用於 GPU 實例）和英特爾 MKL（用於 CPU 實例）庫實現優化環境。Amazon ECR 中提供適用於 Deep Learning Containers 的容器映像，以便在 Amazon ECS 任務定義中進行參考。您可以使用 Deep Learning Containers 和 Amazon Elastic Inference 來降低推論成本。

若要在 Amazon ECS 上開始使用 Deep Learning Containers 而不使用 Elastic Inference，請參閱 AWS Deep Learning AMI 開發人員指南中的 Amazon ECS 上的 Deep Learning Containers。

Deep Learning Containers 與 Amazon ECS 上的 Elastic Inference

> **Note**
>
> 自 2023 年 4 月 15 日起，不 AWS 會將新客戶加入 Amazon Elastic Inference (EI)，並協助目前客戶將工作負載遷移到提供更優惠價格和效能的選項。2023 年 4 月 15 日之後，新客戶將無法在 Amazon、Amazon ECS 或 Amazon Amazon EC2 使用亞馬遜 SageMaker EI 加速器啟動執行個體。但是，在過去 30 天內至少使用過一次 Amazon EI 的客戶將被視為目前客戶，並且可以繼續使用該服務。

AWS Deep Learning Containers 提供一組 Docker 映像，以便在中服務模型，以 TensorFlow 及利用 Amazon Elastic Inference 加速器的 Apache MXNet (孵化) 功能。Amazon ECS 提供任務定義參數，以便將 Elastic Inference 加速器連接到您的容器。當您在任務定義中指定 Elastic Inference 加速器類型時，Amazon ECS 會管理加速器的生命週期和組態。使用此功能時，需要 Amazon ECS 服務連結角色。如需 Elastic Inference 加速器的詳細資訊，請參閱 Amazon Elastic Inference 基本概念。

> **Important**
>
> 您的 Amazon ECS 容器執行個體至少需要 `1.30.0` 版的容器代理程式。如需檢查代理程式版本及更新至最新版本的資訊，請參閱「更新 Amazon ECS 容器代理程式」。

若要在 Amazon ECS 上開始使用 Deep Learning Containers 和 Elastic Inference，請參閱 Amazon Elastic Inference 開發人員指南中的 Amazon ECS 上的 Deep Learning Containers 和 Elastic Inference。

Amazon ECS 服務配額

下表提供 AWS 帳戶的 Amazon ECS 預設服務配額，也稱為限制。如需有關您可以與 Amazon ECS 搭配使用的其他 AWS 服務 (如 Elastic Load Balancing 和 Auto Scaling) 之服務配額的詳細資訊，請參閱 Amazon Web Services 一般參考 中的 AWS 服務配額。如需有關 Amazon ECS API 中 API 限流的資訊，請參閱請求 Amazon ECS API 限流。

Amazon ECS 服務配額

以下是 Amazon ECS 服務配額。

新 AWS 帳戶的初始配額可能會隨著時間而增加。Amazon ECS 會持續監控每個區域內的帳戶使用情況，再根據您的用量自動增加配額。針對顯示為可調整的值，您可以申請提高配額，請參閱《Service Quotas 使用指南》中的請求提高配額。

名稱	預設	可調整	描述
每個叢集的容量提供者	每個受支援的區域：20	否	能與叢集相關聯的容量提供者數量上限。
每個服務的 Classic Load Balancer	每個受支援的區域：1	否	每個服務的 Classic Load Balancer 數量上限
每個帳戶的叢集數量	每個受支援的區域：10,000	是	每個帳戶的叢集數目
每個叢集的容器執行個體數量	每個受支援的區域：5,000	否	每個叢集的容器執行個體數量
每個 start-task 的容器執行個體	每個受支援的區域：10	否	StartTask API 動作中指定的容器執行個體數目上限。
每個任務定義的容器數量	每個受支援的區域：10	否	任務定義內的容器定義數量上限。

名稱	預設	可調整	描述
ECS Exec 工作階段	每個支援的區域：1,000	否	每個容器的 ECS Exec 工作階段數量上限。
AWS Fargate 上服務啟動的任務速率	每個受支援的區域：500	否	Amazon ECS 服務排程器在 Fargate 上每分鐘可為每項服務佈建的任務數量上限。
由 Amazon EC2 或外部執行個體上的服務所啟動之任務比率	每個受支援的區域：500	否	Amazon ECS 服務排程器在 Amazon EC2 或外部執行個體上每分鐘可為每項服務佈建的最大任務數量。
每個任務定義系列的修訂數目	每個受支援的區域：1,000,000	否	每個任務定義系列的修訂數目上限。取消註冊或刪除任務定義修訂版本並不會將其排除在此限制中。
每個 awsvpcConfiguration 的安全群組數量	每個受支援的區域：5	否	awsvpcConfiguration 內指定的安全群組數量上限。
每個叢集的服務數量	每個受支援的區域：5,000	否	每個叢集的服務數量上限
每個命名空間的服務	每個受支援的區域：100	是	可在命名空間內執行的服務數目上限。
每個 awsvpcConfiguration 的子網路數量	每個受支援的區域：16	否	awsvpcConfiguration 內指定的子網路數量上限。
每個資源的標籤	每個受支援的區域：50	否	每個資源的標籤數量上限。這適用於任務定義、叢集、任務和服務。

名稱	預設	可調整	描述
每個服務的目標群組	每個受支援的區域：5	否	每個服務的目標群組數量上限 (如果使用 Application Load Balancer 或 Network Load Balancer)。
任務定義大小	每個受支援的區域：64 KB	否	任務定義的大小上限 (KiB)。
每個叢集於「佈建」狀態下的任務數量	每個受支援的區域：500	否	每個叢集於「佈建」狀態下等候的任務的上限數量。此配額僅適用於使用 EC2 Auto Scaling 群組容量提供者啟動的任務。
每個 run-task 啟動的任務數量	每個受支援的區域：10	否	每個 RunTask API 動作可啟動的工作數目上限。
每個服務的任務	每個受支援的區域：5,000	否	每個服務的任務數量上限 (所需的計數)。

> **Note**
>
> 預設值是由設定的初始配額 AWS，與實際套用的配額值和可能的服務配額上限分開。如需詳細資訊，請參閱《Service Quotas 使用者指南》中的 Service Quotas 術語。

> **Note**
>
> 被設定使用 Amazon ECS 服務探索的服務有每個服務 1,000 項任務的限制。這是因為每個服務有執行個體數目的 AWS Cloud Map 服務配額。如需詳細資訊，請參閱《AWS Cloud Map》中的 Amazon Web Services 一般參考服務配額。

> **Note**
>
> 實際上，任務啟動速率還取決於其他注意事項，例如要下載和解壓縮的容器映像、運作狀態檢查以及啟用的其他整合 (例如向負載平衡器註冊任務)。與此處呈現的配額相比，您可能會看到任務啟動率的變化。這些變化是由您用於服務的功能造成的。如需詳細資訊，請參閱 Amazon ECS 服務參數的最佳實務 。

> **Note**
>
> 設定使用 Amazon ECS Service Connect 的服務有每個服務 1,000 項任務的限制。這是因為每個 AWS Cloud Map 服務的執行個體數目有多少服務的服務配額。如需詳細資訊，請參閱《AWS Cloud Map 》中的 Amazon Web Services 一般參考服務配額。

AWS Fargate 服務配額

以下是 Ser AWS Fargate vice Quotas 上的 Amazon ECS，列於服AWS Fargate務配額主控台中的服務下方。

新 AWS 帳戶的初始配額可能會隨著時間而增加。Fargate 會持續監控每個區域內的帳戶使用情況，再根據您的用量自動增加配額。您也可以申請增加配額，以獲取顯示為可調整的值，請參閱《Service Quotas 使用指南》中的請求提高配額。

名稱	預設	可調整	描述
Fargate 隨需 vCPU 資源計數	每個受支援的區域：6	是	在當前區域中，此帳戶中以 Fargate 隨需同時執行的 Fargate vCPU 數量。
Fargate Spot vCPU 資源計數	每個受支援的區域：6	是	在當前區域中，此帳戶中以 Fargate Spot 同時執行的 Fargate vCPUs 數量。

> **Note**
>
> 預設值是由設定的初始配額 AWS，與實際套用的配額值和可能的服務配額上限分開。如需詳細資訊，請參閱《Service Quotas 使用者指南》中的 Service Quotas 術語。

> **Note**
>
> Fargate 還強制執行 Amazon ECS 任務和 Amazon EKS Pod 啟動速率限制。如需詳細資訊，請參閱 Fargate 調節限制。

管理您的 Amazon ECS 和 AWS Fargate 服務配額 AWS Management Console

Amazon ECS 已與 Service Quotas 整合，這項 AWS 服務可讓您從中央位置檢視和管理配額。如需詳細資訊，請參閱Service Quotas 使用者指南中的什麼是 Service Quotas？。

Service Quotas 可讓您輕鬆查詢 Amazon ECS 服務配額的值。

AWS Management Console

使用 AWS Management Console檢視 Amazon ECS 和 Fargate 服務配額

1. 開啟 Service Quotas 主控台，網址為 https://console.aws.amazon.com/servicequotas/。
2. 在導覽窗格中，選擇 AWS services (AWS 服務)。
3. 從 AWS services (服務) 清單中，搜尋並選取 Amazon Elastic Container Service (Amazon ECS) 或 AWS Fargate。

   在 Service quotas (服務配額) 清單中，您可以看到服務配額名稱、套用的值 (如果有的話)、AWS 預設配額，以及配額值是否可調整。
4. 若要檢視服務配額的其他資訊 (例如說明)，請選擇配額名稱。
5. (選用) 若要請求增加配額，請選取您要增加的配額、選取 Request quota increase (請求增加配額)、輸入或選取必要資訊，然後選取 Request (請求)。

若要進一步處理 Service Quotas， AWS Management Console 請參閱《服務配額使用指南》。若要請求提高配額，請參閱《Service Quotas 使用者指南》https://docs.aws.amazon.com/servicequotas/latest/userguide/request-quota-increase.html中的請求提高配額。

AWS CLI

使用 AWS CLI檢視 Amazon ECS 和 Fargate 服務配額

執行下列命令，以檢視預設 Amazon ECS 配額。

```
aws service-quotas list-aws-default-service-quotas \
    --query 'Quotas[*].
{Adjustable:Adjustable,Name:QuotaName,Value:Value,Code:QuotaCode}' \
    --service-code ecs \
    --output table
```

執行下列命令，以檢視預設 Fargate 配額。

```
aws service-quotas list-aws-default-service-quotas \
    --query 'Quotas[*].
{Adjustable:Adjustable,Name:QuotaName,Value:Value,Code:QuotaCode}' \
    --service-code fargate \
    --output table
```

執行下列命令，以檢視您的套用 Fargate 配額。

```
aws service-quotas list-service-quotas \
    --service-code fargate
```

> **Note**
>
> Amazon ECS 不支援套用的配額。

如需使用使用 Service Quotas 的詳細資訊 AWS CLI，請參閱服務配額 AWS CLI 命令參考。若要請求提升配額，請參閱 AWS CLI 命令參考中的 `request-service-quota-increase` 命令。

處理 Amazon ECS 服務配額和 API 節流限制

Amazon ECS 與包括 Elastic Load Balancing 和 Amazon EC2 在內的幾個 AWS 服務集成。 AWS Cloud Map透過這種緊密整合，Amazon ECS 包含多項功能，例如服務負載平衡、服務 Connect、任務聯網和叢集 auto 擴展。Amazon ECS 和其他 AWS 服務 與所有維護服務配額和 API 速率限制整合的其他項目，以確保效能和使用率一致。這些服務配額還可以防止意外佈建超出需要的資源，並防止可能增加帳單的惡意動作。

熟悉您的服務配額和 AWS API 速率限制，您可以規劃擴展工作負載，而不必擔心意外的效能降低。如需詳細資訊，請參閱 Amazon ECS API 的請求節流。

在 Amazon ECS 上擴展工作負載時，建議您考慮下列服務配額。

- AWS Fargate 具有限制每個工作中同時執行的工作數量的配額 AWS 區域。Amazon ECS 上的隨需任務和 Fargate 競價型任務都有配額。每個服務配額還包括您在 Fargate 上執行的任何 Amazon EKS 網繭。
- 對於在 Amazon EC2 執行個體上執行的任務，您可以為每個叢集註冊的 Amazon EC2 執行個體數目上限為 5,000 個。如果您將 Amazon ECS 叢集自動擴展與 Auto Scaling 群組容量提供者搭配使用，或者如果您自行管理叢集的 Amazon EC2 執行個體，則此配額可能會成為部署瓶頸。如果您需要更多容量，可以建立更多叢集或要求增加服務配額。
- 如果您搭配 auto Scaling 群組容量提供者使用 Amazon ECS 叢集自動擴展，則在擴展服務時，請考慮`Tasks in the PROVISIONING state per cluster`配額。此配額是容量提供者可以增加容量之每個叢集的`PROVISIONING`狀態工作數目上限。當您同時啟動大量任務時，可以輕鬆滿足此配額。一個例子是，如果您同時部署數十個服務，每個服務都有數百個任務。發生這種情況時，容量提供者需要啟動新的容器執行個體，以便在叢集容量不足時放置工作。當容量供應商啟動額外的 Amazon EC2 執行個體時，Amazon ECS 服務排程器可能會繼續 parallel 啟動任務。但是，由於叢集容量不足，此活動可能會受到限制。Amazon ECS 服務排程器會在啟動新的容器執行個體時，實作退出和指數限制策略，以重試任務放置。因此，您可能會遇到較慢的部署或向外延展的時間。若要避免這種情況，您可以使用下列其中一種方式來規劃服務部署。部署大量工作不需要增加叢集容量，或是保留備用叢集容量來啟動新工作。

除了在擴展工作負載時考慮 Amazon ECS 服務配額之外，還要考慮與 Amazon ECS 整合 AWS 服務的其他服務配額。

Elastic Load Balancing

您可以將 Amazon ECS 服務設定為使用 Elastic Load Balancing 將流量平均分配到任務中。如需如何選擇負載平衡器的詳細資訊和建議的最佳作法，請參閱使用負載平衡來分配 Amazon ECS 服務流量。

Elastic Load Balancing 服務配額

擴展工作負載時，請考慮下列 Elastic Load Balancing 服務配額。大部分的 Elastic Load Balancing Service Quotas 都是可調整的，而且您可以要求增加服務配額主控台。

Application Load Balancer

使用應用程式負載平衡器時，視您的使用案例而定，您可能需要針對下列項目要求增加配額：

- `Targets per Application Load Balancer`配額，即「Application Load Balancer」背後的目標數目。
- `Targets per Target Group per Region`配額，即「目標群組」背後的目標數目。

如需詳細資訊，請參閱應用程式負載平衡器使用者指南中的應用程式負載平衡器配額。

Network Load Balancer

您可以向 Network Load Balancer 註冊的目標數目有更嚴格的限制。使用 Network Load Balancer 時，您通常會想要啟用跨區域支援，這會對每個 Network Load Balancer 每個可用區域的目標數目上`Targets per Availability Zone Per Network Load Balancer`限提供額外的擴展限制。如需詳細資訊，請參閱網路負載平衡器的配額中的網路負載平衡器使用者指南。

Elastic Load Balancing API 節流

當您將 Amazon ECS 服務設定為使用負載平衡器時，必須通過目標群組運作狀態檢查，才能將服務視為正常狀態。為了執行這些運作狀態檢查，Amazon ECS 會代表您叫用 Elastic Load Balancing API 操作。如果您的帳戶中使用負載平衡器設定了大量服務，則可能會降低服務部署速度，因為、和 E `DescribeTargetHealth` lastic Load Balancing API 作業專用的潛在節流。`RegisterTarget` `DeregisterTarget`發生節流時，Amazon ECS 服務事件訊息中會發生節流錯誤。

如果您遇到 AWS Cloud Map API 節流，可以聯絡以 AWS Support 取得如何增加 AWS Cloud Map API 節流限制的指引。如需監視和疑難排解此類節流錯誤的詳細資訊，請參閱。處理 Amazon ECS 節流問題

彈性網絡介面

當您的任務使用`awsvpc`網路模式時，Amazon ECS 會為每個任務佈建唯一的 elastic network interface (ENI)。當 Amazon ECS 服務使用 Elastic Load Balancing 負載平衡器時，這些網路界面也會註冊為服務中定義之適當目標群組的目標。

彈性網路介面服務配額

當您執行使用awsvpc網路模式的工作時，每項工作都會附加唯一的 elastic network interface。如果必須透過網際網路達成這些工作，請為這些工作的 elastic network interface 指派公用 IP 位址。擴展 Amazon ECS 工作負載時，請考慮以下兩個重要配額：

- `Network interfaces per Region`配額，即您帳戶中網路介面 AWS 區域 的最大數量。
- `Elastic IP addresses per Region`配額，即彈性 IP 位址的數目上限 AWS 區域。

這兩個 Service Quotas 都是可調整的，您可以從服務配額控制台要求增加這些配額。如需詳細資訊，請參閱 Amazon 虛擬私有雲端使用者指南中的 Amazon VPC 服務配額。

對於 Amazon EC2 執行個體上託管的 Amazon ECS 工作負載，執行使用awsvpc網路模式的任務時，請考慮`Maximum network interfaces`服務配額，即每個 Amazon EC2 執行個體的網路執行個體數目上限。此配額限制了您可以在執行個體上放置的工作數量。您無法調整配額，也無法在「Service Quotas」主控台中使用配額。如需詳細資訊，請參閱 Amazon EC2 使用者指南中每個執行個體類型的每個網路界面的 IP 地址。

雖然您無法變更可連接到 Amazon EC2 執行個體的網路界面數量，但您可以使用 elastic network interface 中繼功能來增加可用網路界面的數量。例如，`c5.large`執行個體預設最多可以有三個網路介面。執行個體的主要網路介面視為一個配額。因此，您可以將另外兩個網路介面附加至執行個體。由於使用awsvpc網路模式的每項工作都需要網路介面，因此您通常只能在此執行個體類型上執行兩項此類工作。這可能會導致叢集容量的使用率不足。如果您啟用 elastic network interface 主幹連線，您可以增加網路介面密度，以便在每個執行個體上放置更多的工作。啟用中繼功能後，`c5.large`執行個體最多可以有 12 個網路介面。執行個體具有主要網路界面，Amazon ECS 會建立「中繼」網路界面並將其附加到執行個體。因此，透過此設定，您可以在執行個體上執行 10 項工作，而不是預設的兩項工作。如需詳細資訊，請參閱 增加 Amazon ECS Linux 容器實例網絡界面。

彈性網路介面 API 節流

當您執行使用awsvpc網路模式的任務時，Amazon ECS 會仰賴下列 Amazon EC2 API。這些 API 中的每一個都有不同的 API 節流。如需詳細資訊，請參閱 Amazon EC2 API 參考中針對 Amazon EC2 API 的請求節流。

- CreateNetwork介面
- AttachNetwork介面
- DetachNetwork介面
- DeleteNetwork介面

- DescribeNetwork介面
- DescribeVpcs
- DescribeSubnets
- DescribeSecurity群組
- DescribeInstances

如果 Amazon EC2 API 呼叫在 elastic network interface 佈建工作流程期間受到限制，Amazon ECS 服務排程器會以指數式回溯自動重試。這些自動淘汰有時可能會導致啟動工作延遲，進而導致部署速度較慢。發生 API 節流時，您會在服務事件訊息`Operations are being throttled. Will try again later.`上看到訊息。如果您始終符合 Amazon EC2 API 節流的要求，可以聯絡以 AWS Support 取得如何增加 API 節流限制的指引。如需監視及疑難排解節流錯誤的詳細資訊，請參閱處理節流問題。

AWS Cloud Map

Amazon ECS 服務探索和服務 Connect 使用 AWS Cloud Map API 來管理 Amazon ECS 服務的命名空間。如果您的服務有大量工作，請考慮下列建議。

AWS Cloud Map 服務配額

當 Amazon ECS 服務設定為使用服務探索或服務 Connect 時，服務的任務數目上限的`Tasks per service`配額會受到服務配額 (該 AWS Cloud Map `Instances per service`服務的執行個體數目上限) 的影響。特別是， AWS Cloud Map 服務配額會減少您可以執行的工作量，達到最多 1,0000 項服務的工作。您無法變更 AWS Cloud Map 配額。如需詳細資訊，請參閱 AWS Cloud Map 服務配額。

AWS Cloud Map API 節流設定

Amazon ECS 會`ListInstances`代表您呼叫`GetInstancesHealthStatus`、`RegisterInstance`、和 `DeregisterInstance` AWS Cloud Map API。它們可協助您探索服務，並在您啟動工作時執行健康狀態檢查。當同時部署使用大量工作之服務探索的多個服務時，這可能會導致超出 AWS Cloud Map API 節流限制。發生這種情況時，您可能會看到以下訊息：`Operations are being throttled. Will try again later`在 Amazon ECS 服務事件訊息中，部署和任務啟動速度較慢。 AWS Cloud Map 不記錄這些 API 的節流限制。如果您遇到這些限制的情況，可以聯絡以 AWS Support 取得增加 API 節流限制的指引。如需監視和疑難排解此類節流錯誤的詳細建議，請參閱。處理 Amazon ECS 節流問題

Amazon ECS API 參考

除了 AWS Management Console 和 AWS Command Line Interface (AWS CLI)，Amazon ECS 還提供了一個 API。您可以使用 API，將管理 Amazon ECS 資源的任務自動化。

- 如需 Amazon ECS 資源的 API 操作清單，請參閱 Amazon ECS 資源執行的動作。
- 如需依字母排序的 API 操作清單，請參閱動作。
- 如需依字母排序的資料類型清單，請參閱資料類型。
- 如需常用查詢參數的清單，請參閱常用參數。
- 如需錯誤碼的說明，請參閱常見錯誤。

如需有關的詳細資訊 AWS CLI，請AWS Command Line Interface 參閱 Amazon ECS 的參考資料。

文件歷史紀錄

下表說明 Amazon Elastic Container Service 開發人員指南的主要更新及和新功能。我們也會經常更新文件，以處理您傳送給我們的意見回饋。

變更	描述	日期
適用於 Fargate 支援 Linux 容器的 GMSA	Amazon ECS 透過一種稱為群組受管服務帳戶 (gMSA) 的特殊類型服務帳戶，支援 Fargate 上 Linux 容器的作用中目錄身份驗證。如需詳細資訊，請參閱gMSA針對 Fargate 上的Linux容器使用。	2024年3月5日
CloudWatch 為附加至任務的 Amazon EBS 磁碟區新增指標	Amazon ECS 現在會針對已連接 Amazon EBS 磁碟區的任務發佈 Amazon EBS 儲存使用率的 CloudWatch 指標。如需詳細資訊，請參閱 Amazon ECS CloudWatch 指標。	2024年2月8日
服務 Connect TLS	您現在可以搭配服務 Connect 使用 TLS。	2024年1月22日
服務 Connect TLS 受管理的原則	增加了新的 AmazonECS 政策InfrastructureRolePolicyForServiceConnectTransportLayerSecurity。	2024年1月22日
服務 Connect 逾時組態更新	服務 Connect 逾時設定現在可以更新，並包含兩個選用參數`-idleTimeout` 和`perRequestTimeout` 。	2024年1月22日
Amazon ECS 受管實例排水	您可以使用 Amazon ECS 受管執行個體排空功能，以促進 Amazon ECS 執行個體的正常終止。	2024 年 1 月 19 日
為 ECS 任何地方添加了 Ubuntu 22 支持	對 Ubuntu 22 操作系統的 Support 已添加到 ECS 任何地方。如需詳細資訊，請參閱 支援的作業系統和系統架構。	2024年1月16日
新增 `AmazonECSInfrastructureRolePolicyForVolumes` IAM 政策	已新增 `AmazonECSInfrastructureolePolicyForVolumes` 。該政策授予 Amazon ECS 進行 AWS API 呼叫以管理與 Amazon ECS 工作負載相關聯的 Amazon EBS 磁碟區所需的許可。	2024年1月11日

變更	描述	日期
Amazon EBS 資料磁碟區適用於 Amazon ECS 任務	您可以在部署期間為每個任務設定 1 個 Amazon EBS 資料磁碟區，以便將附件連接到獨立的 Amazon ECS 任務或由 ECS 服務管理的任務。在部署時設定磁碟區可讓您建立不受特定磁碟區類型或設定限制的可繼續工作定義。Amazon EBS 磁碟區為資料密集型容器化工作負載提供高可用性、經濟實惠、耐用的高效能區塊儲存。	2024年1月11日
Amazon ECS 經典控制台達到生命週期結束	Amazon ECS 控制台已經到了生命週期結束。	2023 年 12 月 4 日
更新的政策	AmazonECS ServiceRolePolicy 受管身分與存取權管理政策已更新為新`events`許可、其他`autoscaling` 許可和許可。`autoscaling-plans`	2023 年 12 月 4 日
運行時監視支持	您可以使用執行階段監控來監控 Amazon ECS 工作負載，以識別惡意或未經授權的行為。如需詳細資訊，請參閱執行階段監視。	2023 年 11 月 26 日
更新的政策	受`AmazonECSServiceRolePolicy` 管 IAM 政策已更新，以允許存取 AWS Cloud Map `DiscoverInstancesRevision` API。	2023 年 10 月 4 日
AWS Fargate 工作淘汰組態	您可以設定淘汰 Fargate 任務之前的等待期。如需詳細資訊，請參閱AWS Fargate 任務維護。	2023 年 9 月 5 日
中的其他任務定義參數 AWS Fargate	AWS Fargate `systemControls` 在 Linux 平台版本中添加了對`pidMode`和的支持`1.4.0`。如需詳細資訊，請參閱任務定義。	2023 年 8 月 9 日
Amazon ECS 主控台任務定義頁面重新設計	Amazon ECS 主控台中的任務定義頁面經過重新設計，並包含其他選項。如需詳細資訊，請參閱使用主控台建立任務定義。	2023 年 7 月 26 日

變更	描述	日期
Fargate 支援使用 Seekable OCI 索引的延遲載入	AWS Fargate 正在引入可搜索的 OCI（SOCI）索引。使用 SOCI 時，容器只需耗費幾秒鐘就可以開始提取映像，因此在背景下載影像時，可為環境設定和應用程式執行個體化提供時間。如需詳細資訊，請參閱 Amazon ECS 法爾蓋特使用者指南中的使用可搜尋 OCI (SOCI) 延遲載入容器映像。 AWS	2023 年 7 月 17 日
已改善對 Linux 和 Windows 上的 gMSA 支援	任務定義具有適用於 Linux 和 Windows 的 gMSA 新 `credentialSpecs` 欄位。已新增關於 Windows 上無網域 gMSA 的完整教學課程，請參閱 Tutorial: Using Windows Containers with Domainless gMSA using the AWS CLI。如需詳細資訊，請參閱針對 Linux 容器使用 gMSA 和針對 Windows 容器使用 gMSA。	2023 年 7 月 14 日
已改善 ECS 代理程式版本文件	Amazon ECS 代理程式版本的文件已更新。我們建議您使用 `v20.10.13` 版本以上的 Docker 和最新版本的 Amazon ECS 容器代理程式。已發行版本和代理程式的變更可在中取得 GitHub。如需詳細資訊，請參閱 Amazon ECS Linux 容器代理程式版本。	2023 年 6 月 20 日
更新了 Fargate ARM64 支援的區域可用性	Fargate ARM64 支援的區域可用性已更新。如需詳細資訊，請參閱 考量事項。	2023 年 6 月 19 日
改善叢集自動擴展文件	Amazon EC2 Auto Scaling 的 Amazon ECS 擴展文件在準確性和可讀性方面有重大改善。如需詳細資訊，請參閱 Amazon ECS 叢集自動擴展。	2023 年 5 月 4 日
資源建立的標記授權。	使用者必須擁有建立資源的動作許可，例如 `ecsCreateCluster` 。當您建立資源並指定該資源的標籤時，請 AWS 執行其他授權以確認是否具有建立標籤的權限。如需詳細資訊，請參閱標記授權和建立時授予標記資源的許可。	2023 年 4 月 18 日
支援 EC2 上 Linux 容器的 gMSA	您可以使用 gMSA 向 EC2 上 Linux 容器的 Active Directory 進行驗證。如需詳細資訊，請參閱針對 Linux 容器使用 gMSA。	2023 年 4 月 14 日

變更	描述	日期
支援 AWS Fargate上 Windows 容器的暫時性儲存。	您可以使用 AWS Fargate上 Windows 容器的暫時性儲存。如需詳細資訊，請參閱 Fargate 任務儲存。	2023 年 4 月 14 日
AWS Cost Management 支援工作層級的 CUR 資料	您可以在「成本與用量報告」中開啟任務層級的成本和資源用量。這會為在 AWS Fargate 和 EC2 上執行的任務新增分割成本分配資料。如需詳細資訊，請參閱 任務層級成本和用量報告。	2023 年 4 月 12 日
Amazon Linux 2023 Amazon ECS 最佳化 AMI	您可在 Amazon Linux 2023 Amazon ECS 最佳化 AMI 上部署工作負載。如需詳細資訊，請參閱 Amazon ECS 優化的 Linux AMI。	2023 年 4 月 10 日
AWS Fargate 聯邦資訊處理標準 (FIPS) 140	您可以使用符合聯邦資訊處理標準 (FIPS) 140 的方式 AWS Fargate 在 Amazon ECS 上部署工作負載。如需詳細資訊，請參閱 AWS Fargate 美國聯邦資訊處理標準 (FIPS-140)。	2023 年 4 月 10 日
任務定義刪除	您可以使用 Amazon ECS 主控台、SDK 和 AWS CLI刪除任務定義。如需詳細資訊，請參閱使用主控台刪除任務定義修訂版和任務定義。	2023 年 2 月 24 日
AWS Fargate Compute Optimizer 中的服務	AWS 運 Compute Optimizer 器會根據在 AWS Fargate 上 Amazon ECS 服務中執行任務的使用率，產生任務和容器大小建議。如需詳細資訊，請參閱檢視針對 Fargate 上 Amazon ECS 服務的建議。	2023 年 1 月 27 日
Amazon ECS 主控台	新版 Amazon ECS 主控台為預設主控台。如需詳細資訊，請參閱新版 Amazon ECS 主控台。	2023 年 1 月 19 日
更新的 `AmazonECS_FullAccess` IAM 政策	`AmazonECS_FullAccess` IAM 政策已更新，以包含在建立期間向負載平衡器新增標籤的許可。如需詳細資訊，請參閱 亞馬遜 FullAccess。	2023 年 1 月 4 日
使用 CloudWatch 警示偵測 Amazon ECS 服務部署失敗	您可以設定 Amazon ECS，在偵測到指定的警示進入 CloudWatch 警示狀態時，將部署設定為失敗。如需詳細資訊，請參閱 the section called “故障檢測 ”。	2022 年 12 月 19 日

變更	描述	日期
支援容器連接埠映射	您可以在綁定到動態映射主機連接埠範圍的容器上設定連接埠號碼範圍。如需詳細資訊，請參閱 the section called “連接埠映射”。	2022 年 12 月 15 日
Amazon ECS Service Connect 全面上市	此功能新增由 Amazon ECS 服務部署控制的服務探索和服務網格。如需詳細資訊，請參閱 the section called “ Service Connect”。	2022 年 11 月 27 日
新 Amazon ECS 主控台版本更新了任務定義	新 Amazon ECS 主控台版本現提供用於任務定義的 JSON 編輯器。如需詳細資訊，請參閱 the section called “使用主控台建立任務定義”。	2022 年 10 月 27 日
新 Amazon ECS 主控台版本更新了任務定義	新 Amazon ECS 主控台版本現提供用於任務定義的 JSON 編輯器。如需詳細資訊，請參閱 the section called “使用主控台建立任務定義”。	2022 年 10 月 27 日
更新了全新的 Amazon ECS 主控台體驗	新 Amazon ECS 主控台版本新增其他服務和任務參數。如需詳細資訊，請參閱 the section called “建立服務” 及 the section called “將應用程式作為工作執行”。	2022 年 10 月 7 日
任務中繼資料端點版本 4 中的新資訊	任務中繼資料端點版本 4 現在會納入 VPC ID 和服務名稱。如需詳細資訊，請參閱 the section called “任務中繼資料端點第 4 版”。	2022 年 10 月 7 日
新的任務定義大小	Amazon ECS on Fargate 現在支援 8 個 vCPU 和 16 個 vCPU 任務大小。如需詳細資訊，請參閱 the section called “任務大小”。	2022 年 9 月 16 日
已封存 ECS CLI 頁面	已封存 ECS CLI 文件。我們建議您使用 AWS Copilot 來滿足您的命令行工具需求。如需詳細資訊，請參閱 使用 AWS 副駕駛命令列界面建立 Amazon ECS 資源。	2022 年 9 月 15 日
新的 Fargate 配額	Fargate 正在從任務計數型配額轉換為 vCPU 型配額。如需詳細資訊，請參閱 the section called “AWS Fargate 服務配額”。	2022 年 9 月 8 日

變更	描述	日期
對 Amazon EC2 Auto Scaling 暖集區的支援。	您現在可以利用 Amazon EC2 Auto Scaling 暖集區來更快地水平擴展應用程式並節省成本。如需詳細資訊，請參閱 為您的 Amazon ECS Auto Scaling 群組設定預先初始化的執行個體。	2022 年 3 月 23 日
對 ECS Anywhere 中 Windows 執行個體的支援。	ECS Anywhere 現在支援 Windows 執行個體。如需詳細資訊，請參閱 適用於外部啟動類型的 Amazon ECS 叢集。	2022 年 3 月 3 日
新增了對外部執行個體的 ECS Exec 支援	ECS Exec 現支援外部執行個體。如需詳細資訊，請參閱 使用 ECS 執行監控 Amazon ECS 容器。	2022 年 1 月 24 日
更新了全新的 Amazon ECS 主控台體驗	全新 Amazon ECS 主控台體驗支援建立和刪除叢集、更新任務定義以及取消註冊任務定義。如需詳細資訊，請參閱 為 Fargate 啟動類型建立 Amazon ECS 叢集、刪除 Amazon ECS 叢集、使用主控台更新 Amazon ECS 任務定義 和 使用主控台取消註冊 Amazon ECS 任務定義修訂。	2021 年 12 月 8 日
更新了全新的 Amazon ECS 主控台體驗	全新 Amazon ECS 主控台體驗支援建立任務定義。如需詳細資訊，請參閱 使用主控台建立 Amazon ECS 任務定義。	2021 年 11 月 23 日
Amazon ECS 支援適用於 Linux 的 64 位元 ARM 架構。	Amazon ECS 支援 Linux 作業系統的 64 位元 ARM CPU 架構。如需詳細資訊，請參閱 the section called “64 位元 ARM 工作負載的工作定義”。	2021 年 11 月 23 日
Amazon ECS 對流暢 log-driver-buffer-limit 選項的支持	Amazon ECS 支援 fluentd `log-driver-buffer-limit` 選項。如需詳細資訊，請參閱 將 Amazon ECS 日誌傳送到 AWS 服務或 AWS Partner。	2021 年 11 月 22 日
Amazon ECS 最佳化 Linux AMI 建置指令碼	Amazon ECS 已採用開放原始碼的形式提供建置指令碼，可用於建置 Amazon ECS 最佳化 AMI 的 Linux 變體。如需詳細資訊，請參閱 Amazon ECS 最佳化 Linux AMI 建置指令碼。	2021 年 11 月 19 日

變更	描述	日期
容器執行個體運作狀態	Amazon ECS 新增對容器執行個體運作狀態監控的支援。如需詳細資訊，請參閱 監控 Amazon ECS 容器執行個體健康。	2021 年 11 月 10 日
支援 Windows Amazon ECS Exec	Amazon ECS Exec 支援 Windows。如需詳細資訊，請參閱 使用 ECS 執行監控 Amazon ECS 容器。	2021 年 11 月 1 日
支援 Fargate 上的 Windows 容器。	Amazon ECS 支援 Fargate 上的 Windows 容器。如需詳細資訊，請參閱 Amazon ECS 的 Fargate 視窗平台版本。	2021 年 10 月 28 日
針對 Amazon ECS Anywhere 上的外部執行個體提供 GPU 支援	Amazon ECS 支援為外部執行個體上執行之任務在任務定義中指定 GPU 要求。如需詳細資訊，請參閱GPU 工作負載的 Amazon ECS 任務定義及將外部執行個體註冊到 Amazon ECS 叢集。	2021 年 10 月 8 日
支援 Windows 上的 awsvpc 網路模式	Amazon ECS 在 Windows 上支援 awsvpc 網路模式。如需詳細資訊，請參閱 為 Amazon ECS 任務分配網絡界面。	2021 年 7 月 15 日
Bottlerocket 正式發行	Amazon ECS 支援 Bottlerocket 作業系統的 Amazon ECS 最佳化 AMI 變體，作為 AMI 提供。如需詳細資訊，請參閱 Amazon ECS 最佳化 Bottlerocket AMI。	2021 年 6 月 30 日
Amazon ECS 已排程的任務更新	Amazon 在創建觸發 Amazon ECS 計劃任務的規則時 EventBridge 添加了對其他參數的支持。	2021 年 6 月 25 日
AWS Amazon ECS 的受管政策	Amazon ECS 針對服務連結角色新增 AWS 受管政策的文件。如需詳細資訊，請參閱 AWS Amazon 彈性容器服務的受管政策。	2021 年 6 月 8 日
開始使用 AWS CDK	添加了 AWS CDK 與 Amazon ECS 一起使用的入門指南。如需詳細資訊，請參閱 使用建立 Amazon ECS 資源 AWS CDK。	2021 年 5 月 27 日

變更	描述	日期
Amazon ECS Anywhere	Amazon ECS 增加了向叢集註冊內部部署伺服器或虛擬機器 (VM) 的支援。如需詳細資訊，請參閱 適用於外部啟動類型的 Amazon ECS 叢集。	2021 年 5 月 25 日
Amazon ECS 最佳化 Windows Server 20H2 Core AMI	Amazon ECS 新增支援 Windows Server 20H2 Core 的新 Windows Amazon ECS 最佳化 AMI 變體。如需詳細資訊，請參閱 Amazon ECS 優化的 Linux AMI。	2021 年 4 月 19 日
Amazon ECS Exec	Amazon ECS 發佈了一個新的除錯工具，名為 ECS Exec。如需詳細資訊，請參閱 使用 ECS 執行監控 Amazon ECS 容器。	2021 年 3 月 15 日
支援 VPC 端點政策	Amazon ECS 現在支援 Amazon VPC 端點政策。如需詳細資訊，請參閱 為 Amazon ECS 建立 VPC 端點政策。	2021 年 1 月 11 日
全新主控台體驗	Amazon ECS 發行了新的主控台體驗，可支援建立或更新服務或執行獨立任務。如需更多詳細資訊，請參閱 使用主控台建立 Amazon ECS 服務 和 以 Amazon ECS 任務的形式執行應用程式。	2020 年 12 月 28 日
容量提供者更新	Amazon ECS 新增了更新現有 Auto Scaling 群組容量提供者的支援。	2020 年 11 月 23 日
對於 Windows 任務，ECS 現在支援 Amazon FSx for Windows File Server。	Amazon ECS 新增了在 Windows 任務定義中指定 Amazon FSx for Windows File Server 磁碟區的支援。如需詳細資訊，請參閱 將 FSx 用於 FSx for Windows File Server 磁碟區與 Amazon ECS。	2020 年 11 月 11 日
新增 VPC 雙堆疊模式支援	Amazon ECS 新增支援在雙堆疊模式下使用 VPC，以及使用 `awsvpc` 網路模式的任務，可提供對 IPv6 地址的支援。如需詳細資訊，請參閱 在雙堆疊模式下使用 VPC。	2020 年 11 月 5 日
任務中繼資料端點 v4 更新	Amazon ECS 將其他中繼資料新增至任務中繼資料端點 v4 輸出。如需詳細資訊，請參閱 Amazon ECS 任務中繼資料端點版本 4。	2020 年 11 月 5 日

變更	描述	日期
支援 Local Zones 和 Wavelength 區域	Amazon ECS 已新增 Local Zones 和 Wavelength 區域中工作負載的支援。如需詳細資訊，請參閱 共用子網路、Local Zones 和 Wavelength 區域中的 Amazon ECS 應用程式。	2020 年 9 月 4 日
Bottlerocket AMI 的 Amazon ECS 變體	Bottlerocket 是一個以 Linux 為基礎的開放原始碼作業系統，專為執行容器而建置。 AWS Bottlerocket 作業系統的 Amazon ECS 最佳化 AMI 變體作為 AMI 提供，可在啟動 Amazon ECS 容器執行個體時使用它。如需詳細資訊，請參閱 Amazon ECS 最佳化 Bottlerocket AMI。	2020 年 8 月 31 日
任務中繼資料端點第 4 版已更新網路速度統計資訊	任務中繼資料端點第 4 版已更新，為在 Amazon EC2 執行個體 (執行容器代理的至少 `1.43.0` 版本) 上託管的使用 `awsvpc` 或 `bridge` 網路模式的 Amazon ECS 任務提供網路速度統計資訊。如需詳細資訊，請參閱 Amazon ECS 任務中繼資料端點版本 4。	2020 年 8 月 10 日
Fargate 用量指標	AWS Fargate 提供 CloudWatch 使用指標，可讓您了解 Fargate 隨需和 Fargate Spot 資源的帳戶使用情況。如需詳細資訊，請參閱 用量指標。	2020 年 8 月 3 日
AWS 副駕駛版本 0.1.0	新推出的 C AWS opilot CLI 提供高階命令，可簡化從本機開發環境在 Amazon ECS 上建模、建立、發行和管理容器化應用程式的程式。如需詳細資訊，請參閱 使用 AWS 副駕駛命令列界面建立 Amazon ECS 資源。	2020 年 7 月 9 日
AWS Fargate 平台版本棄用時間表	已新增 Fargate 平台版本取代排程。如需詳細資訊，請參閱 AWS Fargate Linux 平台版本棄用。	2020 年 7 月 8 日
AWS Fargate 區域擴張	AWS Fargate 上的 Amazon ECS 已擴展到歐洲（米蘭）區域。	2020 年 6 月 25 日

變更	描述	日期
Amazon ECS 最佳化 Amazon Linux 2 (Neuron) AMI 已發佈	Amazon ECS 針對推理工作負載發佈了 Amazon ECS 最佳化 Amazon Linux 2 (Neuron) AMI。 如需詳細資訊，請參閱 Amazon ECS 優化的 Linux AMI。	2020 年 6 月 24 日
新增對刪除容量提供者的支援	Amazon ECS 已新增刪除 Auto Scaling 群組容量提供者的支援。	2020 年 6 月 11 日
AWS Fargate 平台 1.4.0 更新	從 2020 年 5 月 28 日開始，任何使用平台 1.4.0 版啟動的新 Fargate 任務，都會以使用 AWS Fargate受管加密金鑰的 AES-256 加密演算法加密其 20 GB 的暫時性儲存。如需詳細資訊，請參閱 適用於 Amazon ECS 的 Fargate 任務暫時儲存。	2020 年 5 月 28 日
環境變數檔案支援	新增在工作定義中指定環境變數檔案的支援，可讓您大批新增環境變數至容器。如需詳細資訊，請參閱 將個別環境變數傳遞至 Amazon ECS 容器。	2020 年 5 月 18 日
AWS Fargate 區域擴張	AWS Fargate 與 Amazon ECS 已經擴展到非洲（開普敦）地區。	2020 年 5 月 11 日
服務配額已更新	下列服務配額已更新： • 每個帳戶的叢集已從 2,000 提高到 10,000。 如需詳細資訊，請參閱 Amazon ECS 服務配額。	2020 年 4 月 17 日

變更	描述	日期
AWS Fargate 平台 1.4.0 版	AWS Fargate 平台版本 1.4.0 發布，其中包含以下功能： • 新增了對使用 Amazon EFS 檔案系統磁碟區的支援，以便實現持久性任務儲存。如需詳細資訊，請參閱 搭配 Amazon ECS 使用 Amazon EFS 磁碟區。 • 暫時性任務儲存體已增加至 20 GB。如需詳細資訊，請參閱 適用於 Amazon ECS 的 Fargate 任務暫時儲存。 • 進出任務的網路流量行為已更新。從平台版本 1.4 開始，所有 Fargate 任務會收到單一彈性網路介面 (也稱為任務 ENI)，所有網路流量會流經 VPC 中的 ENI，而且您可以透過 VPC 流程日誌查看。如需詳細資訊，請參閱 AWS Fargate的 Amazon Elastic Container Service 使用者指南中的 Fargate 任務聯網。 • 任務 ENI 新增 Jumbo Frame 的支援。網路界面皆以最大傳輸單位 (MTU) 來設定，這是適合單一框架的最大酬載大小。MTU 越大，單一框架能容納的應用程式酬載越多，可降低每個框架的額外負荷並提高效率。當任務和目的地之間的網路路徑支援 Jumbo Frame (例如，VPC 中剩餘的所有流量) 時，支援 Jumbo Frame 會降低額外負荷。 • CloudWatch 容器洞見將包括 Fargate 工作的網路效能指標。如需詳細資訊，請參閱 使用容器洞察來監控 Amazon ECS 容器。 • 已新增任務中繼資料端點 v4 的支援，該端點為您的 Fargate 任務提供其他資訊，包括任務的網路統計資料，以及執行中任務所在的可用區域。如需詳細資訊，請參閱 Amazon ECS 任務中繼資料端點版本 4。	2020 年 4 月 8 日

變更	描述	日期
	• 在容器定義中，新增對 SYS_PTRACE Linux 參數的支援。如需詳細資訊，請參閱 Linux 參數。 • Fargate 容器代理會取代所有 Fargate 任務的 Amazon ECS 容器代理的使用。這項變更應該不會對您的任務執行方式產生影響。 • 容器執行時間目前使用 Containerd，而非 Docker。這項變更應該不會對您的任務執行方式產生影響。您會注意到，容器執行時間產生的一些錯誤訊息將從提及 Docker 變更為較普通的錯誤。 如需詳細資訊，請參閱 適用於 Amazon ECS 的 Fargate Linux 平台版本。	
Amazon EFS 檔案系統支援任務磁碟區	Amazon EFS 檔案系統可做為 Amazon ECS 和 Fargate 任務的資料磁碟區。如需詳細資訊，請參閱 搭配 Amazon ECS 使用 Amazon EFS 磁碟區。	2020 年 4 月 8 日
Amazon ECS 任務中繼資料端點第 4 版	從 Amazon ECS 容器代理程式版本 1.39.0 和 Fargate 平台版本 1.4.0 開始，名為 ECS_CONTAINER_METADATA_URI_V4 的環境變數會插入任務中的每個容器。當您查詢任務中繼資料第 4 版端點時，有各種任務中繼資料和 Docker 統計資訊可供任務使用。如需詳細資訊，請參閱 Amazon ECS 任務中繼資料端點版本 4。	2020 年 4 月 8 日
支援特定版本的 Secrets Manager 秘密作為環境變數插入。	新增使用特定 Secrets Manager 秘密版本指定敏感資料的支援。如需詳細資訊，請參閱 將敏感資料傳遞至 Amazon ECS 容器。	2020 年 2 月 24 日
新增藍/綠部 CodeDeploy 署的其他部署設定選項	該 CodeDeploy 服務為 Amazon ECS 部署類型添加了新的初期測試和線性部署組態。也可以定義自訂部署組態。如需詳細資訊，請參閱 在部署之前驗證 Amazon ECS 服務的狀態 。	2020 年 2 月 6 日

變更	描述	日期
添加了 efsVolume Configuration 任務定義參數	`efsVolumeConfiguration` 任務定義參數目前開放預覽，可讓您更輕鬆地搭配使用 Amazon EFS 檔案系統與 Amazon ECS 任務。如需詳細資訊，請參閱 搭配 Amazon ECS 使用 Amazon EFS 磁碟區。	2020 年 1 月 17 日
Amazon ECS 容器代理程式記錄行為已更新	Amazon ECS 容器代理程式記錄位置和輪換行為已更新。如需詳細資訊，請參閱 Amazon ECS 容器代理程式日誌組態參數。	2020 年 1 月 13 日
Fargate Spot	Amazon ECS 新增使用 Fargate Spot 執行任務的支援。如需詳細資訊，請參閱 適用於 Fargate 啟動類型的 Amazon ECS 叢集 。	2019 年 12 月 3 日
叢集 Auto Scaling	Amazon ECS 叢集自動擴展可讓您更多地控制叢集內擴展任務的方式。如需詳細資訊，請參閱 透過叢集自動擴展自動管理 Amazon ECS 容量。	2019 年 12 月 3 日
叢集容量提供者	Amazon ECS 叢集容量提供者會決定要用於您任務的基礎設施。如需詳細資訊，請參閱 Amazon ECS 叢集。	2019 年 12 月 3 日
在上建立叢集 AWS Outposts	Amazon ECS 現在支援在 AWS Outposts上建立叢集。如需詳細資訊，請參閱 the section called “Amazon 彈性容器服務 AWS Outposts”。	2019 年 12 月 3 日
服務動作事件	Amazon ECS 現在會在發生特定服務動作 EventBridge 時，將事件傳送至 Amazon。如需詳細資訊，請參閱 Amazon ECS 服務動作事件。	2019 年 11 月 25 日
Amazon ECS GPU 最佳化 AMI 支援 G4 執行個體	Amazon ECS 在使用 Amazon ECS GPU 最佳化 AMI 時新增支援 g4 執行個體類型系列。如需詳細資訊，請參閱 GPU 工作負載的 Amazon ECS 任務定義。	2019 年 10 月 8 日
FireLens 對於 Amazon ECS	FireLens Amazon ECS 正式上市。 FireLens Amazon ECS 可讓您使用任務定義參數將日誌路由到 AWS 服務或合作夥伴目的地，以進行日誌儲存和分析。如需詳細資訊，請參閱 將 Amazon ECS 日誌傳送到 AWS 服務或 AWS Partner。	2019 年 9 月 30 日

變更	描述	日期
AWS Fargate 區域擴張	AWS Fargate 與 Amazon ECS 合作，已擴展至歐洲 (巴黎)、歐洲 (斯德哥爾摩) 和中東 (巴林) 等地區。	2019 年 9 月 30 日
Deep Learning Containers 與 Amazon ECS 上的 Elastic Inference	Amazon ECS 支援將 Amazon Elastic Inference 加速器連接到您的容器，讓執行深度學習推論工作負載更有效率。如需詳細資訊，請參閱 Deep Learning Containers 與 Amazon ECS 上的 Elastic Inference。	2019 年 9 月 3 日
FireLens 對於 Amazon ECS	FireLens Amazon ECS 處於公開預覽狀態。 FireLens Amazon ECS 可讓您使用任務定義參數將日誌路由到 AWS 服務或合作夥伴目的地，以進行日誌儲存和分析。如需詳細資訊，請參閱 將 Amazon ECS 日誌傳送到 AWS 服務或 AWS Partner。	2019 年 8 月 30 日
CloudWatch 容器洞察	CloudWatch 容器深入解析現已正式推出。它可讓您從容器化應用程式和微型服務收集、彙總和總結指標和日誌。如需詳細資訊，請參閱 使用容器洞察來監控 Amazon ECS 容器。	2019 年 8 月 30 日
容器層級交換組態	Amazon ECS 已新增支援，可在容器層級控制 Linux 容器執行個體上的交換記憶體空間用量。使用每個容器交換組態，任務定義內的每個容器都可以啟用或停用交換，並且對於已啟用交換的容器，可以限制所使用的交換空間數量上限。如需詳細資訊，請參閱 管理 Amazon ECS 上的容器交換記憶體空間。	2019 年 8 月 16 日
AWS Fargate 區域擴張	AWS Fargate 與 Amazon ECS 合作已擴展到亞太地區（香港）區域。	2019 年 8 月 6 日
彈性網路介面中繼	針對 ENI 中繼功能新增了額外支援的 Amazon EC2 執行個體類型。如需詳細資訊，請參閱 支援增加 Amazon ECS 容器網路界面的執行個體。	2019 年 8 月 1 日
透過服務註冊多個目標群組	已新增在服務定義中指定多個目標群組的支援。如需詳細資訊，請參閱 使用 Amazon ECS 服務註冊多個目標群組。	2019 年 7 月 30 日

變更	描述	日期
使用 Secrets Manager 秘密指定敏感資料	新增了使用 Secrets Manager 秘密指定敏感資料的教學課程。如需詳細資訊，請參閱 使用 Amazon ECS 中的 Secrets Manager 密碼指定敏感資料。	2019 年 7 月 20 日
CloudWatch 容器洞察	Amazon ECS 已新增對 CloudWatch 容器洞見的支援。如需詳細資訊，請參閱 使用容器洞察來監控 Amazon ECS 容器。	2019 年 7 月 9 日
Amazon ECS 服務和任務集的資源層級許可	Amazon ECS 已擴大對 Amazon ECS 服務和任務的資源層級許可支援。如需詳細資訊，請參閱 Amazon Elastic Container Service 如何與 IAM 搭配使用。	2019 年 6 月 27 日
新的 Amazon ECS 優化 AMI 補丁為 -2019-005 AWS	Amazon ECS 已更新 Amazon ECS 最佳化 AMI 以解決 AWS-2019-005 中所述的安全漏洞。	2019 年 6 月 17 日
彈性網路介面中繼	Amazon ECS 支援使用受支援的 Amazon EC2 執行個體類型來啟動容器執行個體，這些類型具有更高的彈性網路介面 (ENI) 密度。使用這些執行個體類型並加入 `awsvpcTrunking` 帳戶設定，可在新啟動之容器執行個體上提供更高的 ENI 密度，允許您在每個容器執行個體上放置更多任務。如需詳細資訊，請參閱 增加 Amazon ECS Linux 容器實例網絡界面。	2019 年 6 月 6 日
AWS Fargate 平台版本 1.3.0 更新	從 2019 年 5 月 1 日開始，啟動的任何新 Fargate 任務除了支援 `awslogs` 日誌驅動程式外，還支援 `splunk` 日誌驅動程式。如需詳細資訊，請參閱 儲存與記錄。	2019 年 5 月 1 日
AWS Fargate 平台版本 1.3.0 更新	從 2019 年 5 月 1 日開始，啟動的任何新 Fargate 任務支援使用 `secretOptions` 容器定義參數，以參考容器日誌組態中的敏感資料。如需詳細資訊，請參閱 將敏感資料傳遞至 Amazon ECS 容器。	2019 年 5 月 1 日

變更	描述	日期
AWS Fargate 平台版本 1.3.0 更新	從 2019 年 4 月 2 日開始，任何啟動的新 Fargate 工作都支援將敏感資料注入容器，方法是將敏感資料儲存在機 AWS Secrets Manager 密或參數存放區 AWS Systems Manager 參數中，然後在容器定義中參照這些參數。如需詳細資訊，請參閱 將敏感資料傳遞至 Amazon ECS 容器。	2019 年 4 月 2 日
AWS Fargate 平台版本 1.3.0 更新	從 2019 年 3 月 27 日開始，啟動的任何新的 Fargate 任務可以使用額外的任務定義參數，讓您定義代理組態、容器啟動和關閉的相依性，以及每一容器的啟動和停止逾時值。如需詳細資訊，請參閱代理組態、容器相依性及容器逾時。	2019 年 3 月 27 日
Amazon ECS 推出外部部署類型	外部部署類型可讓您使用任何第三方部署控制器，以完全控制 Amazon ECS 服務的部署程序。如需詳細資訊，請參閱 使用第三方控制器部署 Amazon ECS 服務。	2019 年 3 月 27 日
AWS Amazon ECS 上的 Deep Learning Containers	AWS Deep Learning Containers 是一組 Docker 映像檔，用於在 Amazon 彈性容器服務 (Amazon ECS) TensorFlow 上訓練和服務模型。Deep Learning Containers 提供優化的環境 TensorFlow，包括 Nvidia CUDA（適用於 GPU 實例）和英特爾 MKL（用於 CPU 實例）庫，並可在 Amazon ECR 中使用。如需詳細資訊，請參閱 在 Amazon ECS 上使用 AWS Deep Learning Containers。	2019 年 3 月 27 日
Amazon ECS 推出增強型容器相依性管理	Amazon ECS 推出其他任務定義參數，讓您可以定義容器啟動和關閉的相依性，以及每個容器的啟動和停止逾時值。如需詳細資訊，請參閱 容器相依性。	2019 年 3 月 7 日

變更	描述	日期
Amazon ECS 推出 `PutAccountSettingDefault` API	Amazon ECS 推出 `PutAccountSettingDefault` API，可讓使用者為帳戶上所有使用者和角色設定預設 ARN/ID 格式選擇加入狀態。在過去，必須具有帳戶擁有者身分才能設定帳戶的預設選擇狀態。 如需詳細資訊，請參閱 Amazon Resource Name (ARN) 和 ID。	2019 年 2 月 8 日
Amazon ECS 支援 GPU 工作負載	Amazon ECS 透過讓您建立具啟用 GPU 容器執行個體的叢集來支援 GPU 工作負載。您可以在任務定義中指定所需 GPU 和 ECS 代理程式的數量會將實體 GPU 釘選至容器。 如需詳細資訊，請參閱 GPU 工作負載的 Amazon ECS 任務定義。	2019 年 2 月 4 日
Amazon ECS 擴充秘密支援	Amazon ECS 擴展了直接在任務定義中使用機 AWS Secrets Manager 密的支援，將敏感資料插入您的容器。 如需詳細資訊，請參閱 將敏感資料傳遞至 Amazon ECS 容器。	2019 年 1 月 21 日
介面 VPC 端點 (AWS PrivateLink)	新增支援設定採用 AWS PrivateLink技術的介面 VPC 端點。這可讓您在 VPC 與 Amazon ECS 之間建立私有連線，而不需要透過網際網路、NAT 執行個體、VPN 連接或 AWS Direct Connect進行存取。 如需詳細資訊，請參閱介面 VPC 端點 (AWS PrivateLink)。	2018 年 12 月 26 日

變更	描述	日期
AWS Fargate 平台 1.3.0 版本	新的 AWS Fargate 平台版本發布，其中包含： • 已新增使用 AWS Systems Manager 參數存放區參數將敏感資料插入容器的支援。 如需詳細資訊，請參閱 將敏感資料傳遞至 Amazon ECS 容器。 • 增加了對 Fargate 任務的任務回收機制，此過程將重新整理屬於 Amazon ECS 服務的各項任務。 如需詳細資訊，請參閱 AWS Fargate的 Amazon Elastic Container Service 使用者指南中的任務維護。 如需詳細資訊，請參閱 適用於 Amazon ECS 的 Fargate Linux 平台版本。	2018 年 12 月 17 日
更新服務限制	以下服務限制已更新： • 每個帳戶每一區域的叢集數目已從 1000 提高到 2000。 • 每個叢集的容器執行個體數目已從 1000 提高到 2000。 • 每個叢集的服務數目已從 500 提高到 1000。 如需詳細資訊，請參閱 Amazon ECS 服務配額。	2018 年 12 月 14 日
AWS Fargate 區域擴張	AWS Fargate 與 Amazon ECS 合作，已擴展至亞太區域 (孟買) 和加拿大 (中部) 區域。 如需詳細資訊，請參閱 支援 AWS Fargate 上 Amazon ECS 的區域。	2018 年 12 月 7 日

變更	描述	日期
Amazon ECS 藍/綠部署	Amazon ECS 使用添加了對藍/綠部署的支持。CodeDeploy此部署類型可在傳送生產流量到服務的新部署之前，讓您先驗證新部署。 如需詳細資訊，請參閱 在部署之前驗證 Amazon ECS 服務的狀態 。	2018 年 11 月 27 日
Amazon ECS 最佳化 Amazon Linux 2 (arm64) AMI 已發佈	Amazon ECS 已發佈 Amazon ECS 最佳化 Amazon Linux 2 AMI，適用於 arm64 架構。 如需詳細資訊，請參閱 Amazon ECS 優化的 Linux AMI。	2018 年 11 月 26 日
在任務定義中增加支援其他 Docker 旗標	Amazon ECS 在任務定義中支援下列 Docker 標記： • IPC 模式 • PID 模式	2018 年 11 月 16 日
支援 Amazon ECS 秘密	Amazon ECS 新增了使 AWS Systems Manager 用參數存放區參數將敏感資料插入容器的支援。 如需詳細資訊，請參閱 將敏感資料傳遞至 Amazon ECS 容器。	2018 年 11 月 15 日
資源標記	Amazon ECS 新增支援將中繼資料標籤新增到您的服務、任務定義、任務、叢集和容器執行個體。 如需詳細資訊，請參閱 標記 Amazon ECS 資源。	2018 年 11 月 15 日
AWS Fargate 區域擴張	AWS Fargate 與 Amazon ECS 合作，已擴展至美國西部 (加利佛尼亞北部) 和亞太區域 (首爾)。 如需詳細資訊，請參閱 AWS Fargate對於 Amazon ECS。	2018 年 11 月 7 日

變更	描述	日期
更新服務限制	以下服務限制已更新： • 每個帳戶每個區域使用 Fargate 啟動類型的任務數量從 20 增加到 50。 • 使用 Fargate 啟動類型之任務的公有 IP 地址數量從 20 增加到 50。 如需詳細資訊，請參閱 Amazon ECS 服務配額。	2018 年 10 月 31 日
AWS Fargate 區域擴張	AWS Fargate 與 Amazon ECS 已經擴展到歐洲（倫敦）地區。 如需詳細資訊，請參閱 AWS Fargate對於 Amazon ECS。	2018 年 10 月 26 日
Amazon ECS 最佳化 Amazon Linux 2 AMI 已發佈	Amazon ECS 銷售兩款針對服務而最佳化的 Linux AMI。最新和建議的版本以 x; 為基礎。Amazon ECS 還會出售以下項目為基礎的 AMI，但我們建議您將工作負載遷移到 Amazon Linux 2 變體，因為對 Amazon Linux AMI 的支援將在 2020 年 6 月 30 日之前結束。 如需詳細資訊，請參閱 Amazon ECS 優化的 Linux AMI。	2018 年 10 月 18 日
Amazon ECS 任務中繼資料端點第 3 版	從 Amazon ECS 容器代理程式 1.21.0 版開始，代理程式會將稱為 ECS_CONTAINER_METADATA_URI 的環境變數插入任務中的每個容器。查詢任務中繼資料第 3 版端點時，在 Amazon ECS 容器代理提供之 HTTP 端點使用 awsvpc 網路模式的任務，皆可使用各種任務中繼資料和 Docker 統計資訊。如需詳細資訊，請參閱 使用 Amazon ECS 中繼資料監控工作負載。	2018 年 10 月 18 日

變更	描述	日期
Amazon ECS 服務探索區域擴展	Amazon ECS 服務探索已將支援擴展至加拿大 (中部)、南美洲 (聖保羅)、亞太區域 (首爾)、亞太區域 (孟買) 和歐洲 (巴黎) 區域。 如需詳細資訊，請參閱 使用服務探索將 Amazon ECS 服務與 DNS 名稱連接。	2018 年 9 月 27 日
新增對容器定義中額外 Docker 標記的支援	Amazon ECS 支援容器定義中的下列 Docker 標記： • 系統控制 • 互動性 • 虛擬終端機	2018 年 9 月 17 日
使用 AWS Fargate 任務為 Amazon ECS 提供私有登錄身份驗證支援	Amazon ECS 透過 AWS Secrets Manager支援使用私有登錄檔身分驗證的 Fargate 任務。此功能可讓您安全地存放您的登入資料，然後在您的容器定義中加以參考，如此可讓您的任務使用私有映像。 如需詳細資訊，請參閱 在 Amazon ECS 中使用非AWS 容器映像。	2018 年 9 月 10 日
Amazon ECS 服務探索區域擴展	Amazon ECS 服務探索已將支援擴展至亞太區域 (新加坡)、亞太區域 (雪梨)、亞太區域 (東京)、歐洲 (法蘭克福) 和歐洲 (倫敦) 區域。 如需詳細資訊，請參閱 使用服務探索將 Amazon ECS 服務與 DNS 名稱連接。	2018 年 8 月 30 日
支援 Fargate 任務的排程任務	Amazon ECS 推出了對 Fargate 啟動類型排程任務的支援。	2018 年 8 月 28 日

變更	描述	日期
使用 AWS Secrets Manager 支援的私人登錄驗證	Amazon ECS 支援使用 AWS Secrets Manager進行私有登錄檔身分驗證。此功能可讓您安全地存放您的登入資料，然後在您的容器定義中加以參考，如此可讓您的任務使用私有映像。 如需詳細資訊，請參閱 在 Amazon ECS 中使用非AWS 容器映像。	2018 年 8 月 16 日
新增 Docker 磁碟區支援	Amazon ECS 推出對 Docker 磁碟區的支援。 如需詳細資訊，請參閱 Amazon ECS 任務的儲存選項。	2018 年 8 月 9 日
AWS Fargate 區域擴張	AWS Fargate 與 Amazon ECS 合作，已擴展至歐洲 (法蘭克福)、亞太區域 (新加坡) 和亞太區域 (雪梨)。 如需詳細資訊，請參閱 AWS Fargate對於 Amazon ECS。	2018 年 7 月 19 日

變更	描述	日期
已新增 Amazon ECS 服務排程器策略	Amazon ECS 推出服務排程器策略概念。 現已推出兩個服務排程器策略概念： • REPLICA - 複本排程策略會置放並在整個叢集中維持所需的任務數量。根據預設，服務排程器會將任務分散至各個可用區域。您可以使用任務置放策略和限制條件，來自訂任務置放決策。如需詳細資訊，請參閱複本策略。 • DAEMON - 常駐程式排程策略會在每個符合您於叢集中所指定所有任務置放限制條件的作用中容器執行個體上，準確地部署一個任務。使用這項策略時，不需指定所需的任務數量、任務置放策略或使用 Service Auto Scaling 政策。如需詳細資訊，請參閱 守護進程策。 Note Fargate 任務不支援 DAEMON 排程策略。 .	2018 年 6 月 12 日
Amazon ECS 容器代理程式 v1.18.0	發行了新版 Amazon ECS 容器代理程式，其新增了下列功能： • 新增使用 ECS_IMAGE_PULL_BEHAVIOR 參數自訂容器代理程式映像提取行為的支援。如需詳細資訊，請參閱 Amazon ECS 容器代理程式組態。 如需詳細資訊，請參閱 amazon-ecs-agent github。	2018 年 5 月 24 日

變更	描述	日期
在設定服務探索時，新增對 `bridge` 和 `host` 網路模式的支援	使用會指定 `bridge` 或 `host` 網路模式的任務定義，新增對 Amazon ECS 服務之服務探索設定的支援。如需詳細資訊，請參閱 使用服務探索將 Amazon ECS 服務與 DNS 名稱連接。	2018 年 5 月 22 日
新增對更多 Amazon ECS 最佳化 AMI 中繼資料參數的支援	新增子參數，讓您可以程式設計的方式擷取 Amazon ECS 最佳化 AMI ID、映像名稱、作業系統、容器代理程式版本，以及執行時間版本。使用 Systems Manager 參數存放 API 查詢中繼資料。如需詳細資訊，請參閱 擷取 Amazon EC 最佳化的 Linux AMI 中繼資料。	2018 年 5 月 9 日
AWS Fargate 區域擴張	AWS Fargate 與 Amazon ECS 合作，已擴展至美國東部 (俄亥俄)、美國西部 (奧勒岡) 和歐洲西部 (愛爾蘭) 區域。 如需詳細資訊，請參閱 AWS Fargate對於 Amazon ECS。	2018 年 4 月 26 日
Amazon ECS 最佳化 AMI 中繼資料擷取	新增的功能可透過 Systems Manager 參數存放 API，以程式設計的方式來擷取 Amazon ECS 最佳化 AMI 中繼資料。如需詳細資訊，請參閱 擷取 Amazon EC 最佳化的 Linux AMI 中繼資料。	2018 年 4 月 10 日
AWS Fargate 平台版	新的 AWS Fargate 平台版本發布，其中包含： • 新增了對 使用 Amazon ECS 中繼資料監控工作負載 的支援。 • 新增了對 運作狀態檢查 的支援。 • 新增對 使用服務探索將 Amazon ECS 服務與 DNS 名稱連接 的支援 如需詳細資訊，請參閱 適用於 Amazon ECS 的 Fargate Linux 平台版本。	2018 年 3 月 26 日

變更	描述	日期
Amazon ECS 服務探索	新增與 Route 53 的整合，以支援 Amazon ECS 服務探索。如需詳細資訊，請參閱 使用服務探索將 Amazon ECS 服務與 DNS 名稱連接。	2018 年 3 月 22 日
支援 Docker shm-size 和 tmpfs	在 Amazon ECS 任務定義中，新增對 Docker shm-size 和 tmpfs 參數的支援。 如需更新之 ECS CLI 語法的詳細資訊，請參閱「Linux 參數」。	2018 年 3 月 20 日
容器運作狀態檢查	在容器定義中，新增對 Docker 運作狀態檢查的支援。如需詳細資訊，請參閱 運作狀態檢查。	2018 年 3 月 8 日
AWS Fargate	添加了 Amazon ECS 與 AWS Fargate 的概述。如需詳細資訊，請參閱 AWS Fargate對於 Amazon ECS。	2018 年 2 月 22 日
Amazon ECS 任務中繼資料端點	從 Amazon ECS 容器代理程式 1.17.0 版開始，在 Amazon ECS 容器代理提供之 HTTP 端點使用 awsvpc 網路模式的任務，皆可使用各種任務中繼資料和 Docker 統計資訊。如需詳細資訊，請參閱 使用 Amazon ECS 中繼資料監控工作負載。	2018 年 2 月 8 日
使用目標追蹤政策的 Amazon ECS Service Auto Scaling	在 Amazon ECS 主控台中新增了對使用目標追蹤政策的 ECS Service Auto Scaling 的支援。如需詳細資訊，請參閱 使用目標指標值擴展您的 Amazon ECS 服務。 在先前 ECS 初次執行精靈的教學課程中，移除擴展步驟，並用新的目標追蹤教學課程取而代之。	2018 年 2 月 8 日
Docker 17.09 支援	新增對 Docker 17.09 的支援。如需詳細資訊，請參閱 Amazon ECS 優化的 Linux AMI。	2018 年 1 月 18 日
新的服務排程器行為	更新了無法啟動之服務任務的行為資訊。記錄了當服務任務連續失敗時觸發的新服務事件消息。	2018 年 1 月 11 日

變更	描述	日期
Elastic Load Balancing 運作狀態檢查初始化等待期間	新增了指定運作狀態檢查等待期間的能力。	2017 年 12 月 27 日
任務層級 CPU 和記憶體	新增了在任務定義中於任務層級指定 CPU 和記憶體的支援。如需詳細資訊，請參閱TaskDefinition。	2017 年 12 月 12 日
任務執行角色	Amazon ECS 容器代理程式會代您呼叫 Amazon ECS API 動作，因此它需要 IAM 政策和角色，才能讓服務知道該代理程式為您所有。任務執行角色涵蓋以下動作： • 呼叫 Amazon ECR 以提取容器映像 • 呼叫儲存 CloudWatch 容器應用程式記錄檔 如需詳細資訊，請參閱 Amazon ECS 任務執行 IAM 角色。	2017 年 12 月 7 日
Windows 容器支援 GA	新增了 Windows Server 2016 容器的支援。如需詳細資訊，請參閱 Amazon ECS 最佳化 AMI 變體。	2017 年 12 月 5 日
AWS Fargate GA	新增了使用 Fargate 啟動類型啟動 Amazon ECS 服務的支援。如需詳細資訊，請參閱 Amazon ECS 啟動類型。	2017 年 11 月 29 日
Amazon ECS 名稱變更	Amazon Elastic Container Service 已重新命名 (之前稱為 Amazon EC2 Container Service)。	2017 年 11 月 21 日

變更	描述	日期
任務聯網	awsvpc 網路模式提供的任務聯網功能可為 Amazon ECS 任務提供與 Amazon EC2 執行個體相同的聯網屬性。當您在任務定義中使用 awsvpc 網路模式時，從該任務定義啟動的所有任務都會取得自己的彈性網路介面、主要私有 IP 地址和內部 DNS 主機名稱。任務聯網功能可簡化容器聯網，讓您更能控制容器化應用程式彼此之間以及與您 VPC 內其他服務的通訊方式。如需詳細資訊，請參閱 適用於 EC2 啟動類型的 Amazon ECS 任務聯網選項。	2017 年 11 月 14 日
Amazon ECS 容器中繼資料	Amazon ECS 容器現在可存取中繼資料，例如其 Docker 容器或映像 ID、聯網組態或 Amazon ARN。如需詳細資訊，請參閱 Amazon ECS 容器中繼資料檔案。	2017 年 11 月 2 日
Docker 17.06 支援	新增對 Docker 17.06 的支援。如需詳細資訊，請參閱 Amazon ECS 優化的 Linux AMI。	2017 年 11 月 2 日
Docker 旗標 device 和 init 的支援	新增了在任務定義中使用 `LinuxParameters` 參數 (`devices` 和 `initProcessEnabled`) 之 Docker device 和 init 功能的支援。如需詳細資訊，請參閱LinuxParameters。	2017 年 11 月 2 日
Docker 旗標 cap-add 和 cap-drop 支援	新增了在任務定義中使用 `LinuxParameters` 參數 (`capabilities`) 之 Docker cap-add 和 cap-drop 功能的支援。如需詳細資訊，請參閱LinuxParameters。	2017 年 9 月 22 日
Network Load Balancer 支援	Amazon ECS 在 Amazon ECS 主控台中新增了對 Network Load Balancer 的支援。	2017 年 9 月 7 日
RunTask 覆蓋	新增了執行任務時的任務定義覆寫支援。這可讓您在變更任務定義時執行任務，而不需要建立新的任務定義修訂。如需詳細資訊，請參閱 以 Amazon ECS 任務的形式執行應用程式。	2017 年 6 月 27 日

變更	描述	日期
Amazon ECS 排程任務	新增了使用 cron 排程任務的支援。	2017 年 6 月 7 日
Amazon ECS 主控台中的 Spot 執行個體	新增了在 Amazon ECS 主控台內建立 Spot 機群容器執行個體的支援。如需詳細資訊，請參閱 啟動 Amazon ECS Linux 容器執行個體。	2017 年 6 月 6 日
新的 Amazon ECS 最佳化 AMI 版本的 Amazon SNS 通知	新增了訂閱新 Amazon ECS 最佳化 AMI 版本之 SNS 通知的能力。	2017 年 3 月 23 日
微服務和批次任務	新增了兩個 Amazon ECS 常用案例的文件：微服務和批次任務。如需詳細資訊，請參閱 Amazon ECS 相關信息。	2017 年 2 月 日
容器執行個體耗盡	新增了容器執行個體耗盡的支援，其提供從叢集移除容器執行個體的方法。如需詳細資訊，請參閱 排空 Amazon ECS 容器實例。	2017 年 1 月 24 日
Docker 1.12 支援	新增對 Docker 1.12 的支援。如需詳細資訊，請參閱 Amazon ECS 優化的 Linux AMI。	2017 年 1 月 24 日
新的任務置放策略	新增了任務置放策略的支援：屬性型置放、分箱封裝、可用區域分散，以及每個主機一個的置放。如需詳細資訊，請參閱 使用策略定義 Amazon ECS 任務放置。	2016 年 12 月 29 日
Beta 版 Windows 容器支援	新增了 Windows Server 2016 容器 (Beta) 的支援。如需詳細資訊，請參閱 Amazon ECS 最佳化 AMI 變體。	2016 年 12 月 20 日
Blox OSS 支援	新增了允許自訂任務排程器的 Blox OSS 支援。如需詳細資訊，請參閱 在 Amazon ECS 上安排您的容器。	2016 年 12 月 1 日
Amazon ECS 事件流活動 CloudWatch	Amazon ECS 現在會將容器執行個體和任務狀態變更傳送至 CloudWatch 事件。如需詳細資訊，請參閱 使用以下方式自動回應 Amazon ECS 錯誤 EventBridge。	2016 年 11 月 21 日

變更	描述	日期
Amazon ECS 容器記錄到 CloudWatch 日誌	已新增 awslogs 驅動程式將容器記錄資料流傳送至記錄的支援。CloudWatch 如需詳細資訊，請參閱 將 Amazon ECS 日誌傳送到 CloudWatch 。	2016 年 9 月 12 日
具有 Elastic Load Balancing 的 Amazon ECS 服務支援動態連接埠	新增了負載平衡器的支援，以為每個接聽程式支援多個 instance:port 組合，這可增加容器的彈性。您現在已可讓 Docker 動態定義容器的主機連接埠，而 ECS 排程器會向負載平衡器註冊 instance:port。如需詳細資訊，請參閱 使用負載平衡來分配 Amazon ECS 服務流量。	2016 年 8 月 11 日
Amazon ECS 任務的 IAM 角色	新增了將 IAM 角色與任務建立關聯的支援。相對於整個容器執行個體的單一角色，這可為容器提供更精細定義的許可。如需詳細資訊，請參閱 Amazon ECS 任務 IAM 角色。	2016 年 7 月 13 日
Docker 1.11 支援	新增對 Docker 1.11 的支援。如需詳細資訊，請參閱 Amazon ECS 優化的 Linux AMI。	2016 年 5 月 31 日
任務自動調整規模	Amazon ECS 新增了自動擴展服務執行之任務的支援。如需詳細資訊，請參閱 自動擴展您的 Amazon ECS 服務。	2016 年 5 月 18 日
依任務系列篩選的任務定義	新增了根據任務定義系列篩選任務定義清單的支援。如需詳細資訊，請參閱ListTaskDefinitions。	2016 年 5 月 17 日
Docker 容器和 Amazon ECS 代理記錄	Amazon ECS 新增了將 ECS 代理程式和 Docker 容器日誌從容器執行個體傳送到日誌的功能，以 CloudWatch 簡化故障排除問題。	2016 年 5 月 5 日
ECS 最佳化 AMI 現已支援 Amazon Linux 2016.03。	ECS 最佳化 AMI 新增了對 Amazon Linux 2016.03 的支援。如需詳細資訊，請參閱 Amazon ECS 優化的 Linux AMI。	2016 年 4 月 5 日
Docker 1.9 支援	新增對 Docker 1.9 的支援。如需詳細資訊，請參閱 Amazon ECS 優化的 Linux AMI。	2015 年 12 月 22 日

變更	描述	日期
CloudWatch 叢集 CPU 和記憶體保留的指標	Amazon ECS 為 CPU 和內存保留添加了自定義 CloudWatch 指標。	2015 年 12 月 22 日
新的 Amazon ECS 初次執行體驗	Amazon ECS 主控台初次執行體驗新增了角色的免按建立。	2015 年 23 月 11 日
跨可用區域的任務置放	Amazon ECS 服務排程器新增了跨可用區域之任務置放的支援。	2015 年 10 月 8 日
CloudWatch Amazon ECS 叢集和服務的指標	Amazon ECS 針對叢集中的每個容器執行個體、服務和任務定義系列新增 CPU 和記憶體使用率的自訂 CloudWatch 指標。這些新指標可用於使用 Auto Scaling 群組擴展叢集中的容器執行個體，或建立自訂 CloudWatch 警示。	2015 年 8 月 17 日
UDP 連接埠支援	新增了任務定義中 UDP 連接埠的支援。	2015 年 7 月 7 日
環境變數覆寫	增加了對 runTask 的支援 deregisterTaskDefinition 和環境變數覆寫。	2015 年 6 月 18 日
自動化 Amazon ECS 代理更新	新增了查看容器執行個體上所執行之 ECS 代理版本的能力。也能夠從 AWS Management Console、 AWS CLI和 SDK 更新 ECS 代理程式。	2015 年 6 月 11 日
Amazon ECS 服務排程器和 Elastic Load Balancing 整合	新增了定義服務並將該服務與 Elastic Load Balancing 負載平衡器建立關聯的能力。	2015 年 4 月 9 日
Amazon ECS GA	Amazon ECS 在美國東部 (維吉尼亞北部)、美國西部 (奧勒岡)、亞太區域 (東京) 及歐洲 (愛爾蘭) 區域全面發行。	2015 年 4 月 9 日

本文為英文版的機器翻譯版本，如內容有任何歧義或不一致之處，概以英文版為準。