



開發人員指南

Amazon Elastic Container Service



Amazon Elastic Container Service: 開發人員指南

Copyright © 2026 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
術語和要素	1
功能	3
佈建中	3
定價	4
相關服務	4
開始使用	5
設定	5
AWS 管理主控台	5
註冊 AWS 帳戶	6
建立具有管理存取權的使用者	6
建立 Virtual Private Cloud	7
建立安全群組	8
建立憑證來連線至 EC2 執行個體	11
安裝 AWS CLI	12
使用 Amazon ECS 的後續步驟	12
建立容器映像	13
先決條件	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
了解如何為 Amazon ECS 受管執行個體建立任務	19
先決條件	19
步驟 1：建立叢集	20
步驟 2：建立任務定義	20
步驟 3：建立服務	22
步驟 4：檢視服務	22
步驟 5：清除	22
了解如何使用 為 Amazon ECS 受管執行個體建立任務 AWS CLI	23
先決條件	24
步驟 1：建立叢集	24
步驟 2：建立 Amazon ECS 受管執行個體容量提供者	25
步驟 3：設定叢集的預設容量提供者策略	26

步驟 4：註冊 Linux 任務定義	27
步驟 5：列出任務定義	28
步驟 6：建立服務	29
步驟 7：列出服務	29
步驟 8：描述正在執行中的服務	30
步驟 9：測試	31
步驟 10：清除	33
了解如何為 Fargate 建立 Linux 任務	34
先決條件	34
步驟 1：建立叢集	35
步驟 2：建立任務定義	35
步驟 3：建立服務	37
步驟 4：檢視服務	37
步驟 5：清除	37
了解如何為 Fargate 建立 Windows 任務	38
先決條件	38
步驟 1：建立叢集	39
步驟 2：註冊 Windows 任務定義	39
步驟 3：以您的任務定義建立服務	41
步驟 4：檢視服務	41
步驟 5：清除	42
了解如何為 EC2 建立 Windows 任務	42
先決條件	42
步驟 1：建立叢集	43
步驟 2：註冊任務定義	45
步驟 3：建立服務	46
步驟 4：檢視服務	46
步驟 5：清除	47
使用 AWS CDK	48
步驟 1：設定您的 AWS CDK 專案	49
步驟 2：使用 AWS CDK 在 Fargate 上定義容器化 Web 伺服器	51
步驟 3：測試 Web 伺服器	58
步驟 4：清理	58
後續步驟	59
使用 AWS Copilot CLI 建立資源	59
安裝 AWS Copilot CLI	60

使用 AWS Copilot CLI 部署範例 Amazon ECS 應用程式	69
使用的 Amazon ECS CloudFormation	72
使用 CloudFormation 主控台建立 Amazon ECS 資源	72
先決條件	73
步驟 1：建立堆疊範本	73
步驟 2：建立 Amazon ECS 資源的堆疊	85
步驟 3：驗證	85
步驟 4：清除資源	86
使用的 AWS CLI 命令建立 Amazon ECS 資源 CloudFormation	86
先決條件	86
步驟 1：建立堆疊	87
步驟 2：驗證 Amazon ECS 資源建立	99
步驟 3：清理	101
範本範例 CloudFormation	101
任務定義	102
容量提供者	114
叢集	116
服務	128
適用於 Amazon ECS 的 IAM 角色	202
最佳實務	206
規劃 Amazon ECS 解決方案架構	209
Capacity	209
服務端點	210
聯網	211
功能存取	211
IAM 角色	212
日誌	213
容器映像最佳實務	213
啟動類型和容量提供者	213
最佳實務	213
服務可變性	214
使用雙堆疊端點	216
使用雙堆疊端點 (AWS CLI)	216
使用雙堆疊端點AWS SDKs)	217
從 REST API 使用雙堆疊端點	218
共用子網路、Local Zone 與 Wavelength Zone 中的應用程式	218

共用子網路	219
本機區域	219
Wavelength 區域	220
上的 Amazon Elastic Container Service AWS Outposts	220
考量事項	221
先決條件	221
在 上建立叢集的概觀 AWS Outposts	221
容器映像的最佳實務	224
聯網最佳實務	225
將應用程式連線至網際網路	226
接收 Amazon ECS 傳入連線的最佳實務	229
連線至 AWS 服務的最佳實務	233
連接服務的最佳實務	236
跨 AWS 帳戶與 VPC 部署服務的聯網最佳實務	240
AWS 用於聯網故障診斷的 服務	240
最佳化任務啟動時間	241
大規模運作	242
Amazon ECS 服務配額與 API 限流限制	243
因應限流問題	247
自動擴展與容量管理最佳實務	249
決定任務大小	250
最佳化服務自動擴展	251
容量與可用性	255
叢集容量	258
選擇 Fargate 任務大小	259
加速叢集自動擴展	259
透過帳戶設定使用功能	260
Amazon Resource Name (ARN) 和 ID	262
ARN 和資源 ID 格式時間表	264
Container Insights	264
AWS Fargate 聯邦資訊處理標準 (FIPS-140) 合規	266
標記授權	267
標記授權時間表	268
AWS Fargate 任務淘汰等待時間	268
AWS Fargate 使用 EC2 事件時段淘汰任務	269
增加 Linux 容器執行個體網路介面數量	270

執行時期監控 (Amazon GuardDuty 整合)	270
雙堆疊 IPv6 VPC	271
預設日誌驅動程式模式	271
使用主控台檢視帳戶設定	272
修改帳戶設定	273
還原為預設的帳戶設定	274
使用 管理帳戶設定 AWS CLI	274
適用於 Amazon ECS 的 IAM 角色	276
Amazon ECS 受管執行個體	279
開始使用	279
容量提供者	279
執行個體選擇與最佳化	280
任務定義	281
作業系統與 CPU 架構	281
主要功能	281
IAM 角色	282
安全和合規	282
聯網	282
執行個體儲存體	283
本機儲存體	284
服務負載平衡	286
監控與可觀測性	286
定價與成本最佳化	286
Service Quotas	287
遷移考量事項	287
限制及考量	287
組織控制	288
封鎖所有 AMIs SCP 範例，但 Amazon ECS 受管執行個體 AMIs 除外	288
Amazon ECS 受管執行個體 AMI 帳戶	289
Amazon ECS 受管執行個體的執行個體類型	290
Amazon ECS 受管執行個體的執行個體系列	291
執行個體選擇方法	292
特定執行個體類型選取	292
屬性型執行個體類型選項	293
帳單和購買選項	295
Amazon ECS 受管執行個體的執行個體選擇最佳實務	295

Amazon ECS 受管執行個體提取行為	296
Amazon ECS 受管執行個體中的修補	297
執行個體生命週期	297
事件時段與維護排程	297
維護序列	298
Amazon ECS 受管執行個體的安全考量	282
安全模型	298
安全性功能	298
合規與法規支援	300
Amazon ECS 受管執行個體 FIPS-140 考量	300
在 Amazon ECS 受管執行個體上停用 FIPS	300
安全考量	301
安全最佳實務	302
啟用 Amazon ECS 受管執行個體的 VPC 加密控制	302
先決條件	303
識別相容的執行個體類型	303
建立支援 VPC 加密的叢集	303
考量事項	304
疑難排解	304
基礎結構最佳化	304
閒置執行個體偵測	305
未充分利用的執行個體偵測	305
ScaleInAfter	305
從 Fargate 遷移至 Amazon ECS 受管執行個體	306
遷移考量事項	306
先決條件	307
步驟 1：更新叢集以使用 Amazon ECS 受管執行個體	307
步驟 2：更新任務定義以取得 Amazon ECS 受管執行個體功能	307
步驟 3：更新服務以使用 Amazon ECS 受管執行個體容量提供者	307
步驟 4：遷移獨立任務	308
從 EC2 遷移至 Amazon ECS 受管執行個體	308
遷移考量事項	308
先決條件	309
步驟 1：更新叢集以使用 Amazon ECS 受管執行個體	309
步驟 2：更新任務定義以取得 Amazon ECS 受管執行個體功能	309
步驟 3：更新服務以使用 Amazon ECS 受管執行個體容量提供者	309

步驟 4：遷移獨立任務	309
Amazon ECS 受管協助程式	310
Managed Daemons 的運作方式	310
主要優點	310
開始使用	311
常駐程式任務定義	311
建立協助程式任務定義	311
協助程式任務定義參數	312
網路模式	313
特殊權限功能	313
磁碟區	314
支援的容器參數	315
建立和管理協助程式	316
先決條件	316
建立協助程式	316
驗證協助程式部署	318
更新協助程式	319
刪除協助程式	319
協助程式部署	320
協助程式部署的運作方式	320
部署組態參數	321
監控部署	321
協助程式部署案例	322
協助程式事件	325
協助程式服務動作事件	325
協助程式部署狀態變更事件	327
建立協助程式事件的規則	328
協助程式指標	329
協助程式使用率指標	330
檢視協助程式指標	331
協助程式自動修復	332
AWS Fargate	333
逐步解說	333
容量提供者	333
任務定義	334
平台版本	334

服務負載平衡	334
用量指標	335
Fargate 適用時機安全考量	335
Fargate 安全最佳實務	335
使用 AWS KMS 加密 Fargate 的暫時性儲存	335
Fargate 核心系統呼叫追蹤的 SYS_PTRACE 功能	336
搭配 Fargate 執行時期監控使用 Amazon GuardDuty	336
Fargate 安全考量	336
Fargate 平台版本	337
遷移至 Linux 平台版本 1.4.0	338
Linux 平台版本變更日誌	339
Linux 平台版本棄用	341
Windows 平台版本變更日誌	343
為 Amazon ECS 選用 Fargate Windows 容器的安全考量	344
Fargate 容器映像上 Linux 容器的提取行為	345
Fargate 容器映像上 Windows 容器的提取行為	346
Fargate 任務暫時性儲存	347
Fargate Linux 容器平台版本	347
Fargate Windows 容器平台版本	348
AWS Fargate 暫時性儲存的客戶自管金鑰	348
任務淘汰與維護	361
任務淘汰通知概觀	361
我可以選擇退出任務淘汰機制嗎？	365
我可以透過其他服務取得任務淘汰通知 AWS 嗎？	365
我可以在任務排定後變更任務淘汰嗎？	365
Amazon ECS 如何處理屬於服務組成部分的任務？	366
Amazon ECS 是否可以自動處理獨立任務？	366
對任務淘汰期間的服務可用性進行疑難排解	366
準備在 Amazon ECS 上淘汰 AWS Fargate 任務	366
AWS Fargate 區域	371
AWS Fargate 上的 Linux 容器	371
AWS Fargate 上的 Windows 容器	374
EC2	376
EC2 與外部執行個體的容器映像提取行為	377
Amazon ECS Anywhere	379
Amazon ECS Express 模式	380

Amazon ECS Express Mode 服務的優點	380
定價	381
可用性	381
建立快速模式服務	381
先決條件	381
逐步解說	382
自訂	382
在主控台中建立您的第一個快速模式服務	383
程序	383
後續步驟	384
自訂	384
使用 建立您的第一個快速模式服務 AWS CLI	385
先決條件	385
步驟 1：建立 IAM 角色	386
步驟 2：建立您的第一個快速模式服務應用程式	387
步驟 3：使用自訂設定建立快速模式服務應用程式	382
步驟 4：監控您的部署	388
步驟 5：存取您的應用程式	389
Express 模式服務建立的資源	389
叢集預設值	390
任務定義預設值	390
容器定義預設值	391
記錄預設值	391
服務預設值	392
網路組態預設值	393
IAM 角色預設值	394
Application Auto Scaling 預設值	395
應用程式擴展目標預設值	395
應用程式擴展政策預設值	396
Application Load Balancer 預設	396
目標群組預設值	397
資源共用和成本最佳化	397
檢視快速模式服務的詳細資訊	398
更新 Amazon ECS Express Mode 服務	399
部署行為	399
程序	399

刪除快速模式服務	401
刪除快速模式服務和 服務之間的差異	402
程序	402
Amazon ECS Express Mode 服務的最佳實務	402
安全最佳實務	403
效能和運算最佳化	404
操作最佳實務	405
疑難排解	406
部署問題	406
連線問題	408
效能問題	408
監控和偵錯工具	410
在快速模式之外更新資源	410
了解共同責任模型	410
在快速模式服務中利用 Amazon ECS 功能之前，是否需要執行任何動作？	411
快速模式外的範例更新	411
任務定義	414
任務定義狀態	415
可以封鎖刪除的 Amazon ECS 資源	416
架構應用程式	416
任務大小的最佳實務	418
Amazon ECS 受管執行個體的任務聯網	419
EC2 的任務聯網	423
Fargate 任務聯網	436
任務的儲存選項	442
管理容器交換記憶體空間	522
Amazon ECS 受管執行個體的任務定義差異	523
Fargate 的任務定義差異	525
EC2 Windows 執行個體的任務定義差異	533
使用主控台建立任務定義	534
JSON 驗證	534
CloudFormation 堆疊	534
程序	534
使用 Amazon Q Developer 在 Amazon ECS 主控台中提供任務定義建議	557
先決條件	558
程序	558

使用主控台更新任務定義	560
JSON 驗證	561
程序	561
使用主控台取消註冊任務定義修訂版	562
CloudFormation 堆疊	534
程序	563
使用主控台刪除任務定義修訂版	563
可以封鎖刪除的 Amazon ECS 資源	416
程序	564
任務定義使用案例	564
GPU 工作負載的任務定義	565
影片轉碼工作負載的任務定義	577
AWS Neuron 機器學習工作負載的任務定義	589
深度學習執行個體的任務定義	597
64 位元 ARM 工作負載的任務定義	599
將日誌傳送至 CloudWatch	601
將日誌傳送至 AWS 服務或 AWS Partner	605
使用非AWS 容器映像	629
在任務中重新啟動個別容器	632
將敏感資料傳遞至容器	634
Amazon ECS 受管執行個體的任務定義參數	654
系列	654
Capacity	654
任務 角色	655
任務執行 角色	655
網路模式	656
執行時間平台	656
任務大小	657
其他任務定義參數	658
容器定義	664
Fargate 的任務定義參數	703
系列	703
Capacity	704
任務 角色	704
任務執行 角色	704
網路模式	705

執行時間平台	705
任務大小	706
容器定義	709
Elastic Inference 加速器名稱 (已棄用)	749
代理組態	750
磁碟區	752
Tags (標籤)	758
其他任務定義參數	759
EC2 的任務定義參數	761
系列	761
Capacity	761
任務 角色	762
任務執行 角色	762
網路模式	763
執行時間平台	764
任務大小	765
容器定義	766
Elastic Inference 加速器名稱 (已棄用)	809
任務置放限制條件	810
代理組態	811
磁碟區	813
Tags (標籤)	819
其他任務定義參數	820
任務定義範本	821
任務定義範例	832
Web 伺服器	833
splunk 日誌驅動程式	835
fluentd 日誌驅動程式	835
gelf 日誌驅動程式	836
外部執行個體上的工作負載	837
Amazon ECR 映像與任務定義 IAM 角色	838
進入點與命令	839
容器相依性	839
任務定義中的磁碟區	841
Windows 任務定義範例	842
叢集	843

選擇合適的叢集類型	843
叢集元件	844
叢集概念	844
容量提供者	845
Amazon ECS 受管執行個體叢集	847
為 Amazon ECS 受管執行個體建立叢集	847
更新叢集以使用 Amazon ECS 受管執行個體	851
受管執行個體容量提供者	854
任務縮減保護	862
Fargate 的叢集	869
Fargate Spot 終止通知	870
為 Fargate 工作負載建立叢集	871
EC2 工作負載的 Amazon ECS 容量提供者	874
EC2 容器執行個體安全	875
為 Amazon EC2 工作負載建立叢集	876
叢集自動擴展	881
Amazon EC2 容器執行個體	911
外部執行個體的叢集	1052
支援的作業系統和系統架構	1052
考量事項	1052
為外部執行個體工作負載建立叢集	1056
將外部執行個體註冊到 Amazon ECS 叢集	1058
取消註冊外部執行個體	1063
更新 AWS Systems Manager 代理程式和 Amazon ECS 容器代理程式	1068
更新叢集	1073
刪除叢集	1074
取消註冊容器執行個體	1075
程序	1076
容器執行個體耗盡	1076
服務的耗盡行為	1076
獨立任務的耗盡行為	1077
Amazon ECS 受管執行個體的排空行為	905
程序	1078
在 Auto Scaling 群組之外變更執行個體類型或大小	1079
Amazon EC2 容器執行個體	1079
生命週期	1080

Docker 支援	1081
Amazon ECS 最佳化 AMI	1081
其他資訊	1081
容器代理程式組態	1082
安裝 Amazon ECS 容器代理程式	1084
容器代理程式日誌組態參數	1090
為私有 Docker 映像檔設定容器執行個體	1091
清理任務與映像	1095
為容器排程	1098
運算選項	1099
任務生命週期	1100
生命週期狀態	1101
Amazon ECS 如何在容器執行個體上置放任務	1102
Amazon ECS 受管執行個體	1103
EC2	1103
Fargate	1104
Amazon ECS 受管執行個體自動擴展與任務置放	1104
使用策略來定義任務置放	1108
群組相關任務	1112
定義用於任務的容器執行個體	1113
獨立任務	1123
任務工作流程	1123
將應用程式作為任務執行	1123
使用 Amazon EventBridge 排程器排程任務	1132
停止任務	1137
服務	1138
基礎結構運算選項	1139
服務自動擴展	1140
服務負載平衡	1140
互連服務	1141
服務部署控制器與策略	1141
服務部署	1312
服務修訂版	1320
可用區域重新平衡	1328
透過負載平衡分佈服務流量	1335
服務自動擴展	1347

互連服務	1380
任務縮減保護	1445
搭配 Amazon ECS 與 Fargate 使用故障注入	1453
更新 Amazon ECS 服務參數	1461
刪除服務	1485
遷移服務短 ARN	1486
服務調節邏輯	1491
服務定義參數	1492
服務定義範本	1518
標記資源	1526
如何標記資源	1526
在建立期間標記資源	1529
限制	1530
Amazon ECS 受管標籤	1530
將標籤用於計費	1531
將標籤新增至資源	1531
Amazon ECS 受管執行個體的標籤	1533
由 新增的標籤 AWS	1534
自訂標籤	1534
將標籤新增至 EC2 容器執行個體	1535
將標籤新增至外部容器執行個體	1537
用量報告	1537
任務層級成本和用量	1538
監控	1540
監控 Amazon ECS 的最佳實務	1540
監控工具	1541
自動化工具	1541
手動工具	1542
使用 CloudWatch 監控 Amazon ECS	1543
考量事項	1544
建議的指標	1544
檢視 Amazon ECS 指標	1545
Amazon ECS CloudWatch 指標	1546
AWS Fargate 用量指標	1567
Amazon ECS 叢集保留量指標	1568
Amazon ECS 叢集使用率指標	1570

Amazon ECS 服務使用率指標	1571
使用 EventBridge 自動化因應 Amazon ECS 錯誤	1575
Amazon ECS 事件	1576
處理事件	1598
使用具有增強可觀測性的 Container Insights 監控 Amazon ECS	1602
Amazon ECS 受管執行個體的其他指標	1603
監控 Amazon ECS 受管執行個體	1604
Container Insights 監控	1604
執行個體監控	1604
使用容器運作狀態檢查來判斷任務運作狀態	1606
如何判斷任務運作狀態	1607
運作狀態檢查與代理程式中斷連線	1608
檢視容器運作狀態	1609
監控 Amazon ECS 容器執行個體運作狀態	1609
容器執行個體運作狀態問題	1610
使用應用程式追蹤資料識別 Amazon ECS 最佳化機會	1610
AWS Distro for OpenTelemetry 與 整合所需的 IAM 許可 AWS X-Ray	1611
指定 AWS Distro for OpenTelemetry 附屬項目以在任務定義中 AWS X-Ray 整合	1611
使用應用程式指標關聯 Amazon ECS 應用程式效能	1613
將應用程式指標匯出至 Amazon CloudWatch	1613
將應用程式指標匯出至 Amazon Managed Service for Prometheus	1617
使用 記錄 Amazon ECS API 呼叫 AWS CloudTrail	1620
CloudTrail 中的 Amazon ECS 管理事件	1621
CloudTrail 中的 Amazon ECS 資料事件	1621
Amazon ECS 事件範例	1622
檢視 Amazon ECS 服務和任務的 CloudWatch Logs Live Tail	1624
所需的許可	1624
程序	1624
AWS CLI 參考	1625
使用中繼資料監控工作負載	1626
環境變數	1627
容器中繼資料檔案	1627
Amazon ECS 受管執行個體上任務可用的任務中繼資料	1633
EC2 上 Amazon ECS 任務的任務中繼資料	1645
可用於 Fargate 上任務的任務中繼資料	1686
容器自我檢查	1708

使用執行時期監控識別未經授權的行為	1711
執行時期監控如何與 Amazon ECS 搭配運作	1711
考量事項	1712
資源使用率	1712
Fargate 工作負載的執行時期監控	1713
EC2 工作負載的執行時期監控	1716
對執行時期監控進行疑難排解	1720
使用 ECS Exec 監控 Amazon ECS 容器	1723
考量事項	1723
Architecture	1725
設定 ECS Exec	1726
使用 ECS Exec 記錄	1728
使用 IAM 政策限制 ECS Exec 的存取	1732
ECS Exec 疑難排解	1736
使用 ECS Exec 執行命令	1736
Compute Optimizer 建議	1738
針對 Fargate 任務大小提出的建議	1739
Amazon ECS MCP 伺服器	1740
概要	1740
開始使用	1741
開始使用	1741
先決條件	1741
設定	1741
常見組態和最佳實務	1747
工具組態	1748
工具組態	1748
環境變數	1749
引數	1749
工具	1750
疑難排解	1754
解決已停止任務錯誤	1758
已停止任務錯誤訊息更新	1758
檢視已停止任務錯誤	1763
已停止任務錯誤訊息	1764
驗證任務連線狀態	1782
檢視 IAM 角色請求	1786

檢視服務事件訊息	1787
Amazon ECS 服務事件訊息	1788
Amazon ECS 服務運作狀態不良事件訊息	1798
Amazon ECS 可用區域服務重新平衡服務事件訊息	1799
對 Amazon ECS 中的服務負載平衡器進行疑難排解	1800
Amazon ECS 中的服務自動擴展疑難排解	1802
事件擷取	1802
運作方式	1802
Event types (事件類型)	1802
日誌群組組態	1803
考量事項	1803
事件型疑難排解	1804
開啟現有叢集的事件擷取	1804
檢視服務和任務狀態變更事件	1805
對任務定義無效的 CPU 或記憶體錯誤進行疑難排解	1807
檢視容器代理程式日誌	1809
使用 Amazon ECS 日誌收集器收集容器日誌	1810
代理程式自我檢查	1812
Amazon ECS 中的 Docker 診斷	1815
在 Amazon ECS 中列出 Docker 容器	1815
在 Amazon ECS 中檢視 Docker 日誌	1816
在 Amazon ECS 中檢查 Docker 容器	1817
在 Amazon ECS 中設定 Docker 常駐程式的詳細輸出	1818
在 Amazon ECS 中對 Docker API error (500): devmapper 進行疑難排解	1819
對 ECS Exec 問題進行疑難排解	1820
透過 Exec 檢查程式進行驗證	1820
呼叫 execute-command 時發生錯誤	1821
對 Amazon ECS Anywhere 問題進行疑難排解	1821
外部執行個體註冊問題	1821
外部執行個體網路問題	1822
執行任務的問題	1822
對 Fargate 上的 Java 類別載入問題進行疑難排解	1822
徵狀	1823
原因	1823
Resolution	1823
預防	1824

AWS Fargate 調節配額	1824
在 Fargate 中對 RunTask API 進行限流	1825
在 Fargate 中調整速率配額	1825
對 Amazon ECS 受管執行個體進行疑難排解	1826
先決條件	1826
常見的疑難排解案例：	1826
Amazon ECS 受管執行個體中的日誌分析	1833
使用 EC2 AWS CLI 從 Amazon ECS 受管執行個體取得主控台輸出	1833
清除	1833
其他資源	1764
對 Amazon ECS 受管執行個體進行疑難排解	1834
Amazon ECS 受管執行個體錯誤	1838
API 故障原因	1841
使用 Amazon Q Developer 進行故障診斷	1848
所需的許可	1848
程序	1848
考量事項	1849
進一步了解	1849
安全	1850
身分和存取權管理	1850
目標對象	1851
使用身分驗證	1851
使用政策管理存取權	1853
Amazon Elastic Container Service 如何與 IAM 搭配使用	1854
身分型政策範例	1863
AWS Amazon ECS 的 受管政策	1874
使用服務連結角色	1891
適用於 Amazon ECS 的 IAM 角色	1897
Amazon ECS 主控台必要的許可	1957
Amazon ECS 藍/綠部署中 Lambda 函式所需的許可	1967
Amazon ECS 服務自動擴展所需的 IAM 許可	1968
在建立期間標記資源	1969
疑難排解	1973
IAM 最佳實務	1975
記錄和監控	1977
法規遵循驗證	1978

合規與安全最佳實務	1979
AWS Fargate FIPS-140 合規	1980
AWS Fargate FIPS-140 考量事項	1980
在 Fargate 上使用 FIPS	1981
使用 CloudTrail 進行 Fargate FIPS-140 稽核	1981
基礎設施安全性	1983
介面 VPC 端點 (AWS PrivateLink)	1983
共同責任模型	1989
Fargate	1989
EC2	1990
Amazon ECS 受管執行個體共同責任	1991
AWS 責任	1991
客戶責任	1992
網路安全最佳實務	1992
傳輸中加密	1992
任務聯網	1993
AWS PrivateLink 和 Amazon ECS	1994
容器代理程式設定	1995
網路安全建議	1995
任務與容器安全最佳實務	1996
建立最小或使用 distroless 映像	1996
掃描您的映像是否有漏洞	1997
從映像移除特殊權限	1998
建立一組精選映像	1999
掃描應用程式套件和程式庫是否有漏洞	1997
執行靜態程式碼分析	1999
以非根使用者身分執行容器	1999
使用唯讀的根檔案系統	2000
使用 CPU 和記憶體限制來設定任務 (Amazon EC2)	2000
不可變標籤與 Amazon ECR 一起使用	2000
避免以特權方式執行容器 (Amazon EC2)	2001
從容器中移除不必要的 Linux 功能	2001
使用客戶自管金鑰 (CMK) 來加密推送至 Amazon ECR 的映像	2001
教學	2002
使用 為 Fargate 建立 Linux 任務 AWS CLI	2004
先決條件	2004

步驟 1：建立叢集	2005
步驟 2：註冊 Linux 任務定義	2006
步驟 3：列出任務定義	2007
步驟 4：建立服務	2008
步驟 5：列出服務	2008
步驟 6：描述執行中的服務	2009
步驟 7：測試	2011
步驟 8：清除	2015
使用 為 Fargate 建立 Windows 任務 AWS CLI	2015
先決條件	2016
步驟 1：建立叢集	2016
步驟 2：註冊 Windows 任務定義	2017
步驟 3：列出任務定義	2018
步驟 4：建立服務	2019
步驟 5：列出服務	2019
步驟 6：描述執行中的服務	2020
步驟 7：清除	2022
使用 為 EC2 建立任務 AWS CLI	2022
先決條件	2023
建立 叢集	2023
使用 Amazon ECS AMI 啟動容器執行個體	2024
列出容器執行個體	2027
描述容器執行個體	2027
註冊任務定義	2030
列出任務定義	2032
建立服務	2032
列出服務	2033
描述服務	2033
描述執行中的任務	2034
測試 Web 伺服器	2035
清除資源	2036
設定 Amazon ECS 監聽 CloudWatch Events 事件	2038
先決條件：設定測試叢集	2038
步驟 1：建立 Lambda 函數	2038
步驟 2：註冊事件規則	2039
步驟 3：建立任務定義	2040

步驟 4：測試您的規則	2041
針對任務停止事件傳送 Amazon Simple Notification Service 提醒	2042
先決條件：設定測試叢集	2042
先決條件：設定 Amazon SNS 的許可	2042
步驟 1：建立並訂閱 Amazon SNS 主題	2042
步驟 2：註冊事件規則	2043
步驟 3：測試您的規則	2044
串連多行或堆疊追蹤的日誌訊息	2045
所需的 IAM 許可	2046
確定何時使用多行日誌設定	2046
剖析並串連選項	2047
在 Windows 容器上部署 Fluent Bit	2066
先決條件	2068
步驟 1：建立 IAM 存取角色	2068
步驟 2：建立 Amazon ECS Windows 容器執行個體	2069
步驟 3：設定 Fluent Bit	2071
步驟 4：註冊會將日誌路由到 CloudWatch 的 Windows Fluent Bit 任務定義	2072
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	2074
步驟 6：註冊會產生日誌的 Windows 任務定義	2075
步驟 7：執行 windows-app-task 任務定義	2077
步驟 8：驗證 CloudWatch 上的日誌	2077
步驟 9：清除	2078
為 EC2 Linux 容器使用 gMSA	2079
考量事項	2080
先決條件	2080
設定	2081
CredSpec file	2088
在 Fargate 上為 Linux 容器使用 gMSA	2089
考量事項	2089
先決條件	2090
設定	2090
CredSpec file	2093
gMSA 使用 搭配無網域使用 Windows 容器 AWS CLI	2094
先決條件	2095
步驟 1：在 Active Directory Domain Services (AD DS) 上建立和設定 gMSA 帳戶	2096

步驟 2：將憑證上傳至 Secrets Manager	2098
步驟 3：修改您的 CredSpec JSON 以包含無網域 gMSA 資訊	2099
步驟 4：將 CredSpec 上傳至 Amazon S3	2100
步驟 5：(選用) 建立 Amazon ECS 叢集	2101
步驟 6：針對容器執行個體建立 IAM 角色	2101
步驟 7：建立自訂任務執行角色	2101
步驟 8：為 Amazon ECS Exec 建立任務角色	2103
步驟 9：註冊任務定義	2104
步驟 10：註冊 Windows 容器執行個體	2106
步驟 11：驗證容器執行個體	2106
步驟 12：執行 Windows 任務	2107
步驟 13：驗證容器具有 gMSA 憑證	2108
步驟 14：清除	2108
除錯	2110
了解如何為 EC2 Windows 容器使用 gMSA	2110
考量事項	2111
先決條件	2112
設定	2113
透過 Image Builder 建置自訂的 Amazon ECS 最佳化 AMI	2118
清理 Amazon ECS 最佳化 AMI	2119
在基礎結構即程式碼 (IaC) 中使用映像 ARN	2120
搭配 使用映像 ARN CloudFormation	2122
在 Terraform 中使用映像 ARN	2123
使用 AWS 深度學習容器	2124
教學課程：使用藍/綠部署建立服務	2124
先決條件	2124
步驟 1：建立 Application Load Balancer	2125
步驟 2：建立 Amazon ECS 叢集	2126
步驟 3：註冊任務定義	2126
步驟 4：建立 Amazon ECS 服務	2127
步驟 5：建立 AWS CodeDeploy 資源	2129
步驟 6：建立與監控 CodeDeploy 部署	2131
步驟 7：清除	2135
Service Quotas	2137
在 中管理您的服務配額 AWS 管理主控台	2137
因應服務配額與 API 限流限制	2138

Elastic Load Balancing	244
彈性網路介面	245
AWS Cloud Map	246
Amazon ECS API 參考	2144
文件歷史紀錄	2145
.....	mmclxxxiv

什麼是 Amazon Elastic Container Service ?

Tip

加入我們即將推出的容器研討會系列，了解 Amazon ECS 和 的最佳實務 AWS Fargate。 [按一下此處註冊](#)。

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 Amazon Elastic Container Registry 等 AWS 工具以及 Docker 等第三方工具整合。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 中跨 執行和擴展容器工作負載，而無需管理控制平面的複雜性。

術語和要素

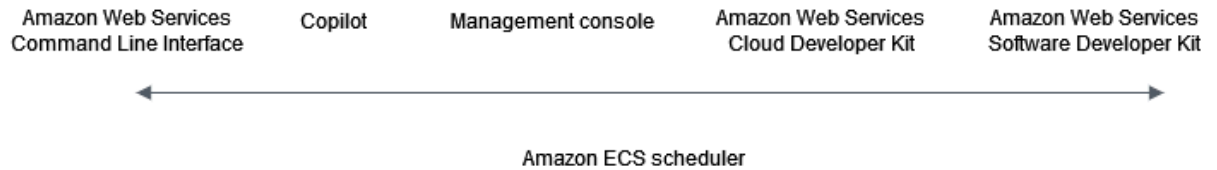
Amazon ECS 有三層：

- 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 Elastic Container Service Layers

Provisioning



Controller



Capacity options



容量即容器執行所在的基礎結構。以下是容量選項概觀：

- Amazon ECS 受管執行個體是 Amazon ECS 的一種運算選項，讓您能在多種 Amazon EC2 執行個體類型上執行容器化工作負載，同時將基礎結構管理工作交由 AWS 負責。透過 Amazon ECS 受管執行個體，您可以存取特定運算功能，例如 GPU 加速、特定 CPU 架構、高網路效能和特殊執行個體類型，同時 AWS 處理基礎基礎設施的佈建、修補、擴展和維護。

- 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。

- AWS 雲端中的無伺服器

Fargate 是一種無伺服器且按使用量付費的運算引擎。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。

Amazon ECS 排程器是管理您應用程式的軟體。

功能

Amazon ECS 提供以下進階功能：

任務定義

應用程式的藍圖。

叢集

應用程式執行所在的基礎結構。

任務

諸如批次作業這類執行工作後即停止的應用程式。

服務

長時間執行的無狀態應用程式。

帳戶設定

允許存取功能。

叢集自動擴展

Amazon ECS 會管理已註冊至您叢集的 Amazon EC2 執行個體的擴展作業。

服務自動擴展

Amazon ECS 會自動增加或減少您服務中所需的任務數量。

佈建中

有多種佈建 Amazon ECS 的選項：

- AWS 管理主控台 - 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) — 為廣泛的 AWS 服務提供命令，包括 Amazon ECS。Windows、Mac 和 Linux 均支援此介面。如需詳細資訊，請參閱[AWS Command Line Interface](#)。
- AWS SDKs — 提供特定語言 APIs，並負責許多連線詳細資訊。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](#)。
- 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。

定價

Amazon ECS 定價視您為容器選擇的容量選項而定。

- [Amazon ECS 定價](#) – Amazon ECS 定價資訊。
- [AWS Fargate 定價](#) – Fargate 定價資訊。

相關服務

搭配 Amazon ECS 一起使用的服務

您可以使用其他 AWS 服務 來協助您在 Amazon ECS 上部署任務和服務。

[Amazon EC2 Auto Scaling](#)

協助確保您有正確的 Amazon EC2 執行個體數量可應付應用程式的負載。

[Amazon CloudWatch](#)

監控您的服務與任務。

[Amazon Elastic Container Registry](#)

儲存與管理容器映像。

[Elastic Load Balancing](#)

自動分配傳入的服務流量。

[Amazon GuardDuty](#)

偵測容器執行個體與工作負載中可能存在的未經授權使用或惡意使用行為。

了解如何建立與使用 Amazon ECS 資源

下列指南介紹可用於存取 Amazon ECS 的工具以及執行容器的入門程序。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 AWS Copilot 命令列界面和 AWS 管理主控台 來完成在 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 Amazon ECS 受管執行個體建立任務](#)
- [了解如何使用 為 Amazon ECS 受管執行個體建立任務 AWS CLI](#)
- [了解如何為 Fargate 建立 Amazon ECS Linux 任務](#)
- [了解如何為 Fargate 建立 Amazon ECS Windows 任務](#)
- [了解如何為 EC2 建立 Amazon ECS Windows 任務](#)
- [使用 建立 Amazon ECS 資源 AWS CDK](#)
- [使用 AWS Copilot 命令列界面建立 Amazon ECS 資源](#)

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。

AWS 管理主控台

AWS 管理主控台 是以瀏覽器為基礎的介面，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 [了解如何建立與使用 Amazon ECS 資源](#)。

一開始，許多客戶偏好使用 主控台，因為它會針對他們採取的動作是否成功提供即時視覺化意見回饋。熟悉 AWS 的客戶可以輕鬆管理相關資源 AWS 管理主控台，例如負載平衡器和 Amazon EC2 執行個體。

從 開始 AWS 管理主控台。

註冊 AWS 帳戶

如果您沒有 AWS 帳戶，請完成下列步驟來建立一個。

註冊 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或簡訊，並在電話鍵盤輸入驗證碼。

當您註冊時 AWS 帳戶，AWS 帳戶根使用者會建立。根使用者有權存取該帳戶中的所有 AWS 服務和資源。作為安全最佳實務，請將管理存取權指派給使用者，並且僅使用根使用者來執行 [需要根使用者存取權的任務](#)。

AWS 會在註冊程序完成後傳送確認電子郵件給您。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。

建立具有管理存取權的使用者

註冊後 AWS 帳戶，請保護 AWS 帳戶根使用者、啟用 AWS IAM Identity Center 和建立管理使用者，以免將根使用者用於日常任務。

保護您的 AWS 帳戶根使用者

1. 選擇根使用者並輸入 AWS 帳戶 您的電子郵件地址，以帳戶擁有者 [AWS 管理主控台](#) 身分登入。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)。

如需說明，請參閱《IAM 使用者指南》中的 [為您的 AWS 帳戶根使用者（主控台）啟用虛擬 MFA 裝置](#)。

建立具有管理存取權的使用者

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 Identity Center 中，將管理存取權授予使用者。

如需使用 IAM Identity Center 目錄 做為身分來源的教學課程，請參閱AWS IAM Identity Center 《使用者指南》中的[使用預設值設定使用者存取 IAM Identity Center 目錄](#)。

以具有管理存取權的使用者身分登入

- 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 Identity Center 使用者登入的說明，請參閱AWS 登入 《使用者指南》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 Identity Center 中，建立一個許可集來遵循套用最低權限的最佳實務。

如需指示，請參閱《AWS IAM Identity Center 使用者指南》中的[建立許可集](#)。

2. 將使用者指派至群組，然後對該群組指派單一登入存取權。

如需指示，請參閱《AWS IAM Identity Center 使用者指南》中的[新增群組](#)。

建立 Virtual Private Cloud

您可以使用 Amazon Virtual Private Cloud (Amazon VPC) 將 AWS 資源啟動至您定義的虛擬網路。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 VPC User Guide 中的 [Work with your default VPC and default subnets](#)。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 User Guide 中的 [Create a 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[區域與可用區域](#)。

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。

如需有關如何建立安全群組的詳細資訊，請參閱 Amazon EC2 User Guide 中的 [Create a security group for your Amazon EC2 instance](#)，並使用下表判斷要選取哪些選項。

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。

如需有關要為使用案例新增的傳出規則的資訊，請參閱 Amazon EC2 User Guide 中的 [Security group rules for different use cases](#)。

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。

將以下三個傳入規則新增至安全群組。如需有關如何建立安全群組的資訊，請參閱 Amazon EC2 User Guide 中的 [Configure security group rules](#)。

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	

選項	Value	
	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

對於 Amazon ECS, 只有當您想要使用 EC2 時, 才需要金鑰對。

AWS 使用公有金鑰密碼編譯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。如需有關區域的詳細資訊, 請參閱 Amazon EC2 User Guide 中的 [Regions and Availability Zones](#)。

建立一組金鑰對

- 使用 Amazon EC2 主控台來建立金鑰對。如需有關建立金鑰對的詳細資訊，請參閱 Amazon EC2 User Guide 中的 [Create a key pair](#)。

如需有關連線至執行個體的資訊，請參閱 Amazon EC2 User Guide 中的 [Connect to your Linux instance](#)。

安裝 AWS CLI

AWS 管理主控台 可用於使用 Amazon ECS 手動管理所有操作。不過，您可以在 AWS CLI 本機桌面或開發人員方塊中安裝，以便建置可在 Amazon ECS 中自動化常見管理任務的指令碼。

若要 AWS CLI 搭配 Amazon ECS 使用，請安裝 AWS CLI 最新版本。如需安裝 AWS CLI 或將其升級至最新版本的詳細資訊，請參閱 AWS Command Line Interface 《使用者指南》中的 [安裝或更新至最新版本的 AWS CLI](#)。

AWS Command Line Interface (AWS CLI) 是您可以用來管理 AWS 服務的統一工具。僅使用這個工具，您就可以控制多個 AWS 服務，並透過指令碼自動化這些服務。中的 Amazon ECS 命令 AWS CLI 是 Amazon ECS API 的反射。

AWS CLI 適合偏好且用來編寫指令碼和與命令列工具互動的客戶，並確切知道他們想要在 Amazon ECS 資源上執行哪些動作。對於想要熟悉 Amazon ECS APIs 的客戶 AWS CLI 來說，也很有幫助。客戶可以使用 AWS CLI 直接從命令列界面對 Amazon ECS 資源執行多項操作，包括建立、讀取、更新和刪除操作。

AWS CLI 如果您已經或想要熟悉 Amazon ECS APIs 和對應的 CLI 命令，並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 也提供命令列工具 [AWS Tools for Windows PowerShell](#)。如需詳細資訊，請參閱「[AWS Tools for Windows PowerShell 使用者指南](#)」。

使用 Amazon ECS 的後續步驟

安裝之後 AWS CLI，當您繼續使用 Amazon ECS 時，可以使用許多不同的工具。下列連結介紹了其中一些工具，並提供了如何將它們與 Amazon ECS 搭配使用的範例。

- 使用 Docker [建立您的第一個容器映像](#)，並將其推送至 Amazon ECR，以便在 Amazon ECS 任務定義中使用。
- [了解如何為 Fargate 建立 Amazon ECS Linux 任務](#)。

- [了解如何為 Fargate 建立 Amazon ECS Windows 任務](#).
- [了解如何為 EC2 建立 Amazon ECS Windows 任務](#).
- 使用您偏好的程式設計語言，利用 [使用 建立 Amazon ECS 資源 AWS CDK](#) 將基礎設施或架構定義為程式碼。
- 使用以自動化部署來定義和管理環境中的所有 AWS 資源 [搭配使用 Amazon ECS AWS CloudFormation](#)。
- 使用完整的 [使用 AWS Copilot 命令列界面建立 Amazon ECS 資源](#) end-to-end 開發人員工作流程來建立、發行和操作符合基礎設施 AWS 最佳實務的容器應用程式。

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。

Amazon ECS 會將容器化應用程式排程在到容器執行個體，或在到 AWS Fargate。容器化應用程式會被封裝為容器映像。此範例說明如何建立 Web 伺服器的容器映像。

您可以建立您的第一個 Docker 映像檔，並將該映像推送至 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。如有關 Docker 的詳細資訊，請參閱 [什麼是 Docker?](#) 與 [Docker documentation](#)。

先決條件

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 Container Registry User Guide 中的 [Moving an image through its lifecycle in 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 Amazon Linux 2023 的 Docker 安裝步驟，請參閱 [在 AL2023 上安裝 Docker](#)。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 AWS CLI 安裝並設定。如需詳細資訊，請參閱 AWS Command Line Interface User Guide 中的 [Installing or updating to the latest version of the AWS CLI](#)。

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，我們提供下列步驟來使用 Amazon Linux 2023 啟動 Amazon EC2 執行個體並安裝 Docker Engine 與 Docker CLI。

在 AL2023 上安裝 Docker

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 User Guide 中的 [Launch an EC2 instance using the launch instance wizard in the console](#)。
2. 連線到您的執行個體。如需詳細資訊，請參閱 Amazon EC2 User Guide 中的 [Connect to your EC2 instance](#)。
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 任務定義使用容器映像啟動您叢集中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

# Update installed packages and install Apache
RUN yum update -y && \
    yum install -y httpd

# 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
```

此 Dockerfile 使用 Amazon ECR Public 上託管的公有 Amazon Linux 2023 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。EXPOSE 指令指容器上的連接埠 80 為接聽連接埠，而 CMD 指令會啟動 Web 伺服器。

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。

如果您在基於 ARM 的系統中執行命令，例如 [Apple Silicon](#)，請使用 `--platform option "--platform linux/amd64"`。

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

4. 列出您的容器映像。

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

輸出：

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

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。

```
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/>。

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 是受管的 AWS 受管映像登錄服務。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。

region 以取代 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 User Guide 中的[Installing or updating to the latest version of the AWS CLI](#)。

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至 Amazon ECR 之後，您便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 “了解如何為 Fargate 建立 Linux 任務”](#)
- [the section called “了解如何為 Fargate 建立 Windows 任務”](#)
- [使用 為 Fargate 建立 Amazon ECS Linux 任務 AWS CLI](#)

了解如何為 Amazon ECS 受管執行個體建立任務

了解如何將 Amazon ECS 受管執行個體與 Amazon ECS 搭配使用來執行容器化應用程式。

先決條件

開始本教學之前，先完成下列動作：

- 您已完成[設定使用 Amazon ECS](#) 一節中的步驟。
- 已完成「[the section called “設定”](#)」中的步驟。
- 您有 Amazon ECS 受管執行個體所需的 IAM 角色。其中包含：
 - 基礎設施角色 - 允許 Amazon ECS 代表您呼叫 AWS 服務，以管理 Amazon ECS 受管執行個體基礎設施。

如需詳細資訊，請參閱[Amazon ECS 基礎結構 IAM 角色](#)。

- 執行個體設定檔 – 為在受管執行個體上執行的 Amazon ECS 容器代理程式與 Docker 常駐程式提供許可。

執行個體角色名稱必須包含 `ecsInstanceRole` 做為字首，以符合基礎設施角色中的 `iam:PassRole` 動作。

如需詳細資訊，請參閱[Amazon ECS 受管執行個體執行個體設定檔](#)。

- 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，因此您的執行個體必須有網際網路存取權。若要將網際網路的路由提供給執行個體，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，並將公有 IP 位址指派給執行個體。

如需詳細資訊，請參閱[the section called “建立 Virtual Private Cloud”](#)。

如需有關安全群組與規則的資訊，請參閱 Amazon Virtual Private Cloud User Guide 中的 [Default security groups for your VPCs](#) 和 [Example rules](#)。

- (選用) AWS CloudShell 是一種工具，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《[AWS CloudShell 使用者指南](#)》中的[什麼是 AWS CloudShell](#)？。

步驟 1：建立叢集

1. 開啟 Amazon ECS 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欄位中，輸入叢集的唯一名稱。
6. 在基礎結構下，選擇 Fargate 與受管 EC2。
7. 設定受管執行個體：
 - a. 在基礎結構角色欄位中，選取您為受管執行個體基礎結構管理工作建立的 IAM 角色。
 - b. 在執行個體設定檔欄位中，選取您建立的 `ecsInstanceRole`。
 - c. 在執行個體屬性欄位中，選擇使用 ECS 預設值。
8. 在聯網下設定受管執行個體的 VPC 與子網路：
 - a. 在 VPC 欄位中，選取託管受管執行個體的 VPC。
 - b. 在子網路欄位中，選取一個或多個將在其中啟動受管執行個體子網路。
 - c. 針對 Security groups (安全群組)，選取一個或多個安全群組。
9. (選用) 若要將標籤新增至叢集，請展開標籤區段，然後設定標籤。
10. 選擇建立。

步驟 2：建立任務定義

任務定義是您應用程式的藍圖。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。請遵循下列步驟建立任務定義：

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. 將下列 JSON 複製並貼到編輯器中，取代預先填入的 JSON：

account-id 將取代為 AWS 您的帳戶 ID，並將 **region** 取代為您正在使用的區域。

```
{
  "family": "managed-instance-tutorial",
  "networkMode": "awsvpc",
  "executionRoleArn": "arn:aws:iam::account-id:role/ecsTaskExecutionRole",
  "containerDefinitions": [
    {
      "name": "sample-app",
      "image": "public.ecr.aws/docker/library/httpd:latest",
      "essential": true,
      "portMappings": [
        {
          "containerPort": 80,
          "hostPort": 80,
          "protocol": "tcp"
        }
      ],
      "logConfiguration": {
        "logDriver": "awslogs",
        "options": {
          "awslogs-group": "/ecs/managed-instance-tutorial",
          "awslogs-region": "region",
          "awslogs-stream-prefix": "ecs"
        }
      }
    }
  ],
  "requiresCompatibilities": [
    "MANAGED_INSTANCES"
  ],
  "cpu": "1024",
  "memory": "2048"
}
```

4. 選擇建立。

步驟 3：建立服務

Amazon ECS 服務可讓您在 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。請遵循下列步驟建立服務：

1. 在導覽窗格中選擇叢集，然後選取 **managed-instance-tutorial** 叢集。
2. 在 Services (服務) 索引標籤上，選擇 Create (建立)。
3. 針對任務定義系列，選擇 managed-instance-tutorial。
4. 對於服務名稱，輸入 **managed-instance-tutorial-service**。
5. 在環境下，選擇容量提供者策略。
6. 在聯網下設定下列項目：
 - a. 選擇現有的 VPC 或建立新的 VPC。
 - b. 對於子網路，選擇要使用的子網路。
 - c. 對於安全群組，請選擇現有的安全群組，或建立新的安全群組，以允許連接埠 80 上的傳入流量。
7. 選擇建立。

步驟 4：檢視服務

服務啟動後，您可以檢視它以了解它的更多資訊並進行測試。

1. 選擇 **managed-instance-tutorial-service** 服務。
2. 從任務索引標籤中，選擇執行中任務的任務 ID。
3. 在網路下的公有 IP 中，選擇開啟位址。
4. 您應該會看到 Apache HTTP 伺服器測試頁面，該頁面可確認 Web 伺服器是否正在正常執行。

步驟 5：清除

完成本教學課程後，您應該清理相關資源，以免未使用的資源產生費用。

1. 在導覽窗格中，選擇叢集。
2. 在叢集頁面上，選取 **managed-instance-tutorial** 叢集。
3. 選擇 Services (服務) 索引標籤。

4. 選取 **managed-instance-tutorial-service** 服務，然後選擇刪除。
5. 在確認提示中，輸入 **delete**，然後選擇刪除。
6. 刪除服務之後，在導覽窗格中選擇叢集。
7. 在叢集頁面上，選取 **managed-instance-tutorial** 叢集，然後選擇刪除叢集。
8. 在確認提示中，輸入 **delete managed-instance-tutorial**，然後選擇刪除。

了解如何使用 為 Amazon ECS 受管執行個體建立任務 AWS CLI

下列步驟會協助您使用 AWS CLI 在 Amazon ECS 中搭配 Amazon ECS 受管執行個體設定叢集、建立容量提供者、註冊任務定義、執行 Linux 任務以及執行其他常見案例。使用 AWS CLI 的最新版本。如需有關如何升級至最新版本的詳細資訊，請參閱 [Installing or updating to the latest version of the AWS CLI](#)。

Note

您可以使用雙堆疊服務端點，透過 IPv4 和 IPv6 從 AWS CLI、SDK 和 Amazon ECS API 與 Amazon ECS 互動。如需詳細資訊，請參閱 [the section called “使用雙堆疊端點”](#)。

主題

- [the section called “先決條件”](#)
- [the section called “步驟 1：建立叢集”](#)
- [the section called “步驟 2：建立 Amazon ECS 受管執行個體容量提供者”](#)
- [the section called “步驟 3：設定叢集的預設容量提供者策略”](#)
- [the section called “步驟 4：註冊 Linux 任務定義”](#)
- [the section called “步驟 5：列出任務定義”](#)
- [the section called “步驟 6：建立服務”](#)
- [the section called “步驟 7：列出服務”](#)
- [the section called “步驟 8：描述正在執行中的服務”](#)
- [the section called “步驟 9：測試”](#)
- [the section called “步驟 10：清除”](#)

先決條件

開始本教學之前，先完成下列動作：

- 您已完成[設定使用 Amazon ECS](#) 一節中的步驟。
- AWS CLI 已安裝並設定最新版本的。如需安裝或升級的詳細資訊 AWS CLI，請參閱AWS Command Line Interface 《使用者指南》中的[安裝或更新至最新版本的 AWS CLI](#)。
- 已完成「[the section called “設定”](#)」中的步驟。
- 您有 Amazon ECS 受管執行個體所需的 IAM 角色。其中包含：
 - 基礎設施角色 - 允許 Amazon ECS 代表您呼叫 AWS 服務，以管理 Amazon ECS 受管執行個體基礎設施。

如需詳細資訊，請參閱[Amazon ECS 基礎結構 IAM 角色](#)。

- 執行個體設定檔 – 為在受管執行個體上執行的 Amazon ECS 容器代理程式與 Docker 常駐程式提供許可。

執行個體角色名稱必須包含 `ecsInstanceRole` 做為字首，以符合基礎設施角色中的 `iam:PassRole` 動作。

如需詳細資訊，請參閱[Amazon ECS 受管執行個體執行個體設定檔](#)。

- 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，因此您的執行個體必須有網際網路存取權。若要將網際網路的路由提供給執行個體，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，並將公有 IP 位址指派給執行個體。

如需詳細資訊，請參閱[the section called “建立 Virtual Private Cloud”](#)。

如需有關安全群組與規則的資訊，請參閱 Amazon Virtual Private Cloud User Guide 中的 [Default security groups for your VPCs](#) 和 [Example rules](#)。

- (選用) AWS CloudShell 是一種工具，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《[AWS CloudShell 使用者指南](#)》中的[什麼是 AWS CloudShell ?](#)。

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 managed-instances-cluster
```

輸出：

```
{
  "cluster": {
    "status": "ACTIVE",
    "defaultCapacityProviderStrategy": [],
    "statistics": [],
    "capacityProviders": [],
    "tags": [],
    "clusterName": "managed-instances-cluster",
    "settings": [
      {
        "name": "containerInsights",
        "value": "disabled"
      }
    ],
    "registeredContainerInstancesCount": 0,
    "pendingTasksCount": 0,
    "runningTasksCount": 0,
    "activeServicesCount": 0,
    "clusterArn": "arn:aws:ecs:region:aws_account_id:cluster/managed-instances-cluster"
  }
}
```

步驟 2：建立 Amazon ECS 受管執行個體容量提供者

您必須先建立容量提供者來定義基礎結構組態，才能使用 Amazon ECS 受管執行個體執行任務。容量提供者會為受管執行個體指定 IAM 角色、網路組態以及其他設定。

使用容量提供者組態建立 JSON 檔案。使用您的實際資源識別碼取代預留位置的值：

```
{
  "name": "managed-instances-cp",
  "cluster": "managed-instances-cluster",
  "managedInstancesProvider": {
    "infrastructureRoleArn":
"arn:aws:iam::aws_account_id:role/ecsInfrastructureRole",
    "instanceLaunchTemplate": {
      "ec2InstanceProfileArn": "arn:aws:iam::aws_account_id:instance-
profile/ecsInstanceRole",
      "networkConfiguration": {
        "subnets": [
          "subnet-abcdef01234567890",
          "subnet-1234567890abcdef0"
        ],
        "securityGroups": [
          "sg-0123456789abcdef0"
        ]
      },
      "storageConfiguration": {
        "storageSizeGiB": 100
      },
      "monitoring": "basic"
    }
  }
}
```

將此組態儲存為 managed-instances-cp.json 並建立容量提供者：

```
aws ecs create-capacity-provider --cli-input-json file://managed-instances-cp.json
```

該命令會在容量提供者建立完成後傳回其描述。

步驟 3：設定叢集的預設容量提供者策略

更新叢集以使用 Amazon ECS 受管執行個體容量提供者作為預設容量提供者策略。這可讓任務和服務自動使用 Amazon ECS 受管執行個體，而無需明確指定容量提供者。

使用叢集容量提供者組態建立 JSON 檔案：

```
{
```



```
{
  "cluster": "managed-instances-cluster",
  "capacityProviders": [
    "managed-instances-cp"
  ],
  "defaultCapacityProviderStrategy": [
    {
      "capacityProvider": "managed-instances-cp",
      "weight": 1
    }
  ]
}
```

將此組態儲存為 `cluster-cp-strategy.json` 並更新叢集：

```
aws ecs put-cluster-capacity-providers --cli-input-json file://cluster-cp-strategy.json
```

步驟 4：註冊 Linux 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。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 [「the section called “Fargate 的任務定義參數”」](#)。

```
{
  "family": "sample-managed-instances",
  "networkMode": "awsvpc",
  "containerDefinitions": [
    {
      "name": "managed-instances-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 using Amazon ECS Managed Instances.</p></div></body></html>' > /usr/local/apache2/htdocs/index.html && httpd-foreground\""
    ],
    "requiresCompatibilities": [
        "MANAGED_INSTANCES"
    ],
    "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/managed-instances-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 5：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-managed-instances:1"
  ]
}

```

步驟 6：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-managed-instances: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 managed-instances-cluster --service-name managed-instances-service --task-definition sample-managed-instances:1 --desired-count 1 --network-configuration "awsvpcConfiguration={subnets=[subnet-abcd1234],securityGroups=[sg-abcd1234]}"
```

使用公有子網路的範例：

```
aws ecs create-service --cluster managed-instances-cluster --service-name managed-instances-service --task-definition sample-managed-instances:1 --desired-count 1 --network-configuration "awsvpcConfiguration={subnets=[subnet-abcd1234],securityGroups=[sg-abcd1234],assignPublicIp=ENABLED}"
```

`create-service` 命令會在服務建立完成後傳回服務的描述。

步驟 7：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 managed-instances-cluster
```

輸出：

```
{
  "serviceArns": [
    "arn:aws:ecs:region:aws_account_id:service/managed-instances-cluster/managed-instances-service"
  ]
}
```

步驟 8：描述正在執行中的服務

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 managed-instances-cluster --services managed-
instances-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。

輸出將顯示服務正在使用 Amazon ECS 受管執行個體容量提供者：

```
{
  "services": [
    {
      "capacityProviderStrategy": [
        {
          "capacityProvider": "managed-instances-cp",
          "weight": 1,
          "base": 0
        }
      ],
      "networkConfiguration": {
        "awsvpcConfiguration": {
          "subnets": [
            "subnet-abcd1234"
          ],
          "securityGroups": [
            "sg-abcd1234"
          ],
          "assignPublicIp": "ENABLED"
        }
      },
      "enableECSTags": false,
      "loadBalancers": [],
      "deploymentController": {
        "type": "ECS"
      },
      "desiredCount": 1,
      "clusterArn": "arn:aws:ecs:region:aws_account_id:cluster/managed-instances-
cluster",
    }
  ]
}
```

```
        "serviceArn": "arn:aws:ecs:region:aws_account_id:service/managed-instances-  
service",  
        "serviceName": "managed-instances-service",  
        "taskDefinition": "arn:aws:ecs:region:aws_account_id:task-definition/  
sample-managed-instances:1"  
    }  
],  
    "failures": []  
}
```

步驟 9：測試

若要測試您的部署，您需要找到執行任務之受管執行個體的公有 IP 位址。

使用公有子網路部署的測試任務

首先，從服務中取得任務 ARN：

```
aws ecs list-tasks --cluster managed-instances-cluster --service managed-instances-  
service
```

輸出包含任務 ARN：

```
{  
  "taskArns": [  
    "arn:aws:ecs:region:aws_account_id:task/managed-instances-cluster/EXAMPLE"  
  ]  
}
```

描述任務以取得容器執行個體 ARN。將任務 ARN 用於 tasks 參數：

```
aws ecs describe-tasks --cluster managed-instances-cluster --tasks  
arn:aws:ecs:region:aws_account_id:task/managed-instances-cluster/EXAMPLE
```

輸出顯示任務正在 Amazon ECS 受管執行個體上執行，並包含容器執行個體 ARN：

```
{  
  "tasks": [  
    {  
      "launchType": "MANAGED_INSTANCES",  
      "capacityProviderName": "managed-instances-cp",  

```

```

        "containerInstanceArn": "arn:aws:ecs:region:aws_account_id:container-
instance/managed-instances-cluster/CONTAINER_INSTANCE_ID",
        "taskArn": "arn:aws:ecs:region:aws_account_id:task/managed-instances-
cluster/EXAMPLE",
        "taskDefinitionArn": "arn:aws:ecs:region:aws_account_id:task-definition/
sample-managed-instances:1"
    }
]
}

```

描述容器執行個體以取得 EC2 執行個體 ID：

```
aws ecs describe-container-instances --cluster managed-instances-cluster --container-
instances CONTAINER_INSTANCE_ID
```

輸出包含 EC2 執行個體 ID：

```

{
  "containerInstances": [
    {
      "ec2InstanceId": "i-1234567890abcdef0",
      "capacityProviderName": "managed-instances-cp",
      "containerInstanceArn": "arn:aws:ecs:region:aws_account_id:container-
instance/managed-instances-cluster/CONTAINER_INSTANCE_ID"
    }
  ]
}

```

描述 EC2 執行個體以取得公有 IP 位址：

```
aws ec2 describe-instances --instance-ids i-1234567890abcdef0
```

輸出中包含公有 IP 位址：

```

{
  "Reservations": [
    {
      "Instances": [
        {
          "PublicIpAddress": "198.51.100.2",
          "InstanceId": "i-1234567890abcdef0"
        }
      ]
    }
  ]
}

```

```
    ]  
  }  
]  
}
```

在 Web 瀏覽器中輸入公有 IP 位址，然後您應該會看到一個網頁，其中顯示正在 Amazon ECS 受管執行個體上執行的 Amazon ECS 範例應用程式。

使用私有子網路部署的測試任務

對於部署在私有子網路中的任務，您可以使用 Amazon ECS Exec 連線至容器，並從執行個體內測試部署。請遵循上述相同步驟取得任務 ARN，然後使用 ECS Exec：

```
aws ecs execute-command --cluster managed-instances-cluster \  
  --task arn:aws:ecs:region:aws_account_id:task/managed-instances-cluster/EXAMPLE \  
  --container managed-instances-app \  
  --interactive \  
  --command "/bin/sh"
```

在互動式 Shell 執行後，您可以測試該 Web 伺服器：

```
curl localhost
```

您應該會看到相當於 Amazon ECS 範例應用程式網頁的 HTML。

步驟 10：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 managed-instances-cluster --service managed-instances-  
service --force
```

等待刪除服務並停止所有任務，然後刪除容量提供者：

```
aws ecs delete-capacity-provider --capacity-provider managed-instances-cp
```

刪除叢集：

```
aws ecs delete-cluster --cluster managed-instances-cluster
```

 Note

容量提供者刪除後，受管執行個體會自動終止。您無需手動終止 EC2 執行個體。

了解如何為 Fargate 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。如需有關 Fargate 的詳細資訊，請參閱 [為 Amazon ECS 規劃 AWS Fargate 架構](#)。

將 AWS Fargate 用於 Amazon ECS 支援 AWS Fargate 的區域中的任務，以在 上開始使用 Amazon ECS。

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

先決條件

開始之前，請完成 [設定以使用 Amazon ECS](#) 中的步驟，並確認 IAM 使用者擁有 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 [建立角色以將許可委派給 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。如需有關如何建立安全群組的資訊，請參閱 Amazon EC2 User Guide 中的 [Create a security group for your Amazon EC2 instance](#)。

- 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 叢集範例”](#)。

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：

- 在金鑰欄位中，輸入金鑰名稱。
- 在值欄位中，輸入金鑰值。

[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. 在服務詳細資訊下，指定應用程式部署的方式。
 - a. 針對任務定義系列，選擇您在 中建立的任務定義 [步驟 2：建立任務定義](#)。
 - b. 針對 Service name (服務名稱)，輸入服務的名稱。
4. 在環境下，選擇啟動類型，然後選擇 FARGATE。
5. 在部署組態下，針對所需任務，輸入 1。
6. 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 [先決條件](#) 下方列出傳入規則。
7. 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 主控台中手動清除或刪除建立它們的 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 在刪除服務之前代表您縮減服務規模。

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 建立 Amazon ECS Windows 任務

將 AWS Fargate 用於 Amazon ECS 支援 AWS Fargate 的區域中的任務，以在 上開始使用 Amazon ECS。

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

先決條件

開始之前，請完成[設定以使用 Amazon ECS](#)。中的步驟，並確認 IAM 使用者擁有 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[建立角色以將許可委派給 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。如需有關如何建立安全群組的資訊，請參閱 Amazon EC2 User Guide 中的[Create a security group for your Amazon EC2 instance](#)。

- 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。

使用 建立叢集 AWS 管理主控台

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 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：

- 在金鑰欄位中，輸入金鑰名稱。
- 在值欄位中，輸入金鑰值。

[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。

向 註冊範例任務定義 AWS 管理主控台

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 [先決條件](#) 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 主控台中手動清除或刪除建立它們的 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 建立 Amazon ECS Windows 任務

在主控台中註冊任務定義、建立叢集與服務，以透過 EC2 來開始使用 Amazon ECS。

完成下列步驟，以使用 EC2 啟動類型來開始使用 Amazon ECS。

先決條件

開始之前，請完成[設定以使用 Amazon ECS](#)。中的步驟，並確認 IAM 使用者擁有 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[建立角色以將許可委派給 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。如需有關如何建立安全群組的資訊，請參閱 Amazon EC2 User Guide 中的[Create a security group for your Amazon EC2 instance](#)。

- 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 叢集範例”](#)。

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 執行個體在您的 AWS 資源中存在時會產生成本。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：

- 在金鑰欄位中，輸入金鑰名稱。
- 在值欄位中，輸入金鑰值。

[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

向 註冊範例任務定義 AWS 管理主控台

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](#)。

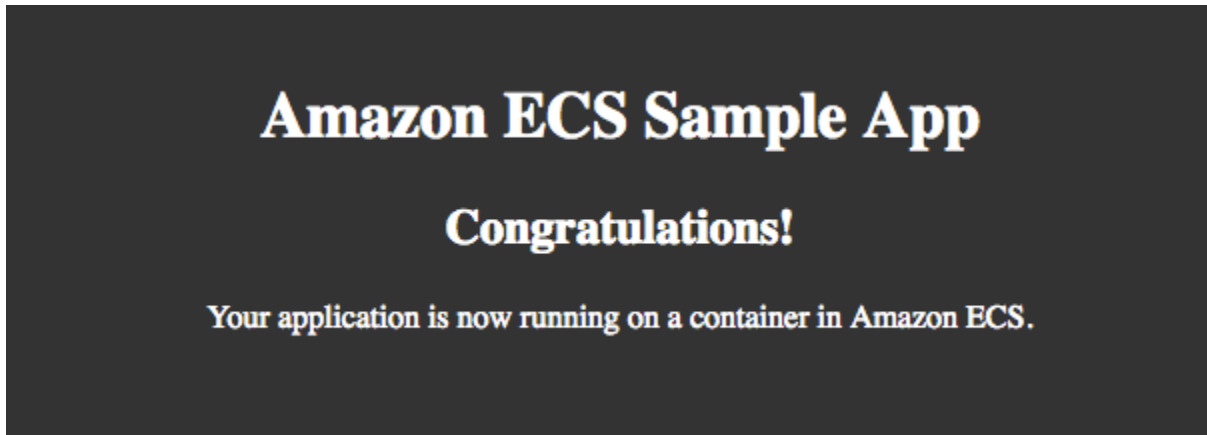
建立服務

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)。以下螢幕擷取畫面顯示了預期輸出。



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 主控台中手動清除或刪除建立它們的 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。

- 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。

使用 建立 Amazon ECS 資源 AWS CDK

AWS Cloud Development Kit (AWS CDK) 是 Infrastructure-as-Code (IAC) 架構，您可以使用您選擇的程式設計語言來定義 AWS 雲端基礎設施。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，您將它合成到 CloudFormation 範本，並將您的資源部署到 AWS 帳戶。請依照本主題中的步驟，使用 Amazon Elastic Container Service (Amazon ECS) 和 Fargate AWS CDK 上的部署容器化 Web 伺服器。

隨附於 CDK 的 AWS 建構程式庫提供模組，可讓您用來建立 AWS 服務所提供資源的模型。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。

開始此任務之前，請先設定您的 AWS CDK 開發環境，然後執行下列命令 AWS CDK 來安裝。如需如何設定開發 AWS CDK 環境的說明，請參閱 [AWS CDK - 先決條件入門](#)。

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

Note

這些指示假設您使用 AWS CDK v2。

主題

- [步驟 1：設定您的 AWS CDK 專案](#)
- [步驟 2：使用 AWS CDK 在 Fargate 上定義容器化 Web 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 應用程式建立目錄並初始化專案。

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，請啟用專案的虛擬環境，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 包含名為 `aws-cdk-lib` 的單一套件 AWS 服務 中所有 的穩定建構 `aws-cdk-lib`。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 AWS CDK 在 Fargate 上定義容器化 Web 伺服器

使用容器映像 [amazon-ecs-sample](#)。此映像包含在 Nginx 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，以類似下列其中一個範例。

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。AWS CDK 應用程式可以包含多個堆疊，而一個堆疊中的資源可以參考另一個堆疊中的資源。

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('public.ecr.aws/ecs-sample-image/  
amazon-ecs-sample:latest'),  
      },  
    },
```

```
        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 Amazon ECR 公共映像庫中取得的容器映像：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。會 AWS CDK 執行應用程式並從中合成 CloudFormation 範本，然後顯示範本。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
```

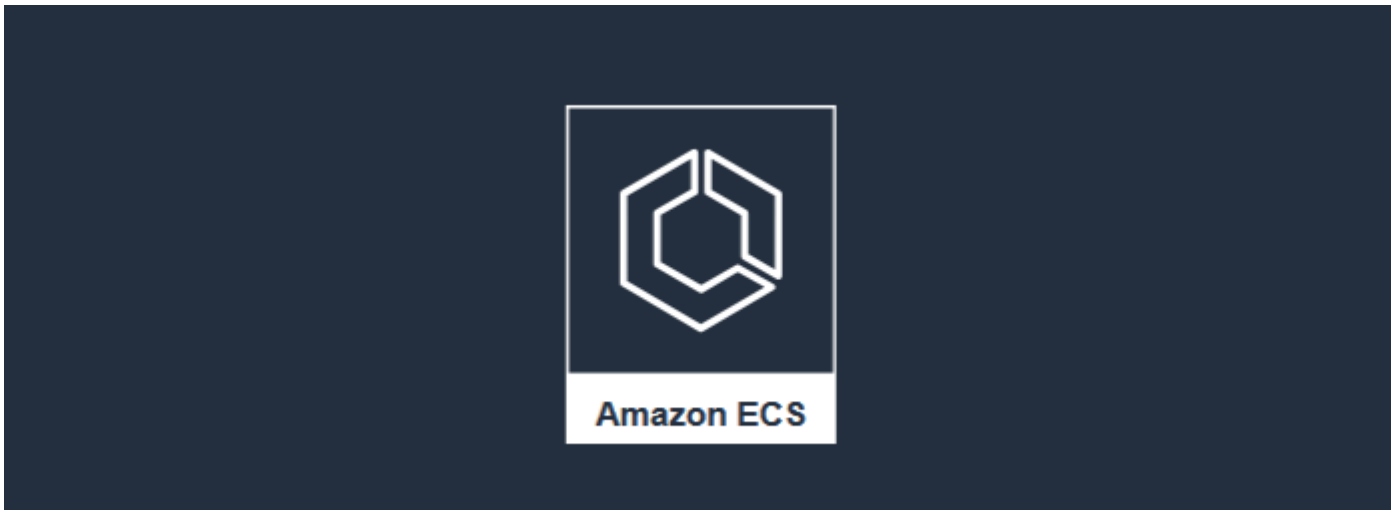
若要在 中部署服務 AWS 帳戶，請在應用程式的主目錄中執行 `cdk deploy` 命令。系統會要求您核准 AWS CDK 產生的 IAM 政策。

部署需要幾分鐘的時間，在此期間 AWS CDK 會建立數個資源。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.MyWebServerLoadBalancerDNSXXXXXXXX = Hello-MyWeb-ZZZZZZZZZZZZZZ-
ZZZZZZZZZZ.us-west-2.elb.amazonaws.com
HelloEcsStack.MyWebServerServiceURLYYYYYYYYY = http://Hello-MyWeb-ZZZZZZZZZZZZZZ-
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 基礎設施 AWS CDK，請參閱 [AWS CDK 開發人員指南](#)。

如需以您選擇的語言撰寫 AWS CDK 應用程式的資訊，請參閱以下內容：

TypeScript

[以 TypeScript 使用 AWS CDK](#)

JavaScript

[以 JavaScript 使用 AWS CDK](#)

Python

[在 Python AWS CDK 中使用](#)

Java

[在 Java AWS CDK 中使用](#)

C#

[在 C# AWS CDK 中使用](#)

Go

[在 Go AWS CDK 中使用](#)

如需本主題中使用的 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 AWS Copilot 命令列界面建立 Amazon ECS 資源

AWS Copilot CLI End-of-Support通知

AWS Copilot CLI 將於 2026 年 6 月 12 日 end-of-support。在此日期之後，工具將不再收到更新、安全修補程式或技術支援。

- 2026 年 6 月 12 日之後沒有新功能或增強功能

- 沒有安全性更新或錯誤修正
- 無技術支援
- 現有的部署將繼續運作，但不需要持續維護

需要協助嗎？如需詳細的遷移指引，請聯絡 AWS Support 或造訪我們的[部落格文章](#)。

AWS Copilot 命令列界面 (CLI) 命令可簡化從本機開發環境在 Amazon ECS 上建置、發佈和操作生產就緒的容器化應用程式。AWS Copilot 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：從使用基礎設施做為程式碼，到建立代表使用者佈建的 CI/CD 管道。使用 AWS Copilot CLI 作為日常開發和測試週期的一部分，作為的替代方案 AWS 管理主控台。

AWS Copilot 目前支援 Linux、macOS 和 Windows 系統。如需 AWS Copilot CLI 最新版本的詳細資訊，請參閱[版本](#)。

Note

AWS Copilot CLI 的原始碼可在 [GitHub](#) 上取得。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。透過在 [Gitter](#) 或 [GitHub](#) 上聯絡我們來報告 AWS Copilot 的問題，您可以在其中開啟問題、提供意見回饋和報告錯誤。

如需有關 Copilot 開發的資訊，請參閱 [What's the future of Copilot?](#)。

主題

- [安裝 AWS Copilot CLI](#)
- [使用 AWS Copilot CLI 部署範例 Amazon ECS 應用程式](#)

AWS Copilot [AWS 網站提供 Copilot](#) CLI 的其他文件

安裝 AWS Copilot CLI

AWS Copilot CLI End-of-Support通知

AWS Copilot CLI 將於 2026 年 6 月 12 日 end-of-support。在此日期之後，工具將不再收到更新、安全修補程式或技術支援。

- 2026 年 6 月 12 日之後沒有新功能或增強功能
- 沒有安全性更新或錯誤修正
- 無技術支援
- 現有的部署將繼續運作，但不需要持續維護

需要協助嗎？如需詳細的遷移指引，請聯絡 AWS Support 或造訪我們的[部落格文章](#)。

您可以使用 Homebrew 或依照下列步驟手動下載二進位檔來安裝 AWS Copilot CLI。

使用 Homebrew

以下命令用於使用 Homebrew 在 macOS 或 Linux 系統上安裝 AWS Copilot 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
```

下載二進位檔

除了 Homebrew，您可以在 macOS、Windows 或 Linux 系統上手動安裝 AWS Copilot CLI。使用下列命令讓作業系統下載二進位檔。macOS 與 Linux 作業系統範例也包含將執行許可套用至二進位檔的命令，並列出了說明功能表來驗證安裝是否能夠正常運作。

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 Copilot CLI 可執行檔使用 PGP 簽章進行密碼編譯簽署。PGP 簽章可用來驗證 AWS Copilot 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。

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 後，請關閉並重新打開管理員 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 TextEdit 開啟檔案。

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+qPpHqdArRgDZAtn2pXY5fEipHgar4CP8QgRnRM02f174lmavr4Vg
7K/KH8VH1q2uRw32/B94XLEgRbGTMDWfDKuxoPCttBQaMj3LGn6Pe+6xVWRkChQu
BoQAhjBQ+bEm0kNy0LjNgjN1nL3UMAG56t8E3LANIgGgEnpNsB1UwfwluPoGZoTx
N+6pHBjRkIL/1v/ETU4FXpYw2zvhwNahxeNRnoYj3uycHkeLiCrw4kj0+skizBg0
2K7oVX80c3j5+Zilhl/qDLXmUCb2az5cMM1m0oF8EKX5HaNuq1KfwJxqXE6NNic0
lFTTrT7QwD5fMNld3FanLgv/ZnIrsSaqJ0L6zRSq804LN10WBVBndExk2Kr+5kFxn
5lBPgfPgRj5hQ+KTHMa9Y8Z7yUc64BJiN6F9N17FJuSsfqbdkvRLsQRbcBG9qxX3
rJAEhieJzVMEUNl+EgeCkxj5xuSkNU7zw2c3hQZqEcrADLV+hvFJkt0z9Gm6xzbq
lTnWWCz4xrIWtuEBA2qE+MlDheVd78a3gIsEaSTfQq0osYXaQbvlnSW0oc1y/5Zb
zizHTJiHltUyls9WisP2s0emeHZicVMfw61EgPrJAIupgc7kyZvFt4YwfwARAQAB
tCRBbWF6b24gRUNTIDx1Y3Mtc2VjdXJpdHlAYW1hem9uLmNvbT6JAHwEEAECAAYF
AlrjL0YACgkQHivRXs0TaQrg1g/+JppwPqHn1VPmv7lessB8I5UqZeD6p6uVpHd7
Bs3pcPp8BV7BdRbs3sPLt5bV1+rkq0lw+0gZ4Q/ue/YbWt0At4qY00cEo0HgcnaX
lsB827QIfZIVtGWMhuh94xzm/SJkvngml6KB3YJNnWP61A9qJ37/VbVVLzvcmaZ
McWB4HUMNrh0JgBCo0gIpqCbpJEvUc02Bjn23eEJsS9kc70UAHyQkVnx4d9UzXF
40oISF6hmQKIBoLnRrAlj5Qvs3GhvHQ0ThYq0Grk/KMJJX2CSqt7tWJ8gk1n3H3Y
SReRXJRnv7DsDDBwFgT6r5Q2HW1TBuvaoZy5hF6maD09nHcNnvBjqADzeT8Tr/Qu
bBCLzkNSYqqkpgtwv7seoD2P4n1giRvDA0EfMZpVku+C252IaH1HZFEz+TvBVQM
Y80WwXmIJW+J6evjo3N1e019UHv71jvoF8zljBI4bsL2c+QTJm0v7nRqzDQgCWyp
Id/v2dUUVVtk1j9omuLBBwNJzQCB+72LcIzJhYmaP1HC4LcKQG+/f4lexuItenatK
lEJQhYtyVXcBlh6Yn/wzNg2NW0wb3vqY/F7m6u9ixAwgtIMgPCDE4aJ86zrrXYFz
N2HqkTSQh77Z8KPKmyGopsmN/reMuilPdINb249nA0dzoN+nj+ttFOYCIaLaFyjs
Z0r1QA0JAjkeEEwECACMFAlq1SasCGwMHCwkIBwMCAQYVCAIJCgsEFgIDAQIeAQIX
gAAKCRC86dmkLVF4T9iFEACEnkm1dNXsWUx34R3c0vamHrPxvfkyI1F1EUen8D1h
uX9xy6jCER0HWEp0rjGK4QDPgM93sWJ+s1UAKg214QRVzft0y9/DdR+twApA0fzy
uavIthGd6+03jAAo6udYDE+cZC3P7XBbDiYEWk4XAF9I1JjB8hTZUgvXBL046JhG
```

```
eM17+crgUyQeetki0QemLbsbXQ40Bd9V7zf7XJraFd8VrwNUwNb+9KFtgAsc9rk+
YIT/PEf+Y0PysgcxI4sTWghtyCulVnuGoskgDv4v73PALU0ieUrvvQvqWMrVhVx1
0X90J7cC1K0yh1EQQ1aFTgmQjmXexVTwIBm8LvysFK6YXM41Kj0r1z3+6xBIm/qe
bFyLUnf4Woiu0PlAaJhK9pRY+XENGNxdtN4D26Kd0F+PLkm3Tr3Hy3b10k34F1Gr
KVHUq1T2D7cvMnnKEELTUcKX+1mV3an16nmAg/my1JSUt6BNK2rJpY1s/kkSGSE
XQ4zuF2IGCpvBFhYAlt5Un5zwqkwQR3/n2kwAoDzonJcehDw/C/cGos5D0aIU7I
K2X2aTD3+pA7Mx3IME2hqmYqRt9X42yF1PIEVRneBRJ3HDezAgJrNh0GQWRQkhIx
gz6/cTR+ekr5TptVszS9few2GpI5bCgBKBisZIst89aw7mAKWut0Gcm4qm9/yK6
1bkCDQRatUmrARAAxNPvVwreJ2yAiFcUpdR1Vhsu0gnxvs1QgsIw3H7+Pacr9Hpe
8uftYzqdC82KeSKhpHq7c8gMTMucIINTH25x9BCc73E33EjCL9Lqov1TL7+QkgHe
T+JIhZwdD8Mx2K+LVVVu/aWkNrfMuNwyDUciSI4D5QHa8T+F8fgN40TpWYjirze1
5yoICMr9hVcbzDNv/ozKCxjx+XKgnFc3wrnDfJfntfDAT7ecwbUTL+viQKJ646s+
psiqXRYtVvYInEhLVrJ0aV6zHfoigE/Bils6/g7ru1Q6CEHqEw++APs5CcE8VzJu
WAGSVHZgun5Y9N4quR/M9Vm+IPMhTxrAg7r0vyRN9cAXfeSMf77I+XTifigNna8x
t/M0djXr1fjF4pThEi5u6WsuRdFwjY2azEv3vevodTi4HoJReH6dFRa6y8c+UDgl
2iHi0KIPIqQlBHEfQmHcDd2fix+AaJKMnPGNku9qCFEMbgSRJpXz6BfwnY1QuKE+I
R6jA0frUNT2jhiGG/F8RceXzohaaC/Cx7LUCUFwc0n7z32C9/Dtj7I1PM0acdZzz
bjJzRK0/ZDv+UN/c9dwAk1llzAyPMwGBkUaY68EBstnIliW34aWm6IiHhxioVPKSp
VJfyiXP00EXqujtHLAeChfjcns3I12YshT1dv2PafG53fp33ZdzeUgsBo+EAEQEA
AYkCHwQYAQIACQUCWrvJqWibDAAKCRC86dmkLVF4T+ZdD/9x/8APzgNjF3o3STrF
jvnV1ycyhWYGaEBJiu7wjsNWwzMF0v15tLjB7AqeVxZn+WKDD/mIOQ450ZvnYZuy
X7DR0JszaH9wrYTxZLVruAu+t6UL0y/XQ4L1GZ9QR6+r+7t1Mvbfy7B1HbvX/gYt
Rwe/uwdibI0CagEzyX+2D3kT01H05XThbXaNf8AN8zha91Jt2Q2UR2X5T6JcwtMz
FBvZn13LSmZyE0EQehS2iUurU4uW0pGppuqVnbi0jbCvCHKgDGrqZ0smKNAQng54
F365W3g8AfY48s8XQwzmcliowYX9bT8PZiEi0J4QmQh0aXkpqZyFefuWeOL2R94S
XKzr+gRh3BAULoqF+qK+IUMxTip9KTPNvYDpiC66yBiT6gFDji5Ca9pGpJXrC3xe
TXiKQ8DBWDhBPVPrruLIaenTtZE0sPc4I85yt5U9RoPTStc0r34s3w5yEaJagt6S
Gc5r9ysjkfH6+6rbi1ujxMgR0Sqtqr+RyB+V9A5/0gtNZc811K6u4Uo0Cde8jUuW
vqWKvjJB/Kz3u4zaeNu2ZyyHa0q0uH+TETcw+jsY9IhbEzqN5yQYGi4pVmDkY5vu
lXbJnbqPKpRXgM9BecV9AMbPgbDq/5LnHJJXg+G8YQ0gp4lR/hC1TEFdIp5wM8AK
CWsENyt2o1rjgMXiZOMF8A5oBLkCDQRatUuSARAAr77kj7j2QR2SZe0S1FBvV7oS
mFeSNnz9xZssqrsm6bTwSHM6YLDwc7Sdf2esDdyz0NETwqrVCg+FxgL8hmo9hS4c
rR6tmrP0m0mptr+xLLsKcaP7ogIXsyZnrEAEsvW8PnfayoiPCdc3cMCR/1TnHFGA
7EuR/XLBmi7Qg9tByVYQ5Yj5wB9V4B2yeCt3XtzPqeLKvaxl7PNe1aHGJQY/xo+m
V0bndxf9IY+4oFJ4b1D32WqvYxESo7vW6WBh7oqv3Zbm0yQrr8a6mDBpqLkvWwNI
3kpJR974tg5o5LfDu1BeeyHWPSGm4U/G4JB+JIG1ADy+RmoWEt4BqTCZ/knnoGvw
D5sTCxbKdmu0mhGyTssoG+300cGYHV7pWYP hazKHMPm201xKCjH1RfzRULzGKjD+
yMLT1I3AXFmLmZJXika01vE3/wgMqCXscbycbLjLD/bXIuFWo3rzoezeXjgi/DJx
jKBAyBTY05nMctH109oaFd9d0Hbs0UDkIMnsgGBE766Piro6MHo0T0rXl07Tp4pI
rwuS0sc6XzCzdImj0Wc6axS/HeUKRXWdXJwno5awTwXKRJMXGfhCvSvbcbc2Wx+L
IKvmb7EB4K3fmjFFE67yolmiw2qRcUBfygtH3eL5XZU28MiCpue8Y8GKJoBAUyvf
KeM1r08Jm3iRAc5a/D0AEQEAAyKEPgQYAQIACQUCWrvLkgIbAgIpCRC86dmkLVF4
T8FdIAQZAQIABGUCWrvLkgAKCRDePL1hra+LjtHYD/9MucxdFe6bX01dQR4tKhhQ
P0LRqy6z1BY9ILCLowNdGZdqorogUiUymgn3VhEhVtxT0oHcN7q0uM01PNsRn0eS
```

EYjf8Xrb1clzkD6xULwm0clTb9bBxnBc/4PFvHABZW3QzusaZniNgkuxt6BTf1oS
Of4inq71kjmGK+T1zQ6mUMQUg228NUQC+a84EPqYyAeY1sgvgB7hJBhYL0QAxhcW
6m20Rd8iEc6HyZJ3yCOCsKip/nRWAbf00vfHfRBp0+m0ZwnJM8cPRFj0qqzFpKH9
HpDmTrC4wKP1+TL52LyEqNh4yZitXmZNV7giSRIkk0eDSko+bFy6VbMzKUMkUJK3
D3eHFAMkujmbfJmSMTJOPGn5SB1HyjCZNx6bhIibQyEUB9gKCMUfaQXKwKpF6rj0
iQXAJxLR/shZ5Rk96Vxz0phU17T90m/PnUEEPwq8KsBhnMRgxa0RFidDP+n9fgtv
HLmr0Qx9zBCVXh0mdWYLrWvmzQFWzG7AoE55fkf8nAEPsalrCdtanUBHRXA00QxG
AHM0dJQQvBsmqMvuAdjKDWPfu5y0My5ddU+hiUzUyQLjL5Hhd5L0UDdewlZgIw1j
xrEAUzDKetnemM8GkHxDgg8koev5frmShJuce7vSjKpCNg3EIJsgqMOPFjJuLWtZ
vjHeDNbJy6uNL65ckJy6WhGjEADS2WAW1D6Tfekkc21SsIXk/LqEpLMR/0g50Uif
wcEN1rS9IJBWiy8Me1N9qr5KcKQLmfdFBNEyyceBhyV10MDyH0KC+7PofMtkGBq
13QieRHv5GJ8LB3fclqHV8pwTTo3Bc8z2g0TjmUYAN/ixETdReDoKavWJYSE9yoM
aaJu279ioVTrwpECse0XkiRyKToTjw0b73CGkBZZpJyqux/rmCV/fp4ALdSW8zbz
FJV0RaivhoWwzjpfQKhwcU9LABXi2UvVm14v0AfeI7oiJPSU1zM4fEny4oiIBX1R
zhFNih1UjIu82X16mTm3BwbIga/s1fnQRGzyhqUIMii+mWra23EwjChaxpvjjcUH
5i1Lc5Zq781aCYRygYQw+hu5nFk0H1R+Z50Ubxjd/auUfngIAX7kPMD3Lof4KldD
Q8ppQriUvxVo+4nPV6rpTy/PyqCLWDjkguHpJsEFsMkwajrAz0QNSAU5CJ0G2Zu4
yxvYlumHCE17nbFrm0vIiA75Sa8KnywTDsyZsu3Xc0cf3g+g1xWtpjJqy2bYX1qz
9uD0WtArWH0is6bq819RE6xr1RBVXS6uqqQIZFBGyq66b0dIq4D2JdsUvgEMaHbc
e7tBfeB1CMBdA64e9Rq7bFR7Tvt8gasCZY1Nr3lydh+dFHIEkH53HzQe6l88HEic
+0jVnLkCDQRA55wJARAaYLYa2Lx6gyoWoJN1a6740q3o8e9d4KggQ0fGMTcf1meq
ivuzgN+3DZHN+9ty2KxXMtn0mhHBERZdbNJyjMNT1gAgrhPNB4HtXBxum2wS57WK
DNmade914L7FWTPAWBG2Wn4480EHTqsC1ICXXWy9IICgc1AEyIq0Yq5mAdTEgRJS
Z8t4GpwtDL9gNQyFXaWQmDmkAsCygQMvhAlmu9x0IzQG5CxSnZFk7zcuL60k14Z3
Cmt49k4T/7ZU8goWi8tt+rU78/IL3J/ff9+1civ10wuUidgfPCSv0UW1JojsdCQA
L+RZJcoXq71f0Fj/eNje0SstCTDPfTCL+kThE6E5neDtbQHBYkEX1BRiTedsV4+M
ucgiTrdQFWKf89G72xdv8ut9AyyQ2BbEYU+JAYhUH8rYYui2dHKJIgjNvJscuUWb
+QEqJIRleJRhr0+/CHgMs4fZAKWF1VFhKBkcKmeJLn1f7EJJUW84ZhKXj0/AUPX
1CHsNjzirceJCJYox1cwsoq6jTE50GiNzcIxTn9xUc0UMKFeggNAFys1K+TDTm3
Bzo8H5ucjCUemUm9lhkGwqTZg0LRX5eqPX+JBoSa0bqhggCa5IPinKR6MgoFPHK
6sYKqroYwBGgZm6Js5chpNchvJMs/3WXN0EVg0J3z3vP0DMhxqWm+r+n9z1w8qsA
EQEAAYKEPqQYAQgACQUCWuecCQIBagIpCRC86dmkLVF4T8FdIAQZAQgABgUCWuec
CQAKCRBQ3szEcQ5hr+ykD/4t0LRHFHXuKucxgGaubUcVtsFrwBKma1cYjqaPms8u
6Sk0wfGRI32G/Gh0rp0Ts/M0kb0bq6VLTh8N5Yc/53ME18zQFw9Y5AmRoW4PZXER
ujs5s7p4oR7xHMiHmJCCBn1bvrR+34YPfgzTcgLi0EFHYT8UTxwnGmX0vNkMM7md
xD3CV5q6VAte8WKBo/220II3fcQ1c9r/oWX4kXXkb0v9hoGwKbDJ1tzqTPrp/xFt
yohqnvImpnlz+Q9zXmbrWYL9/g8VCmW/NN2gju2G3Lu/T1FUWIT4v/50PK6TdeNb
VKJ04+S8bTayqSG9CML1S57KSgCo5HUHQWeSNHI+fpe5oX6FALPT9JLDce80Zz1i
cZZ0MELP37m00Qun0AlmHm/hVzf0f311PtzbzcqWaE51tJvgUR/nZf06Ta305Ezhs
3V1EJNQ1IjF/6DH87SxvAoRIARCuZd0qxBCDK0avpFzUtbJd241RA3WJpkEiMqKv
RDVZkE4b6TW61f0o+LaVfK6E8oLpixegS4fiqC16mFrOdyRk+RJJfIUyz0WTDVmt
g0U1C01ezokMSqkJ7724pyjr2xf/r9/sC6a0JwB/1KgZkJfC6NqL7T1xVA31dUga
LE0vEJTTE4gl+tYtfsCDvALCtqL0jduSkUo+RXcBItmXhA+tShW0pbS2Rtx/ixua
KohVD/0R4QxiSwQmICNtm9mw9ydI11yjYXX5a9x4wMJracNY/LBybJPFnZnT4dYR


```

z4XjqysDwvvYZByaWoIe3QxjX84V6M1I2IdAT/xImu8gbaCI8tmyfpIrLnPKiR9D
VFYfGBXuAX7+HgPPSFtrHQ0NCALxxz1bNpS+zxt9r0MiLgcLyspWxSdmoYGZ6nQP
R05Nm/ZVS+u2imPCRzNUZEMa+d1E6kHx0rS0dPiuJ407NtPeYDKkoQtNagspsDvh
cK7CSqAiKMq06UBTxqlTSRkm62e0Ctcs3p30eHu5GRZF1uzTET0ZxYkaPgdrQknx
ozjP5mC7X+451cCfmcVt94TFNL5HwEUVJpm0gmzILCI8yoDTWzloo+i+fPFsXX4f
kynhE83mSEcr5VHFYrTY3mQXGmNJ3bCLuc/jq7ysGq69xiKmTlUeXFm+aojcR05i
zyShIRJZ0GZfuzDYFDbMV9amA/YQGygLw//zP5ju5SW26dNx1f3MdFQE5JJ86rn9
MgZ4gcpazHEVUusbZsgkLizRp9imUiH8ymLqAXnfRG1U/LpNsefnvDFTtEIRcp0Hc
bhayG0bk51Bd4mio0XnIsKy4j63nJXA27x5EVVHQ1sYRN8Ny4Fdr2tMamj20+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 Copilot 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 上執行下列命令，將 AWS Copilot 目錄新增至路徑。

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

使用 AWS Copilot CLI 部署範例 Amazon ECS 應用程式

⚠ AWS Copilot CLI End-of-Support通知

AWS Copilot CLI 將於 2026 年 6 月 12 日 end-of-support。在此日期之後，工具將不再收到更新、安全修補程式或技術支援。

- 2026 年 6 月 12 日之後沒有新功能或增強功能
- 沒有安全性更新或錯誤修正
- 無技術支援
- 現有的部署將繼續運作，但不需要持續維護

需要協助嗎？如需詳細的遷移指引，請聯絡 AWS Support 或造訪我們的[部落格文章](#)。

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，以及清除資源。

先決條件

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 Copilot 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 \
--tag latest \
--deploy
```

2. 部署完成後，AWS Copilot CLI 會傳回 URL，供您用來驗證部署。您也可以使用下列命令來驗證應用程式的狀態。

- 列出所有 AWS Copilot 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，請執行下列命令來清理相關聯的資源及避免未使用的資源產生費用。

```
copilot app delete
```

搭配 使用 Amazon ECS AWS CloudFormation

Amazon ECS 已與 整合 AWS CloudFormation，這項服務可讓您使用您定義的範本來建模和設定 AWS 資源。CloudFormation 會使用屬於 JSON YAML 或格式化文字檔案的 範本。範本就像您要建立之 AWS 資源的藍圖。您建立並提交範本後，CloudFormation 會建立一個堆疊。您可以透過該堆疊來管理在範本中定義的資源。當您想要建立、更新或刪除資源時，可以建立、更新或刪除基於該資源建立的堆疊。當涉及到更新堆疊時，您需要先建立一個變更集。變更集會在您進行變更之前向您展示變更帶來的影響。例如，這可防止您透過變更資料庫名稱而意外刪除資料庫。如需範本、堆疊和變更集的詳細資訊，請參閱 AWS CloudFormation 《使用者指南》中的 [如何 CloudFormation 運作](#)。

透過使用 CloudFormation，您可以減少建立和管理資源和基礎設施的時間。您可以建立範本來描述您想要的所有 AWS 資源，例如 Amazon ECS 叢集、任務定義、服務。然後，CloudFormation 負責為您佈建和設定這些資源。

CloudFormation 也可讓您重複使用範本，以一致且可重複的方式設定 Amazon ECS 資源。您描述一次資源，然後在多個 AWS 帳戶 和 之間再次佈建相同的資源 AWS 區域。

CloudFormation 範本可與 AWS 管理主控台 或 搭配使用 AWS Command Line Interface，以建立 資源。

若要進一步了解 CloudFormation，請參閱下列資源：

- [AWS CloudFormation](#)
- [AWS CloudFormation 使用者指南](#)
- [AWS CloudFormation 命令列界面使用者指南](#)

主題

- [使用 CloudFormation 主控台建立 Amazon ECS 資源](#)
- [使用的 AWS CLI 命令建立 Amazon ECS 資源 CloudFormation](#)
- [CloudFormation Amazon ECS 的範例範本](#)

使用 CloudFormation 主控台建立 Amazon ECS 資源

搭配 使用 Amazon ECS 的一種方法是 CloudFormation 透過 AWS 管理主控台。使用此方法，您可以建立任務定義、叢集與服務等 Amazon ECS 元件的 CloudFormation 堆疊，並直接在主控台中部署它們。下列教學課程說明如何使用 CloudFormation 主控台來透過範本建立 Amazon ECS 資源。

先決條件

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 IAM 使用者擁有 [AmazonECS_FullAccess](#) IAM 政策範例中指定的所需許可。

步驟 1：建立堆疊範本

使用下列步驟為 Amazon ECS 服務和其他相關資源建立 CloudFormation 堆疊範本。

1. 使用所選的文字編輯器建立一個名為 `ecs-tutorial-template.yaml` 的檔案。
2. 在 `ecs-tutorial-template.yaml` 檔案中，貼上下列範本並儲存變更。

```
AWSTemplateFormatVersion: '2010-09-09'
Description: '[AWS Docs] ECS: load-balanced-web-application'

Parameters:
  VpcCidr:
    Type: String
    Default: '10.0.0.0/16'
    Description: CIDR block for the VPC
  ContainerImage:
    Type: String
    Default: 'public.ecr.aws/ecs-sample-image/amazon-ecs-sample:latest'
    Description: Container image to use in task definition

  PublicSubnet1Cidr:
    Type: String
    Default: '10.0.1.0/24'
    Description: CIDR block for public subnet 1

  PublicSubnet2Cidr:
    Type: String
    Default: '10.0.2.0/24'
    Description: CIDR block for public subnet 2

  PrivateSubnet1Cidr:
    Type: String
    Default: '10.0.3.0/24'
    Description: CIDR block for private subnet 1
```

```
PrivateSubnet2Cidr:
  Type: String
  Default: '10.0.4.0/24'
  Description: CIDR block for private subnet 2

ServiceName:
  Type: String
  Default: 'tutorial-app'
  Description: Name of the ECS service

ContainerPort:
  Type: Number
  Default: 80
  Description: Port on which the container listens

DesiredCount:
  Type: Number
  Default: 2
  Description: Desired number of tasks

MinCapacity:
  Type: Number
  Default: 1
  Description: Minimum number of tasks for auto scaling

MaxCapacity:
  Type: Number
  Default: 10
  Description: Maximum number of tasks for auto scaling

Resources:
  # VPC and Networking
  VPC:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: !Ref VpcCidr
      EnableDnsHostnames: true
      EnableDnsSupport: true
      Tags:
        - Key: Name
          Value: !Sub '${AWS::StackName}-vpc'

  # Internet Gateway
```



```
InternetGateway:
  Type: AWS::EC2::InternetGateway
  Properties:
    Tags:
      - Key: Name
        Value: !Sub '${AWS::StackName}-igw'

InternetGatewayAttachment:
  Type: AWS::EC2::VPCGatewayAttachment
  Properties:
    InternetGatewayId: !Ref InternetGateway
    VpcId: !Ref VPC

# Public Subnets for ALB
PublicSubnet1:
  Type: AWS::EC2::Subnet
  Properties:
    VpcId: !Ref VPC
    AvailabilityZone: !Select [0, !GetAZs '']
    CidrBlock: !Ref PublicSubnet1Cidr
    MapPublicIpOnLaunch: true
    Tags:
      - Key: Name
        Value: !Sub '${AWS::StackName}-public-subnet-1'

PublicSubnet2:
  Type: AWS::EC2::Subnet
  Properties:
    VpcId: !Ref VPC
    AvailabilityZone: !Select [1, !GetAZs '']
    CidrBlock: !Ref PublicSubnet2Cidr
    MapPublicIpOnLaunch: true
    Tags:
      - Key: Name
        Value: !Sub '${AWS::StackName}-public-subnet-2'

# Private Subnets for ECS Tasks
PrivateSubnet1:
  Type: AWS::EC2::Subnet
  Properties:
    VpcId: !Ref VPC
    AvailabilityZone: !Select [0, !GetAZs '']
    CidrBlock: !Ref PrivateSubnet1Cidr
    Tags:
```

```
- Key: Name
  Value: !Sub '${AWS::StackName}-private-subnet-1'

PrivateSubnet2:
  Type: AWS::EC2::Subnet
  Properties:
    VpcId: !Ref VPC
    AvailabilityZone: !Select [1, !GetAZs '']
    CidrBlock: !Ref PrivateSubnet2Cidr
    Tags:
      - Key: Name
        Value: !Sub '${AWS::StackName}-private-subnet-2'

# NAT Gateways for private subnet internet access
NatGateway1EIP:
  Type: AWS::EC2::EIP
  DependsOn: InternetGatewayAttachment
  Properties:
    Domain: vpc
    Tags:
      - Key: Name
        Value: !Sub '${AWS::StackName}-nat-eip-1'

NatGateway2EIP:
  Type: AWS::EC2::EIP
  DependsOn: InternetGatewayAttachment
  Properties:
    Domain: vpc
    Tags:
      - Key: Name
        Value: !Sub '${AWS::StackName}-nat-eip-2'

NatGateway1:
  Type: AWS::EC2::NatGateway
  Properties:
    AllocationId: !GetAtt NatGateway1EIP.AllocationId
    SubnetId: !Ref PublicSubnet1
    Tags:
      - Key: Name
        Value: !Sub '${AWS::StackName}-nat-1'

NatGateway2:
  Type: AWS::EC2::NatGateway
  Properties:
```

```
AllocationId: !GetAtt NatGateway2EIP.AllocationId
SubnetId: !Ref PublicSubnet2
Tags:
  - Key: Name
    Value: !Sub '${AWS::StackName}-nat-2'

# Route Tables
PublicRouteTable:
  Type: AWS::EC2::RouteTable
  Properties:
    VpcId: !Ref VPC
    Tags:
      - Key: Name
        Value: !Sub '${AWS::StackName}-public-routes'

DefaultPublicRoute:
  Type: AWS::EC2::Route
  DependsOn: InternetGatewayAttachment
  Properties:
    RouteTableId: !Ref PublicRouteTable
    DestinationCidrBlock: 0.0.0.0/0
    GatewayId: !Ref InternetGateway

PublicSubnet1RouteTableAssociation:
  Type: AWS::EC2::SubnetRouteTableAssociation
  Properties:
    RouteTableId: !Ref PublicRouteTable
    SubnetId: !Ref PublicSubnet1

PublicSubnet2RouteTableAssociation:
  Type: AWS::EC2::SubnetRouteTableAssociation
  Properties:
    RouteTableId: !Ref PublicRouteTable
    SubnetId: !Ref PublicSubnet2

PrivateRouteTable1:
  Type: AWS::EC2::RouteTable
  Properties:
    VpcId: !Ref VPC
    Tags:
      - Key: Name
        Value: !Sub '${AWS::StackName}-private-routes-1'

DefaultPrivateRoute1:
```

```
Type: AWS::EC2::Route
Properties:
  RouteTableId: !Ref PrivateRouteTable1
  DestinationCidrBlock: 0.0.0.0/0
  NatGatewayId: !Ref NatGateway1

PrivateSubnet1RouteTableAssociation:
  Type: AWS::EC2::SubnetRouteTableAssociation
  Properties:
    RouteTableId: !Ref PrivateRouteTable1
    SubnetId: !Ref PrivateSubnet1

PrivateRouteTable2:
  Type: AWS::EC2::RouteTable
  Properties:
    VpcId: !Ref VPC
    Tags:
      - Key: Name
        Value: !Sub '${AWS::StackName}-private-routes-2'

DefaultPrivateRoute2:
  Type: AWS::EC2::Route
  Properties:
    RouteTableId: !Ref PrivateRouteTable2
    DestinationCidrBlock: 0.0.0.0/0
    NatGatewayId: !Ref NatGateway2

PrivateSubnet2RouteTableAssociation:
  Type: AWS::EC2::SubnetRouteTableAssociation
  Properties:
    RouteTableId: !Ref PrivateRouteTable2
    SubnetId: !Ref PrivateSubnet2

# Security Groups
ALBSecurityGroup:
  Type: AWS::EC2::SecurityGroup
  Properties:
    GroupName: !Sub '${AWS::StackName}-alb-sg'
    GroupDescription: Security group for Application Load Balancer
    VpcId: !Ref VPC
    SecurityGroupIngress:
      - IpProtocol: tcp
        FromPort: 80
        ToPort: 80
```

```

        CidrIp: 0.0.0.0/0
        Description: Allow HTTP traffic from internet
    SecurityGroupEgress:
      - IpProtocol: -1
        CidrIp: 0.0.0.0/0
        Description: Allow all outbound traffic
    Tags:
      - Key: Name
        Value: !Sub '${AWS::StackName}-alb-sg'

ECSSecurityGroup:
  Type: AWS::EC2::SecurityGroup
  Properties:
    GroupName: !Sub '${AWS::StackName}-ecs-sg'
    GroupDescription: Security group for ECS tasks
    VpcId: !Ref VPC
    SecurityGroupIngress:
      - IpProtocol: tcp
        FromPort: !Ref ContainerPort
        ToPort: !Ref ContainerPort
        SourceSecurityGroupId: !Ref ALBSecurityGroup
        Description: Allow traffic from ALB
    SecurityGroupEgress:
      - IpProtocol: -1
        CidrIp: 0.0.0.0/0
        Description: Allow all outbound traffic
    Tags:
      - Key: Name
        Value: !Sub '${AWS::StackName}-ecs-sg'

# Application Load Balancer
ApplicationLoadBalancer:
  Type: AWS::ElasticLoadBalancingV2::LoadBalancer
  Properties:
    Name: !Sub '${AWS::StackName}-alb'
    Scheme: internet-facing
    Type: application
    Subnets:
      - !Ref PublicSubnet1
      - !Ref PublicSubnet2
    SecurityGroups:
      - !Ref ALBSecurityGroup
    Tags:
      - Key: Name

```

```
    Value: !Sub '${AWS::StackName}-alb'

ALBTargetGroup:
  Type: AWS::ElasticLoadBalancingV2::TargetGroup
  Properties:
    Name: !Sub '${AWS::StackName}-tg'
    Port: !Ref ContainerPort
    Protocol: HTTP
    VpcId: !Ref VPC
    TargetType: ip
    HealthCheckIntervalSeconds: 30
    HealthCheckPath: /
    HealthCheckProtocol: HTTP
    HealthCheckTimeoutSeconds: 5
    HealthyThresholdCount: 2
    UnhealthyThresholdCount: 5
    Tags:
      - Key: Name
        Value: !Sub '${AWS::StackName}-tg'

ALBListener:
  Type: AWS::ElasticLoadBalancingV2::Listener
  Properties:
    DefaultActions:
      - Type: forward
        TargetGroupArn: !Ref ALBTargetGroup
    LoadBalancerArn: !Ref ApplicationLoadBalancer
    Port: 80
    Protocol: HTTP

# ECS Cluster
ECSCluster:
  Type: AWS::ECS::Cluster
  Properties:
    ClusterName: !Sub '${AWS::StackName}-cluster'
    CapacityProviders:
      - FARGATE
      - FARGATE_SPOT
    DefaultCapacityProviderStrategy:
      - CapacityProvider: FARGATE
        Weight: 1
      - CapacityProvider: FARGATE_SPOT
        Weight: 4
    ClusterSettings:
```

```
- Name: containerInsights
  Value: enabled
Tags:
- Key: Name
  Value: !Sub '${AWS::StackName}-cluster'

# IAM Roles
ECSTaskExecutionRole:
  Type: AWS::IAM::Role
  Properties:
    RoleName: !Sub '${AWS::StackName}-task-execution-role'
    AssumeRolePolicyDocument:
      Version: '2012-10-17'
      Statement:
        - Effect: Allow
          Principal:
            Service: ecs-tasks.amazonaws.com
          Action: sts:AssumeRole
    ManagedPolicyArns:
      - arn:aws:iam::aws:policy/service-role/AmazonECSTaskExecutionRolePolicy
    Tags:
      - Key: Name
        Value: !Sub '${AWS::StackName}-task-execution-role'

ECSTaskRole:
  Type: AWS::IAM::Role
  Properties:
    RoleName: !Sub '${AWS::StackName}-task-role'
    AssumeRolePolicyDocument:
      Version: '2012-10-17'
      Statement:
        - Effect: Allow
          Principal:
            Service: ecs-tasks.amazonaws.com
          Action: sts:AssumeRole
    Tags:
      - Key: Name
        Value: !Sub '${AWS::StackName}-task-role'

# CloudWatch Log Group
LogGroup:
  Type: AWS::Logs::LogGroup
  Properties:
    LogGroupName: !Sub '/ecs/${AWS::StackName}'
```

```
RetentionInDays: 7

# ECS Task Definition
TaskDefinition:
  Type: AWS::ECS::TaskDefinition
  Properties:
    Family: !Sub '${AWS::StackName}-task'
    Cpu: '256'
    Memory: '512'
    NetworkMode: awsvpc
    RequiresCompatibilities:
      - FARGATE
    ExecutionRoleArn: !GetAtt ECSTaskExecutionRole.Arn
    TaskRoleArn: !GetAtt ECSTaskRole.Arn
    ContainerDefinitions:
      - Name: !Ref ServiceName
        Image: !Ref ContainerImage
        PortMappings:
          - ContainerPort: !Ref ContainerPort
            Protocol: tcp
        Essential: true
        LogConfiguration:
          LogDriver: awslogs
          Options:
            awslogs-group: !Ref LogGroup
            awslogs-region: !Ref AWS::Region
            awslogs-stream-prefix: ecs
        HealthCheck:
          Command:
            - CMD-SHELL
            - curl -f http://localhost/ || exit 1
          Interval: 30
          Timeout: 5
          Retries: 3
          StartPeriod: 60
    Tags:
      - Key: Name
        Value: !Sub '${AWS::StackName}-task'

# ECS Service
ECSService:
  Type: AWS::ECS::Service
  DependsOn: ALBListener
  Properties:
```



```

    ServiceName: !Sub '${AWS::StackName}-service'
    Cluster: !Ref ECSCluster
    TaskDefinition: !Ref TaskDefinition
    DesiredCount: !Ref DesiredCount
    LaunchType: FARGATE
    PlatformVersion: LATEST
    NetworkConfiguration:
      AwsVpcConfiguration:
        AssignPublicIp: DISABLED
        SecurityGroups:
          - !Ref ECSSecurityGroup
        Subnets:
          - !Ref PrivateSubnet1
          - !Ref PrivateSubnet2
    LoadBalancers:
      - ContainerName: !Ref ServiceName
        ContainerPort: !Ref ContainerPort
        TargetGroupArn: !Ref ALBTargetGroup
    DeploymentConfiguration:
      MaximumPercent: 200
      MinimumHealthyPercent: 50
      DeploymentCircuitBreaker:
        Enable: true
        Rollback: true
    EnableExecuteCommand: true # For debugging
    Tags:
      - Key: Name
        Value: !Sub '${AWS::StackName}-service'

# Auto Scaling Target
ServiceScalingTarget:
  Type: AWS::ApplicationAutoScaling::ScalableTarget
  Properties:
    MaxCapacity: !Ref MaxCapacity
    MinCapacity: !Ref MinCapacity
    ResourceId: !Sub 'service/${ECSCluster}/${ECSService.Name}'
    RoleARN: !Sub 'arn:aws:iam:${AWS::AccountId}:role/
aws-service-role/ecs.application-autoscaling.amazonaws.com/
AWSServiceRoleForApplicationAutoScaling_ECSService'
    ScalableDimension: ecs:service:DesiredCount
    ServiceNamespace: ecs

# Auto Scaling Policy - CPU Utilization
ServiceScalingPolicy:

```

```
Type: AWS::ApplicationAutoScaling::ScalingPolicy
Properties:
  PolicyName: !Sub '${AWS::StackName}-cpu-scaling-policy'
  PolicyType: TargetTrackingScaling
  ScalingTargetId: !Ref ServiceScalingTarget
  TargetTrackingScalingPolicyConfiguration:
    PredefinedMetricSpecification:
      PredefinedMetricType: ECSServiceAverageCPUUtilization
    TargetValue: 70.0
    ScaleOutCooldown: 300
    ScaleInCooldown: 300
```

Outputs:**VPCId:**

```
Description: VPC ID
Value: !Ref VPC
Export:
  Name: !Sub '${AWS::StackName}-VPC-ID'
```

LoadBalancerURL:

```
Description: URL of the Application Load Balancer
Value: !Sub 'http://${ApplicationLoadBalancer.DNSName}'
Export:
  Name: !Sub '${AWS::StackName}-ALB-URL'
```

ECSClusterName:

```
Description: Name of the ECS Cluster
Value: !Ref ECSCluster
Export:
  Name: !Sub '${AWS::StackName}-ECS-Cluster'
```

ECSServiceName:

```
Description: Name of the ECS Service
Value: !GetAtt ECSService.Name
Export:
  Name: !Sub '${AWS::StackName}-ECS-Service'
```

PrivateSubnet1:

```
Description: Private Subnet 1 ID
Value: !Ref PrivateSubnet1
Export:
  Name: !Sub '${AWS::StackName}-Private-Subnet-1'
```

PrivateSubnet2:

```

Description: Private Subnet 2 ID
Value: !Ref PrivateSubnet2
Export:
  Name: !Sub '${AWS::StackName}-Private-Subnet-2'

```

本教學課程中使用的範本會建立 Amazon ECS 服務，其中包含兩項在 Fargate 上執行的任務。每項任務都會執行範例 Amazon ECS 應用程式。範本也會建立一個 Application Load Balancer (用於分配應用程式流量)，以及一個根據 CPU 使用率擴展應用程式的應用程式自動擴展政策。該範本還會建立部署應用程式所需的聯網資源、容器日誌的記錄資源，以及 Amazon ECS 任務執行 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](#)」。如需有關自動擴展的詳細資訊，請參閱[自動擴展您的 Amazon ECS 服務](#)。

步驟 2：建立 Amazon ECS 資源的堆疊

為範本建立檔案之後，您可以依照下列步驟，使用 CloudFormation 主控台建立具有範本的堆疊。

如需有關如何使用 CloudFormation 主控台建立堆疊的資訊，請參閱AWS CloudFormation 《使用者指南》中的[在 CloudFormation 主控台上建立堆疊](#)，並使用下表來決定要指定哪些選項。

選項	Value
先決條件 – 準備範本	選擇現有範本
指定範本	上傳範本檔案
選擇檔案	ecs-tutorial-template.yaml
Stack name (堆疊名稱)	ecs-tutorial-stack
Parameters	將所有參數值保留為預設值。
功能	選擇 我確認此範本可能會建立 IAM 資源，以確認 CloudFormation 建立 IAM 資源。

步驟 3：驗證

遵循下列步驟，使用提供的範本來驗證 Amazon ECS 資源是否已建立。

如需有關如何檢視堆疊資訊與資源的資訊，請參閱 AWS CloudFormation User Guide 中的 [Viewing stack information from the CloudFormation console](#)，並使用下表來決定要驗證的內容。

堆疊詳細資訊欄位	要尋找的內容
堆疊資訊	狀態為 CREATE_COMPLETE。
Resources	已建立的資源清單，其中包含指向服務主控台的連結。選擇 ECSCluster、ECSService、TaskDefinition 的連結，以在 Amazon ECS 主控台中檢視有關已建立服務、叢集與任務定義的更多詳細資訊。
輸出	LoadBalancerURL。將 URL 貼到 Web 瀏覽器中，以檢視顯示範例 Amazon ECS 應用程式的網頁。

步驟 4：清除資源

若要清理資源並避免產生更多成本，請遵循 CloudFormation user guide 中 [Delete a stack from the CloudFormation console](#) 一節的步驟。

使用的 AWS CLI 命令建立 Amazon ECS 資源 CloudFormation

搭配使用 Amazon ECS 的另一種方法是 CloudFormation 透過 AWS CLI。您可以使用命令為任務定義、叢集和服務等 Amazon ECS 元件建立 CloudFormation 堆疊，並進行部署。下列教學課程說明如何使用範本 AWS CLI 來建立 Amazon ECS 資源 CloudFormation。

先決條件

- 已完成「[設定以使用 Amazon ECS。](#)」中的步驟。
- 您的 IAM 使用者擁有 [AmazonECS_FullAccess](#) IAM 政策範例中指定的所需許可。

步驟 1：建立堆疊

若要使用 AWS CLI 儲存在名為 的檔案中的 建立堆疊`ecs-tutorial-template.yaml`，請執行下列命令。

```
cat << 'EOF' > ecs-tutorial-template.yaml
AWSTemplateFormatVersion: '2010-09-09'
Description: '[AWS Docs] ECS: load-balanced-web-application'
Parameters:
  VpcCidr:
    Type: String
    Default: '10.0.0.0/16'
    Description: CIDR block for the VPC
  ContainerImage:
    Type: String
    Default: 'public.ecr.aws/ecs-sample-image/amazon-ecs-sample:latest'
    Description: Container image to use in task definition

  PublicSubnet1Cidr:
    Type: String
    Default: '10.0.1.0/24'
    Description: CIDR block for public subnet 1

  PublicSubnet2Cidr:
    Type: String
    Default: '10.0.2.0/24'
    Description: CIDR block for public subnet 2

  PrivateSubnet1Cidr:
    Type: String
    Default: '10.0.3.0/24'
    Description: CIDR block for private subnet 1

  PrivateSubnet2Cidr:
    Type: String
    Default: '10.0.4.0/24'
    Description: CIDR block for private subnet 2

  ServiceName:
    Type: String
    Default: 'tutorial-app'
    Description: Name of the ECS service
```

```
ContainerPort:
  Type: Number
  Default: 80
  Description: Port on which the container listens

DesiredCount:
  Type: Number
  Default: 2
  Description: Desired number of tasks

MinCapacity:
  Type: Number
  Default: 1
  Description: Minimum number of tasks for auto scaling

MaxCapacity:
  Type: Number
  Default: 10
  Description: Maximum number of tasks for auto scaling

Resources:
  # VPC and Networking
  VPC:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: !Ref VpcCidr
      EnableDnsHostnames: true
      EnableDnsSupport: true
      Tags:
        - Key: Name
          Value: !Sub '${AWS::StackName}-vpc'

  # Internet Gateway
  InternetGateway:
    Type: AWS::EC2::InternetGateway
    Properties:
      Tags:
        - Key: Name
          Value: !Sub '${AWS::StackName}-igw'

  InternetGatewayAttachment:
    Type: AWS::EC2::VPCGatewayAttachment
    Properties:
      InternetGatewayId: !Ref InternetGateway
```

```
VpcId: !Ref VPC

# Public Subnets for ALB
PublicSubnet1:
  Type: AWS::EC2::Subnet
  Properties:
    VpcId: !Ref VPC
    AvailabilityZone: !Select [0, !GetAZs '']
    CidrBlock: !Ref PublicSubnet1Cidr
    MapPublicIpOnLaunch: true
    Tags:
      - Key: Name
        Value: !Sub '${AWS::StackName}-public-subnet-1'

PublicSubnet2:
  Type: AWS::EC2::Subnet
  Properties:
    VpcId: !Ref VPC
    AvailabilityZone: !Select [1, !GetAZs '']
    CidrBlock: !Ref PublicSubnet2Cidr
    MapPublicIpOnLaunch: true
    Tags:
      - Key: Name
        Value: !Sub '${AWS::StackName}-public-subnet-2'

# Private Subnets for ECS Tasks
PrivateSubnet1:
  Type: AWS::EC2::Subnet
  Properties:
    VpcId: !Ref VPC
    AvailabilityZone: !Select [0, !GetAZs '']
    CidrBlock: !Ref PrivateSubnet1Cidr
    Tags:
      - Key: Name
        Value: !Sub '${AWS::StackName}-private-subnet-1'

PrivateSubnet2:
  Type: AWS::EC2::Subnet
  Properties:
    VpcId: !Ref VPC
    AvailabilityZone: !Select [1, !GetAZs '']
    CidrBlock: !Ref PrivateSubnet2Cidr
    Tags:
      - Key: Name
```

```
    Value: !Sub '${AWS::StackName}-private-subnet-2'

# NAT Gateways for private subnet internet access
NatGateway1EIP:
  Type: AWS::EC2::EIP
  DependsOn: InternetGatewayAttachment
  Properties:
    Domain: vpc
    Tags:
      - Key: Name
        Value: !Sub '${AWS::StackName}-nat-eip-1'

NatGateway2EIP:
  Type: AWS::EC2::EIP
  DependsOn: InternetGatewayAttachment
  Properties:
    Domain: vpc
    Tags:
      - Key: Name
        Value: !Sub '${AWS::StackName}-nat-eip-2'

NatGateway1:
  Type: AWS::EC2::NatGateway
  Properties:
    AllocationId: !GetAtt NatGateway1EIP.AllocationId
    SubnetId: !Ref PublicSubnet1
    Tags:
      - Key: Name
        Value: !Sub '${AWS::StackName}-nat-1'

NatGateway2:
  Type: AWS::EC2::NatGateway
  Properties:
    AllocationId: !GetAtt NatGateway2EIP.AllocationId
    SubnetId: !Ref PublicSubnet2
    Tags:
      - Key: Name
        Value: !Sub '${AWS::StackName}-nat-2'

# Route Tables
PublicRouteTable:
  Type: AWS::EC2::RouteTable
  Properties:
    VpcId: !Ref VPC
```


Tags:

- Key: Name
Value: !Sub '\${AWS::StackName}-public-routes'

DefaultPublicRoute:

Type: AWS::EC2::Route
DependsOn: InternetGatewayAttachment
Properties:
RouteTableId: !Ref PublicRouteTable
DestinationCidrBlock: 0.0.0.0/0
GatewayId: !Ref InternetGateway

PublicSubnet1RouteTableAssociation:

Type: AWS::EC2::SubnetRouteTableAssociation
Properties:
RouteTableId: !Ref PublicRouteTable
SubnetId: !Ref PublicSubnet1

PublicSubnet2RouteTableAssociation:

Type: AWS::EC2::SubnetRouteTableAssociation
Properties:
RouteTableId: !Ref PublicRouteTable
SubnetId: !Ref PublicSubnet2

PrivateRouteTable1:

Type: AWS::EC2::RouteTable
Properties:
VpcId: !Ref VPC
Tags:
- Key: Name
Value: !Sub '\${AWS::StackName}-private-routes-1'

DefaultPrivateRoute1:

Type: AWS::EC2::Route
Properties:
RouteTableId: !Ref PrivateRouteTable1
DestinationCidrBlock: 0.0.0.0/0
NatGatewayId: !Ref NatGateway1

PrivateSubnet1RouteTableAssociation:

Type: AWS::EC2::SubnetRouteTableAssociation
Properties:
RouteTableId: !Ref PrivateRouteTable1
SubnetId: !Ref PrivateSubnet1

```
PrivateRouteTable2:
  Type: AWS::EC2::RouteTable
  Properties:
    VpcId: !Ref VPC
    Tags:
      - Key: Name
        Value: !Sub '${AWS::StackName}-private-routes-2'

DefaultPrivateRoute2:
  Type: AWS::EC2::Route
  Properties:
    RouteTableId: !Ref PrivateRouteTable2
    DestinationCidrBlock: 0.0.0.0/0
    NatGatewayId: !Ref NatGateway2

PrivateSubnet2RouteTableAssociation:
  Type: AWS::EC2::SubnetRouteTableAssociation
  Properties:
    RouteTableId: !Ref PrivateRouteTable2
    SubnetId: !Ref PrivateSubnet2

# Security Groups
ALBSecurityGroup:
  Type: AWS::EC2::SecurityGroup
  Properties:
    GroupName: !Sub '${AWS::StackName}-alb-sg'
    GroupDescription: Security group for Application Load Balancer
    VpcId: !Ref VPC
    SecurityGroupIngress:
      - IpProtocol: tcp
        FromPort: 80
        ToPort: 80
        CidrIp: 0.0.0.0/0
        Description: Allow HTTP traffic from internet
    SecurityGroupEgress:
      - IpProtocol: -1
        CidrIp: 0.0.0.0/0
        Description: Allow all outbound traffic
    Tags:
      - Key: Name
        Value: !Sub '${AWS::StackName}-alb-sg'

ECSSecurityGroup:
```

```
Type: AWS::EC2::SecurityGroup
Properties:
  GroupName: !Sub '${AWS::StackName}-ecs-sg'
  GroupDescription: Security group for ECS tasks
  VpcId: !Ref VPC
  SecurityGroupIngress:
    - IpProtocol: tcp
      FromPort: !Ref ContainerPort
      ToPort: !Ref ContainerPort
      SourceSecurityGroupId: !Ref ALBSecurityGroup
      Description: Allow traffic from ALB
  SecurityGroupEgress:
    - IpProtocol: -1
      CidrIp: 0.0.0.0/0
      Description: Allow all outbound traffic
  Tags:
    - Key: Name
      Value: !Sub '${AWS::StackName}-ecs-sg'
```

Application Load Balancer

```
ApplicationLoadBalancer:
  Type: AWS::ElasticLoadBalancingV2::LoadBalancer
  Properties:
    Name: !Sub '${AWS::StackName}-alb'
    Scheme: internet-facing
    Type: application
    Subnets:
      - !Ref PublicSubnet1
      - !Ref PublicSubnet2
    SecurityGroups:
      - !Ref ALBSecurityGroup
    Tags:
      - Key: Name
        Value: !Sub '${AWS::StackName}-alb'
```

```
ALBTargetGroup:
  Type: AWS::ElasticLoadBalancingV2::TargetGroup
  Properties:
    Name: !Sub '${AWS::StackName}-tg'
    Port: !Ref ContainerPort
    Protocol: HTTP
    VpcId: !Ref VPC
    TargetType: ip
    HealthCheckIntervalSeconds: 30
```

```
HealthCheckPath: /
HealthCheckProtocol: HTTP
HealthCheckTimeoutSeconds: 5
HealthyThresholdCount: 2
UnhealthyThresholdCount: 5
Tags:
  - Key: Name
    Value: !Sub '${AWS::StackName}-tg'
```

```
ALBListener:
  Type: AWS::ElasticLoadBalancingV2::Listener
  Properties:
    DefaultActions:
      - Type: forward
        TargetGroupArn: !Ref ALBTargetGroup
    LoadBalancerArn: !Ref ApplicationLoadBalancer
    Port: 80
    Protocol: HTTP
```

```
# ECS Cluster
ECSCluster:
  Type: AWS::ECS::Cluster
  Properties:
    ClusterName: !Sub '${AWS::StackName}-cluster'
    CapacityProviders:
      - FARGATE
      - FARGATE_SPOT
    DefaultCapacityProviderStrategy:
      - CapacityProvider: FARGATE
        Weight: 1
      - CapacityProvider: FARGATE_SPOT
        Weight: 4
    ClusterSettings:
      - Name: containerInsights
        Value: enabled
    Tags:
      - Key: Name
        Value: !Sub '${AWS::StackName}-cluster'
```

```
# IAM Roles
ECSTaskExecutionRole:
  Type: AWS::IAM::Role
  Properties:
    RoleName: !Sub '${AWS::StackName}-task-execution-role'
```

```

AssumeRolePolicyDocument:
  Version: '2012-10-17'
  Statement:
    - Effect: Allow
      Principal:
        Service: ecs-tasks.amazonaws.com
      Action: sts:AssumeRole
ManagedPolicyArns:
  - arn:aws:iam::aws:policy/service-role/AmazonECSTaskExecutionRolePolicy
Tags:
  - Key: Name
    Value: !Sub '${AWS::StackName}-task-execution-role'

```

```

ECSTaskRole:
  Type: AWS::IAM::Role
  Properties:
    RoleName: !Sub '${AWS::StackName}-task-role'
    AssumeRolePolicyDocument:
      Version: '2012-10-17'
      Statement:
        - Effect: Allow
          Principal:
            Service: ecs-tasks.amazonaws.com
          Action: sts:AssumeRole
    Tags:
      - Key: Name
        Value: !Sub '${AWS::StackName}-task-role'

```

```

# CloudWatch Log Group
LogGroup:
  Type: AWS::Logs::LogGroup
  Properties:
    LogGroupName: !Sub '/ecs/${AWS::StackName}'
    RetentionInDays: 7

```

```

# ECS Task Definition
TaskDefinition:
  Type: AWS::ECS::TaskDefinition
  Properties:
    Family: !Sub '${AWS::StackName}-task'
    Cpu: '256'
    Memory: '512'
    NetworkMode: awsvpc
    RequiresCompatibilities:

```

```

- FARGATE
ExecutionRoleArn: !GetAtt ECSTaskExecutionRole.Arn
TaskRoleArn: !GetAtt ECSTaskRole.Arn
ContainerDefinitions:
  - Name: !Ref ServiceName
    Image: !Ref ContainerImage
    PortMappings:
      - ContainerPort: !Ref ContainerPort
        Protocol: tcp
    Essential: true
    LogConfiguration:
      LogDriver: awslogs
      Options:
        awslogs-group: !Ref LogGroup
        awslogs-region: !Ref AWS::Region
        awslogs-stream-prefix: ecs
    HealthCheck:
      Command:
        - CMD-SHELL
        - curl -f http://localhost/ || exit 1
      Interval: 30
      Timeout: 5
      Retries: 3
      StartPeriod: 60
Tags:
  - Key: Name
    Value: !Sub '${AWS::StackName}-task'

```

```

# ECS Service
ECSService:
  Type: AWS::ECS::Service
  DependsOn: ALBListener
  Properties:
    ServiceName: !Sub '${AWS::StackName}-service'
    Cluster: !Ref ECSCluster
    TaskDefinition: !Ref TaskDefinition
    DesiredCount: !Ref DesiredCount
    LaunchType: FARGATE
    PlatformVersion: LATEST
    NetworkConfiguration:
      AwsvpcConfiguration:
        AssignPublicIp: DISABLED
      SecurityGroups:
        - !Ref ECSSecurityGroup

```

```

    Subnets:
      - !Ref PrivateSubnet1
      - !Ref PrivateSubnet2
  LoadBalancers:
    - ContainerName: !Ref ServiceName
      ContainerPort: !Ref ContainerPort
      TargetGroupArn: !Ref ALBTargetGroup
  DeploymentConfiguration:
    MaximumPercent: 200
    MinimumHealthyPercent: 50
    DeploymentCircuitBreaker:
      Enable: true
      Rollback: true
  EnableExecuteCommand: true # For debugging
  Tags:
    - Key: Name
      Value: !Sub '${AWS::StackName}-service'

# Auto Scaling Target
ServiceScalingTarget:
  Type: AWS::ApplicationAutoScaling::ScalableTarget
  Properties:
    MaxCapacity: !Ref MaxCapacity
    MinCapacity: !Ref MinCapacity
    ResourceId: !Sub 'service/${ECSCluster}/${ECSService.Name}'
    RoleARN: !Sub 'arn:aws:iam::${AWS::AccountId}:role/
aws-service-role/ecs.application-autoscaling.amazonaws.com/
AWSServiceRoleForApplicationAutoScaling_ECSService'
    ScalableDimension: ecs:service:DesiredCount
    ServiceNamespace: ecs

# Auto Scaling Policy - CPU Utilization
ServiceScalingPolicy:
  Type: AWS::ApplicationAutoScaling::ScalingPolicy
  Properties:
    PolicyName: !Sub '${AWS::StackName}-cpu-scaling-policy'
    PolicyType: TargetTrackingScaling
    ScalingTargetId: !Ref ServiceScalingTarget
    TargetTrackingScalingPolicyConfiguration:
      PredefinedMetricSpecification:
        PredefinedMetricType: ECSServiceAverageCPUUtilization
      TargetValue: 70.0
      ScaleOutCooldown: 300
      ScaleInCooldown: 300

```

Outputs:**VPCId:**

Description: VPC ID

Value: !Ref VPC

Export:

Name: !Sub '\${AWS::StackName}-VPC-ID'

LoadBalancerURL:

Description: URL of the Application Load Balancer

Value: !Sub 'http://\${ApplicationLoadBalancer.DNSName}'

Export:

Name: !Sub '\${AWS::StackName}-ALB-URL'

ECSClusterName:

Description: Name of the ECS Cluster

Value: !Ref ECSCluster

Export:

Name: !Sub '\${AWS::StackName}-ECS-Cluster'

ECSServiceName:

Description: Name of the ECS Service

Value: !GetAtt ECSService.Name

Export:

Name: !Sub '\${AWS::StackName}-ECS-Service'

PrivateSubnet1:

Description: Private Subnet 1 ID

Value: !Ref PrivateSubnet1

Export:

Name: !Sub '\${AWS::StackName}-Private-Subnet-1'

PrivateSubnet2:

Description: Private Subnet 2 ID

Value: !Ref PrivateSubnet2

Export:

Name: !Sub '\${AWS::StackName}-Private-Subnet-2'

EOF

本教學課程中使用的範本會建立 Amazon ECS 服務，其中包含兩項在 Fargate 上執行的任務。每項任務都會執行範例 Amazon ECS 應用程式。範本也會建立一個 Application Load Balancer (用於分配應用程式流量)，以及一個根據 CPU 使用率擴展應用程式的應用程式自動擴展政策。該範本還會建立部署應用程式所需的聯網資源、容器日誌的記錄資源，以及 Amazon ECS 任務執行 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](#)」。如需有關自動擴展的詳細資訊，請參閱[自動擴展您的 Amazon ECS 服務](#)。

建立範本檔案之後，請使用下列命令來建立堆疊。建立範本中指定的 Amazon ECS 任務執行角色需要 `--capabilities` 旗標。您也可以指定 `--parameters` 旗標來自訂範本參數。

```
aws cloudformation create-stack \  
  --stack-name ecs-tutorial-stack \  
  --template-body file://ecs-tutorial-template.yaml \  
  --region aws-region \  
  --capabilities CAPABILITY_NAMED_IAM
```

執行 `create-stack` 命令後，您可以使用 `describe-stacks` 來檢查堆疊建立的狀態。

```
aws cloudformation describe-stacks \  
  --stack-name ecs-tutorial-stack \  
  --region aws-region
```

步驟 2：驗證 Amazon ECS 資源建立

若要確保正確建立 Amazon ECS 資源，請遵循下列步驟。

1. 執行下列指令以列出 AWS 區域中的所有任務定義。

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

此命令會傳回任務定義 Amazon Resource Name (ARN) 的清單。您使用範本建立的任務定義的 ARN 會以下列格式顯示。

```
{  
  "taskDefinitionArns": [  
    .....  
    "arn:aws:ecs:aws-region:111122223333:task-definition/ecs-tutorial-stack-  
task:1",  
    .....  
  ]  
}
```

2. 執行下列命令以列出 AWS 區域中的所有叢集。

```
aws ecs list-clusters
```

該命令會傳回叢集 ARN 的清單。您使用範本建立的叢集的 ARN 會以下列格式顯示。

```
{
  "clusterArns": [
    .....
    "arn:aws:ecs:aws-region:111122223333:cluster/ecs-tutorial-stack-cluster",
    .....
  ]
}
```

3. 執行下列指令以列出叢集 `ecs-tutorial-stack-cluster` 中的所有服務。

```
aws ecs list-services \
  --cluster ecs-tutorial-stack-cluster
```

該命令會傳回服務 ARN 的清單。您使用範本建立的服務的 ARN 會以下列格式顯示。

```
{
  "serviceArns": [
    "arn:aws:ecs:aws-region:111122223333:service/ecs-tutorial-stack-cluster/
    ecs-tutorial-stack-service"
  ]
}
```

您也可以取得已建立之 Application Load Balancer 的 DNS 名稱，並使用它來驗證資源是否已建立。若要取得 DNS 名稱，請執行下列命令：

執行下列命令以擷取已建立堆疊的輸出。

```
aws cloudformation describe-stacks \
  --stack-name ecs-tutorial-stack \
  --region aws-region \
  --query 'Stacks[0].Outputs[?OutputKey==`LoadBalancerURL`].OutputValue' \
  --output text
```

輸出：

```
http://ecs-tutorial-stack-alb-0123456789.aws-region.elb.amazonaws.com
```

將 DNS 名稱貼到瀏覽器中，以檢視顯示範例 Amazon ECS 應用程式的網頁。

步驟 3：清理

若要清理您建立的資源，請執行下列命令。

```
aws cloudformation delete-stack \  
  --stack-name ecs-stack
```

delete-stack 命令會啟動刪除在本教學課程中建立的 CloudFormation 堆疊，並刪除堆疊中的所有資源。若要驗證是否已刪除，您可以重複[步驟 2：驗證 Amazon ECS 資源建立](#)中的程序。輸出中的 ARN 清單將不再包含名為 ecs-tutorial-stack-task 的任務定義或名為 ecs-tutorial-stack-cluster 的叢集。list-services 呼叫將會失敗。

CloudFormation Amazon ECS 的範例範本

您可以使用 建立 Amazon ECS 叢集、任務定義和服務 CloudFormation。下列主題包括示範如何建立具有不同組態的資源的範本。您可以使用 CloudFormation 主控台或 `aws` CLI，使用這些範本建立這些資源。

CloudFormation 範本是 JSON 或 YAML 格式的文字檔案，描述您要在 CloudFormation 堆疊中佈建的資源。如果您不熟悉 JSON 或 YAML 格式或兩者，您可以使用 AWS Infrastructure Composer 開始使用 CloudFormation 範本。如需詳細資訊，請參閱 AWS CloudFormation User Guide 中的 [Create templates visually with Infrastructure Composer](#)。

下列主題列出了 Amazon ECS 任務定義、叢集與服務的範例範本。

主題

- [任務定義](#)
- [容量提供者](#)
- [叢集](#)
- [服務](#)
- [適用於 Amazon ECS 的 IAM 角色](#)

任務定義

任務定義是應用程式的藍圖，描述構成應用程式的參數和一或多個容器。以下是 Amazon ECS 任務定義的範例 CloudFormation 範本。如需有關 Amazon ECS 任務定義的詳細資訊，請參閱 [Amazon ECS 任務定義](#)。

Fargate Linux 任務定義

您可以使用下列範本來建立範例 Fargate Linux 任務。

JSON

```
{
  "AWSTemplateFormatVersion": "2010-09-09",
  "Description": "ECS Task Definition with parameterized values",
  "Parameters": {
    "ContainerImage": {
      "Type": "String",
      "Default": "public.ecr.aws/docker/library/httpd:2.4",
      "Description": "The container image to use for the task"
    },
    "ContainerCpu": {
      "Type": "Number",
      "Default": 256,
      "Description": "The number of CPU units to reserve for the container",
      "AllowedValues": [256, 512, 1024, 2048, 4096]
    },
    "ContainerMemory": {
      "Type": "Number",
      "Default": 512,
      "Description": "The amount of memory (in MiB) to reserve for the container",
      "AllowedValues": [512, 1024, 2048, 3072, 4096, 5120, 6144, 7168, 8192]
    },
    "TaskFamily": {
      "Type": "String",
      "Default": "task-definition-cfn",
      "Description": "The name of the task definition family"
    },
    "ContainerName": {
      "Type": "String",
      "Default": "sample-fargate-app",
      "Description": "The name of the container"
    }
  },
```

```

"ContainerPort": {
  "Type": "Number",
  "Default": 80,
  "Description": "The port number on the container"
},
"HostPort": {
  "Type": "Number",
  "Default": 80,
  "Description": "The port number on the host"
},
"ExecutionRoleArn": {
  "Type": "String",
  "Default": "arn:aws:iam::aws_account_id:role/ecsTaskExecutionRole",
  "Description": "The ARN of the task execution role"
},
"LogGroup": {
  "Type": "String",
  "Default": "/ecs/fargate-task-definition",
  "Description": "The CloudWatch log group for container logs"
},
"NetworkMode": {
  "Type": "String",
  "Default": "awsvpc",
  "Description": "The Docker networking mode to use",
  "AllowedValues": ["awsvpc", "bridge", "host", "none"]
},
"OperatingSystemFamily": {
  "Type": "String",
  "Default": "LINUX",
  "Description": "The operating system for the task",
  "AllowedValues": ["LINUX", "WINDOWS_SERVER_2019_FULL",
"WINDOWS_SERVER_2019_CORE", "WINDOWS_SERVER_2022_FULL", "WINDOWS_SERVER_2022_CORE"]
},
"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 && -
foreground\"
    ],
    "EntryPoint": [
        "sh",
        "-c"
    ],
    "Essential": true,
    "Image": {"Ref": "ContainerImage"},
    "LogConfiguration": {
        "LogDriver": "awslogs",
        "Options": {
            "mode": "non-blocking",
            "max-buffer-size": "25m",
            "awslogs-create-group": "true",
            "awslogs-group": {"Ref": "LogGroup"},
            "awslogs-region": {"Ref": "AWS::Region"},
            "awslogs-stream-prefix": "ecs"
        }
    },
    "Name": {"Ref": "ContainerName"},
    "PortMappings": [
        {
            "ContainerPort": {"Ref": "ContainerPort"},
            "HostPort": {"Ref": "HostPort"},
            "Protocol": "tcp"
        }
    ]
    }
    ],
    "Cpu": {"Ref": "ContainerCpu"},
    "ExecutionRoleArn": {"Ref": "ExecutionRoleArn"},
    "Family": {"Ref": "TaskFamily"},
    "Memory": {"Ref": "ContainerMemory"},
    "NetworkMode": {"Ref": "NetworkMode"},
    "RequiresCompatibilities": [
        "FARGATE"
    ],
    "RuntimePlatform": {
        "OperatingSystemFamily": {"Ref": "OperatingSystemFamily"}
    }
    }
}

```

```

},
"Outputs": {
  "TaskDefinitionArn": {
    "Description": "The ARN of the created task definition",
    "Value": {"Ref": "ECSTaskDefinition"}
  }
}
}

```

YAML

```

AWSTemplateFormatVersion: 2010-09-09
Description: 'ECS Task Definition to deploy a sample app'
Parameters:
  ContainerImage:
    Type: String
    Default: 'public.ecr.aws/docker/library/httpd:2.4'
    Description: The container image to use for the task
  ContainerCpu:
    Type: Number
    Default: 256
    Description: The number of CPU units to reserve for the container
    AllowedValues: [256, 512, 1024, 2048, 4096]
  ContainerMemory:
    Type: Number
    Default: 512
    Description: The amount of memory (in MiB) to reserve for the container
    AllowedValues: [512, 1024, 2048, 3072, 4096, 5120, 6144, 7168, 8192]
  TaskFamily:
    Type: String
    Default: 'task-definition-cfn'
    Description: The name of the task definition family
  ContainerName:
    Type: String
    Default: 'sample-fargate-app'
    Description: The name of the container
  ContainerPort:
    Type: Number
    Default: 80
    Description: The port number on the container
  HostPort:
    Type: Number
    Default: 80

```

```

    Description: The port number on the host
ExecutionRoleArn:
  Type: String
  Default: 'arn:aws:iam::111122223333:role/ecsTaskExecutionRole'
  Description: The ARN of the task execution role
LogGroup:
  Type: String
  Default: '/ecs/fargate-task-definition'
  Description: The CloudWatch log group for container logs
NetworkMode:
  Type: String
  Default: 'awsvpc'
  Description: The Docker networking mode to use
  AllowedValues: ['awsvpc', 'bridge', 'host', 'none']
OperatingSystemFamily:
  Type: String
  Default: 'LINUX'
  Description: The operating system for the task
  AllowedValues: ['LINUX', 'WINDOWS_SERVER_2019_FULL', 'WINDOWS_SERVER_2019_CORE',
'WINDOWS_SERVER_2022_FULL', 'WINDOWS_SERVER_2022_CORE']
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: !Ref ContainerImage
        LogConfiguration:
          LogDriver: awslogs
          Options:
            mode: non-blocking
            max-buffer-size: 25m

```



```

        awslogs-create-group: 'true'
        awslogs-group: !Ref LogGroup
        awslogs-region: !Ref AWS::Region
        awslogs-stream-prefix: ecs
    Name: !Ref ContainerName
    PortMappings:
      - ContainerPort: !Ref ContainerPort
        HostPort: !Ref HostPort
        Protocol: tcp
    Cpu: !Ref ContainerCpu
    ExecutionRoleArn: !Ref ExecutionRoleArn
    Family: !Ref TaskFamily
    Memory: !Ref ContainerMemory
    NetworkMode: !Ref NetworkMode
    RequiresCompatibilities:
      - FARGATE
    RuntimePlatform:
      OperatingSystemFamily: !Ref OperatingSystemFamily
  Outputs:
    TaskDefinitionArn:
      Description: The ARN of the created task definition
      Value: !Ref ECSTaskDefinition

```

Amazon EFS 任務定義

您可以使用下列範本來建立使用您建立之 Amazon EFS 檔案系統的任務。如需有關將 Amazon EFS 磁碟區與 Amazon ECS 搭配使用的詳細資訊，請參閱[將 Amazon EFS 磁碟區與 Amazon ECS 搭配使用](#)。

JSON

```

{
  "AWSTemplateFormatVersion": "2010-09-09",
  "Description": "Create a task definition for a web server with parameterized values.",
  "Parameters": {
    "ExecutionRoleArn": {
      "Type": "String",
      "Default": "arn:aws:iam::123456789012:role/ecsTaskExecutionRole",
      "Description": "The ARN of the task execution role"
    },
    "NetworkMode": {

```

```
    "Type": "String",
    "Default": "awsvpc",
    "Description": "The Docker networking mode to use",
    "AllowedValues": ["awsvpc", "bridge", "host", "none"]
  },
  "TaskFamily": {
    "Type": "String",
    "Default": "my-ecs-task",
    "Description": "The name of the task definition family"
  },
  "ContainerCpu": {
    "Type": "String",
    "Default": "256",
    "Description": "The number of CPU units to reserve for the container",
    "AllowedValues": ["256", "512", "1024", "2048", "4096"]
  },
  "ContainerMemory": {
    "Type": "String",
    "Default": "512",
    "Description": "The amount of memory (in MiB) to reserve for the container",
    "AllowedValues": ["512", "1024", "2048", "3072", "4096", "5120", "6144",
"7168", "8192"]
  },
  "ContainerName": {
    "Type": "String",
    "Default": "nginx",
    "Description": "The name of the container"
  },
  "ContainerImage": {
    "Type": "String",
    "Default": "public.ecr.aws/nginx/nginx:latest",
    "Description": "The container image to use for the task"
  },
  "ContainerPort": {
    "Type": "Number",
    "Default": 80,
    "Description": "The port number on the container"
  },
  "InitProcessEnabled": {
    "Type": "String",
    "Default": "true",
    "Description": "Whether to enable the init process inside the container",
    "AllowedValues": ["true", "false"]
  },
}
```

```
"EfsVolumeName": {
  "Type": "String",
  "Default": "efs-volume",
  "Description": "The name of the EFS volume"
},
"EfsContainerPath": {
  "Type": "String",
  "Default": "/usr/share/nginx/html",
  "Description": "The path in the container where the EFS volume will be
mounted"
},
"LogGroup": {
  "Type": "String",
  "Default": "LogGroup",
  "Description": "The CloudWatch log group for container logs"
},
"LogStreamPrefix": {
  "Type": "String",
  "Default": "efs-task",
  "Description": "The prefix for the log stream"
},
"EfsFileSystemId": {
  "Type": "String",
  "Default": "fs-1234567890abcdef0",
  "Description": "The ID of the EFS filesystem"
},
"EfsRootDirectory": {
  "Type": "String",
  "Default": "/",
  "Description": "The root directory in the EFS filesystem"
},
"EfsTransitEncryption": {
  "Type": "String",
  "Default": "ENABLED",
  "Description": "Whether to enable transit encryption for EFS",
  "AllowedValues": ["ENABLED", "DISABLED"]
},
"Resources": {
  "ECSTaskDefinition": {
    "Type": "AWS::ECS::TaskDefinition",
    "Properties": {
      "ExecutionRoleArn": {"Ref": "ExecutionRoleArn"},
      "NetworkMode": {"Ref": "NetworkMode"},
```

```

"RequiresCompatibilities": ["FARGATE"],
"Family": {"Ref": "TaskFamily"},
"Cpu": {"Ref": "ContainerCpu"},
"Memory": {"Ref": "ContainerMemory"},
"ContainerDefinitions": [
  {
    "Name": {"Ref": "ContainerName"},
    "Image": {"Ref": "ContainerImage"},
    "Essential": true,
    "PortMappings": [
      {
        "ContainerPort": {"Ref": "ContainerPort"},
        "Protocol": "tcp"
      }
    ],
    "LinuxParameters": {
      "InitProcessEnabled": {"Ref": "InitProcessEnabled"}
    },
    "MountPoints": [
      {
        "SourceVolume": {"Ref": "EfsVolumeName"},
        "ContainerPath": {"Ref": "EfsContainerPath"}
      }
    ],
    "LogConfiguration": {
      "LogDriver": "awslogs",
      "Options": {
        "mode": "non-blocking",
        "max-buffer-size": "25m",
        "awslogs-group": {"Ref": "LogGroup"},
        "awslogs-region": {"Ref": "AWS::Region"},
        "awslogs-create-group": "true",
        "awslogs-stream-prefix": {"Ref": "LogStreamPrefix"}
      }
    }
  }
],
"Volumes": [
  {
    "Name": {"Ref": "EfsVolumeName"},
    "EFSVolumeConfiguration": {
      "FileSystemId": {"Ref": "EfsFileSystemId"},
      "RootDirectory": {"Ref": "EfsRootDirectory"},
      "TransitEncryption": {"Ref": "EfsTransitEncryption"}
    }
  }
]

```

```

    }
  }
]
}
},
"Outputs": {
  "TaskDefinitionArn": {
    "Description": "The ARN of the created task definition",
    "Value": {"Ref": "ECSTaskDefinition"}
  }
}
}

```

YAML

```

AWSTemplateFormatVersion: 2010-09-09
Description: Create a task definition for a web server with parameterized values.
Parameters:
  ExecutionRoleArn:
    Type: String
    Default: arn:aws:iam::123456789012:role/ecsTaskExecutionRole
    Description: The ARN of the task execution role
  NetworkMode:
    Type: String
    Default: awsvpc
    Description: The Docker networking mode to use
    AllowedValues: [awsvpc, bridge, host, none]
  TaskFamily:
    Type: String
    Default: my-ecs-task
    Description: The name of the task definition family
  ContainerCpu:
    Type: String
    Default: "256"
    Description: The number of CPU units to reserve for the container
    AllowedValues: ["256", "512", "1024", "2048", "4096"]
  ContainerMemory:
    Type: String
    Default: "512"
    Description: The amount of memory (in MiB) to reserve for the container
    AllowedValues: ["512", "1024", "2048", "3072", "4096", "5120", "6144", "7168",
"8192"]

```

ContainerName:
Type: String
Default: nginx
Description: The name of the container

ContainerImage:
Type: String
Default: public.ecr.aws/nginx/nginx:latest
Description: The container image to use for the task

ContainerPort:
Type: Number
Default: 80
Description: The port number on the container

InitProcessEnabled:
Type: String
Default: "true"
Description: Whether to enable the init process inside the container
AllowedValues: ["true", "false"]

EfsVolumeName:
Type: String
Default: efs-volume
Description: The name of the EFS volume

EfsContainerPath:
Type: String
Default: /usr/share/nginx/html
Description: The path in the container where the EFS volume will be mounted

LogGroup:
Type: String
Default: LogGroup
Description: The CloudWatch log group for container logs

LogStreamPrefix:
Type: String
Default: efs-task
Description: The prefix for the log stream

EfsFileSystemId:
Type: String
Default: fs-1234567890abcdef0
Description: The ID of the EFS filesystem

EfsRootDirectory:
Type: String
Default: /
Description: The root directory in the EFS filesystem

EfsTransitEncryption:
Type: String
Default: ENABLED

Description: Whether to enable transit encryption for EFS

AllowedValues: [ENABLED, DISABLED]

Resources:

ECSTaskDefinition:

Type: AWS::ECS::TaskDefinition

Properties:

ExecutionRoleArn: !Ref ExecutionRoleArn

NetworkMode: !Ref NetworkMode

RequiresCompatibilities:

- FARGATE

Family: !Ref TaskFamily

Cpu: !Ref ContainerCpu

Memory: !Ref ContainerMemory

ContainerDefinitions:

- Name: !Ref ContainerName

Image: !Ref ContainerImage

Essential: true

PortMappings:

- ContainerPort: !Ref ContainerPort

Protocol: tcp

LinuxParameters:

InitProcessEnabled: !Ref InitProcessEnabled

MountPoints:

- SourceVolume: !Ref EfsVolumeName

ContainerPath: !Ref EfsContainerPath

LogConfiguration:

LogDriver: awslogs

Options:

mode: non-blocking

max-buffer-size: 25m

awslogs-group: !Ref LogGroup

awslogs-region: !Ref AWS::Region

awslogs-create-group: "true"

awslogs-stream-prefix: !Ref LogStreamPrefix

Volumes:

- Name: !Ref EfsVolumeName

EFSVolumeConfiguration:

FileSystemId: !Ref EfsFileSystemId

RootDirectory: !Ref EfsRootDirectory

TransitEncryption: !Ref EfsTransitEncryption

Outputs:

TaskDefinitionArn:

Description: The ARN of the created task definition

```
Value: !Ref ECSTaskDefinition
```

容量提供者

容量提供者與 Amazon ECS 叢集相關聯，用於管理工作負載的運算容量。

為 Amazon ECS 受管執行個體建立容量提供者

依預設，Amazon ECS 提供的容量提供者會自動選取最具成本效益的一般用途執行個體類型。不過，您可以建立自訂容量提供者來指定執行個體需求，例如執行個體類型、CPU 製造商、加速器類型與其他需求。您可以使用下列範本，為滿足指定記憶體與 CPU 需求的 Amazon ECS 受管執行個體建立容量提供者。

JSON

```
{
  "AWSTemplateFormatVersion": "2010-09-09",
  "Resources": {
    "MyCapacityProvider": {
      "Type": "AWS::ECS::CapacityProvider",
      "Properties": {
        "ManagedInstancesProvider": {
          "InfrastructureRoleArn": "arn:aws:iam::123456789012:role/
ECSInfrastructureRole",
          "InstanceLaunchTemplate": {
            "Ec2InstanceProfileArn":
"arn:aws:iam::123456789012:instance-profile/ecsInstanceProfile",
            "NetworkConfiguration": null,
            "Subnets": [
              "subnet-12345678"
            ],
            "SecurityGroups": [
              "sg-87654321"
            ]
          },
          "StorageConfiguration": {
            "StorageSizeGiB": 30
          },
          "InstanceRequirements": {
            "VCpuCount": {
              "Min": 1,
```



```
    "Max": 4  
  },  
  "MemoryMiB": {  
    "Min": 2048,  
    "Max": 8192  
  }  
}  
  
}  
  
}  
  
}  
  
}
```

YAML

```

AWSTemplateFormatVersion: 2010-09-09
Resources:
  MyCapacityProvider:
    Type: AWS::ECS::CapacityProvider
    Properties:
      ManagedInstancesProvider:
        InfrastructureRoleArn: "arn:aws:iam::123456789012:role/
ECSInfrastructureRole"
      InstanceLaunchTemplate:
        Ec2InstanceProfileArn: "arn:aws:iam::123456789012:instance-profile/
ecsInstanceProfile"
        NetworkConfiguration:
          Subnets:
            - "subnet-12345678"
          SecurityGroups:
            - "sg-87654321"
        StorageConfiguration:
          StorageSizeGiB: 30
      InstanceRequirements:
        VCpuCount:
          Min: 1
          Max: 4
        MemoryMiB:
          Min: 2048
          Max: 8192

```

叢集

Amazon ECS 叢集是任務或服務的邏輯分組。您可以使用下列範本來建立具有不同組態的叢集。如需有關 Amazon ECS 叢集的詳細資訊，請參閱 [Amazon ECS 叢集](#)。

建立具有預設設定的空叢集

您可以使用下列範本來建立具有預設設定的空叢集。

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
```

建立具有受管儲存加密與增強型 Container Insights 的空叢集

您可以使用下列範本來建立啟用了叢集層級受管儲存與增強型 Container Insights 的叢集。叢集層級加密適用於 Amazon ECS 受管資料磁碟區，例如 Amazon EBS 磁碟區。如需有關 Amazon EBS 加密的詳細資訊，請參閱[加密儲存於連接至 Amazon ECS 任務之 Amazon EBS 磁碟區中的資料](#)。如需有關使用具有增強可觀測性的 Container Insights 的詳細資訊，請參閱[使用具有增強可觀測性的 Container Insights 監控 Amazon ECS](#)。

JSON

```
{
  "AWSTemplateFormatVersion": "2010-09-09",
  "Resources": {
    "Cluster": {
      "Type": "AWS::ECS::Cluster",
      "Properties": {
        "ClusterName": "EncryptedEnhancedCluster",
        "ClusterSettings": [
          {
            "Name": "containerInsights",
            "Value": "enhanced"
          }
        ],
        "Configuration": {
          "ManagedStorageConfiguration": {
            "KmsKeyId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"
          }
        }
      }
    }
  }
}
```

YAML

```
AWSTemplateFormatVersion: 2010-09-09
Resources:
  Cluster:
    Type: AWS::ECS::Cluster
    Properties:
      ClusterName: EncryptedEnhancedCluster
      ClusterSettings:
        - Name: containerInsights
          Value: enhanced
      Configuration:
        ManagedStorageConfiguration:
          KmsKeyId: a1b2c3d4-5678-90ab-cdef-EXAMPLE11111
```

建立具有 AL2023 Amazon ECS-Optimized-AMI 的叢集

您可以使用下列範本來建立叢集，該叢集使用在 Amazon EC2 上啟動 AL2023 執行個體的容量提供者。

Important

如需取得最新的 AMI ID，請參閱《Amazon Elastic Container Service 開發人員指南》中的 [Amazon ECS 最佳化 AMI](#)。

JSON

```
{
  "AWSTemplateFormatVersion": "2010-09-09",
  "Description": "EC2 ECS cluster that starts out empty, with no EC2 instances yet. An ECS capacity provider automatically launches more EC2 instances as required on the fly when you request ECS to launch services or standalone tasks.",
  "Parameters": {
    "InstanceType": {
      "Type": "String",
      "Description": "EC2 instance type",
      "Default": "t2.medium",
      "AllowedValues": [
        "t1.micro",
        "t2.2xlarge",
        "t2.large",
        "t2.medium",
        "t2.micro",
        "t2.nano",
        "t2.small",
        "t2.xlarge",
        "t3.2xlarge",
        "t3.large",
        "t3.medium",
        "t3.micro",
        "t3.nano",
        "t3.small",
        "t3.xlarge"
      ]
    },
    "DesiredCapacity": {
```

```

        "Type": "Number",
        "Default": "0",
        "Description": "Number of EC2 instances to launch in your ECS cluster."
    },
    "MaxSize": {
        "Type": "Number",
        "Default": "100",
        "Description": "Maximum number of EC2 instances that can be launched in
your ECS cluster."
    },
    "ECSAMI": {
        "Description": "The Amazon Machine Image ID used for the cluster",
        "Type": "AWS::SSM::Parameter::Value<AWS::EC2::Image::Id>",
        "Default": "/aws/service/ecs/optimized-ami/amazon-linux-2023/
recommended/image_id"
    },
    "VpcId": {
        "Type": "AWS::EC2::VPC::Id",
        "Description": "VPC ID where the ECS cluster is launched",
        "Default": "vpc-1234567890abcdef0"
    },
    "SubnetIds": {
        "Type": "List<AWS::EC2::Subnet::Id>",
        "Description": "List of subnet IDs where the EC2 instances will be
launched",
        "Default": "subnet-021345abcdef67890"
    }
},
"Resources": {
    "ECSCluster": {
        "Type": "AWS::ECS::Cluster",
        "Properties": {
            "ClusterSettings": [
                {
                    "Name": "containerInsights",
                    "Value": "enabled"
                }
            ]
        }
    },
    "ECSAutoScalingGroup": {
        "Type": "AWS::AutoScaling::AutoScalingGroup",
        "DependsOn": [
            "ECSCluster",

```

```

        "EC2Role"
    ],
    "Properties": {
        "VPCZoneIdentifier": {
            "Ref": "SubnetIds"
        },
        "LaunchTemplate": {
            "LaunchTemplateId": {
                "Ref": "ContainerInstances"
            },
            "Version": {
                "Fn::GetAtt": [
                    "ContainerInstances",
                    "LatestVersionNumber"
                ]
            }
        },
        "MinSize": 0,
        "MaxSize": {
            "Ref": "MaxSize"
        },
        "DesiredCapacity": {
            "Ref": "DesiredCapacity"
        },
        "NewInstancesProtectedFromScaleIn": true
    },
    "UpdatePolicy": {
        "AutoScalingReplacingUpdate": {
            "WillReplace": "true"
        }
    }
},
"ContainerInstances": {
    "Type": "AWS::EC2::LaunchTemplate",
    "Properties": {
        "LaunchTemplateName": "asg-launch-template-2",
        "LaunchTemplateData": {
            "ImageId": {
                "Ref": "ECSAMI"
            },
            "InstanceType": {
                "Ref": "InstanceType"
            },
            "IamInstanceProfile": {

```

```

        "Name": {
            "Ref": "EC2InstanceProfile"
        },
        "SecurityGroupIds": [
            {
                "Ref": "ContainerHostSecurityGroup"
            }
        ],
        "UserData": {
            "Fn::Base64": {
                "Fn::Sub": "#!/bin/bash -xe\n echo ECS_CLUSTER=
${ECSCluster} >> /etc/ecs/ecs.config\n yum install -y aws-cfn-bootstrap\n /opt/aws/
bin/cfn-init -v --stack ${AWS::StackId} --resource ContainerInstances --configsets
full_install --region ${AWS::Region} &\n"
            }
        },
        "MetadataOptions": {
            "HttpEndpoint": "enabled",
            "HttpTokens": "required"
        }
    }
},
"EC2InstanceProfile": {
    "Type": "AWS::IAM::InstanceProfile",
    "Properties": {
        "Path": "/",
        "Roles": [
            {
                "Ref": "EC2Role"
            }
        ]
    }
},
"CapacityProvider": {
    "Type": "AWS::ECS::CapacityProvider",
    "Properties": {
        "AutoScalingGroupProvider": {
            "AutoScalingGroupArn": {
                "Ref": "ECSAutoScalingGroup"
            },
            "ManagedScaling": {
                "InstanceWarmupPeriod": 60,

```

```

        "MinimumScalingStepSize": 1,
        "MaximumScalingStepSize": 100,
        "Status": "ENABLED",
        "TargetCapacity": 100
    },
    "ManagedTerminationProtection": "ENABLED"
}
},
"CapacityProviderAssociation": {
    "Type": "AWS::ECS::ClusterCapacityProviderAssociations",
    "Properties": {
        "CapacityProviders": [
            {
                "Ref": "CapacityProvider"
            }
        ],
        "Cluster": {
            "Ref": "ECSCluster"
        },
        "DefaultCapacityProviderStrategy": [
            {
                "Base": 0,
                "CapacityProvider": {
                    "Ref": "CapacityProvider"
                },
                "Weight": 1
            }
        ]
    }
},
"ContainerHostSecurityGroup": {
    "Type": "AWS::EC2::SecurityGroup",
    "Properties": {
        "GroupDescription": "Access to the EC2 hosts that run containers",
        "VpcId": {
            "Ref": "VpcId"
        }
    }
},
"EC2Role": {
    "Type": "AWS::IAM::Role",
    "Properties": {
        "AssumeRolePolicyDocument": {

```



```

        "Statement": [
            {
                "Effect": "Allow",
                "Principal": {
                    "Service": [
                        "ec2.amazonaws.com"
                    ]
                },
                "Action": [
                    "sts:AssumeRole"
                ]
            }
        ],
        "Path": "/",
        "ManagedPolicyArns": [
            "arn:aws:iam::aws:policy/service-role/AmazonEC2ContainerServiceforEC2Role",
            "arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore"
        ]
    },
    "ECSTaskExecutionRole": {
        "Type": "AWS::IAM::Role",
        "Properties": {
            "AssumeRolePolicyDocument": {
                "Statement": [
                    {
                        "Effect": "Allow",
                        "Principal": {
                            "Service": [
                                "ecs-tasks.amazonaws.com"
                            ]
                        },
                        "Action": [
                            "sts:AssumeRole"
                        ],
                        "Condition": {
                            "ArnLike": {
                                "aws:SourceArn": {
                                    "Fn::Sub": "arn:${AWS::Partition}:ecs:
${AWS::Region}:${AWS::AccountId}:*"
                                }
                            }
                        }
                    }
                ]
            }
        }
    },

```

```

        "StringEquals": {
            "aws:SourceAccount": {
                "Fn::Sub": "${AWS::AccountId}"
            }
        }
    },
    "Path": "/",
    "ManagedPolicyArns": [
        "arn:aws:iam::aws:policy/service-role/AmazonECSTaskExecutionRolePolicy"
    ]
}
},
"Outputs": {
    "ClusterName": {
        "Description": "The ECS cluster into which to launch resources",
        "Value": "ECSCluster"
    },
    "ECSTaskExecutionRole": {
        "Description": "The role used to start up a task",
        "Value": "ECSTaskExecutionRole"
    },
    "CapacityProvider": {
        "Description": "The cluster capacity provider that the service should use to request capacity when it wants to start up a task",
        "Value": "CapacityProvider"
    }
}
}

```

YAML

```

AWSTemplateFormatVersion: '2010-09-09'
Description: EC2 ECS cluster that starts out empty, with no EC2 instances yet. An ECS capacity provider automatically launches more EC2 instances as required on the fly when you request ECS to launch services or standalone tasks.
Parameters:
  InstanceType:
    Type: String

```

Description: EC2 instance type

Default: t2.medium

AllowedValues:

- t1.micro
- t2.2xlarge
- t2.large
- t2.medium
- t2.micro
- t2.nano
- t2.small
- t2.xlarge
- t3.2xlarge
- t3.large
- t3.medium
- t3.micro
- t3.nano
- t3.small
- t3.xlarge

DesiredCapacity:

Type: Number

Default: '0'

Description: Number of EC2 instances to launch in your ECS cluster.

MaxSize:

Type: Number

Default: '100'

Description: Maximum number of EC2 instances that can be launched in your ECS cluster.

ECSAMI:

Description: The Amazon Machine Image ID used for the cluster

Type: AWS::SSM::Parameter::Value<AWS::EC2::Image::Id>

Default: /aws/service/ecs/optimized-ami/amazon-linux-2023/recommended/image_id

VpcId:

Type: AWS::EC2::VPC::Id

Description: VPC ID where the ECS cluster is launched

Default: vpc-1234567890abcdef0

SubnetIds:

Type: List<AWS::EC2::Subnet::Id>

Description: List of subnet IDs where the EC2 instances will be launched

Default: subnet-021345abcdef67890

Resources:

ECSCluster:

Type: AWS::ECS::Cluster

Properties:

ClusterSettings:

```

    - Name: containerInsights
      Value: enabled
ECSAutoScalingGroup:
  Type: AWS::AutoScaling::AutoScalingGroup
  DependsOn:
    - ECSCluster
    - EC2Role
  Properties:
    VPCZoneIdentifier: !Ref SubnetIds
    LaunchTemplate:
      LaunchTemplateId: !Ref ContainerInstances
      Version: !GetAtt ContainerInstances.LatestVersionNumber
    MinSize: 0
    MaxSize: !Ref MaxSize
    DesiredCapacity: !Ref DesiredCapacity
    NewInstancesProtectedFromScaleIn: true
  UpdatePolicy:
    AutoScalingReplacingUpdate:
      WillReplace: 'true'
ContainerInstances:
  Type: AWS::EC2::LaunchTemplate
  Properties:
    LaunchTemplateName: asg-launch-template-2
    LaunchTemplateData:
      ImageId: !Ref ECSAMI
      InstanceType: !Ref InstanceType
      IamInstanceProfile:
        Name: !Ref EC2InstanceProfile
      SecurityGroupIds:
        - !Ref ContainerHostSecurityGroup
      UserData: !Base64
        Fn::Sub: |
          #!/bin/bash -xe
          echo ECS_CLUSTER=${ECSCluster} >> /etc/ecs/ecs.config
          yum install -y aws-cfn-bootstrap
          /opt/aws/bin/cfn-init -v --stack ${AWS::StackId} --resource
ContainerInstances --configsets full_install --region ${AWS::Region} &
      MetadataOptions:
        HttpEndpoint: enabled
        HttpTokens: required
EC2InstanceProfile:
  Type: AWS::IAM::InstanceProfile
  Properties:
    Path: /

```

```

Roles:
  - !Ref EC2Role
CapacityProvider:
  Type: AWS::ECS::CapacityProvider
  Properties:
    AutoScalingGroupProvider:
      AutoScalingGroupArn: !Ref ECSAutoScalingGroup
    ManagedScaling:
      InstanceWarmupPeriod: 60
      MinimumScalingStepSize: 1
      MaximumScalingStepSize: 100
      Status: ENABLED
      TargetCapacity: 100
    ManagedTerminationProtection: ENABLED
CapacityProviderAssociation:
  Type: AWS::ECS::ClusterCapacityProviderAssociations
  Properties:
    CapacityProviders:
      - !Ref CapacityProvider
    Cluster: !Ref ECSCluster
    DefaultCapacityProviderStrategy:
      - Base: 0
      CapacityProvider: !Ref CapacityProvider
      Weight: 1
ContainerHostSecurityGroup:
  Type: AWS::EC2::SecurityGroup
  Properties:
    GroupDescription: Access to the EC2 hosts that run containers
    VpcId: !Ref VpcId
EC2Role:
  Type: AWS::IAM::Role
  Properties:
    AssumeRolePolicyDocument:
      Statement:
        - Effect: Allow
          Principal:
            Service:
              - ec2.amazonaws.com
          Action:
            - sts:AssumeRole
    Path: /
    ManagedPolicyArns:
      - arn:aws:iam::aws:policy/service-role/AmazonEC2ContainerServiceforEC2Role
      - arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore

```

```

ECSTaskExecutionRole:
  Type: AWS::IAM::Role
  Properties:
    AssumeRolePolicyDocument:
      Statement:
        - Effect: Allow
          Principal:
            Service:
              - ecs-tasks.amazonaws.com
          Action:
            - sts:AssumeRole
          Condition:
            ArnLike:
              aws:SourceArn: !Sub arn:${AWS::Partition}:ecs:${AWS::Region}:
${AWS::AccountId}:*
            StringEquals:
              aws:SourceAccount: !Sub ${AWS::AccountId}
      Path: /
    ManagedPolicyArns:
      - arn:aws:iam::aws:policy/service-role/AmazonECSTaskExecutionRolePolicy
Outputs:
  ClusterName:
    Description: The ECS cluster into which to launch resources
    Value: ECSCluster
  ECSTaskExecutionRole:
    Description: The role used to start up a task
    Value: ECSTaskExecutionRole
  CapacityProvider:
    Description: The cluster capacity provider that the service should use to
request capacity when it wants to start up a task
    Value: CapacityProvider

```

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。這有助於維持服務中所需的任務數量。下列範本可用於部署服務。如需有關 Amazon ECS 服務的詳細資訊，請參閱 [Amazon ECS 服務](#)。

部署負載平衡的 Web 應用程式

下列範本會建立 Amazon ECS 服務，其中包含兩項在 Fargate 上執行的任務。每項任務都有一個 NGINX 容器。範本也會建立一個 Application Load Balancer (用於分配應用程式流量)，以及一個根據 CPU 使用率擴展應用程式的應用程式自動擴展政策。該範本還會建立部署應用程式所需的聯網資源、容器日誌的記錄資源，以及 Amazon ECS 任務執行 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](#)」。如需有關自動擴展的詳細資訊，請參閱[自動擴展您的 Amazon ECS 服務](#)。

JSON

```
{
  "AWSTemplateFormatVersion": "2010-09-09",
  "Description": "[AWSDocs] ECS: load-balanced-web-application",
  "Parameters": {
    "VpcCidr": {
      "Type": "String",
      "Default": "10.0.0.0/16",
      "Description": "CIDR block for the VPC"
    },
    "ContainerImage": {
      "Type": "String",
      "Default": "public.ecr.aws/ecs-sample-image/amazon-ecs-sample:latest",
      "Description": "Container image to use in task definition"
    },
    "PublicSubnet1Cidr": {
      "Type": "String",
      "Default": "10.0.1.0/24",
      "Description": "CIDR block for public subnet 1"
    },
    "PublicSubnet2Cidr": {
      "Type": "String",
      "Default": "10.0.2.0/24",
      "Description": "CIDR block for public subnet 2"
    },
    "PrivateSubnet1Cidr": {
      "Type": "String",
      "Default": "10.0.3.0/24",
      "Description": "CIDR block for private subnet 1"
    },
    "PrivateSubnet2Cidr": {
      "Type": "String",
      "Default": "10.0.4.0/24",

```

```

        "Description": "CIDR block for private subnet 2"
    },
    "ServiceName": {
        "Type": "String",
        "Default": "tutorial-app",
        "Description": "Name of the ECS service"
    },
    "ContainerPort": {
        "Type": "Number",
        "Default": 80,
        "Description": "Port on which the container listens"
    },
    "DesiredCount": {
        "Type": "Number",
        "Default": 2,
        "Description": "Desired number of tasks"
    },
    "MinCapacity": {
        "Type": "Number",
        "Default": 1,
        "Description": "Minimum number of tasks for auto scaling"
    },
    "MaxCapacity": {
        "Type": "Number",
        "Default": 10,
        "Description": "Maximum number of tasks for auto scaling"
    }
},
"Resources": {
    "VPC": {
        "Type": "AWS::EC2::VPC",
        "Properties": {
            "CidrBlock": {
                "Ref": "VpcCidr"
            },
            "EnableDnsHostnames": true,
            "EnableDnsSupport": true,
            "Tags": [
                {
                    "Key": "Name",
                    "Value": {
                        "Fn::Sub": "${AWS::StackName}-vpc"
                    }
                }
            ]
        }
    }
}

```



```

    ]
  },
  "InternetGateway": {
    "Type": "AWS::EC2::InternetGateway",
    "Properties": {
      "Tags": [
        {
          "Key": "Name",
          "Value": {
            "Fn::Sub": "${AWS::StackName}-igw"
          }
        }
      ]
    }
  },
  "InternetGatewayAttachment": {
    "Type": "AWS::EC2::VPCGatewayAttachment",
    "Properties": {
      "InternetGatewayId": {
        "Ref": "InternetGateway"
      },
      "VpcId": {
        "Ref": "VPC"
      }
    }
  },
  "PublicSubnet1": {
    "Type": "AWS::EC2::Subnet",
    "Properties": {
      "VpcId": {
        "Ref": "VPC"
      },
      "AvailabilityZone": {
        "Fn::Select": [
          0,
          {
            "Fn::GetAZs": ""
          }
        ]
      },
      "CidrBlock": {
        "Ref": "PublicSubnet1Cidr"
      }
    }
  },

```

```

        "MapPublicIpOnLaunch": true,
        "Tags": [
            {
                "Key": "Name",
                "Value": {
                    "Fn::Sub": "${AWS::StackName}-public-subnet-1"
                }
            }
        ]
    },
    "PublicSubnet2": {
        "Type": "AWS::EC2::Subnet",
        "Properties": {
            "VpcId": {
                "Ref": "VPC"
            },
            "AvailabilityZone": {
                "Fn::Select": [
                    1,
                    {
                        "Fn::GetAZs": ""
                    }
                ]
            },
            "CidrBlock": {
                "Ref": "PublicSubnet2Cidr"
            },
            "MapPublicIpOnLaunch": true,
            "Tags": [
                {
                    "Key": "Name",
                    "Value": {
                        "Fn::Sub": "${AWS::StackName}-public-subnet-2"
                    }
                }
            ]
        }
    },
    "PrivateSubnet1": {
        "Type": "AWS::EC2::Subnet",
        "Properties": {
            "VpcId": {
                "Ref": "VPC"
            },

```

```

    },
    "AvailabilityZone": {
      "Fn::Select": [
        0,
        {
          "Fn::GetAZs": ""
        }
      ]
    },
    "CidrBlock": {
      "Ref": "PrivateSubnet1Cidr"
    },
    "Tags": [
      {
        "Key": "Name",
        "Value": {
          "Fn::Sub": "${AWS::StackName}-private-subnet-1"
        }
      }
    ]
  }
},
"PrivateSubnet2": {
  "Type": "AWS::EC2::Subnet",
  "Properties": {
    "VpcId": {
      "Ref": "VPC"
    },
    "AvailabilityZone": {
      "Fn::Select": [
        1,
        {
          "Fn::GetAZs": ""
        }
      ]
    },
    "CidrBlock": {
      "Ref": "PrivateSubnet2Cidr"
    },
    "Tags": [
      {
        "Key": "Name",
        "Value": {
          "Fn::Sub": "${AWS::StackName}-private-subnet-2"
        }
      }
    ]
  }
}

```

```

    }
  }
]
}
},
"NatGateway1EIP": {
  "Type": "AWS::EC2::EIP",
  "DependsOn": "InternetGatewayAttachment",
  "Properties": {
    "Domain": "vpc",
    "Tags": [
      {
        "Key": "Name",
        "Value": {
          "Fn::Sub": "${AWS::StackName}-nat-eip-1"
        }
      }
    ]
  }
},
"NatGateway2EIP": {
  "Type": "AWS::EC2::EIP",
  "DependsOn": "InternetGatewayAttachment",
  "Properties": {
    "Domain": "vpc",
    "Tags": [
      {
        "Key": "Name",
        "Value": {
          "Fn::Sub": "${AWS::StackName}-nat-eip-2"
        }
      }
    ]
  }
},
"NatGateway1": {
  "Type": "AWS::EC2::NatGateway",
  "Properties": {
    "AllocationId": {
      "Fn::GetAtt": [
        "NatGateway1EIP",
        "AllocationId"
      ]
    }
  }
},

```

```

        "SubnetId": {
            "Ref": "PublicSubnet1"
        },
        "Tags": [
            {
                "Key": "Name",
                "Value": {
                    "Fn::Sub": "${AWS::StackName}-nat-1"
                }
            }
        ]
    },
    "NatGateway2": {
        "Type": "AWS::EC2::NatGateway",
        "Properties": {
            "AllocationId": {
                "Fn::GetAtt": [
                    "NatGateway2EIP",
                    "AllocationId"
                ]
            },
            "SubnetId": {
                "Ref": "PublicSubnet2"
            },
            "Tags": [
                {
                    "Key": "Name",
                    "Value": {
                        "Fn::Sub": "${AWS::StackName}-nat-2"
                    }
                }
            ]
        }
    },
    "PublicRouteTable": {
        "Type": "AWS::EC2::RouteTable",
        "Properties": {
            "VpcId": {
                "Ref": "VPC"
            },
            "Tags": [
                {
                    "Key": "Name",

```

```

        "Value": {
            "Fn::Sub": "${AWS::StackName}-public-routes"
        }
    ]
},
"DefaultPublicRoute": {
    "Type": "AWS::EC2::Route",
    "DependsOn": "InternetGatewayAttachment",
    "Properties": {
        "RouteTableId": {
            "Ref": "PublicRouteTable"
        },
        "DestinationCidrBlock": "0.0.0.0/0",
        "GatewayId": {
            "Ref": "InternetGateway"
        }
    }
},
"PublicSubnet1RouteTableAssociation": {
    "Type": "AWS::EC2::SubnetRouteTableAssociation",
    "Properties": {
        "RouteTableId": {
            "Ref": "PublicRouteTable"
        },
        "SubnetId": {
            "Ref": "PublicSubnet1"
        }
    }
},
"PublicSubnet2RouteTableAssociation": {
    "Type": "AWS::EC2::SubnetRouteTableAssociation",
    "Properties": {
        "RouteTableId": {
            "Ref": "PublicRouteTable"
        },
        "SubnetId": {
            "Ref": "PublicSubnet2"
        }
    }
},
"PrivateRouteTable1": {
    "Type": "AWS::EC2::RouteTable",

```

```

    "Properties": {
      "VpcId": {
        "Ref": "VPC"
      },
      "Tags": [
        {
          "Key": "Name",
          "Value": {
            "Fn::Sub": "${AWS::StackName}-private-routes-1"
          }
        }
      ]
    }
  },
  "DefaultPrivateRoute1": {
    "Type": "AWS::EC2::Route",
    "Properties": {
      "RouteTableId": {
        "Ref": "PrivateRouteTable1"
      },
      "DestinationCidrBlock": "0.0.0.0/0",
      "NatGatewayId": {
        "Ref": "NatGateway1"
      }
    }
  },
  "PrivateSubnet1RouteTableAssociation": {
    "Type": "AWS::EC2::SubnetRouteTableAssociation",
    "Properties": {
      "RouteTableId": {
        "Ref": "PrivateRouteTable1"
      },
      "SubnetId": {
        "Ref": "PrivateSubnet1"
      }
    }
  },
  "PrivateRouteTable2": {
    "Type": "AWS::EC2::RouteTable",
    "Properties": {
      "VpcId": {
        "Ref": "VPC"
      },
      "Tags": [

```

```

        {
            "Key": "Name",
            "Value": {
                "Fn::Sub": "${AWS::StackName}-private-routes-2"
            }
        }
    ]
}
},
"DefaultPrivateRoute2": {
    "Type": "AWS::EC2::Route",
    "Properties": {
        "RouteTableId": {
            "Ref": "PrivateRouteTable2"
        },
        "DestinationCidrBlock": "0.0.0.0/0",
        "NatGatewayId": {
            "Ref": "NatGateway2"
        }
    }
},
"PrivateSubnet2RouteTableAssociation": {
    "Type": "AWS::EC2::SubnetRouteTableAssociation",
    "Properties": {
        "RouteTableId": {
            "Ref": "PrivateRouteTable2"
        },
        "SubnetId": {
            "Ref": "PrivateSubnet2"
        }
    }
},
"ALBSecurityGroup": {
    "Type": "AWS::EC2::SecurityGroup",
    "Properties": {
        "GroupName": {
            "Fn::Sub": "${AWS::StackName}-alb-sg"
        },
        "GroupDescription": "Security group for Application Load Balancer",
        "VpcId": {
            "Ref": "VPC"
        },
        "SecurityGroupIngress": [
            {

```



```

        "IpProtocol": "tcp",
        "FromPort": 80,
        "ToPort": 80,
        "CidrIp": "0.0.0.0/0",
        "Description": "Allow HTTP traffic from internet"
    }
],
"SecurityGroupEgress": [
    {
        "IpProtocol": -1,
        "CidrIp": "0.0.0.0/0",
        "Description": "Allow all outbound traffic"
    }
],
"Tags": [
    {
        "Key": "Name",
        "Value": {
            "Fn::Sub": "${AWS::StackName}-alb-sg"
        }
    }
]
},
{
    "ECSecurityGroup": {
        "Type": "AWS::EC2::SecurityGroup",
        "Properties": {
            "GroupName": {
                "Fn::Sub": "${AWS::StackName}-ecs-sg"
            },
            "GroupDescription": "Security group for ECS tasks",
            "VpcId": {
                "Ref": "VPC"
            },
            "SecurityGroupIngress": [
                {
                    "IpProtocol": "tcp",
                    "FromPort": {
                        "Ref": "ContainerPort"
                    },
                    "ToPort": {
                        "Ref": "ContainerPort"
                    },
                    "SourceSecurityGroupId": {

```

```

        "Ref": "ALBSecurityGroup"
    },
    "Description": "Allow traffic from ALB"
  }
],
"SecurityGroupEgress": [
  {
    "IpProtocol": "-1",
    "CidrIp": "0.0.0.0/0",
    "Description": "Allow all outbound traffic"
  }
],
"Tags": [
  {
    "Key": "Name",
    "Value": {
      "Fn::Sub": "${AWS::StackName}-ecs-sg"
    }
  }
]
},
"ApplicationLoadBalancer": {
  "Type": "AWS::ElasticLoadBalancingV2::LoadBalancer",
  "Properties": {
    "Name": {
      "Fn::Sub": "${AWS::StackName}-alb"
    },
    "Scheme": "internet-facing",
    "Type": "application",
    "Subnets": [
      {
        "Ref": "PublicSubnet1"
      },
      {
        "Ref": "PublicSubnet2"
      }
    ],
    "SecurityGroups": [
      {
        "Ref": "ALBSecurityGroup"
      }
    ],
    "Tags": [

```

```

        {
            "Key": "Name",
            "Value": {
                "Fn::Sub": "${AWS::StackName}-alb"
            }
        }
    ]
}
},
"ALBTargetGroup": {
    "Type": "AWS::ElasticLoadBalancingV2::TargetGroup",
    "Properties": {
        "Name": {
            "Fn::Sub": "${AWS::StackName}-tg"
        },
        "Port": {
            "Ref": "ContainerPort"
        },
        "Protocol": "HTTP",
        "VpcId": {
            "Ref": "VPC"
        },
        "TargetType": "ip",
        "HealthCheckIntervalSeconds": 30,
        "HealthCheckPath": "/",
        "HealthCheckProtocol": "HTTP",
        "HealthCheckTimeoutSeconds": 5,
        "HealthyThresholdCount": 2,
        "UnhealthyThresholdCount": 5,
        "Tags": [
            {
                "Key": "Name",
                "Value": {
                    "Fn::Sub": "${AWS::StackName}-tg"
                }
            }
        ]
    }
},
"ALBListener": {
    "Type": "AWS::ElasticLoadBalancingV2::Listener",
    "Properties": {
        "DefaultActions": [
            {

```

```

        "Type": "forward",
        "TargetGroupArn": {
            "Ref": "ALBTargetGroup"
        }
    },
    ],
    "LoadBalancerArn": {
        "Ref": "ApplicationLoadBalancer"
    },
    "Port": 80,
    "Protocol": "HTTP"
}
},
"ECSCluster": {
    "Type": "AWS::ECS::Cluster",
    "Properties": {
        "ClusterName": {
            "Fn::Sub": "${AWS::StackName}-cluster"
        },
        "CapacityProviders": [
            "FARGATE",
            "FARGATE_SPOT"
        ],
        "DefaultCapacityProviderStrategy": [
            {
                "CapacityProvider": "FARGATE",
                "Weight": 1
            },
            {
                "CapacityProvider": "FARGATE_SPOT",
                "Weight": 4
            }
        ],
        "ClusterSettings": [
            {
                "Name": "containerInsights",
                "Value": "enabled"
            }
        ],
        "Tags": [
            {
                "Key": "Name",
                "Value": {
                    "Fn::Sub": "${AWS::StackName}-cluster"
                }
            }
        ]
    }
}

```

```

    }
  }
]
},
"ECSTaskExecutionRole": {
  "Type": "AWS::IAM::Role",
  "Properties": {
    "RoleName": {
      "Fn::Sub": "${AWS::StackName}-task-execution-role"
    },
    "AssumeRolePolicyDocument": {
      "Version": "2012-10-17",
      "Statement": [
        {
          "Effect": "Allow",
          "Principal": {
            "Service": "ecs-tasks.amazonaws.com"
          },
          "Action": "sts:AssumeRole"
        }
      ]
    },
    "ManagedPolicyArns": [
      "arn:aws:iam::aws:policy/service-role/AmazonECSTaskExecutionRolePolicy"
    ],
    "Tags": [
      {
        "Key": "Name",
        "Value": {
          "Fn::Sub": "${AWS::StackName}-task-execution-role"
        }
      }
    ]
  }
},
"ECSTaskRole": {
  "Type": "AWS::IAM::Role",
  "Properties": {
    "RoleName": {
      "Fn::Sub": "${AWS::StackName}-task-role"
    },
    "AssumeRolePolicyDocument": {

```

```

        "Version": "2012-10-17",
        "Statement": [
            {
                "Effect": "Allow",
                "Principal": {
                    "Service": "ecs-tasks.amazonaws.com"
                },
                "Action": "sts:AssumeRole"
            }
        ],
        "Tags": [
            {
                "Key": "Name",
                "Value": {
                    "Fn::Sub": "${AWS::StackName}-task-role"
                }
            }
        ]
    },
    "LogGroup": {
        "Type": "AWS::Logs::LogGroup",
        "Properties": {
            "LogGroupName": {
                "Fn::Sub": "/ecs/${AWS::StackName}"
            },
            "RetentionInDays": 7
        }
    },
    "TaskDefinition": {
        "Type": "AWS::ECS::TaskDefinition",
        "Properties": {
            "Family": {
                "Fn::Sub": "${AWS::StackName}-task"
            },
            "Cpu": "256",
            "Memory": "512",
            "NetworkMode": "awsvpc",
            "RequiresCompatibilities": [
                "FARGATE"
            ],
            "ExecutionRoleArn": {
                "Fn::GetAtt": [

```

```

        "ECSTaskExecutionRole",
        "Arn"
    ]
},
"TaskRoleArn": {
    "Fn::GetAtt": [
        "ECSTaskRole",
        "Arn"
    ]
},
"ContainerDefinitions": [
    {
        "Name": {
            "Ref": "ServiceName"
        },
        "Image": {
            "Ref": "ContainerImage"
        },
        "PortMappings": [
            {
                "ContainerPort": {
                    "Ref": "ContainerPort"
                },
                "Protocol": "tcp"
            }
        ],
        "Essential": true,
        "LogConfiguration": {
            "LogDriver": "awslogs",
            "Options": {
                "awslogs-group": {
                    "Ref": "LogGroup"
                },
                "awslogs-region": {
                    "Ref": "AWS::Region"
                },
                "awslogs-stream-prefix": "ecs"
            }
        },
        "HealthCheck": {
            "Command": [
                "CMD-SHELL",
                "curl -f http://localhost/ || exit 1"
            ],

```

```

        "Interval": 30,
        "Timeout": 5,
        "Retries": 3,
        "StartPeriod": 60
    }
}
],
"Tags": [
    {
        "Key": "Name",
        "Value": {
            "Fn::Sub": "${AWS::StackName}-task"
        }
    }
]
}
},
"ECSService": {
    "Type": "AWS::ECS::Service",
    "DependsOn": "ALBListener",
    "Properties": {
        "ServiceName": {
            "Fn::Sub": "${AWS::StackName}-service"
        },
        "Cluster": {
            "Ref": "ECSCluster"
        },
        "TaskDefinition": {
            "Ref": "TaskDefinition"
        },
        "DesiredCount": {
            "Ref": "DesiredCount"
        },
        "LaunchType": "FARGATE",
        "PlatformVersion": "LATEST",
        "NetworkConfiguration": {
            "AwsvpcConfiguration": {
                "AssignPublicIp": "DISABLED",
                "SecurityGroups": [
                    {
                        "Ref": "ECSSecurityGroup"
                    }
                ]
            },
            "Subnets": [

```



```

        {
            "Ref": "PrivateSubnet1"
        },
        {
            "Ref": "PrivateSubnet2"
        }
    ]
}
},
"LoadBalancers": [
    {
        "ContainerName": {
            "Ref": "ServiceName"
        },
        "ContainerPort": {
            "Ref": "ContainerPort"
        },
        "TargetGroupArn": {
            "Ref": "ALBTargetGroup"
        }
    }
],
"DeploymentConfiguration": {
    "MaximumPercent": 200,
    "MinimumHealthyPercent": 50,
    "DeploymentCircuitBreaker": {
        "Enable": true,
        "Rollback": true
    }
},
"EnableExecuteCommand": true,
"Tags": [
    {
        "Key": "Name",
        "Value": {
            "Fn::Sub": "${AWS::StackName}-service"
        }
    }
]
}
},
"ServiceScalingTarget": {
    "Type": "AWS::ApplicationAutoScaling::ScalableTarget",
    "Properties": {

```

```

        "MaxCapacity": {
            "Ref": "MaxCapacity"
        },
        "MinCapacity": {
            "Ref": "MinCapacity"
        },
        "ResourceId": {
            "Fn::Sub": "service/${ECSCluster}/${ECSService.Name}"
        },
        "RoleARN": {
            "Fn::Sub": "arn:aws:iam::${AWS::AccountId}:role/
aws-service-role/ecs.application-autoscaling.amazonaws.com/
AWSServiceRoleForApplicationAutoScaling_ECSService"
        },
        "ScalableDimension": "ecs:service:DesiredCount",
        "ServiceNamespace": "ecs"
    }
},
"ServiceScalingPolicy": {
    "Type": "AWS::ApplicationAutoScaling::ScalingPolicy",
    "Properties": {
        "PolicyName": {
            "Fn::Sub": "${AWS::StackName}-cpu-scaling-policy"
        },
        "PolicyType": "TargetTrackingScaling",
        "ScalingTargetId": {
            "Ref": "ServiceScalingTarget"
        },
        "TargetTrackingScalingPolicyConfiguration": {
            "PredefinedMetricSpecification": {
                "PredefinedMetricType": "ECSServiceAverageCPUUtilization"
            },
            "TargetValue": 70,
            "ScaleOutCooldown": 300,
            "ScaleInCooldown": 300
        }
    }
}
},
"Outputs": {
    "VPCId": {
        "Description": "VPC ID",
        "Value": {
            "Ref": "VPC"
        }
    }
}

```

```

    },
    "Export": {
      "Name": {
        "Fn::Sub": "${AWS::StackName}-VPC-ID"
      }
    }
  },
  "LoadBalancerURL": {
    "Description": "URL of the Application Load Balancer",
    "Value": {
      "Fn::Sub": "http://${ApplicationLoadBalancer.DNSName}"
    },
    "Export": {
      "Name": {
        "Fn::Sub": "${AWS::StackName}-ALB-URL"
      }
    }
  },
  "ECSClusterName": {
    "Description": "Name of the ECS Cluster",
    "Value": {
      "Ref": "ECSCluster"
    },
    "Export": {
      "Name": {
        "Fn::Sub": "${AWS::StackName}-ECS-Cluster"
      }
    }
  },
  "ECSServiceName": {
    "Description": "Name of the ECS Service",
    "Value": {
      "Fn::GetAtt": [
        "ECSService",
        "Name"
      ]
    },
    "Export": {
      "Name": {
        "Fn::Sub": "${AWS::StackName}-ECS-Service"
      }
    }
  },
  "PrivateSubnet1": {

```

```

        "Description": "Private Subnet 1 ID",
        "Value": {
            "Ref": "PrivateSubnet1"
        },
        "Export": {
            "Name": {
                "Fn::Sub": "${AWS::StackName}-Private-Subnet-1"
            }
        }
    },
    "PrivateSubnet2": {
        "Description": "Private Subnet 2 ID",
        "Value": {
            "Ref": "PrivateSubnet2"
        },
        "Export": {
            "Name": {
                "Fn::Sub": "${AWS::StackName}-Private-Subnet-2"
            }
        }
    }
}
}
}

```

YAML

```

AWSTemplateFormatVersion: '2010-09-09'
Description: '[AWSDocs] ECS: load-balanced-web-application'

Parameters:
  VpcCidr:
    Type: String
    Default: '10.0.0.0/16'
    Description: CIDR block for the VPC
  ContainerImage:
    Type: String
    Default: 'public.ecr.aws/ecs-sample-image/amazon-ecs-sample:latest'
    Description: Container image to use in task definition

  PublicSubnet1Cidr:
    Type: String
    Default: '10.0.1.0/24'
    Description: CIDR block for public subnet 1

```

PublicSubnet2Cidr:

Type: String

Default: '10.0.2.0/24'

Description: CIDR block for public subnet 2

PrivateSubnet1Cidr:

Type: String

Default: '10.0.3.0/24'

Description: CIDR block for private subnet 1

PrivateSubnet2Cidr:

Type: String

Default: '10.0.4.0/24'

Description: CIDR block for private subnet 2

ServiceName:

Type: String

Default: 'tutorial-app'

Description: Name of the ECS service

ContainerPort:

Type: Number

Default: 80

Description: Port on which the container listens

DesiredCount:

Type: Number

Default: 2

Description: Desired number of tasks

MinCapacity:

Type: Number

Default: 1

Description: Minimum number of tasks for auto scaling

MaxCapacity:

Type: Number

Default: 10

Description: Maximum number of tasks for auto scaling

Resources:

VPC and Networking

VPC:

```
Type: AWS::EC2::VPC
Properties:
  CidrBlock: !Ref VpcCidr
  EnableDnsHostnames: true
  EnableDnsSupport: true
  Tags:
    - Key: Name
      Value: !Sub '${AWS::StackName}-vpc'

# Internet Gateway
InternetGateway:
  Type: AWS::EC2::InternetGateway
  Properties:
    Tags:
      - Key: Name
        Value: !Sub '${AWS::StackName}-igw'

InternetGatewayAttachment:
  Type: AWS::EC2::VPCGatewayAttachment
  Properties:
    InternetGatewayId: !Ref InternetGateway
    VpcId: !Ref VPC

# Public Subnets for ALB
PublicSubnet1:
  Type: AWS::EC2::Subnet
  Properties:
    VpcId: !Ref VPC
    AvailabilityZone: !Select [0, !GetAZs '']
    CidrBlock: !Ref PublicSubnet1Cidr
    MapPublicIpOnLaunch: true
    Tags:
      - Key: Name
        Value: !Sub '${AWS::StackName}-public-subnet-1'

PublicSubnet2:
  Type: AWS::EC2::Subnet
  Properties:
    VpcId: !Ref VPC
    AvailabilityZone: !Select [1, !GetAZs '']
    CidrBlock: !Ref PublicSubnet2Cidr
    MapPublicIpOnLaunch: true
    Tags:
      - Key: Name
```

```
Value: !Sub '${AWS::StackName}-public-subnet-2'

# Private Subnets for ECS Tasks
PrivateSubnet1:
  Type: AWS::EC2::Subnet
  Properties:
    VpcId: !Ref VPC
    AvailabilityZone: !Select [0, !GetAZs '']
    CidrBlock: !Ref PrivateSubnet1Cidr
    Tags:
      - Key: Name
        Value: !Sub '${AWS::StackName}-private-subnet-1'

PrivateSubnet2:
  Type: AWS::EC2::Subnet
  Properties:
    VpcId: !Ref VPC
    AvailabilityZone: !Select [1, !GetAZs '']
    CidrBlock: !Ref PrivateSubnet2Cidr
    Tags:
      - Key: Name
        Value: !Sub '${AWS::StackName}-private-subnet-2'

# NAT Gateways for private subnet internet access
NatGateway1EIP:
  Type: AWS::EC2::EIP
  DependsOn: InternetGatewayAttachment
  Properties:
    Domain: vpc
    Tags:
      - Key: Name
        Value: !Sub '${AWS::StackName}-nat-eip-1'

NatGateway2EIP:
  Type: AWS::EC2::EIP
  DependsOn: InternetGatewayAttachment
  Properties:
    Domain: vpc
    Tags:
      - Key: Name
        Value: !Sub '${AWS::StackName}-nat-eip-2'

NatGateway1:
  Type: AWS::EC2::NatGateway
```

```
Properties:
  AllocationId: !GetAtt NatGateway1EIP.AllocationId
  SubnetId: !Ref PublicSubnet1
  Tags:
    - Key: Name
      Value: !Sub '${AWS::StackName}-nat-1'

NatGateway2:
  Type: AWS::EC2::NatGateway
  Properties:
    AllocationId: !GetAtt NatGateway2EIP.AllocationId
    SubnetId: !Ref PublicSubnet2
    Tags:
      - Key: Name
        Value: !Sub '${AWS::StackName}-nat-2'

# Route Tables
PublicRouteTable:
  Type: AWS::EC2::RouteTable
  Properties:
    VpcId: !Ref VPC
    Tags:
      - Key: Name
        Value: !Sub '${AWS::StackName}-public-routes'

DefaultPublicRoute:
  Type: AWS::EC2::Route
  DependsOn: InternetGatewayAttachment
  Properties:
    RouteTableId: !Ref PublicRouteTable
    DestinationCidrBlock: 0.0.0.0/0
    GatewayId: !Ref InternetGateway

PublicSubnet1RouteTableAssociation:
  Type: AWS::EC2::SubnetRouteTableAssociation
  Properties:
    RouteTableId: !Ref PublicRouteTable
    SubnetId: !Ref PublicSubnet1

PublicSubnet2RouteTableAssociation:
  Type: AWS::EC2::SubnetRouteTableAssociation
  Properties:
    RouteTableId: !Ref PublicRouteTable
    SubnetId: !Ref PublicSubnet2
```



```
PrivateRouteTable1:
  Type: AWS::EC2::RouteTable
  Properties:
    VpcId: !Ref VPC
    Tags:
      - Key: Name
        Value: !Sub '${AWS::StackName}-private-routes-1'

DefaultPrivateRoute1:
  Type: AWS::EC2::Route
  Properties:
    RouteTableId: !Ref PrivateRouteTable1
    DestinationCidrBlock: 0.0.0.0/0
    NatGatewayId: !Ref NatGateway1

PrivateSubnet1RouteTableAssociation:
  Type: AWS::EC2::SubnetRouteTableAssociation
  Properties:
    RouteTableId: !Ref PrivateRouteTable1
    SubnetId: !Ref PrivateSubnet1

PrivateRouteTable2:
  Type: AWS::EC2::RouteTable
  Properties:
    VpcId: !Ref VPC
    Tags:
      - Key: Name
        Value: !Sub '${AWS::StackName}-private-routes-2'

DefaultPrivateRoute2:
  Type: AWS::EC2::Route
  Properties:
    RouteTableId: !Ref PrivateRouteTable2
    DestinationCidrBlock: 0.0.0.0/0
    NatGatewayId: !Ref NatGateway2

PrivateSubnet2RouteTableAssociation:
  Type: AWS::EC2::SubnetRouteTableAssociation
  Properties:
    RouteTableId: !Ref PrivateRouteTable2
    SubnetId: !Ref PrivateSubnet2

# Security Groups
```

```

ALBSecurityGroup:
  Type: AWS::EC2::SecurityGroup
  Properties:
    GroupName: !Sub '${AWS::StackName}-alb-sg'
    GroupDescription: Security group for Application Load Balancer
    VpcId: !Ref VPC
    SecurityGroupIngress:
      - IpProtocol: tcp
        FromPort: 80
        ToPort: 80
        CidrIp: 0.0.0.0/0
        Description: Allow HTTP traffic from internet
    SecurityGroupEgress:
      - IpProtocol: -1
        CidrIp: 0.0.0.0/0
        Description: Allow all outbound traffic
    Tags:
      - Key: Name
        Value: !Sub '${AWS::StackName}-alb-sg'

ECSSecurityGroup:
  Type: AWS::EC2::SecurityGroup
  Properties:
    GroupName: !Sub '${AWS::StackName}-ecs-sg'
    GroupDescription: Security group for ECS tasks
    VpcId: !Ref VPC
    SecurityGroupIngress:
      - IpProtocol: tcp
        FromPort: !Ref ContainerPort
        ToPort: !Ref ContainerPort
        SourceSecurityGroupId: !Ref ALBSecurityGroup
        Description: Allow traffic from ALB
    SecurityGroupEgress:
      - IpProtocol: -1
        CidrIp: 0.0.0.0/0
        Description: Allow all outbound traffic
    Tags:
      - Key: Name
        Value: !Sub '${AWS::StackName}-ecs-sg'

# Application Load Balancer
ApplicationLoadBalancer:
  Type: AWS::ElasticLoadBalancingV2::LoadBalancer
  Properties:

```

```
Name: !Sub '${AWS::StackName}-alb'
Scheme: internet-facing
Type: application
Subnets:
  - !Ref PublicSubnet1
  - !Ref PublicSubnet2
SecurityGroups:
  - !Ref ALBSecurityGroup
Tags:
  - Key: Name
    Value: !Sub '${AWS::StackName}-alb'
```

```
ALBTargetGroup:
  Type: AWS::ElasticLoadBalancingV2::TargetGroup
  Properties:
    Name: !Sub '${AWS::StackName}-tg'
    Port: !Ref ContainerPort
    Protocol: HTTP
    VpcId: !Ref VPC
    TargetType: ip
    HealthCheckIntervalSeconds: 30
    HealthCheckPath: /
    HealthCheckProtocol: HTTP
    HealthCheckTimeoutSeconds: 5
    HealthyThresholdCount: 2
    UnhealthyThresholdCount: 5
    Tags:
      - Key: Name
        Value: !Sub '${AWS::StackName}-tg'
```

```
ALBListener:
  Type: AWS::ElasticLoadBalancingV2::Listener
  Properties:
    DefaultActions:
      - Type: forward
        TargetGroupArn: !Ref ALBTargetGroup
    LoadBalancerArn: !Ref ApplicationLoadBalancer
    Port: 80
    Protocol: HTTP
```

```
# ECS Cluster
ECSCluster:
  Type: AWS::ECS::Cluster
  Properties:
```

```

ClusterName: !Sub '${AWS::StackName}-cluster'
CapacityProviders:
  - FARGATE
  - FARGATE_SPOT
DefaultCapacityProviderStrategy:
  - CapacityProvider: FARGATE
    Weight: 1
  - CapacityProvider: FARGATE_SPOT
    Weight: 4
ClusterSettings:
  - Name: containerInsights
    Value: enabled
Tags:
  - Key: Name
    Value: !Sub '${AWS::StackName}-cluster'

# IAM Roles
ECSTaskExecutionRole:
  Type: AWS::IAM::Role
  Properties:
    RoleName: !Sub '${AWS::StackName}-task-execution-role'
    AssumeRolePolicyDocument:
      Version: '2012-10-17'
      Statement:
        - Effect: Allow
          Principal:
            Service: ecs-tasks.amazonaws.com
          Action: sts:AssumeRole
    ManagedPolicyArns:
      - arn:aws:iam::aws:policy/service-role/AmazonECSTaskExecutionRolePolicy
  Tags:
    - Key: Name
      Value: !Sub '${AWS::StackName}-task-execution-role'

ECSTaskRole:
  Type: AWS::IAM::Role
  Properties:
    RoleName: !Sub '${AWS::StackName}-task-role'
    AssumeRolePolicyDocument:
      Version: '2012-10-17'
      Statement:
        - Effect: Allow
          Principal:
            Service: ecs-tasks.amazonaws.com

```

```

        Action: sts:AssumeRole
    Tags:
      - Key: Name
        Value: !Sub '${AWS::StackName}-task-role'

# CloudWatch Log Group
LogGroup:
  Type: AWS::Logs::LogGroup
  Properties:
    LogGroupName: !Sub '/ecs/${AWS::StackName}'
    RetentionInDays: 7

# ECS Task Definition
TaskDefinition:
  Type: AWS::ECS::TaskDefinition
  Properties:
    Family: !Sub '${AWS::StackName}-task'
    Cpu: '256'
    Memory: '512'
    NetworkMode: awsvpc
    RequiresCompatibilities:
      - FARGATE
    ExecutionRoleArn: !GetAtt ECSTaskExecutionRole.Arn
    TaskRoleArn: !GetAtt ECSTaskRole.Arn
    ContainerDefinitions:
      - Name: !Ref ServiceName
        Image: !Ref ContainerImage
        PortMappings:
          - ContainerPort: !Ref ContainerPort
            Protocol: tcp
        Essential: true
        LogConfiguration:
          LogDriver: awslogs
          Options:
            awslogs-group: !Ref LogGroup
            awslogs-region: !Ref AWS::Region
            awslogs-stream-prefix: ecs
        HealthCheck:
          Command:
            - CMD-SHELL
            - curl -f http://localhost/ || exit 1
          Interval: 30
          Timeout: 5
          Retries: 3

```

```

        StartPeriod: 60
    Tags:
      - Key: Name
        Value: !Sub '${AWS::StackName}-task'

# ECS Service
ECSService:
  Type: AWS::ECS::Service
  DependsOn: ALBListener
  Properties:
    ServiceName: !Sub '${AWS::StackName}-service'
    Cluster: !Ref ECSCluster
    TaskDefinition: !Ref TaskDefinition
    DesiredCount: !Ref DesiredCount
    LaunchType: FARGATE
    PlatformVersion: LATEST
    NetworkConfiguration:
      AwsvpcConfiguration:
        AssignPublicIp: DISABLED
        SecurityGroups:
          - !Ref ECSSecurityGroup
        Subnets:
          - !Ref PrivateSubnet1
          - !Ref PrivateSubnet2
    LoadBalancers:
      - ContainerName: !Ref ServiceName
        ContainerPort: !Ref ContainerPort
        TargetGroupArn: !Ref ALBTargetGroup
    DeploymentConfiguration:
      MaximumPercent: 200
      MinimumHealthyPercent: 50
      DeploymentCircuitBreaker:
        Enable: true
        Rollback: true
    EnableExecuteCommand: true # For debugging
    Tags:
      - Key: Name
        Value: !Sub '${AWS::StackName}-service'

# Auto Scaling Target
ServiceScalingTarget:
  Type: AWS::ApplicationAutoScaling::ScalableTarget
  Properties:
    MaxCapacity: !Ref MaxCapacity

```

```

    MinCapacity: !Ref MinCapacity
    ResourceId: !Sub 'service/${ECSCluster}/${ECSService.Name}'
    RoleARN: !Sub 'arn:aws:iam::${AWS::AccountId}:role/
aws-service-role/ecs.application-autoscaling.amazonaws.com/
AWSServiceRoleForApplicationAutoScaling_ECSService'
    ScalableDimension: ecs:service:DesiredCount
    ServiceNamespace: ecs

# Auto Scaling Policy - CPU Utilization
ServiceScalingPolicy:
  Type: AWS::ApplicationAutoScaling::ScalingPolicy
  Properties:
    PolicyName: !Sub '${AWS::StackName}-cpu-scaling-policy'
    PolicyType: TargetTrackingScaling
    ScalingTargetId: !Ref ServiceScalingTarget
    TargetTrackingScalingPolicyConfiguration:
      PredefinedMetricSpecification:
        PredefinedMetricType: ECSServiceAverageCPUUtilization
      TargetValue: 70.0
      ScaleOutCooldown: 300
      ScaleInCooldown: 300

Outputs:
  VPCId:
    Description: VPC ID
    Value: !Ref VPC
    Export:
      Name: !Sub '${AWS::StackName}-VPC-ID'

  LoadBalancerURL:
    Description: URL of the Application Load Balancer
    Value: !Sub 'http://${ApplicationLoadBalancer.DNSName}'
    Export:
      Name: !Sub '${AWS::StackName}-ALB-URL'

  ECSClusterName:
    Description: Name of the ECS Cluster
    Value: !Ref ECSCluster
    Export:
      Name: !Sub '${AWS::StackName}-ECS-Cluster'

  ECSServiceName:
    Description: Name of the ECS Service
    Value: !GetAtt ECSService.Name

```

```

Export:
  Name: !Sub '${AWS::StackName}-ECS-Service'

PrivateSubnet1:
  Description: Private Subnet 1 ID
  Value: !Ref PrivateSubnet1
  Export:
    Name: !Sub '${AWS::StackName}-Private-Subnet-1'

PrivateSubnet2:
  Description: Private Subnet 2 ID
  Value: !Ref PrivateSubnet2
  Export:
    Name: !Sub '${AWS::StackName}-Private-Subnet-2'

```

部署已啟用 ECS Exec 的服務

您可以使用下列範本來部署已啟用 ECS Exec 的服務。該服務在叢集中執行，使用 KMS 金鑰來加密 ECS Exec 工作階段，並使用日誌組態將執行命令工作階段日誌重新導向至 Amazon S3 儲存貯體。如需詳細資訊，請參閱[使用 ECS Exec 監控 Amazon ECS 容器](#)。

JSON

```

{
  "AWSTemplateFormatVersion": "2010-09-09",
  "Description": "ECS Cluster with Fargate Service and Task Definition and ECS Exec enabled.",
  "Parameters": {
    "ClusterName": {
      "Type": "String",
      "Default": "CFNCluster",
      "Description": "Name of the ECS Cluster"
    },
    "S3BucketName": {
      "Type": "String",
      "Default": "amzn-s3-demo-bucket",
      "Description": "S3 bucket for ECS execute command logs"
    },
    "KmsKeyId": {
      "Type": "String",
      "Default": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
      "Description": "KMS Key ID for ECS execute command encryption"
    }
  }
}

```



```

    },
    "ContainerImage": {
        "Type": "String",
        "Default": "public.ecr.aws/docker/library/httpd:2.4",
        "Description": "Container image to use for the task"
    },
    "ContainerCpu": {
        "Type": "Number",
        "Default": 256,
        "AllowedValues": [256, 512, 1024, 2048, 4096],
        "Description": "CPU units for the container (256 = 0.25 vCPU)"
    },
    "ContainerMemory": {
        "Type": "Number",
        "Default": 512,
        "AllowedValues": [512, 1024, 2048, 3072, 4096, 5120, 6144, 7168, 8192],
        "Description": "Memory for the container (in MiB)"
    },
    "DesiredCount": {
        "Type": "Number",
        "Default": 1,
        "Description": "Desired count of tasks in the service"
    },
    "SecurityGroups": {
        "Type": "List<AWS::EC2::SecurityGroup::Id>",
        "Description": "Security Group IDs for the ECS Service"
    },
    "Subnets": {
        "Type": "List<AWS::EC2::Subnet::Id>",
        "Description": "Subnet IDs for the ECS Service"
    },
    "ServiceName": {
        "Type": "String",
        "Default": "cfn-service",
        "Description": "Name of the ECS service"
    },
    "TaskFamily": {
        "Type": "String",
        "Default": "task-definition-cfn",
        "Description": "Family name for the task definition"
    },
    "TaskExecutionRoleArn": {
        "Type": "String",
        "Description": "ARN of an existing IAM role for ECS task execution",

```

```

        "Default": "arn:aws:iam::111122223333:role/ecsTaskExecutionRole"
    },
    "TaskRoleArn": {
        "Type": "String",
        "Description": "ARN of an existing IAM role for ECS tasks",
        "Default": "arn:aws:iam::111122223333:role/execTaskRole"
    }
},
"Resources": {
    "ECSCluster": {
        "Type": "AWS::ECS::Cluster",
        "Properties": {
            "ClusterName": {"Ref": "ClusterName"},
            "Configuration": {
                "ExecuteCommandConfiguration": {
                    "Logging": "OVERRIDE",
                    "LogConfiguration": {
                        "S3BucketName": {"Ref": "S3BucketName"}
                    },
                    "KmsKeyId": {"Ref": "KmsKeyId"}
                }
            },
            "Tags": [
                {
                    "Key": "Environment",
                    "Value": {"Ref": "AWS::StackName"}
                }
            ]
        }
    },
    "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": {"Ref": "ContainerImage"},
    "LogConfiguration": {
        "LogDriver": "awslogs",
        "Options": {
            "mode": "non-blocking",
            "max-buffer-size": "25m",
            "awslogs-create-group": "true",
            "awslogs-group": {"Fn::Sub": "/ecs/"
${AWS::StackName}"},
            "awslogs-region": {"Ref": "AWS::Region"},
            "awslogs-stream-prefix": "ecs"
        }
    },
    "Name": "sample-fargate-app",
    "PortMappings": [
        {
            "ContainerPort": 80,
            "HostPort": 80,
            "Protocol": "tcp"
        }
    ]
}
],
"Cpu": {"Ref": "ContainerCpu"},
"ExecutionRoleArn": {"Ref": "TaskExecutionRoleArn"},
"TaskRoleArn": {"Ref": "TaskRoleArn"},
"Family": {"Ref": "TaskFamily"},
"Memory": {"Ref": "ContainerMemory"},
"NetworkMode": "awsvpc",
"RequiresCompatibilities": [
    "FARGATE"
],
"RuntimePlatform": {
    "OperatingSystemFamily": "LINUX"
},
"Tags": [
    {
        "Key": "Name",
        "Value": {"Fn::Sub": "${AWS::StackName}-TaskDefinition"}
    }
]

```

```

    ]
  }
},
"ECSService": {
  "Type": "AWS::ECS::Service",
  "Properties": {
    "ServiceName": {"Ref": "ServiceName"},
    "Cluster": {"Ref": "ECSCluster"},
    "DesiredCount": {"Ref": "DesiredCount"},
    "EnableExecuteCommand": true,
    "LaunchType": "FARGATE",
    "NetworkConfiguration": {
      "AwsVpcConfiguration": {
        "AssignPublicIp": "ENABLED",
        "SecurityGroups": {"Ref": "SecurityGroups"},
        "Subnets": {"Ref": "Subnets"}
      }
    },
    "TaskDefinition": {"Ref": "ECSTaskDefinition"},
    "Tags": [
      {
        "Key": "Name",
        "Value": {"Fn::Sub": "${AWS::StackName}-Service"}
      }
    ]
  }
}
},
"Outputs": {
  "ClusterName": {
    "Description": "The name of the ECS cluster",
    "Value": {"Ref": "ECSCluster"}
  },
  "ServiceName": {
    "Description": "The name of the ECS service",
    "Value": {"Ref": "ServiceName"}
  },
  "TaskDefinitionArn": {
    "Description": "The ARN of the task definition",
    "Value": {"Ref": "ECSTaskDefinition"}
  },
  "ClusterArn": {
    "Description": "The ARN of the ECS cluster",
    "Value": {"Fn::GetAtt": ["ECSCluster", "Arn"]}
  }
}

```

```

    },
    "StackName": {
      "Description": "The name of this stack",
      "Value": {"Ref": "AWS::StackName"}
    },
    "StackId": {
      "Description": "The unique identifier for this stack",
      "Value": {"Ref": "AWS::StackId"}
    },
    "Region": {
      "Description": "The AWS Region where the stack is deployed",
      "Value": {"Ref": "AWS::Region"}
    },
    "AccountId": {
      "Description": "The AWS Account ID",
      "Value": {"Ref": "AWS::AccountId"}
    }
  }
}

```

YAML

```

AWSTemplateFormatVersion: '2010-09-09'
Description: ECS Cluster with Fargate Service and Task Definition and ECS Exec
  enabled.
Parameters:
  ClusterName:
    Type: String
    Default: CFNCluster
    Description: Name of the ECS Cluster
  S3BucketName:
    Type: String
    Default: amzn-s3-demo-bucket
    Description: S3 bucket for ECS execute command logs
  KmsKeyId:
    Type: String
    Default: a1b2c3d4-5678-90ab-cdef-EXAMPLE11111
    Description: KMS Key ID for ECS execute command encryption
  ContainerImage:
    Type: String
    Default: public.ecr.aws/docker/library/httpd:2.4
    Description: Container image to use for the task
  ContainerCpu:

```

```

    Type: Number
    Default: 256
    AllowedValues: [256, 512, 1024, 2048, 4096]
    Description: CPU units for the container (256 = 0.25 vCPU)
ContainerMemory:
    Type: Number
    Default: 512
    AllowedValues: [512, 1024, 2048, 3072, 4096, 5120, 6144, 7168, 8192]
    Description: Memory for the container (in MiB)
DesiredCount:
    Type: Number
    Default: 1
    Description: Desired count of tasks in the service
SecurityGroups:
    Type: List<AWS::EC2::SecurityGroup::Id>
    Description: Security Group IDs for the ECS Service
Subnets:
    Type: List<AWS::EC2::Subnet::Id>
    Description: Subnet IDs for the ECS Service
ServiceName:
    Type: String
    Default: cfn-service
    Description: Name of the ECS service
TaskFamily:
    Type: String
    Default: task-definition-cfn
    Description: Family name for the task definition
TaskExecutionRoleArn:
    Type: String
    Description: ARN of an existing IAM role for ECS task execution
    Default: 'arn:aws:iam::111122223333:role/ecsTaskExecutionRole'
TaskRoleArn:
    Type: String
    Description: ARN of an existing IAM role for ECS tasks
    Default: 'arn:aws:iam::111122223333:role/execTaskRole'
Resources:
    ECSCluster:
        Type: AWS::ECS::Cluster
        Properties:
            ClusterName: !Ref ClusterName
            Configuration:
                ExecuteCommandConfiguration:
                    Logging: OVERRIDE
                    LogConfiguration:

```

```

    S3BucketName: !Ref S3BucketName
    KmsKeyId: !Ref KmsKeyId
  Tags:
    - Key: Environment
      Value: !Ref AWS::StackName
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: !Ref ContainerImage
    LogConfiguration:
      LogDriver: awslogs
      Options:
        mode: non-blocking
        max-buffer-size: 25m
        awslogs-create-group: 'true'
        awslogs-group: !Sub /ecs/${AWS::StackName}
        awslogs-region: !Ref AWS::Region
        awslogs-stream-prefix: ecs
    Name: sample-fargate-app
    PortMappings:
      - ContainerPort: 80
        HostPort: 80
        Protocol: tcp
    Cpu: !Ref ContainerCpu
    ExecutionRoleArn: !Ref TaskExecutionRoleArn
    TaskRoleArn: !Ref TaskRoleArn
    Family: !Ref TaskFamily
    Memory: !Ref ContainerMemory
    NetworkMode: awsvpc
    RequiresCompatibilities:

```

```

    - FARGATE
RuntimePlatform:
  OperatingSystemFamily: LINUX
Tags:
  - Key: Name
    Value: !Sub ${AWS::StackName}-TaskDefinition
ECSService:
  Type: AWS::ECS::Service
  Properties:
    ServiceName: !Ref ServiceName
    Cluster: !Ref ECSCluster
    DesiredCount: !Ref DesiredCount
    EnableExecuteCommand: true
    LaunchType: FARGATE
    NetworkConfiguration:
      AwsvpcConfiguration:
        AssignPublicIp: ENABLED
        SecurityGroups: !Ref SecurityGroups
        Subnets: !Ref Subnets
    TaskDefinition: !Ref ECSTaskDefinition
    Tags:
      - Key: Name
        Value: !Sub ${AWS::StackName}-Service
Outputs:
  ClusterName:
    Description: The name of the ECS cluster
    Value: !Ref ECSCluster
  ServiceName:
    Description: The name of the ECS service
    Value: !Ref ServiceName
  TaskDefinitionArn:
    Description: The ARN of the task definition
    Value: !Ref ECSTaskDefinition
  ClusterArn:
    Description: The ARN of the ECS cluster
    Value: !GetAtt ECSCluster.Arn
  StackName:
    Description: The name of this stack
    Value: !Ref AWS::StackName
  StackId:
    Description: The unique identifier for this stack
    Value: !Ref AWS::StackId
  Region:
    Description: The AWS Region where the stack is deployed

```



```
Value: !Ref AWS::Region
AccountId:
  Description: The AWS Account ID
  Value: !Ref AWS::AccountId
```

部署使用 Amazon VPC Lattice 的服務

您可以使用下列範本來開始建立具有 VPC Lattice 的 Amazon ECS 服務。您可能需要完成下列額外步驟才能設定 VPC Lattice：

- 更新安全群組的 VPC Lattice 傳入規則，以允許傳入規則 vpc-lattice 字首與連接埠 80 上的流量。
- 將服務的 VPC 與 VPC Lattice 服務網路建立關聯。
- 使用 Amazon Route 53 設定私有或公有託管區域。
- 在 VPC Lattice 服務中設定接聽程式與接聽程式規則。
- 驗證目標群組的運作狀態檢查組態。

如需有關將 VPC Lattice 與 Amazon ECS 搭配使用的詳細資訊，請參閱[使用 Amazon VPC Lattice 連接、觀察和保護您的 Amazon ECS 服務](#)。

JSON

```
{
  "AWSTemplateFormatVersion": "2010-09-09",
  "Description": "The template used to create an ECS Service with VPC Lattice.",
  "Parameters": {
    "ECSClusterName": {
      "Type": "String",
      "Default": "vpc-lattice-cluster"
    },
    "ECSServiceName": {
      "Type": "String",
      "Default": "vpc-lattice-service"
    },
    "SecurityGroupIDs": {
      "Type": "List<AWS::EC2::SecurityGroup::Id>",
      "Description": "Security Group IDs for the ECS Service"
    },
    "SubnetIDs": {
```

```

    "Type": "List<AWS::EC2::Subnet::Id>",
    "Description": "Subnet IDs for the ECS Service"
  },
  "VpcID": {
    "Type": "AWS::EC2::VPC::Id",
    "Description": "VPC ID for the resources"
  },
  "ContainerImage": {
    "Type": "String",
    "Default": "public.ecr.aws/docker/library/httpd:2.4",
    "Description": "Container image to use for the task"
  },
  "TaskCpu": {
    "Type": "Number",
    "Default": 256,
    "AllowedValues": [256, 512, 1024, 2048, 4096],
    "Description": "CPU units for the task"
  },
  "TaskMemory": {
    "Type": "Number",
    "Default": 512,
    "AllowedValues": [512, 1024, 2048, 4096, 8192, 16384],
    "Description": "Memory (in MiB) for the task"
  },
  "LogGroupName": {
    "Type": "String",
    "Default": "/ecs/vpc-lattice-task",
    "Description": "CloudWatch Log Group name"
  },
  "EnableContainerInsights": {
    "Type": "String",
    "Default": "enabled",
    "AllowedValues": ["enabled", "disabled"],
    "Description": "Enable or disable CloudWatch Container Insights for the cluster"
  }
},
"Resources": {
  "ECSCluster": {
    "Type": "AWS::ECS::Cluster",
    "Properties": {
      "ClusterName": {"Ref": "ECSClusterName"},
      "ClusterSettings": [
        {
          "Name": "containerInsights",

```

```

    "Value": {"Ref": "EnableContainerInsights"}
  }
],
"Tags": [
  {
    "Key": "Name",
    "Value": {"Ref": "ECSClusterName"}
  }
]
},
"ECSTaskExecutionRole": {
  "Type": "AWS::IAM::Role",
  "Properties": {
    "AssumeRolePolicyDocument": {
      "Version": "2012-10-17",
      "Statement": [
        {
          "Effect": "Allow",
          "Principal": {
            "Service": "ecs-tasks.amazonaws.com"
          },
          "Action": "sts:AssumeRole"
        }
      ]
    },
    "ManagedPolicyArns": [
      "arn:aws:iam::aws:policy/service-role/AmazonECSTaskExecutionRolePolicy"
    ]
  }
},
"TaskLogGroup": {
  "Type": "AWS::Logs::LogGroup",
  "DeletionPolicy": "Retain",
  "UpdateReplacePolicy": "Retain",
  "Properties": {
    "LogGroupName": {"Ref": "LogGroupName"},
    "RetentionInDays": 30
  }
},
"VpcLatticeTaskDefinition": {
  "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": {"Ref": "ContainerImage"},
  "LogConfiguration": {
    "LogDriver": "awslogs",
    "Options": {
      "mode": "non-blocking",
      "max-buffer-size": "25m",
      "awslogs-create-group": "true",
      "awslogs-group": {"Ref": "LogGroupName"},
      "awslogs-region": {"Ref": "AWS::Region"},
      "awslogs-stream-prefix": "ecs"
    }
  },
  "Name": "vpc-lattice-container",
  "PortMappings": [
    {
      "ContainerPort": 80,
      "HostPort": 80,
      "Protocol": "tcp",
      "Name": "vpc-lattice-port"
    }
  ],
  "Cpu": {"Ref": "TaskCpu"},
  "ExecutionRoleArn": {"Fn::GetAtt": ["ECSTaskExecutionRole", "Arn"]},
  "Family": "vpc-lattice-task-definition",
  "Memory": {"Ref": "TaskMemory"},
  "NetworkMode": "awsvpc",
  "RequiresCompatibilities": [
    "FARGATE"
  ]
}

```

```

    ],
    "RuntimePlatform": {
      "OperatingSystemFamily": "LINUX"
    }
  },
  "ECSService": {
    "Type": "AWS::ECS::Service",
    "Properties": {
      "Cluster": {"Ref": "ECSCluster"},
      "TaskDefinition": {"Ref": "VpcLatticeTaskDefinition"},
      "LaunchType": "FARGATE",
      "ServiceName": {"Ref": "ECSServiceName"},
      "SchedulingStrategy": "REPLICA",
      "DesiredCount": 2,
      "AvailabilityZoneRebalancing": "ENABLED",
      "NetworkConfiguration": {
        "AwsvpcConfiguration": {
          "AssignPublicIp": "ENABLED",
          "SecurityGroups": {
            "Ref": "SecurityGroupIDs"
          },
          "Subnets": {
            "Ref": "SubnetIDs"
          }
        }
      },
      "PlatformVersion": "LATEST",
      "VpcLatticeConfigurations": [
        {
          "RoleArn": "arn:aws:iam::111122223333:role/ecsInfrastructureRole",
          "PortName": "vpc-lattice-port",
          "TargetGroupArn": {
            "Ref": "TargetGroup1"
          }
        }
      ],
      "DeploymentConfiguration": {
        "DeploymentCircuitBreaker": {
          "Enable": true,
          "Rollback": true
        },
        "MaximumPercent": 200,
        "MinimumHealthyPercent": 100
      }
    }
  }
}

```

```

    },
    "DeploymentController": {
      "Type": "ECS"
    },
    "ServiceConnectConfiguration": {
      "Enabled": false
    },
    "Tags": [],
    "EnableECSTags": true
  }
},
"TargetGroup1": {
  "Type": "AWS::VpcLattice::TargetGroup",
  "Properties": {
    "Type": "IP",
    "Name": "first-target-group",
    "Config": {
      "Port": 80,
      "Protocol": "HTTP",
      "VpcIdentifier": {"Ref": "VpcID"},
      "HealthCheck": {
        "Enabled": true,
        "Path": "/"
      }
    }
  },
  "Tags": [
    {
      "Key": "ecs-application-networking/ServiceName",
      "Value": {"Ref": "ECSServiceName"}
    },
    {
      "Key": "ecs-application-networking/ClusterName",
      "Value": {"Ref": "ECSClusterName"}
    },
    {
      "Key": "ecs-application-networking/TaskDefinition",
      "Value": {"Ref": "VpcLatticeTaskDefinition"}
    },
    {
      "Key": "ecs-application-networking/VpcId",
      "Value": {"Ref": "VpcID"}
    }
  ]
}

```

```

    }
  },
  "Outputs": {
    "ClusterName": {
      "Description": "The cluster used to create the service.",
      "Value": {
        "Ref": "ECSCluster"
      }
    },
    "ClusterArn": {
      "Description": "The ARN of the ECS cluster",
      "Value": {
        "Fn::GetAtt": ["ECSCluster", "Arn"]
      }
    },
    "ECSService": {
      "Description": "The created service.",
      "Value": {
        "Ref": "ECSService"
      }
    },
    "TaskDefinitionArn": {
      "Description": "The ARN of the task definition",
      "Value": {
        "Ref": "VpcLatticeTaskDefinition"
      }
    }
  }
}

```

YAML

```

AWSTemplateFormatVersion: '2010-09-09'
Description: The template used to create an ECS Service with VPC Lattice.

Parameters:
  ECSClusterName:
    Type: String
    Default: vpc-lattice-cluster
  ECSServiceName:
    Type: String
    Default: vpc-lattice-service
  SecurityGroupIDs:

```

```

    Type: List<AWS::EC2::SecurityGroup::Id>
    Description: Security Group IDs for the ECS Service
SubnetIDs:
    Type: List<AWS::EC2::Subnet::Id>
    Description: Subnet IDs for the ECS Service
VpcID:
    Type: AWS::EC2::VPC::Id
    Description: VPC ID for the resources
ContainerImage:
    Type: String
    Default: public.ecr.aws/docker/library/httpd:2.4
    Description: Container image to use for the task
TaskCpu:
    Type: Number
    Default: 256
    AllowedValues: [256, 512, 1024, 2048, 4096]
    Description: CPU units for the task
TaskMemory:
    Type: Number
    Default: 512
    AllowedValues: [512, 1024, 2048, 4096, 8192, 16384]
    Description: Memory (in MiB) for the task
LogGroupName:
    Type: String
    Default: /ecs/vpc-lattice-task
    Description: CloudWatch Log Group name
EnableContainerInsights:
    Type: String
    Default: 'enhanced'
    AllowedValues: ['enabled', 'disabled', 'enhanced']
    Description: Enable or disable CloudWatch Container Insights for the cluster

Resources:
  # ECS Cluster
  ECSCluster:
    Type: AWS::ECS::Cluster
    Properties:
      ClusterName: !Ref ECSClusterName
      ClusterSettings:
        - Name: containerInsights
          Value: !Ref EnableContainerInsights
      Tags:
        - Key: Name
          Value: !Ref ECSClusterName

```



```

# IAM Roles
ECSTaskExecutionRole:
  Type: AWS::IAM::Role
  Properties:
    AssumeRolePolicyDocument:
      Version: '2012-10-17'
      Statement:
        - Effect: Allow
          Principal:
            Service: ecs-tasks.amazonaws.com
          Action: sts:AssumeRole
    ManagedPolicyArns:
      - arn:aws:iam::aws:policy/service-role/AmazonECSTaskExecutionRolePolicy

# CloudWatch Logs
TaskLogGroup:
  Type: AWS::Logs::LogGroup
  DeletionPolicy: Retain
  UpdateReplacePolicy: Retain
  Properties:
    LogGroupName: !Ref LogGroupName
    RetentionInDays: 30

# Task Definition
VpcLatticeTaskDefinition:
  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: !Ref ContainerImage
      LogConfiguration:

```

```

    LogDriver: awslogs
    Options:
      mode: non-blocking
      max-buffer-size: 25m
      awslogs-create-group: 'true'
      awslogs-group: !Ref LogGroupName
      awslogs-region: !Ref 'AWS::Region'
      awslogs-stream-prefix: ecs
    Name: vpc-lattice-container
    PortMappings:
      - ContainerPort: 80
        HostPort: 80
        Protocol: tcp
        Name: vpc-lattice-port
    Cpu: !Ref TaskCpu
    ExecutionRoleArn: !GetAtt ECSTaskExecutionRole.Arn
    Family: vpc-lattice-task-definition
    Memory: !Ref TaskMemory
    NetworkMode: awsvpc
    RequiresCompatibilities:
      - FARGATE
    RuntimePlatform:
      OperatingSystemFamily: LINUX

ECSService:
  Type: AWS::ECS::Service
  Properties:
    Cluster: !Ref ECSCluster
    TaskDefinition: !Ref VpcLatticeTaskDefinition
    LaunchType: FARGATE
    ServiceName: !Ref ECSServiceName
    SchedulingStrategy: REPLICA
    DesiredCount: 2
    AvailabilityZoneRebalancing: ENABLED
    NetworkConfiguration:
      AwsvpcConfiguration:
        AssignPublicIp: ENABLED
        SecurityGroups: !Ref SecurityGroupIDs
        Subnets: !Ref SubnetIDs
    PlatformVersion: LATEST
    VpcLatticeConfigurations:
      - RoleArn: arn:aws:iam::111122223333:role/ecsInfrastructureRole
        PortName: vpc-lattice-port
        TargetGroupArn: !Ref TargetGroup1

```

```
DeploymentConfiguration:
  DeploymentCircuitBreaker:
    Enable: true
    Rollback: true
    MaximumPercent: 200
    MinimumHealthyPercent: 100
  DeploymentController:
    Type: ECS
  ServiceConnectConfiguration:
    Enabled: false
  Tags: []
  EnableECSManagedTags: true
```

```
TargetGroup1:
  Type: AWS::VpcLattice::TargetGroup
  Properties:
    Type: IP
    Name: first-target-group
    Config:
      Port: 80
      Protocol: HTTP
      VpcIdentifier: !Ref VpcID
      HealthCheck:
        Enabled: true
        Path: /
    Tags:
      - Key: ecs-application-networking/ServiceName
        Value: !Ref ECSServiceName
      - Key: ecs-application-networking/ClusterName
        Value: !Ref ECSClusterName
      - Key: ecs-application-networking/TaskDefinition
        Value: !Ref VpcLatticeTaskDefinition
      - Key: ecs-application-networking/VpcId
        Value: !Ref VpcID
```

```
Outputs:
  ClusterName:
    Description: The cluster used to create the service.
    Value: !Ref ECSCluster
  ClusterArn:
    Description: The ARN of the ECS cluster
    Value: !GetAtt ECSCluster.Arn
  ECSService:
    Description: The created service.
```

```
Value: !Ref ECSService
TaskDefinitionArn:
  Description: The ARN of the task definition
  Value: !Ref VpcLatticeTaskDefinition
```

部署具有磁碟區組態的服務

下列範本在服務定義中包含了磁碟區組態。Amazon ECS 支援在啟動時使用磁碟區組態來設定下列資料磁碟區：Amazon EBS 磁碟區。如需 Amazon EBS 磁碟區的詳細資訊，請參閱 [將 Amazon EBS 磁碟區與 Amazon ECS 搭配使用](#)。

JSON

```
{
  "AWSTemplateFormatVersion": "2010-09-09",
  "Description": "The template used to create an ECS Service that includes a
volume configuration. The configuration is used to create Amazon EBS volumes for
attachment to the tasks. One volume is attached per task.",
  "Parameters": {
    "ECSClusterName": {
      "Type": "String",
      "Default": "volume-config-cluster",
      "Description": "Name of the ECS cluster"
    },
    "SecurityGroupIDs": {
      "Type": "List<AWS::EC2::SecurityGroup::Id>",
      "Description": "Security Group IDs for the ECS Service"
    },
    "SubnetIDs": {
      "Type": "List<AWS::EC2::Subnet::Id>",
      "Description": "Subnet IDs for the ECS Service"
    },
    "InfrastructureRoleArn": {
      "Type": "String",
      "Description": "ARN of the IAM role that ECS will use to manage EBS volumes"
    },
    "ContainerImage": {
      "Type": "String",
      "Default": "public.ecr.aws/nginx/nginx:latest",
      "Description": "Container image to use for the task"
    },
    "TaskCpu": {
```

```

    "Type": "String",
    "Default": "2048",
    "Description": "CPU units for the task"
  },
  "TaskMemory": {
    "Type": "String",
    "Default": "4096",
    "Description": "Memory (in MiB) for the task"
  },
  "VolumeSize": {
    "Type": "String",
    "Default": "10",
    "Description": "Size of the EBS volume in GiB"
  },
  "VolumeType": {
    "Type": "String",
    "Default": "gp3",
    "AllowedValues": ["gp2", "gp3", "io1", "io2", "st1", "sc1", "standard"],
    "Description": "EBS volume type"
  },
  "VolumeIops": {
    "Type": "String",
    "Default": "3000",
    "Description": "IOPS for the EBS volume (required for io1, io2, and gp3)"
  },
  "VolumeThroughput": {
    "Type": "String",
    "Default": "125",
    "Description": "Throughput for the EBS volume (only for gp3)"
  },
  "FilesystemType": {
    "Type": "String",
    "Default": "xfs",
    "AllowedValues": ["xfs", "ext4"],
    "Description": "Filesystem type for the EBS volume"
  },
  "EnableContainerInsights": {
    "Type": "String",
    "Default": "enhanced",
    "AllowedValues": ["enabled", "disabled", "enhanced"],
    "Description": "Enable or disable CloudWatch Container Insights for the
cluster"
  }
},

```

```

"Resources": {
  "ECSCluster": {
    "Type": "AWS::ECS::Cluster",
    "Properties": {
      "ClusterName": {"Ref": "ECSClusterName"},
      "ClusterSettings": [
        {
          "Name": "containerInsights",
          "Value": {"Ref": "EnableContainerInsights"}
        }
      ],
      "Tags": [
        {
          "Key": "Name",
          "Value": {"Ref": "ECSClusterName"}
        }
      ]
    }
  },
  "ECSTaskExecutionRole": {
    "Type": "AWS::IAM::Role",
    "Properties": {
      "AssumeRolePolicyDocument": {
        "Version": "2012-10-17",
        "Statement": [
          {
            "Effect": "Allow",
            "Principal": {
              "Service": "ecs-tasks.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
          }
        ]
      },
      "ManagedPolicyArns": [
        "arn:aws:iam::aws:policy/service-role/AmazonECSTaskExecutionRolePolicy"
      ]
    }
  },
  "EBSTaskDefinition": {
    "Type": "AWS::ECS::TaskDefinition",
    "Properties": {
      "Family": "ebs-task-attach-task-def",
      "ExecutionRoleArn": {"Fn::GetAtt": ["ECSTaskExecutionRole", "Arn"]}
    }
  }
}

```

```

    "NetworkMode": "awsvpc",
    "RequiresCompatibilities": [
      "EC2",
      "FARGATE"
    ],
    "Cpu": {"Ref": "TaskCpu"},
    "Memory": {"Ref": "TaskMemory"},
    "ContainerDefinitions": [
      {
        "Name": "nginx",
        "Image": {"Ref": "ContainerImage"},
        "Essential": true,
        "PortMappings": [
          {
            "Name": "nginx-80-tcp",
            "ContainerPort": 80,
            "HostPort": 80,
            "Protocol": "tcp",
            "AppProtocol": "http"
          }
        ],
        "MountPoints": [
          {
            "SourceVolume": "ebs-vol",
            "ContainerPath": "/foo-container-path",
            "ReadOnly": false
          }
        ]
      }
    ],
    "Volumes": [
      {
        "Name": "ebs-vol",
        "ConfiguredAtLaunch": true
      }
    ]
  },
  "ECSService": {
    "Type": "AWS::ECS::Service",
    "Properties": {
      "Cluster": {"Ref": "ECSCluster"},
      "TaskDefinition": {"Ref": "EBSTaskDefinition"},
      "LaunchType": "FARGATE",

```

```

    "ServiceName": "ebs",
    "SchedulingStrategy": "REPLICA",
    "DesiredCount": 1,
    "NetworkConfiguration": {
      "AwsvpcConfiguration": {
        "AssignPublicIp": "ENABLED",
        "SecurityGroups": {"Ref": "SecurityGroupIDs"},
        "Subnets": {"Ref": "SubnetIDs"}
      }
    },
    "PlatformVersion": "LATEST",
    "DeploymentConfiguration": {
      "MaximumPercent": 200,
      "MinimumHealthyPercent": 100,
      "DeploymentCircuitBreaker": {
        "Enable": true,
        "Rollback": true
      }
    },
    "DeploymentController": {
      "Type": "ECS"
    },
    "Tags": [],
    "EnableECSManagedTags": true,
    "VolumeConfigurations": [
      {
        "Name": "ebs-vol",
        "ManagedEBSVolume": {
          "RoleArn": {"Ref": "InfrastructureRoleArn"},
          "VolumeType": {"Ref": "VolumeType"},
          "Iops": {"Ref": "VolumeIops"},
          "Throughput": {"Ref": "VolumeThroughput"},
          "SizeInGiB": {"Ref": "VolumeSize"},
          "FilesystemType": {"Ref": "FilesystemType"},
          "TagSpecifications": [
            {
              "ResourceType": "volume",
              "PropagateTags": "TASK_DEFINITION"
            }
          ]
        }
      }
    ]
  }
}

```



```

    }
  },
  "Outputs": {
    "ClusterName": {
      "Description": "The cluster used to create the service.",
      "Value": {"Ref": "ECSCluster"}
    },
    "ClusterArn": {
      "Description": "The ARN of the ECS cluster",
      "Value": {"Fn::GetAtt": ["ECSCluster", "Arn"]}
    },
    "ECSService": {
      "Description": "The created service.",
      "Value": {"Ref": "ECSService"}
    },
    "TaskDefinitionArn": {
      "Description": "The ARN of the task definition",
      "Value": {"Ref": "EBSTaskDefinition"}
    }
  }
}

```

YAML

```

AWSTemplateFormatVersion: 2010-09-09
Description: The template used to create an ECS Service that includes a volume
  configuration. The configuration is used to create Amazon EBS volumes for
  attachment to the tasks. One volume is attached per task.
Parameters:
  ECSClusterName:
    Type: String
    Default: volume-config-cluster
    Description: Name of the ECS cluster

  SecurityGroupIDs:
    Type: List<AWS::EC2::SecurityGroup::Id>
    Description: Security Group IDs for the ECS Service

  SubnetIDs:
    Type: List<AWS::EC2::Subnet::Id>
    Description: Subnet IDs for the ECS Service

  InfrastructureRoleArn:

```

Type: String
Description: ARN of the IAM role that ECS will use to manage EBS volumes

ContainerImage:

Type: String
Default: public.ecr.aws/nginx/nginx:latest
Description: Container image to use for the task

TaskCpu:

Type: String
Default: "2048"
Description: CPU units for the task

TaskMemory:

Type: String
Default: "4096"
Description: Memory (in MiB) for the task

VolumeSize:

Type: String
Default: "10"
Description: Size of the EBS volume in GiB

VolumeType:

Type: String
Default: gp3
AllowedValues: [gp2, gp3, io1, io2, st1, sc1, standard]
Description: EBS volume type

VolumeIops:

Type: String
Default: "3000"
Description: IOPS for the EBS volume (required for io1, io2, and gp3)

VolumeThroughput:

Type: String
Default: "125"
Description: Throughput for the EBS volume (only for gp3)

FilesystemType:

Type: String
Default: xfs
AllowedValues: [xfs, ext4]
Description: Filesystem type for the EBS volume

```
EnableContainerInsights:
  Type: String
  Default: 'enhanced'
  AllowedValues: ['enabled', 'disabled', 'enhanced']
  Description: Enable or disable CloudWatch Container Insights for the cluster
```

Resources:

```
# ECS Cluster
```

```
ECSCluster:
```

```
  Type: AWS::ECS::Cluster
```

```
  Properties:
```

```
    ClusterName: !Ref ECSClusterName
```

```
    ClusterSettings:
```

- Name: containerInsights
- Value: !Ref EnableContainerInsights

```
    Tags:
```

- Key: Name
- Value: !Ref ECSClusterName

```
# IAM Role for Task Execution
```

```
ECSTaskExecutionRole:
```

```
  Type: AWS::IAM::Role
```

```
  Properties:
```

```
    AssumeRolePolicyDocument:
```

```
      Version: '2012-10-17'
```

```
      Statement:
```

- Effect: Allow
- Principal:
- Service: ecs-tasks.amazonaws.com
- Action: sts:AssumeRole

```
    ManagedPolicyArns:
```

- arn:aws:iam::aws:policy/service-role/AmazonECSTaskExecutionRolePolicy

```
# Task Definition
```

```
EBSTaskDefinition:
```

```
  Type: AWS::ECS::TaskDefinition
```

```
  Properties:
```

```
    Family: ebs-task-attach-task-def
```

```
    ExecutionRoleArn: !GetAtt ECSTaskExecutionRole.Arn
```

```
    NetworkMode: awsvpc
```

```
    RequiresCompatibilities:
```

- EC2
- FARGATE

```
Cpu: !Ref TaskCpu
Memory: !Ref TaskMemory
ContainerDefinitions:
  - Name: nginx
    Image: !Ref ContainerImage
    Essential: true
    PortMappings:
      - Name: nginx-80-tcp
        ContainerPort: 80
        HostPort: 80
        Protocol: tcp
        AppProtocol: http
    MountPoints:
      - SourceVolume: ebs-vol
        ContainerPath: /foo-container-path
        ReadOnly: false
Volumes:
  - Name: ebs-vol
    ConfiguredAtLaunch: true
```

```
ECSService:
  Type: AWS::ECS::Service
  Properties:
    Cluster: !Ref ECSCluster
    TaskDefinition: !Ref EBSTaskDefinition
    LaunchType: FARGATE
    ServiceName: ebs
    SchedulingStrategy: REPLICA
    DesiredCount: 1
    NetworkConfiguration:
      AwsvpcConfiguration:
        AssignPublicIp: ENABLED
        SecurityGroups: !Ref SecurityGroupIDs
        Subnets: !Ref SubnetIDs
    PlatformVersion: LATEST
    DeploymentConfiguration:
      MaximumPercent: 200
      MinimumHealthyPercent: 100
      DeploymentCircuitBreaker:
        Enable: true
        Rollback: true
    DeploymentController:
      Type: ECS
  Tags: []
```

```

    EnableECSManagedTags: true
    VolumeConfigurations:
      - Name: ebs-vol
        ManagedEBSVolume:
          RoleArn: !Ref InfrastructureRoleArn
          VolumeType: !Ref VolumeType
          Iops: !Ref VolumeIops
          Throughput: !Ref VolumeThroughput
          SizeInGiB: !Ref VolumeSize
          FilesystemType: !Ref FilesystemType
          TagSpecifications:
            - ResourceType: volume
              PropagateTags: TASK_DEFINITION

Outputs:
  ClusterName:
    Description: The cluster used to create the service.
    Value: !Ref ECSCluster
  ClusterArn:
    Description: The ARN of the ECS cluster
    Value: !GetAtt ECSCluster.Arn
  ECSService:
    Description: The created service.
    Value: !Ref ECSService
  TaskDefinitionArn:
    Description: The ARN of the task definition
    Value: !Ref EBSTaskDefinition

```

部署具有容量提供者的服務

下列範本定義的服務使用容量提供者來請求 AL2023 容量以供執行。容器會在 AL2023 執行個體上線時啟動到這些執行個體上。

JSON

```

{
  "AWSTemplateFormatVersion": "2010-09-09",
  "Description": "An example service that deploys in AWS VPC networking mode on EC2 capacity. Service uses a capacity provider to request EC2 instances to run on. Service runs with networking in private subnets, but still accessible to the internet via a load balancer hosted in public subnets.",
  "Parameters": {

```

```

    "VpcId": {
      "Type": "String",
      "Description": "The VPC that the service is running inside of"
    },
    "PublicSubnetIds": {
      "Type": "List<AWS::EC2::Subnet::Id>",
      "Description": "List of public subnet ID's to put the load balancer in"
    },
    "PrivateSubnetIds": {
      "Type": "List<AWS::EC2::Subnet::Id>",
      "Description": "List of private subnet ID's that the AWS VPC tasks are in"
    },
    "ClusterName": {
      "Type": "String",
      "Description": "The name of the ECS cluster into which to launch
capacity."
    },
    "ECSTaskExecutionRole": {
      "Type": "String",
      "Description": "The role used to start up an ECS task"
    },
    "CapacityProvider": {
      "Type": "String",
      "Description": "The cluster capacity provider that the service should use
to request capacity when it wants to start up a task"
    },
    "ServiceName": {
      "Type": "String",
      "Default": "web",
      "Description": "A name for the service"
    },
    "ImageUrl": {
      "Type": "String",
      "Default": "public.ecr.aws/docker/library/nginx:latest",
      "Description": "The url of a docker image that contains the application
process that will handle the traffic for this service"
    },
    "ContainerCpu": {
      "Type": "Number",
      "Default": 256,
      "Description": "How much CPU to give the container. 1024 is 1 CPU"
    },
    "ContainerMemory": {
      "Type": "Number",

```

```

        "Default": 512,
        "Description": "How much memory in megabytes to give the container"
    },
    "ContainerPort": {
        "Type": "Number",
        "Default": 80,
        "Description": "What port that the application expects traffic on"
    },
    "DesiredCount": {
        "Type": "Number",
        "Default": 2,
        "Description": "How many copies of the service task to run"
    }
},
"Resources": {
    "TaskDefinition": {
        "Type": "AWS::ECS::TaskDefinition",
        "Properties": {
            "Family": {
                "Ref": "ServiceName"
            },
            "Cpu": {
                "Ref": "ContainerCpu"
            },
            "Memory": {
                "Ref": "ContainerMemory"
            },
            "NetworkMode": "awsvpc",
            "RequiresCompatibilities": [
                "EC2"
            ],
            "ExecutionRoleArn": {
                "Ref": "ECSTaskExecutionRole"
            },
            "ContainerDefinitions": [
                {
                    "Name": {
                        "Ref": "ServiceName"
                    },
                    "Cpu": {
                        "Ref": "ContainerCpu"
                    },
                    "Memory": {
                        "Ref": "ContainerMemory"
                    }
                }
            ]
        }
    }
}

```

```

    },
    "Image": {
        "Ref": "ImageUrl"
    },
    "PortMappings": [
        {
            "ContainerPort": {
                "Ref": "ContainerPort"
            },
            "HostPort": {
                "Ref": "ContainerPort"
            }
        }
    ],
    "LogConfiguration": {
        "LogDriver": "awslogs",
        "Options": {
            "mode": "non-blocking",
            "max-buffer-size": "25m",
            "awslogs-group": {
                "Ref": "LogGroup"
            },
            "awslogs-region": {
                "Ref": "AWS::Region"
            },
            "awslogs-stream-prefix": {
                "Ref": "ServiceName"
            }
        }
    }
}

]
}

},
"Service": {
    "Type": "AWS::ECS::Service",
    "DependsOn": "PublicLoadBalancerListener",
    "Properties": {
        "ServiceName": {
            "Ref": "ServiceName"
        },
        "Cluster": {
            "Ref": "ClusterName"
        }
    },

```



```
    "PlacementStrategies": [
      {
        "Field": "attribute:ecs.availability-zone",
        "Type": "spread"
      },
      {
        "Field": "cpu",
        "Type": "binpack"
      }
    ],
    "CapacityProviderStrategy": [
      {
        "Base": 0,
        "CapacityProvider": {
          "Ref": "CapacityProvider"
        },
        "Weight": 1
      }
    ],
    "NetworkConfiguration": {
      "AwsvpcConfiguration": {
        "SecurityGroups": [
          {
            "Ref": "ServiceSecurityGroup"
          }
        ],
        "Subnets": {
          "Ref": "PrivateSubnetIds"
        }
      }
    },
    "DeploymentConfiguration": {
      "MaximumPercent": 200,
      "MinimumHealthyPercent": 75
    },
    "DesiredCount": {
      "Ref": "DesiredCount"
    },
    "TaskDefinition": {
      "Ref": "TaskDefinition"
    },
    "LoadBalancers": [
      {
        "ContainerName": {
```

```

        "Ref": "ServiceName"
    },
    "ContainerPort": {
        "Ref": "ContainerPort"
    },
    "TargetGroupArn": {
        "Ref": "ServiceTargetGroup"
    }
}
]
}
},
"ServiceSecurityGroup": {
    "Type": "AWS::EC2::SecurityGroup",
    "Properties": {
        "GroupDescription": "Security group for service",
        "VpcId": {
            "Ref": "VpcId"
        }
    }
},
"ServiceTargetGroup": {
    "Type": "AWS::ElasticLoadBalancingV2::TargetGroup",
    "Properties": {
        "HealthCheckIntervalSeconds": 6,
        "HealthCheckPath": "/",
        "HealthCheckProtocol": "HTTP",
        "HealthCheckTimeoutSeconds": 5,
        "HealthyThresholdCount": 2,
        "TargetType": "ip",
        "Port": {
            "Ref": "ContainerPort"
        },
        "Protocol": "HTTP",
        "UnhealthyThresholdCount": 10,
        "VpcId": {
            "Ref": "VpcId"
        },
        "TargetGroupAttributes": [
            {
                "Key": "deregistration_delay.timeout_seconds",
                "Value": 0
            }
        ]
    }
}
]

```

```

    }
  },
  "PublicLoadBalancerSG": {
    "Type": "AWS::EC2::SecurityGroup",
    "Properties": {
      "GroupDescription": "Access to the public facing load balancer",
      "VpcId": {
        "Ref": "VpcId"
      },
      "SecurityGroupIngress": [
        {
          "CidrIp": "0.0.0.0/0",
          "IpProtocol": -1
        }
      ]
    }
  },
  "PublicLoadBalancer": {
    "Type": "AWS::ElasticLoadBalancingV2::LoadBalancer",
    "Properties": {
      "Scheme": "internet-facing",
      "LoadBalancerAttributes": [
        {
          "Key": "idle_timeout.timeout_seconds",
          "Value": "30"
        }
      ],
      "Subnets": {
        "Ref": "PublicSubnetIds"
      },
      "SecurityGroups": [
        {
          "Ref": "PublicLoadBalancerSG"
        }
      ]
    }
  },
  "PublicLoadBalancerListener": {
    "Type": "AWS::ElasticLoadBalancingV2::Listener",
    "Properties": {
      "DefaultActions": [
        {
          "Type": "forward",
          "ForwardConfig": {

```

```

        "TargetGroups": [
            {
                "TargetGroupArn": {
                    "Ref": "ServiceTargetGroup"
                },
                "Weight": 100
            }
        ]
    },
    "LoadBalancerArn": {
        "Ref": "PublicLoadBalancer"
    },
    "Port": 80,
    "Protocol": "HTTP"
},
"ServiceIngressfromLoadBalancer": {
    "Type": "AWS::EC2::SecurityGroupIngress",
    "Properties": {
        "Description": "Ingress from the public ALB",
        "GroupId": {
            "Ref": "ServiceSecurityGroup"
        },
        "IpProtocol": -1,
        "SourceSecurityGroupId": {
            "Ref": "PublicLoadBalancerSG"
        }
    }
},
"LogGroup": {
    "Type": "AWS::Logs::LogGroup"
}
}
}

```

YAML

```

AWSTemplateFormatVersion: '2010-09-09'
Description: >-
  An example service that deploys in AWS VPC networking mode on EC2 capacity.
  Service uses a capacity provider to request EC2 instances to run on. Service

```

runs with networking in private subnets, but still accessible to the internet via a load balancer hosted in public subnets.

Parameters:**VpcId:**

Type: String

Description: The VPC that the service is running inside of

PublicSubnetIds:

Type: 'List<AWS::EC2::Subnet::Id>'

Description: List of public subnet ID's to put the load balancer in

PrivateSubnetIds:

Type: 'List<AWS::EC2::Subnet::Id>'

Description: List of private subnet ID's that the AWS VPC tasks are in

ClusterName:

Type: String

Description: The name of the ECS cluster into which to launch capacity.

ECSTaskExecutionRole:

Type: String

Description: The role used to start up an ECS task

CapacityProvider:

Type: String

Description: >-

The cluster capacity provider that the service should use to request capacity when it wants to start up a task

ServiceName:

Type: String

Default: web

Description: A name for the service

ImageUrl:

Type: String

Default: 'public.ecr.aws/docker/library/nginx:latest'

Description: >-

The url of a docker image that contains the application process that will handle the traffic for this service

ContainerCpu:

Type: Number

Default: 256

Description: How much CPU to give the container. 1024 is 1 CPU

ContainerMemory:

Type: Number

Default: 512

Description: How much memory in megabytes to give the container

ContainerPort:

Type: Number

Default: 80

```

    Description: What port that the application expects traffic on
  DesiredCount:
    Type: Number
    Default: 2
    Description: How many copies of the service task to run
  Resources:
    TaskDefinition:
      Type: 'AWS::ECS::TaskDefinition'
      Properties:
        Family: !Ref ServiceName
        Cpu: !Ref ContainerCpu
        Memory: !Ref ContainerMemory
        NetworkMode: awsvpc
        RequiresCompatibilities:
          - EC2
        ExecutionRoleArn: !Ref ECSTaskExecutionRole
        ContainerDefinitions:
          - Name: !Ref ServiceName
            Cpu: !Ref ContainerCpu
            Memory: !Ref ContainerMemory
            Image: !Ref ImageUrl
            PortMappings:
              - ContainerPort: !Ref ContainerPort
                HostPort: !Ref ContainerPort
            LogConfiguration:
              LogDriver: awslogs
              Options:
                mode: non-blocking
                max-buffer-size: 25m
                awslogs-group: !Ref LogGroup
                awslogs-region: !Ref AWS::Region
                awslogs-stream-prefix: !Ref ServiceName
    Service:
      Type: AWS::ECS::Service
      DependsOn: PublicLoadBalancerListener
      Properties:
        ServiceName: !Ref ServiceName
        Cluster: !Ref ClusterName
        PlacementStrategies:
          - Field: 'attribute:ecs.availability-zone'
            Type: spread
          - Field: cpu
            Type: binpack
        CapacityProviderStrategy:

```

```

    - Base: 0
      CapacityProvider: !Ref CapacityProvider
      Weight: 1
  NetworkConfiguration:
    AwsvpcConfiguration:
      SecurityGroups:
        - !Ref ServiceSecurityGroup
      Subnets: !Ref PrivateSubnetIds
  DeploymentConfiguration:
    MaximumPercent: 200
    MinimumHealthyPercent: 75
  DesiredCount: !Ref DesiredCount
  TaskDefinition: !Ref TaskDefinition
  LoadBalancers:
    - ContainerName: !Ref ServiceName
      ContainerPort: !Ref ContainerPort
      TargetGroupArn: !Ref ServiceTargetGroup
ServiceSecurityGroup:
  Type: 'AWS::EC2::SecurityGroup'
  Properties:
    GroupDescription: Security group for service
    VpcId: !Ref VpcId
ServiceTargetGroup:
  Type: 'AWS::ElasticLoadBalancingV2::TargetGroup'
  Properties:
    HealthCheckIntervalSeconds: 6
    HealthCheckPath: /
    HealthCheckProtocol: HTTP
    HealthCheckTimeoutSeconds: 5
    HealthyThresholdCount: 2
    TargetType: ip
    Port: !Ref ContainerPort
    Protocol: HTTP
    UnhealthyThresholdCount: 10
    VpcId: !Ref VpcId
    TargetGroupAttributes:
      - Key: deregistration_delay.timeout_seconds
        Value: 0
PublicLoadBalancerSG:
  Type: 'AWS::EC2::SecurityGroup'
  Properties:
    GroupDescription: Access to the public facing load balancer
    VpcId: !Ref VpcId
    SecurityGroupIngress:

```

```

      - CidrIp: 0.0.0.0/0
        IpProtocol: -1
PublicLoadBalancer:
  Type: 'AWS::ElasticLoadBalancingV2::LoadBalancer'
  Properties:
    Scheme: internet-facing
    LoadBalancerAttributes:
      - Key: idle_timeout.timeout_seconds
        Value: '30'
    Subnets: !Ref PublicSubnetIds
    SecurityGroups:
      - !Ref PublicLoadBalancerSG
PublicLoadBalancerListener:
  Type: 'AWS::ElasticLoadBalancingV2::Listener'
  Properties:
    DefaultActions:
      - Type: forward
        ForwardConfig:
          TargetGroups:
            - TargetGroupArn: !Ref ServiceTargetGroup
              Weight: 100
    LoadBalancerArn: !Ref PublicLoadBalancer
    Port: 80
    Protocol: HTTP
ServiceIngressfromLoadBalancer:
  Type: 'AWS::EC2::SecurityGroupIngress'
  Properties:
    Description: Ingress from the public ALB
    GroupId: !Ref ServiceSecurityGroup
    IpProtocol: -1
    SourceSecurityGroupId: !Ref PublicLoadBalancerSG
LogGroup:
  Type: 'AWS::Logs::LogGroup'

```

適用於 Amazon ECS 的 IAM 角色

您可以使用 CloudFormation 範本來建立與 Amazon ECS 搭配使用的 IAM 角色。如需有關適用於 Amazon ECS 的 IAM 角色的詳細資訊，請參閱[適用於 Amazon ECS 的 IAM 角色](#)。

Amazon ECS 任務執行角色

任務執行角色會授予 Amazon ECS 容器和 Fargate 代理程式代表您進行 AWS API 呼叫的許可。是否需要此角色視任務需求而定。如需詳細資訊，請參閱[Amazon ECS 任務執行 IAM 角色](#)。

下列範本可用來建立使用 AmazonECSTaskExecutionRolePolicy 受管政策的簡單任務執行角色。

JSON

```
{
  "AWSTemplateFormatVersion": "2010-09-09",
  "Description": "CloudFormation template for ECS Task Execution Role",
  "Resources": {
    "ECSTaskExecutionRole": {
      "Type": "AWS::IAM::Role",
      "Properties": {
        "AssumeRolePolicyDocument": {
          "Statement": [
            {
              "Effect": "Allow",
              "Principal": {
                "Service": ["ecs-tasks.amazonaws.com"]
              },
              "Action": ["sts:AssumeRole"],
              "Condition": {
                "ArnLike": {
                  "aws:SourceArn": {
                    "Fn::Sub": "arn:aws:ecs:${AWS::Region}:${AWS::AccountId}:*"
                  }
                },
                "StringEquals": {
                  "aws:SourceAccount": {
                    "Ref": "AWS::AccountId"
                  }
                }
              }
            }
          ]
        },
        "Path": "/",
        "ManagedPolicyArns": [
          "arn:aws:iam::aws:policy/service-role/AmazonECSTaskExecutionRolePolicy"
        ]
      }
    }
  }
}
```

```

    }
  }
},
"Outputs": {
  "ECSTaskExecutionRoleARN": {
    "Description": "ARN of the ECS Task Execution Role",
    "Value": {
      "Fn::GetAtt": ["ECSTaskExecutionRole", "Arn"]
    },
    "Export": {
      "Name": {
        "Fn::Sub": "${AWS::StackName}-ECSTaskExecutionRoleARN"
      }
    }
  },
  "ECSTaskExecutionRoleName": {
    "Description": "Name of the ECS Task Execution Role",
    "Value": {
      "Ref": "ECSTaskExecutionRole"
    },
    "Export": {
      "Name": {
        "Fn::Sub": "${AWS::StackName}-ECSTaskExecutionRoleName"
      }
    }
  }
}
}
}

```

YAML

```

AWSTemplateFormatVersion: '2010-09-09'
Description: 'CloudFormation template for ECS Task Execution Role'
Resources:
  ECSTaskExecutionRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Statement:
          - Effect: Allow
            Principal:
              Service: [ecs-tasks.amazonaws.com]
            Action: ['sts:AssumeRole']

```

```
    Condition:
      ArnLike:
        aws:SourceArn: !Sub arn:aws:ecs:${AWS::Region}:${AWS::AccountId}:*
      StringEquals:
        aws:SourceAccount: !Ref AWS::AccountId
    Path: /
    ManagedPolicyArns:
      - arn:aws:iam::aws:policy/service-role/AmazonECSTaskExecutionRolePolicy
Outputs:
  ECSTaskExecutionRoleARN:
    Description: ARN of the ECS Task Execution Role
    Value: !GetAtt ECSTaskExecutionRole.Arn
    Export:
      Name: !Sub "${AWS::StackName}-ECSTaskExecutionRoleARN"
  ECSTaskExecutionRoleName:
    Description: Name of the ECS Task Execution Role
    Value: !Ref ECSTaskExecutionRole
    Export:
      Name: !Sub "${AWS::StackName}-ECSTaskExecutionRoleName"
```

Amazon ECS 最佳實務

您可以透過下列任一頁面，了解 Amazon ECS 聯網最重要的操作最佳實務。

最佳實務概觀	進一步了解	
將應用程式連線至網際網路	將 Amazon ECS 應用程式連線至網際網路	
從網際網路接收傳入至 Amazon ECS 的連線。	從網際網路接收 Amazon ECS 傳入連線的最佳實務	
將 Amazon ECS 連線至 VPC AWS 內的其他服務	從 VPC 內部將 Amazon ECS 連線至 AWS 服務的最佳實務	
跨 AWS 帳戶 和 VPCs 的網路服務	跨 AWS 帳戶 和 VPCs 聯網 Amazon ECS 服務的最佳實務	
網路問題疑難排解	AWS Amazon ECS 網路疑難排解的服務	

您可以透過下列任一頁面，了解在 Amazon ECS 上使用 Fargate 最重要的操作最佳實務。

最佳實務概觀	進一步了解	
Fargate 安全	Amazon ECS 中的 Fargate 安全最佳實務	
Fargate 安全考量	為 Amazon ECS 選用 Fargate 的安全考量	
Fargate 容器映像上 Linux 容器的提取行為	Amazon ECS 的 Fargate 容器映像上 Linux 容器的提取行為	
Fargate 容器映像上 Windows 容器的提取行為	Amazon ECS 的 Fargate 容器映像上 Windows 容器的提取行為	

最佳實務概觀	進一步了解	
Fargate 任務淘汰	Amazon ECS 上 AWS Fargate 的任務淘汰和維護	

您可以透過下列任一頁面，了解任務定義最重要的操作最佳實務。

最佳實務概觀	進一步了解	
容器映像	Amazon ECS 容器映像的最佳實務	
任務大小	Amazon ECS 任務大小的最佳實務	
磁碟區最佳實務	Amazon ECS 任務的儲存選項	

您可以透過下列任一頁面，了解叢集與容量最重要的操作最佳實務。

最佳實務概觀	進一步了解	
Fargate 安全	Amazon ECS 中的 Fargate 安全最佳實務	
EC2 容器執行個體安全考量	Amazon ECS 的 Amazon EC2 容器執行個體安全考量	
叢集自動擴展	最佳化 Amazon ECS 叢集自動擴展	
大規模運作	大規模運作 Amazon ECS	
自動擴展與容量管理	Amazon ECS 自動擴展與容量管理最佳實務	
Amazon ECS 受管執行個體的執行個體選擇最佳實務	Amazon ECS 受管執行個體的執行個體選擇最佳實務	

您可以透過下列任一頁面，了解任務與服務最重要的操作最佳實務。

最佳實務概觀	進一步了解	
最佳化任務啟動時間	最佳化 Amazon ECS 任務啟動時間	
服務參數	Amazon ECS 服務參數最佳實務	
最佳化負載平衡器運作狀態檢查參數	針對 Amazon ECS 最佳化負載平衡器運作狀態檢查參數	
最佳化負載平衡器連接耗盡參數	針對 Amazon ECS 最佳化負載平衡器連接耗盡參數	
最佳化服務自動擴展	最佳化 Amazon ECS 服務自動擴展	

您可以透過下列任一頁面，了解安全最重要的操作最佳實務。

最佳實務概觀	進一步了解	
網路安全	Amazon ECS 的網路安全最佳實務	
任務和容器安全	Amazon ECS 任務與容器安全最佳實務	

規劃 Amazon ECS 解決方案架構

您必須規劃應用程式架構，使其能夠在容器上執行。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。容器映像是靜態成品，內含應用程式及其相依性。映像通常由 Dockerfile 建立。Dockerfile 是純文字檔案，內含用於建置容器的指令。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
- 執行時期監控

Capacity

容量即容器執行所在的基礎結構。Amazon ECS 為叢集提供三種基礎結構類型：

- Amazon ECS 受管執行個體 - AWS 完整管理您帳戶中執行的基礎 Amazon EC2 執行個體，包括佈建、修補和擴展。此選項能在效能、成本效益與操作簡易性之間實現最佳平衡。
- 無伺服器 (Fargate) – 只需為任務實際使用的資源付費，無需管理任何基礎結構。適用於變動性工作負載與需要快速上手的案例。

- Amazon EC2 執行個體 – 您可以直接管理底層 Amazon EC2 執行個體，包括執行個體選擇、設定與維護。

使用 Amazon ECS 受管執行個體的時機：

- 您希望兼具 Fargate 的簡易性，同時對底層基礎結構擁有更多控制權。
- 您需要可預測的效能與成本最佳化。
- AWS 您想要在維持彈性的同時處理基礎設施管理。

使用 Fargate 的時機：

- 您希望專注於應用程式，無需管理基礎結構。
- 您的工作負載具有變動性或不可預測性。
- 您剛接觸容器技術，希望有最簡單的部署選項。

使用 Amazon EC2 執行個體的時機：

- 您需要特殊的硬體需求 (GPU 加速、高效能運算)。
- 您需要容量保留或特定類型的執行個體。
- 您需要特殊權限功能或自訂 AMI。

您可以在建立叢集時指定基礎結構。您也可以註冊任務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」。您也可以使用容量提供者。

服務端點

服務端點是 Amazon ECS 的進入點 URL，您可透過網際網路通訊協定第 4 版 (IPv4) 或第 6 版 (IPv6)，以程式設計方式連線至該服務。依預設，您透過程式設計方式連線至 Amazon ECS 的請求，會使用僅支援 IPv4 請求的服務端點。若要透過 IPv4 或 IPv6 請求，以程式設計方式連線至 Amazon ECS，您可使用雙堆疊端點。如需有關雙堆疊端點的資訊，請參閱[使用 Amazon ECS 雙堆疊端點](#)。

聯網

AWS 資源是在子網路中建立的。若使用 EC2 執行個體，Amazon ECS 會在您建立叢集時指定的子網路中啟動執行個體。任務將在執行個體子網路中執行。若使用 Fargate 或內部部署虛擬機器，您需要在執行任務或建立服務時指定子網路。

根據您的應用程式，子網路可以是私有或公有子網路，子網路可以位於下列任何 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 Zone 與 Wavelength Zone 中的 Amazon ECS 應用程式](#) 或 [上的 Amazon Elastic Container Service AWS Outposts](#)。

您可以透過下列任一方法，將應用程式連線至網際網路：

- 具有網際網路閘道的公有子網路

若公開應用程式需要大量頻寬或極低延遲，請使用公有子網路。適用案例包括影片串流與遊戲服務。

- 具有 NAT 閘道的私有子網路

若要保護容器免受外部直接存取，請使用私有子網路。適用案例包括支付處理系統或儲存使用者資料與密碼的容器。

- AWS PrivateLink

使用 AWS PrivateLink 在 VPCs、AWS 服務和內部部署網路之間建立私有連線，而不會將您的流量暴露到公有網際網路。

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型服務收集、彙總及總結指標和日誌。指標包含 CPU、記憶體、磁碟和網路這類資源的使用率。

- awsvpc 主幹功能

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提供額外的網路介面 (ENI)。

- 標記授權

使用者必須擁有建立資源的動作許可，例如 `ecsCreateCluster`。如果在資源建立動作中指定標籤，會對 `ecs:TagResource` 動作 AWS 執行其他授權，以驗證使用者或角色是否具有建立標籤的許可。

- 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 任務淘汰時間變更

您可以設定 Fargate 任務淘汰以進行修補之前的等待期間。

- 雙堆疊 VPC

允許任務透過 IPv4 和/或 IPv6 進行通訊。

- Amazon Resource Name (ARN) 格式

部分功能 (例如標籤授權) 需要使用新的 Amazon Resource Name (ARN) 格式。

如需詳細資訊，請參閱[透過帳戶設定使用 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，將相關許可授予 Amazon ECS 資源 (例如容器或服務)。

部分 Amazon ECS 功能需要角色才能使用。如需詳細資訊，請參閱[適用於 Amazon ECS 的 IAM 角色](#)。

日誌

記錄與監控是維護 Amazon ECS 工作負載可靠性、可用性與效能的重要環節。下列選項可供使用：

- Amazon CloudWatch 日誌 - 將日誌路由到 Amazon CloudWatch
- FireLens for 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。AWS Partner Network 是一個全球合作夥伴社群，利用計劃、專業知識和資源來建置、行銷和銷售客戶產品。

容器映像最佳實務

以下是建構 Amazon ECS 容器映像的核心原則：

- 映像中包含所有相依性
- 每個容器執行一個程序
- 處理 SIGTERM 實現優雅關閉
- 將日誌寫入 stdout 與 stderr
- 使用唯一標籤，避免在生產中使用 latest

Amazon Amazon ECS 啟動類型和容量提供者

Amazon ECS 提供兩種方法來設定工作負載的容量。您可以使用啟動類型或容量提供者。啟動類型包括 EC2、Fargate 和 External。容量提供者為容量管理提供增強的彈性和進階功能。您可以使用 Fargate 和 Fargate Spot 容量提供者在無伺服器運算上執行工作負載、透過 Auto Scaling 群組容量提供者在自我管理的 EC2 執行個體上執行工作負載，或使用結合 Fargate 簡易性和 EC2 運算彈性的 Amazon ECS 受管執行個體容量提供者在全受管運算上執行工作負載。容量提供者可更好地控制資源配置，並有助於最佳化效能和成本。與傳統啟動類型相比，容量提供者是為工作負載設定容量的建議方法。使用下列項目來了解容量提供者和啟動類型之間的差異。

最佳實務

以下是最佳實務：

使用啟動類型來定義基礎設施相容性

啟動類型定義任務和服務執行所在的基礎設施。當您定義任務時，請指定 `RequiresCompatibilities` 以包含與任務相容的一或多個啟動類型。您可以使用下列啟動類

型：EC2、Fargate、外部和 Amazon ECS 受管執行個體。雖然您也可以使用啟動類型來執行您的任務或服務，但我們建議您僅將啟動類型用於在任務定義中定義相容性，並使用容量提供者來啟動任務或服務。請注意，您可以選擇一或多個啟動類型來定義任務的相容性。

使用容量提供者來設定運算容量

當您啟動任務或服務時，請設定容量提供者策略。Amazon ECS 支援下列容量提供者：Fargate 和 FARGATE_SPOT、自我管理 EC2 執行個體的 Auto Scaling 群組，以及 Amazon ECS 受管執行個體。請注意，Spot Fleet 只能作為容量提供者使用，不能作為啟動類型使用。您可以在叢集中建立一或多個 Amazon ECS 受管執行個體或 Auto Scaling 群組容量提供者。Fargate 和 Fargate Spot 容量提供者是由 Amazon ECS 在每個叢集上建立和管理，您不需要建立它們。叢集可以混合所有容量提供者類型，但容量提供者策略不能混合不同的容量提供者類型。

更新 服務的容量

您可以直接更新服務的容量提供者策略，將其從一種運算類型移至另一種。

服務可變性

Amazon ECS 支援更新不同容量提供者之間的服務。這允許：

- 從啟動類型無縫更新至容量提供者
- 不同容量提供者類型之間的轉換
- 在沒有服務重新建立的情況下測試不同的運算選項

以下是程序的高階概觀：

1. 更新任務定義 – 例如確保 `requiresCompatibilities` 包含目標容量提供者 `MANAGED_INSTANCES`。

Note

任務定義必須通過目標容量提供者的相容性驗證。如果任務定義版本的 `requiresCompatibilities` 檢查失敗，`UpdateService` 呼叫會失敗。

2. 建立容量提供者 – 如果您使用自訂 Amazon EC2 Auto Scaling 群組，請建立容量提供者。
3. 更新服務 – 修改服務以使用容量提供者策略，而非啟動類型。
4. 驗證部署 – 確認任務部署成功。

5. 監控和最佳化 – 視需要調整容量提供者設定。

容量提供者至容量提供者

支援容量提供者到容量提供者的所有更新：

- Amazon EC2 Auto Scaling 群組容量提供者至 Amazon ECS 受管執行個體
- Fargate 容量提供者至 Amazon ECS 受管執行個體
- Amazon EC2 Auto Scaling 群組容量提供者至 Fargate 容量提供者
- Amazon ECS 受管執行個體到 Fargate 容量提供者
- Fargate 容量提供者至 Amazon EC2 Auto Scaling 群組容量提供者
- Amazon ECS 受管執行個體到 Amazon EC2 Auto Scaling 群組容量提供者

容量提供者的啟動類型

支援容量提供者更新的所有啟動類型：

- Amazon ECS 受管執行個體的 EC2 啟動類型
- Amazon ECS 受管執行個體的 Fargate 啟動類型
- Fargate 容量提供者的 EC2 啟動類型
- EC2 Auto Scaling 群組容量提供者的 EC2 啟動類型
- Amazon EC2 Auto Scaling 群組容量提供者的 Fargate 啟動類型
- Fargate 啟動類型至 Fargate 容量提供者
- Amazon ECS 受管執行個體的外部啟動類型
- Fargate 容量提供者的外部啟動類型
- Amazon EC2 Auto Scaling 群組容量提供者的外部啟動類型

啟動類型以啟動類型

不支援啟動類型以啟動類型更新：

- 從 EC2 啟動類型到 Fargate 啟動類型（改用 Fargate 容量提供者）
- Fargate 啟動類型為 EC2 啟動類型（改用 Amazon EC2 Auto Scaling 群組容量提供者）

遷移到同等容量提供者，以增強功能和未來相容性，而不是在啟動類型之間遷移。

Note

任務定義必須通過目標容量提供者的相容性驗證。如果任務定義版本的 `requiresCompatibilities` 檢查失敗，`UpdateService` 呼叫將會失敗。

使用 Amazon ECS 雙堆疊端點

Amazon ECS 雙堆疊端點支援透過網際網路通訊協定第 4 版 (IPv4) 與第 6 版 (IPv6) 向 Amazon ECS 發出請求。如需所有 Amazon ECS 端點的清單，請參閱 AWS 一般參考中的 [Amazon ECS endpoints and quotas](#)。

使用 REST API 等同於使用端點名稱 (URI) 直接存取 Amazon ECS 端點。Amazon ECS 僅支援區域雙堆疊端點名稱，亦即指定的名稱必須包含區域。

雙堆疊端點名稱使用以下命名慣例：`ecs.region.api.aws`。

使用 AWS Command Line Interface (AWS CLI) 和 AWS SDKs 時，您可以使用參數或旗標來變更為雙堆疊端點。您也可以直接指定雙堆疊端點來覆寫組態檔中的 Amazon ECS 端點。

下列各節說明如何使用來自 AWS CLI、AWS SDKs 和 REST API 的雙堆疊端點。

主題

- [從 使用雙堆疊端點 AWS CLI](#)
- [從 AWS SDKs 使用雙堆疊端點](#)
- [從 REST API 使用雙堆疊端點](#)

從 使用雙堆疊端點 AWS CLI

本節提供用於向雙堆疊端點提出請求的 AWS CLI 命令範例。如需安裝 AWS CLI 或更新至最新版本的詳細資訊，請參閱《第 2 版使用者指南》中的 [安裝或更新至最新版本 AWS CLI](#) 的。AWS Command Line Interface

若要使用雙堆疊端點，您可以將 `config use_dualstack_endpoint true` 檔案中的組態值設定為 AWS CLI，讓將 `ecs` AWS CLI 命令提出的所有 Amazon ECS 請求導向指定區域的雙堆疊端點。您可以在 `config` 檔案或命令中使用 `--region` 選項指定區域。如需組態檔案的詳細資訊 AWS CLI，請參閱《第 2 版使用者指南》中的 [組態和登入資料檔案設定 AWS CLI](#)。AWS Command Line Interface

如果您想要針對特定 AWS CLI 命令使用雙堆疊端點，您可以使用下列其中一種方法：

- 您可以將任何 ecs 命令的 `--endpoint-url` 參數設定為 `https://ecs.aws-region.api.aws` 或 `http://ecs.aws-region.api.aws`，以在每個命令中使用雙堆疊端點。

下列範例命令會列出所有可用的叢集，且該請求將使用雙堆疊端點。

```
$ aws ecs list-clusters --endpoint-url https://ecs.aws-region.api.aws
```

- 您可以在 AWS Config 檔案中設定個別的設定檔。例如，您可以建立一個設定檔，將 `use_dualstack_endpoint` 設為 `true`，再建立另一個設定檔不設定 `use_dualstack_endpoint`。當您執行命令時，必須依據是否要使用雙堆疊端點來指定所要使用的設定檔。

從 AWS SDKs 使用雙堆疊端點

本節提供如何使用 AWS SDKs 存取雙堆疊端點的範例。

AWS SDK for Java 2.x

下列範例顯示如何使用 AWS SDK for Java 2.x 指定 `us-east-1` 區域的雙堆疊端點。

```
Region region = Region.US_EAST_1
EcsClient client =
    EcsClient.builder().region(region).dualstackEnabled(true).build();
```

適用於 Go 的 AWS SDK

下列範例顯示如何使用適用於 Go 的 AWS SDK 指定 `us-east-1` 區域的雙堆疊端點。

```
sess := session.Must(session.NewSession())
svc := ecs.New(sess, &aws.Config{
    Region: aws.String(endpoints.UsEast1RegionID),
    Endpoint: aws.String("https://ecs.us-east-1.api.aws")
})
```

如需詳細資訊，請參閱 AWS SDKs and Tools Reference Guide 中的 [Dual-stack and FIPS endpoints](#)。

從 REST API 使用雙堆疊端點

使用 REST API 時，您可以透過在請求中指定雙堆疊端點，直接存取該端點。下列範例使用雙堆疊端點，列出 us-east-1 區域中的所有 Amazon ECS 叢集。

```
POST / HTTP/1.1
Host: ecs.us-east-1.api.aws
Accept-Encoding: identity
Content-Length: 2
X-Amz-Target: AmazonEC2ContainerServiceV20141113.ListClusters
X-Amz-Date: 20150429T170621Z
Content-Type: application/x-amz-json-1.1
Authorization: AUTHPARAMS

{}
```

共用子網路、Local Zone 與 Wavelength Zone 中的 Amazon ECS 應用程式

當需要低延遲或本機資料處理時，Amazon ECS 支援使用 Local Zones、Wavelength Zones 和 AWS Outposts 的工作負載。

- 您可以使用 Local Zones 作為的延伸 AWS 區域，將資源放置在靠近最終使用者的多個位置。
- 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 會將標準 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 服務基礎設施和操作模型帶入幾乎所有資料中心、主機代管空間或內部部署設施。

Important

Local Zones、Wavelength Zones 或上的 AWS Fargate 工作負載不支援 Amazon ECS AWS Outposts。

如需 Local Zones、Wavelength Zones 和之間差異的資訊 AWS Outposts，請參閱 [AWS Wavelength FAQs](#) 中的 [我應該如何考慮何時使用 AWS Wavelength、AWS Local Zones 或需要低延遲或本機資料處理 AWS Outposts 的應用程式](#)。

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。

您可將共用 VPC 用於 EC2，但需留意以下考量事項：

- 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。
- 您無法將 VPC 預設安全群組用於容器執行個體，因為預設安全群組屬於擁有者。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](#)。

您可將共用 VPC 用於 Fargate，但需留意以下考量事項：

- 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 的預設安全群組來建立服務或執行任務，因為預設安全群組屬於擁有者。此外，參與者無法使用其他參與者或擁有者所擁有的安全群組來建立服務或執行任務。
- 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

Local Zone 是 AWS 區域的延伸，其地理位置接近您的使用者。Local Zones 有自己的網際網路連線，並支援 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 [AWS Local Zones 使用者指南中的什麼是 Local Zones？](#)。AWS

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 會將標準 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 執行個體等 AWS 資源來執行需要超低延遲和 AWS 服務 區域中 連線的應用程式。

Wavelength Zone 是部署 Wavelength 基礎設施之電信業者位置的隔離區域。Wavelength 區域繫結至 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](#)。

Wavelength 區域不適用於所有 AWS 區域。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](#)。

上的 Amazon Elastic Container Service AWS Outposts

AWS Outposts 在內部部署設施中啟用原生 AWS 服務、基礎設施和操作模型。在 AWS Outposts 環境中，您可以使用您在 中使用的相同 AWS APIs、工具和基礎設施 AWS 雲端。

上的 Amazon ECS AWS Outposts 非常適合需要在接近內部部署資料和應用程式的情況下執行的低延遲工作負載。

如需 的詳細資訊 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 的考量事項：

- Amazon Elastic Container Registry AWS Identity and Access Management 和 Network Load Balancer 會在 AWS 區域中執行，而不是在區域中執行 AWS Outposts。這將提高這些服務和容器之間的延遲。
- AWS Fargate 不適用於 AWS Outposts。

以下是的網路連線考量 AWS Outposts：

- 如果 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。CloudWatch Logs 代理程式將無法更新記錄檔和事件資料。
- 我們建議您在 AWS Outposts 及其 AWS 區域之間提供可靠、高可用性和低延遲的連線。

先決條件

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。

在上建立叢集的概觀 AWS Outposts

下列是組態概觀：

1. 建立具有權限的角色和政策 AWS Outposts。
2. 在 AWS Outposts 上建立具有權限的 IAM 執行執行個體設定檔。
3. 建立 VPC，或使用與 AWS Outposts 位於同一區域的現有 VPC。
4. 建立子網路，或使用與 AWS Outposts 相關聯的現有子網路。

此即容器執行個體執行所在的子網路。

5. 在叢集中建立容器執行個體的安全群組。
6. 建立 Amazon ECS 叢集
7. 定義 Amazon ECS 容器代理程式環境變數，將執行個體啟動至叢集中。

8. 執行一個容器。

如需如何整合 Amazon ECS 與 的詳細資訊 AWS Outposts，請參閱將 [Amazon ECS 延伸到兩個 AWS Outposts 機架](#)。

以下範例會在 AWS Outposts 上建立 Amazon ECS 叢集。

1. 建立具有 權限的角色和政策 AWS Outposts。

role-policy.json 檔案是政策文件，其中包含資源的效果和動作。如需檔案格式的資訊，請參閱《IAM API 參考》中的 [PutRolePolicy](#)

```
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. 建立與 相關聯的子網路 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-xxxxxxxxxxxxxxxx \  
    --availability-zone-id usw2-az1
```

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

```
aws ec2 create-security-group --group-name MyOutpostSG  
aws ec2 authorize-security-group-ingress --group-name MyOutpostSG --protocol tcp \  
    --cidr 10.0.0.0/16
```

```
--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
```

6. 建立叢集。
7. 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。

8. 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。(此步驟與不同 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。如需有關 API 呼叫檔案的資訊，請參閱 Amazon EC2 User Guide 中的 [Run commands on your Linux instance at launch](#)。

```
aws ec2 run-instances --count 1 --image-id ami-xxxxxxx --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。

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。

如果變更容器映像中的某些內容，請建置包含變更的新容器映像。

- 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 會取代異常終止的程序，並決定替代程序的啟動位置。完整的容器映像能有效提升整體部署的復原能力。

- 讓應用程式處理 SIGTERM。

當 Amazon ECS 停止任務時，會先向任務傳送 SIGTERM 訊號，告知應用程式需完成處理並關閉。然後，Amazon ECS 會傳送 SIGKILL 訊息。當應用程式忽略 SIGTERM 時，Amazon ECS 服務必須等待一段時間，再傳送 SIGKILL 訊號來終止程序。

您需要確定應用程式完成自身工作所需的時間，並確保應用程式會處理 SIGTERM 訊號。在應用程式的訊號處理中，您需要停止應用程式接受新工作並完成正在進行的工作；如果工作需要太長時間才能完成，則需將未完成的工作儲存至任務外部的儲存空間。

- 設定容器化應用程式以將日誌寫入 `stdout` 和 `stderr`。

將日誌處理與應用程式程式碼分離，可讓您更靈活地調整基礎結構層級的日誌處理。其中一個範例是變更記錄系統。您可以調整設定，而不是修改服務以及建置與部署新的容器映像。

- 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。相反地，Amazon ECR 強制要求必須將新映像上傳至新標籤。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](#)。

當您建構應用程式以在其中執行時 AWS Fargate，您必須決定要在相同的任務定義中部署多個容器，以及在多個任務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)。
- 您的容器會共用資源。
- 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。這能讓您分別對容器進行擴展、佈建與解除佈建。

Amazon ECS 聯網最佳實務

現代應用程式通常由多個可互相通訊的分散式元件所構成。例如，行動或 Web 應用程式可能會與某個 API 端點通訊，而該 API 可能由透過網際網路通訊的多個微服務提供支援。

如需有關將應用程式連線至網際網路的最佳實務資訊，請參閱[將 Amazon ECS 應用程式連線至網際網路](#)。

如需有關從網際網路接收 Amazon ECS 傳入連線的最佳實務資訊，請參閱[從網際網路接收 Amazon ECS 傳入連線的最佳實務](#)。

如需將 Amazon ECS 連線至其他 AWS 服務之最佳實務的相關資訊，請參閱[從 VPC 內部將 Amazon ECS 連線至 AWS 服務的最佳實務](#)。

如需有關在 VPC 中連接服務的最佳實務資訊，請參閱[在 VPC 中連接 Amazon ECS 服務的最佳實務](#)。

如需跨 AWS 帳戶和 VPCs 聯網服務最佳實務的相關資訊，請參閱[跨 AWS 帳戶和 VPCs 聯網 Amazon ECS 服務的最佳實務](#)。

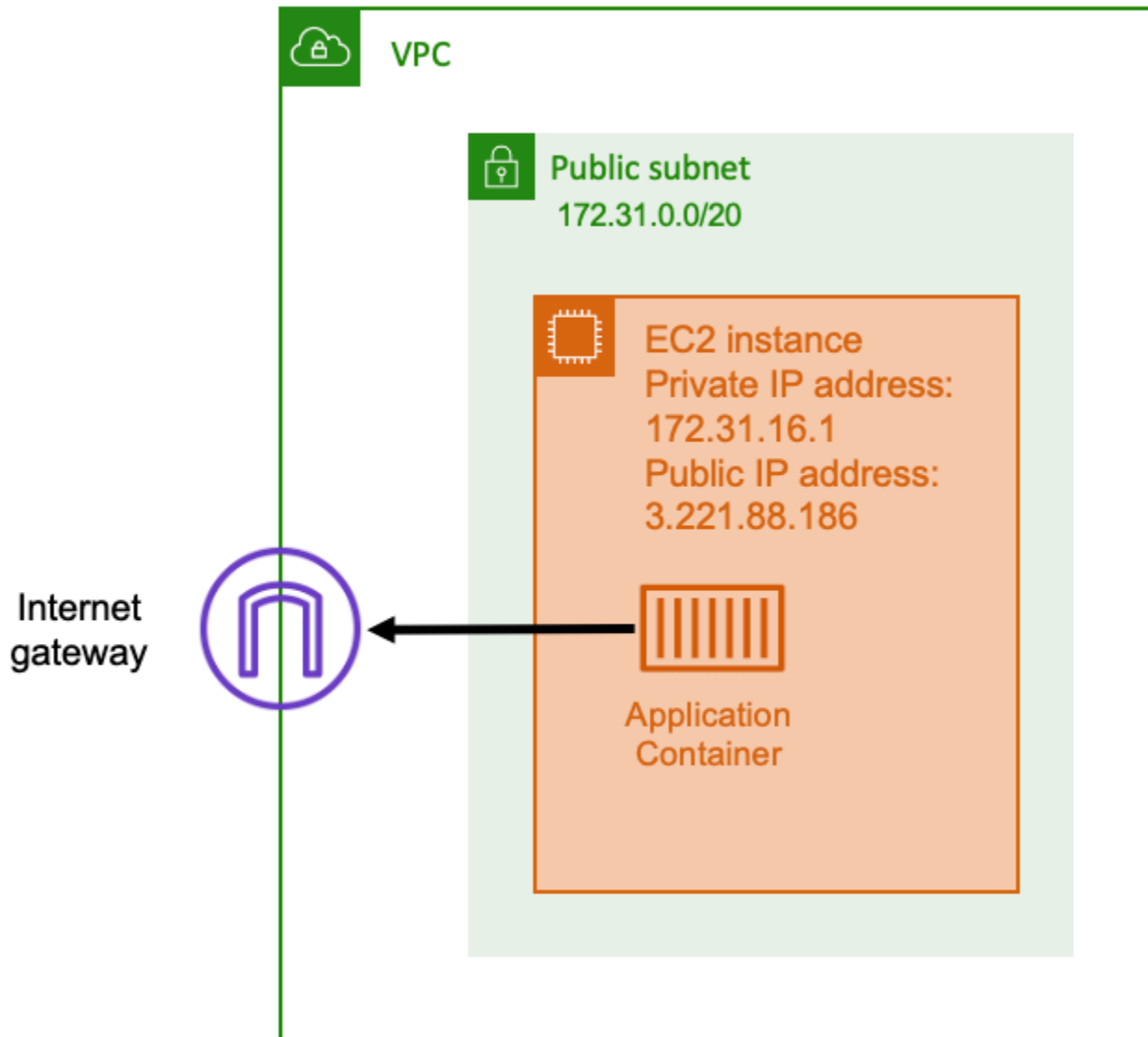
如需有關針對聯網問題進行疑難排解的服務最佳實務資訊，請參閱[AWS Amazon ECS 網路疑難排解的服務](#)。

將 Amazon ECS 應用程式連線至網際網路

大多數容器化應用程式至少有一些元件需要網際網路的傳出存取權。例如，行動應用程式的後端需要推播通知的傳出存取權。

Amazon Virtual Private Cloud 主要有兩種方法來促進 VPC 與網際網路之間的通訊。

公有子網路和網際網路閘道



當您使用具備網際網路閘道路由的公有子網路時，容器化應用程式即可在 VPC 內的公用子網路中執行。執行容器的主機會獲指派公有 IP 位址。此公有 IP 位址可從網際網路進行路由。如需詳細資訊，請參閱《Amazon VPC 使用者指南》中的[網際網路閘道](#)。

此網路架構可促進執行應用程式的主機與網際網路上其他主機之間的直接通訊。通訊是雙向的。這表示您不僅能建立通往網際網路上任何主機的傳出連線，網際網路上的其他主機也可能嘗試連線至您的主機。因此，您應該密切注意自己的安全群組與防火牆規則。這能確保網際網路上的其他主機無法開啟任何您不希望開放的連線。

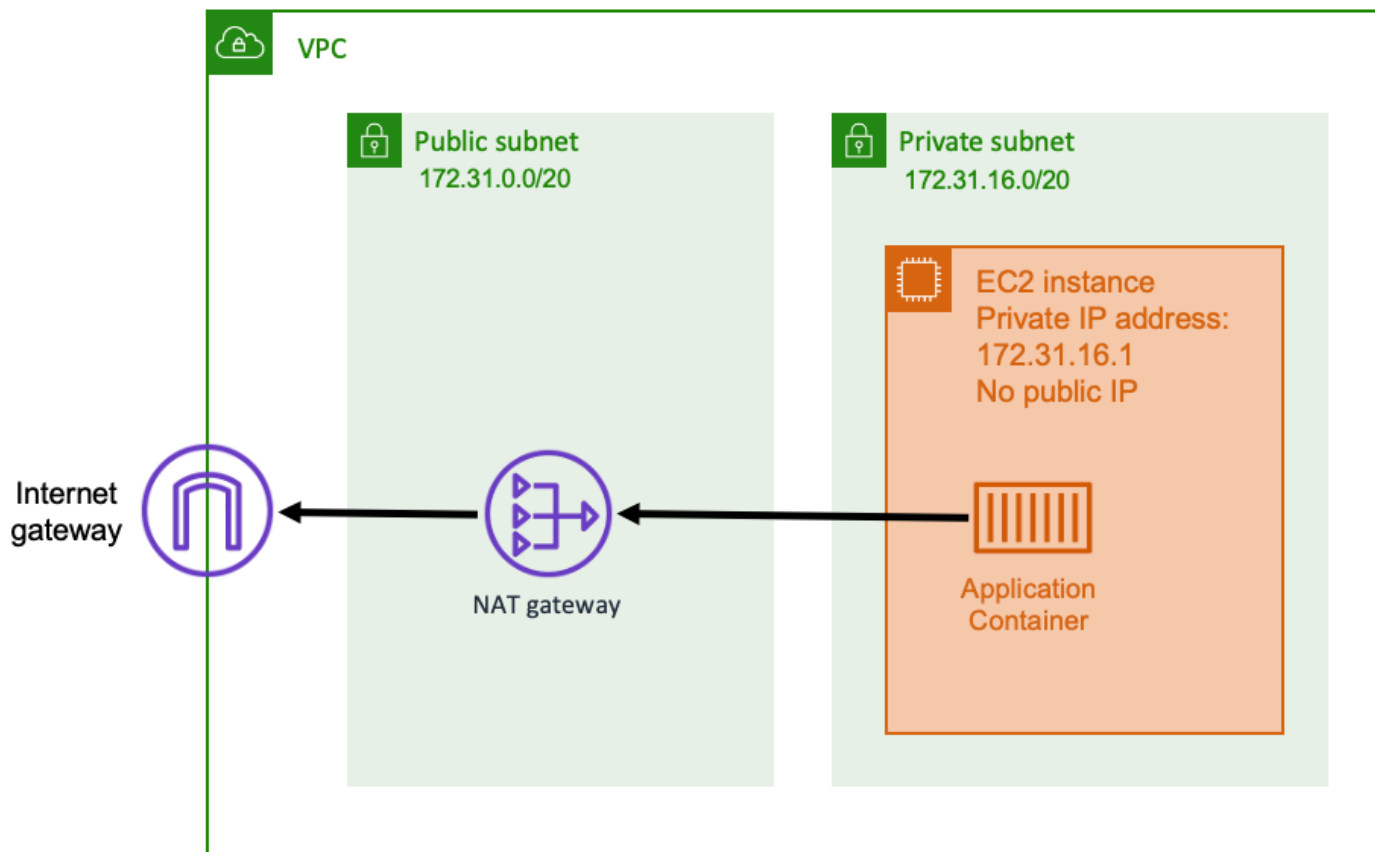
例如，若應用程式在 Amazon EC2 上執行，請確定用於 SSH 存取的連接埠 22 未開啟。否則，您的執行個體可能會不斷收到來自網際網路中惡意機器人的 SSH 連線嘗試。這些機器人會掃描公有 IP 位址。一旦找到開啟的 SSH 連接埠，機器人就會嘗試以暴力破解密碼的方式入侵您的執行個體。因此，許多組織會限制公有子網路的使用，並且偏好將大部分資源 (甚至全部) 置於私有子網路內。

使用公有子網路進行聯網，適用於需要大量頻寬或極低延遲的公開應用程式。適用的使用案例包括影片串流與遊戲服務。

當您在 Amazon EC2 與 AWS Fargate 上使用 Amazon ECS 時，都支援此聯網方案。

- 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 執行個體作為叢集容量，在執行個體上執行的任何容器都可以使用主機的底層公有 IP 位址進行傳出聯網通訊。這同時適用於 host 與 bridge 網路模式。不過，awsipc 網路模式不會為任務 ENI 提供公有 IP 位址。因此，這些 ENI 無法直接使用網際網路閘道。
- Fargate：當您建立 Amazon ECS 服務時，請為服務的聯網組態指定公有子網路，並使用指派公有 IP 位址選項。每個 Fargate 任務都在公有子網路中網路設定，並擁有專屬公有 IP 位址，可直接與網際網路通訊。

私有子網路和 NAT 閘道



當您使用私有子網路與 NAT 閘道時，可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 VPC 內進行路由，但無法從網際網路進行路由的私有 IP 位址。這表示 VPC 內的其他主機可以使用其私有 IP 位址連線至主機，但網際網路上的其他主機無法對主機進行任何傳入通訊。

透過私有子網路，您可以使用網路位址轉譯 (NAT) 閘道，讓私有子網路內的主機能夠連接至網際網路。網際網路上的主機會收到傳入連線，該連線似乎來自公有子網路內 NAT 閘道的公有 IP 位址。NAT 閘道負責作為網際網路與私有子網路之間的橋接器。基於安全性考量，這類組態常被優先採用，因為能確保您的 VPC 免受網際網路中攻擊者的直接存取。如需詳細資訊，請參閱《Amazon VPC 使用者指南》中的 [NAT 閘道](#)。

此類私有聯網方案適用於您希望保護容器免遭直接外部存取的案例。適用案例包括支付處理系統或儲存使用者資料與密碼的容器。您將需要為您帳戶中建立及使用的 NAT 閘道支付費用。NAT 閘道的每小時用量率與資料處理率也同樣適用。基於備援目的，您應該在每個可用區域中部署一個 NAT 閘道。如此，單一可用區域的可用性損失就不會影響傳出連線能力。因此，如果工作負載很小，使用私有子網路與 NAT 閘道可能更具成本效益。

在 Amazon EC2 與 AWS Fargate 上使用 Amazon ECS 時，都支援此聯網方案。

- 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底層主機聯網設定，而傳出請求會透過 NAT 閘道傳輸。
- Fargate：當您建立 Amazon ECS 服務時，請為服務的聯網組態指定私有子網路，但不使用指派公有 IP 位址選項。每個 Fargate 任務皆託管於私有子網路中。其傳出流量會透過您已與該私有子網路關聯的任一 NAT 閘道進行路由。

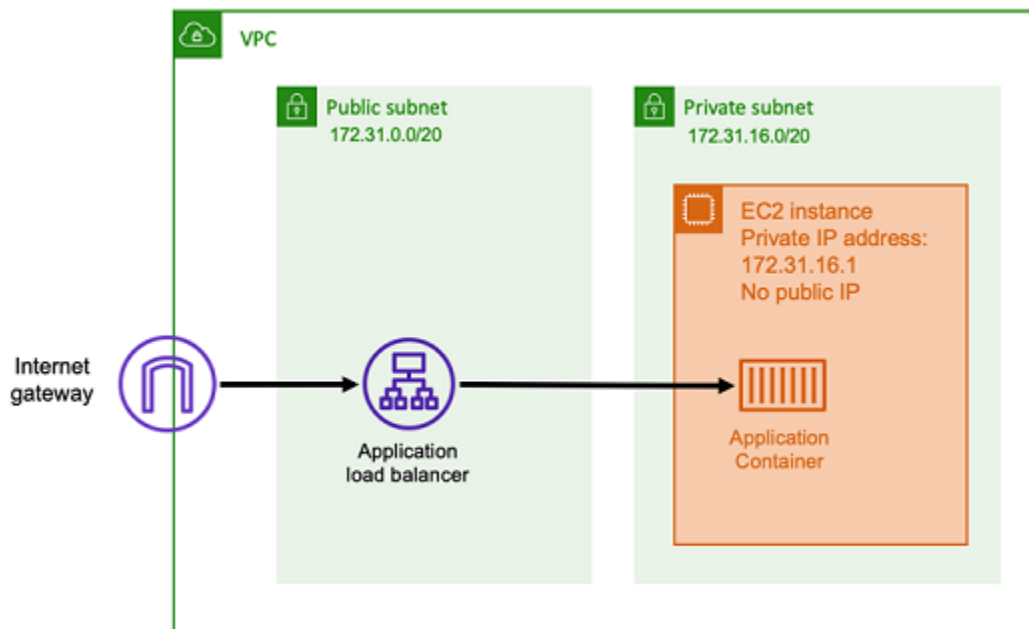
從網際網路接收 Amazon ECS 傳入連線的最佳實務

如果執行公有服務，則必須接受來自網際網路的傳入流量。例如，公有網站必須接受來自瀏覽器的傳入 HTTP 請求。在這種情況下，網際網路上的其他主機也必須啟動與應用程式主機의傳入連線。

解決此問題的一個方案，是在具有公有 IP 位址的公有子網路中的主機上啟動容器。不過，我們不建議將此方案用於大規模應用程式。對於這類應用程式，更優的方案是在網際網路與您的應用程式之間，部署一個可擴展的輸入層。採用此方案時，您可以使用本節所列的任一 AWS 服務作為輸入層。

Application Load Balancer

Application Load Balancer 在應用程式層運作。這是開放式系統互連 (OSI) 模型的第七層。此特性使得 Application Load Balancer 適用於公有 HTTP 服務。若您擁有網站或 HTTP REST API，則 Application Load Balancer 是適合此工作負載的負載平衡器。如需詳細資訊，請參閱 User Guide for Application Load Balancers 中的 [What is an Application Load Balancer?](#) 章節。



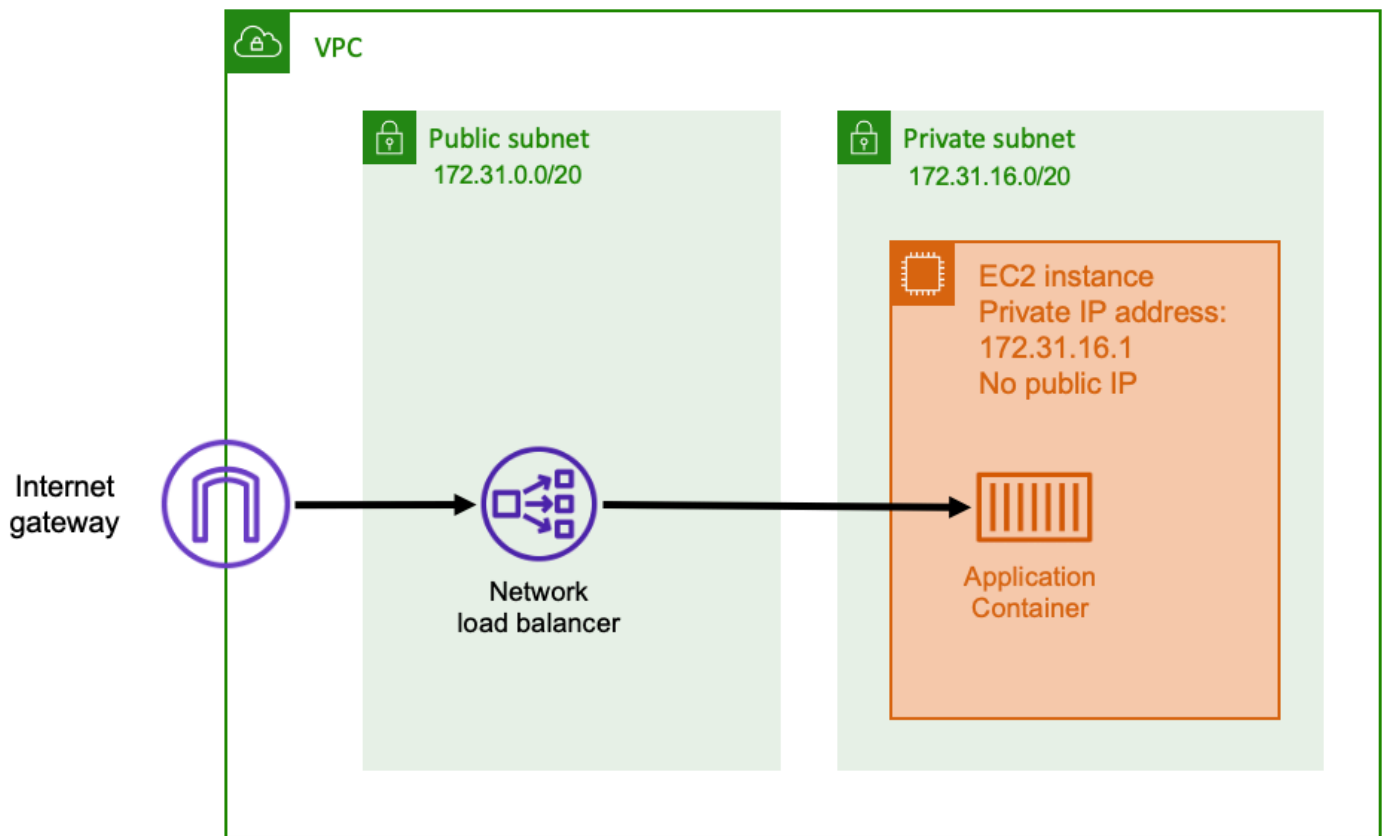
您可以使用此架構在公有子網路中建立 Application Load Balancer，使其具有公有 IP 位址，並可接收來自網際網路的傳入連線。當 Application Load Balancer 收到傳入連線 (更具體地說為 HTTP 請求) 時，會透過私有 IP 位址與應用程式建立連線。接著，該服務會透過這條內部連線轉送該請求。

Application Load Balancer 具備以下優勢。

- **SSL/TLS 終止**：Application Load Balancer 可維持與用戶端通訊所需的安全 HTTPS 通訊及憑證。您可選擇在負載平衡器層級終止 SSL 連線，無需在自身應用程式中處理憑證。
- **進階路由**：Application Load Balancer 可以具有多個 DNS 主機名稱。其還具有進階路由功能，可根據主機名稱或請求路徑等指標，將傳入的 HTTP 請求傳送至不同的目的地。這表示您可以使用單一 Application Load Balancer 作為許多不同內部服務的輸入，甚至是 REST API 不同路徑上微服務的輸入。
- **gRPC 支援與 Websocket**：Application Load Balancer 可處理的不僅限於 HTTP。其也能在支援 HTTP/2 的基礎上，對 gRPC 與 WebSocket 服務進行負載平衡。
- **安全性**：Application Load Balancer 有助於保護您的應用程式免遭惡意流量攻擊。它包含 HTTP 取消同步緩解等功能，並與 AWS Web Application Firewall (AWS WAF) 整合。AWS WAF 可以進一步篩選可能包含攻擊模式的惡意流量，例如 SQL Injection 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。其適用於非 HTTP 通訊協定或需要端到端加密的案例，但不具備 Application Load Balancer 特有的 HTTP 相關功能。因此，Network Load Balancer 最適用於不使用 HTTP 協定的應用程式。如需詳細資訊，請參閱 User Guide for Network Load Balancers 中的 [What is a Network Load Balancer?](#) 章節。



使用 Network Load Balancer 作為輸入時，其運作方式與 Application Load Balancer 類似。這是因為其是在公有子網路中建立，且具有可在網際網路上存取的公有 IP 位址。然後，Network Load Balancer 會開啟與執行您容器之主機私有 IP 位址的連線，並將封包從公有端傳送至私有端。

Network Load Balancer 功能

由於 Network Load Balancer 運作於聯網堆疊的較底層，因此沒有與 Application Load Balancer 相同的功能集。不過，仍具備以下重要功能。

- 端對端加密：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。這使其適用於需端對端加密的通訊負載平衡。
- TLS 加密：除了端對端加密之外，Network Load Balancer 也可以終止 TLS 連線。如此，後端應用程式就無需自行實現 TLS 加密功能。
- UDP 支援：由於 Network Load Balancer 在 OSI 模型的第四層運作，因此適用於非 HTTP 工作負載以及 TCP 以外的通訊協定。

關閉連線

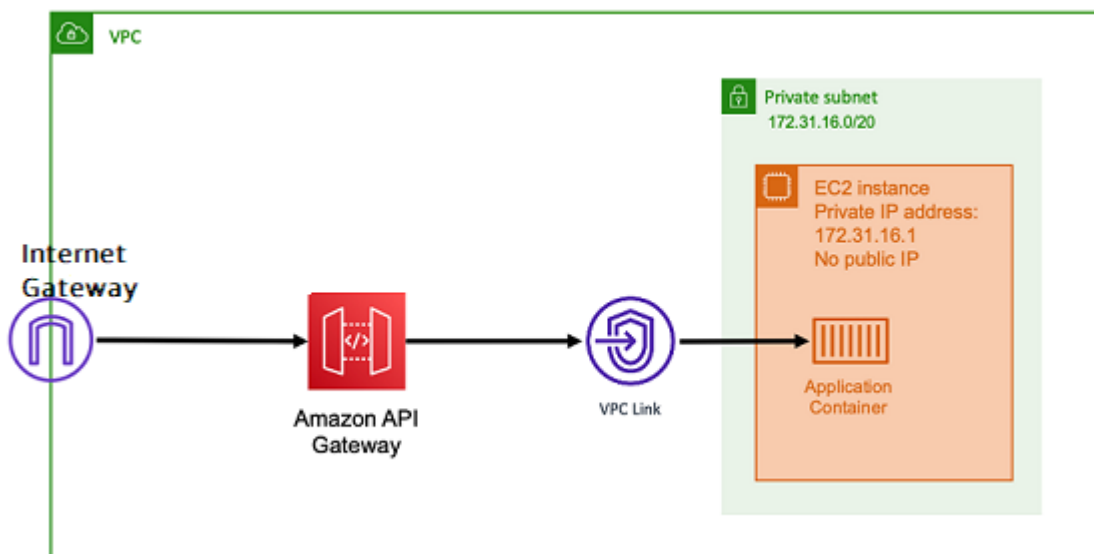
由於 Network Load Balancer 未在 OSI 模型的較高層偵測到應用程式通訊協定，因此無法透過這些協定向用戶端傳送關閉訊息。與 Application Load Balancer 不同，這類連線需要由應用程式自行關閉；您也可以設定 Network Load Balancer，在任務停止或取代時關閉第四層連線。請參閱 [Network Load Balancer documentation](#) 中 Network Load Balancer 目標群組的連線終止設定。

若用戶端未妥善處理第四層連線終止，透過 Network Load Balancer 關閉連線可能導致用戶端顯示非預期的錯誤訊息。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。

關閉連線的方法因應用程式而異，但其中一種方式是確保 Network Load Balancer 目標取消註冊延遲時間長於用戶端連線逾時設定。如此一來，用戶端會先逾時並透過 Network Load Balancer 優雅地重新連線至新任務，而舊任務則能逐步排空所有現有連線。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 documentation](#)。

Amazon API Gateway HTTP API

Amazon API Gateway 適用於請求量突增或請求量偏低的 HTTP 應用程式。如需詳細資訊，請參閱 API Gateway Developer Guide 中的 [What is Amazon API Gateway?](#) 章節。



Application Load Balancer 與 Network Load Balancer 的定價模式，均包含讓負載平衡器隨時保持可用以接收傳入連線的每小時費用。與之相對，API Gateway 採用按請求獨立計價的模式。該模式的作用在於，如果沒有請求傳入，就不會產生任何費用。在高流量負載下，Application Load Balancer 或 Network Load Balancer 能以比 API Gateway 更低的單次請求成本，處理更大規模的請求量。不過，如果整體請求數量較少，或具有低流量時段，則為了維持未充分利用的負載平衡器，使用 API Gateway 的累積價格應該比支付每小時費用更具成本效益。API Gateway 也能快取 API 回應，有助於降低後端請求速率。

使用 VPC 連結的 API Gateway 函數，允許 AWS 受管服務使用其私有 IP 地址連接到 VPC 私有子網路內的主機。它可以透過查看由 Amazon ECS Service Discovery 管理 AWS Cloud Map 的服務探索記錄來偵測這些私有 IP 地址。

API Gateway 支援以下功能。

- API Gateway 的運作方式類似負載平衡器，但具備 API 管理專屬的額外功能
- API Gateway 提供用戶端授權、用量方案與請求/回應修改等額外功能。如需詳細資訊，請參閱 [Amazon API Gateway features](#)。
- API Gateway 可以支援邊緣、區域與私有 API 閘道端點。Edge 端點可透過受管的 CloudFront 分發取得。區域端點與私有端點皆為特定區域的本機端點。
- SSL/TLS 終止
- 將不同的 HTTP 路徑路由至不同的後端微服務

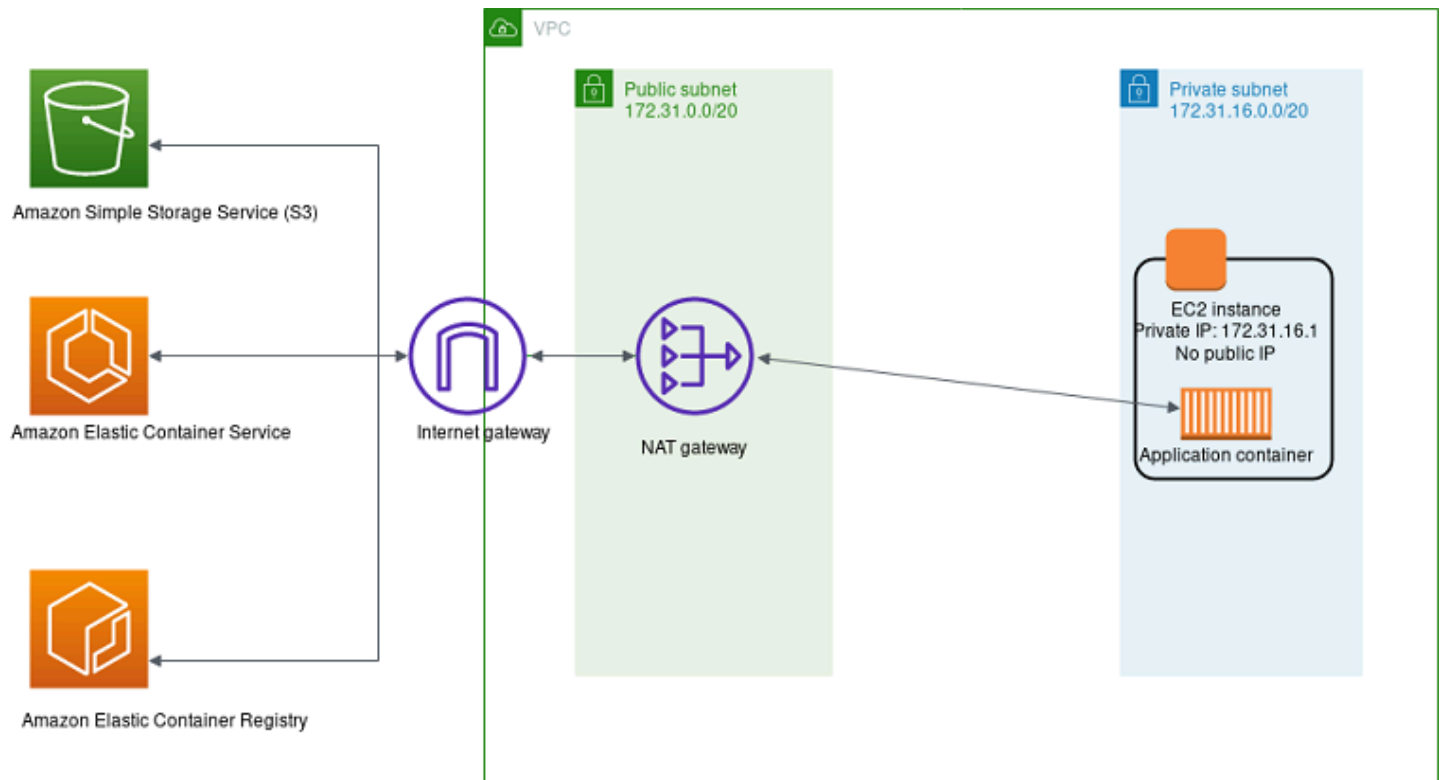
除上述功能外，API Gateway 還支援使用自訂 Lambda 授權方，您可以使用這些授權方來保護 API 免受未經授權的使用。如需詳細資訊，請參閱 [Field Notes: Serverless Container-based APIs with Amazon ECS and Amazon API Gateway](#)。

從 VPC 內部將 Amazon ECS 連線至 AWS 服務的最佳實務

為使 Amazon ECS 正常運作，執行於各主機上的 Amazon ECS 容器代理程式必須與 Amazon ECS 控制平面保持通訊。如果要將容器映像儲存於 Amazon ECR 中，Amazon EC2 主機必須與 Amazon ECR 服務端點以及儲存映像層的 Amazon S3 通訊。如果您將其他 AWS 服務用於容器化應用程式，例如保留存放在 DynamoDB 中的資料，請仔細檢查這些服務是否也具有必要的聯網支援。

NAT 閘道

使用 NAT 閘道是確保您的 Amazon ECS 任務可以存取其他 AWS 服務最簡單的方式。如需有關此方法的詳細資訊，請參閱[私有子網路和 NAT 閘道](#)。



採用此方法具有以下劣勢：

- 您無法限制 NAT 閘道可與哪些目的地通訊。若不中斷 VPC 的所有傳出通訊，您也無法限制後端層可與哪些目的地通訊。
- NAT 閘道會針對每 GB 的傳輸資料量收取費用。如果將 NAT 閘道用於下列任何操作，則需為每 GB 的頻寬付費：
 - 從 Amazon S3 下載大型檔案
 - 對 DynamoDB 執行高頻率資料庫查詢
 - 從 Amazon ECR 提取映像

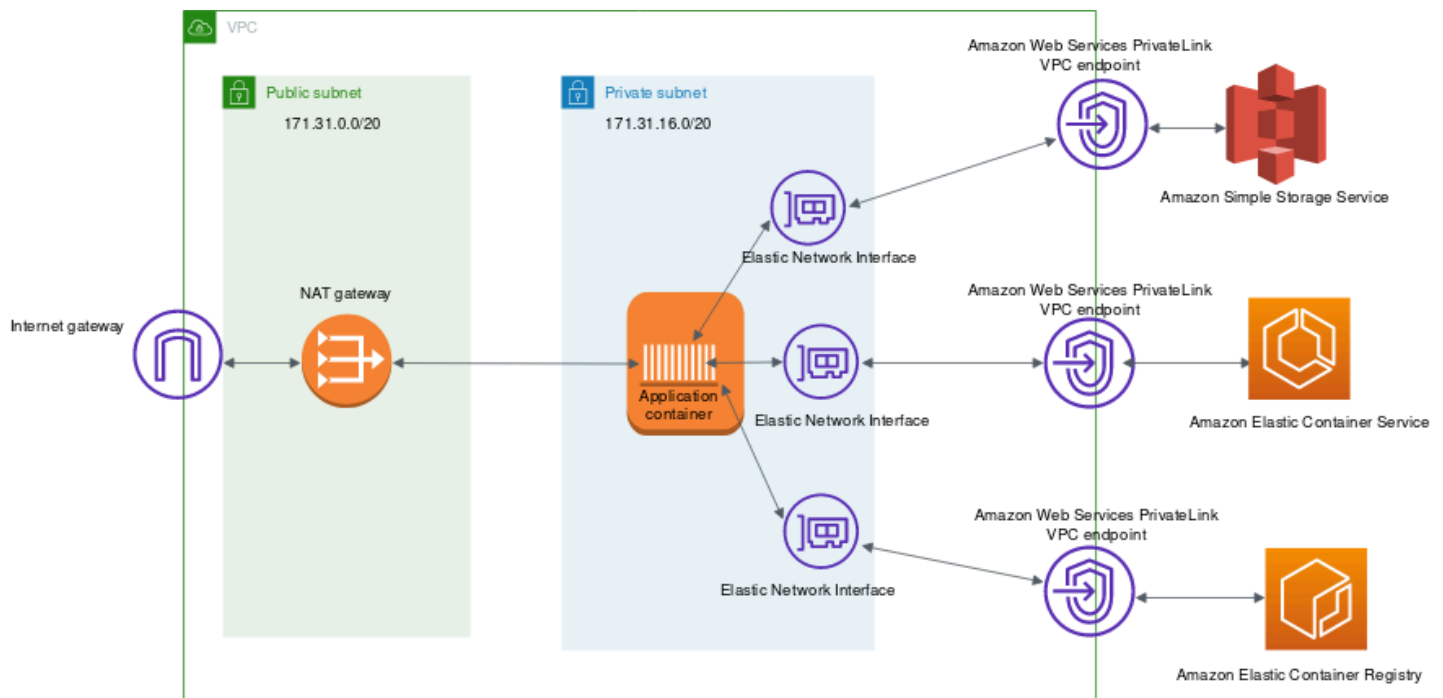
此外，NAT 閘道支援 5 Gbps 的頻寬，並可自動向上擴展至 45 Gbps。如果您透過單一 NAT 閘道進行路由，需要非常高頻寬連線的應用程式可能會遇到聯網限制。作為替代方案，您可以將工作負載分散至多個子網路，並為每個子網路設定專屬的 NAT 閘道。

AWS PrivateLink

AWS PrivateLink 在 VPCs、AWS 服務和內部部署網路之間提供私有連線，而不會將您的流量暴露到公有網際網路。

VPC 端點可讓您的 VPC 與受支援的 AWS 服務及 VPC 端點服務之間進行私有連線。VPC 與另一個服務之間的流量都會保持在 Amazon 網路的範圍內。VPC 端點不需要網際網路閘道、虛擬私有閘道、NAT 裝置、VPN 連線或 Direct Connect 連線。VPC 中的 Amazon EC2 執行個體不需要公有 IP 位址，即可與服務中的資源進行通訊。

下圖顯示當您使用 VPC 端點而非網際網路閘道時，AWS 服務通訊的運作方式。會在子網路內 AWS PrivateLink 佈建彈性網路介面 (ENIs)，而 VPC 路由規則則用來透過 ENI 將任何通訊直接傳送至目的地 AWS 服務。此類流量不再需要經過 NAT 閘道或網際網路閘道。



下列是與 Amazon ECS 服務搭配使用的部分常見 VPC 端點。

- [Amazon S3 閘道端點](#)
- [DynamoDB VPC 端點](#)
- [Amazon ECS VPC 端點](#)
- [Amazon ECR VPC 端點](#)

許多 AWS 其他服務都支援 VPC 端點。若頻繁使用某項 AWS 服務，您應查閱該服務的專屬文件，了解如何為其流量建立 VPC 端點。

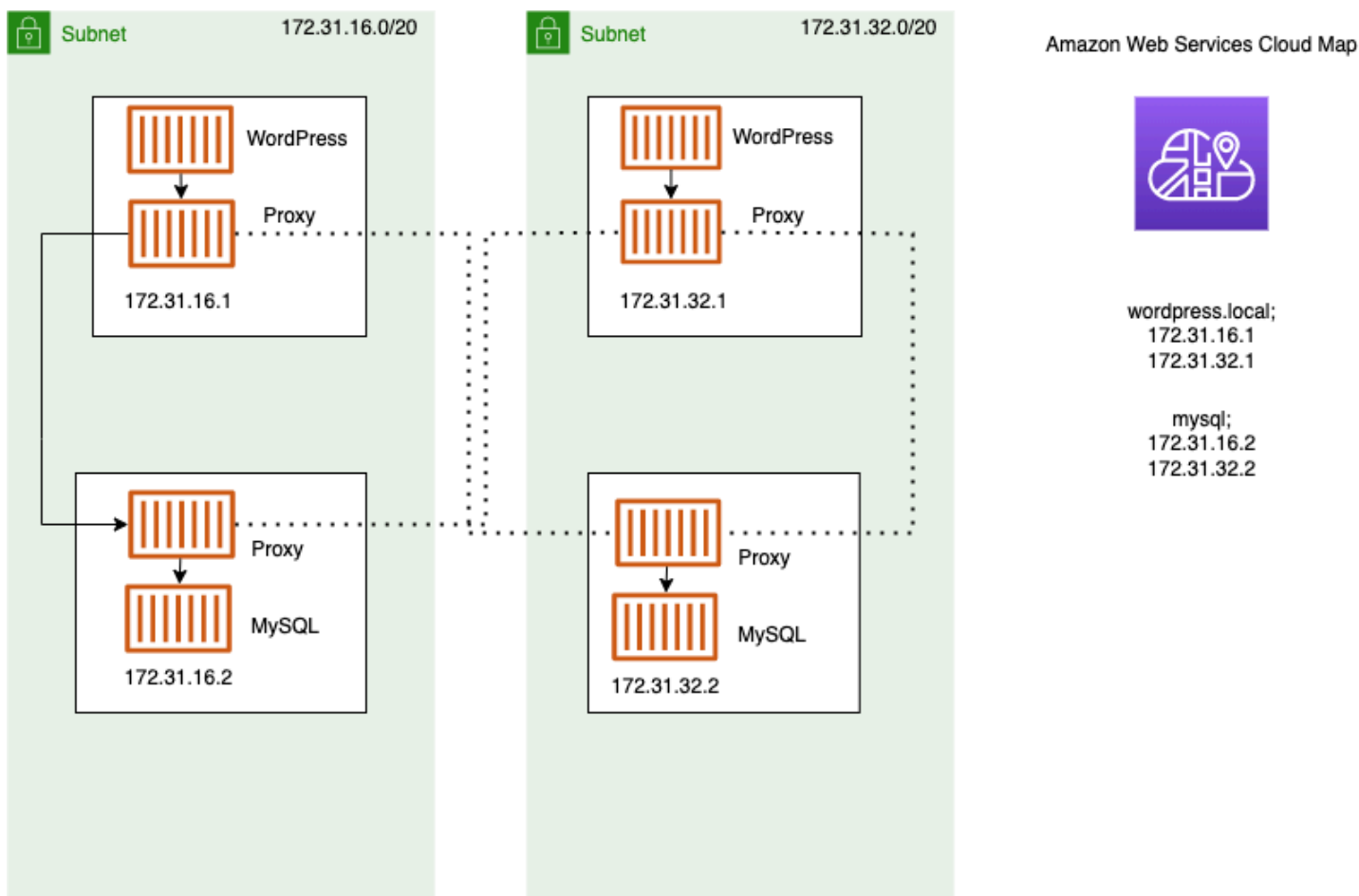
在 VPC 中連接 Amazon ECS 服務的最佳實務

您可以在 VPC 中使用 Amazon ECS 任務，將單體式應用程式分割成可在安全環境中獨立部署與擴展的獨立元件。此架構即稱為服務導向架構 (SOA) 或微服務。然而，要確保 VPC 內外的所有元件都能相互通訊，可能具有挑戰性。實現通訊的方法有多種，每種方法都有不同的優勢與劣勢。

使用 Service Connect

建議使用 Service Connect，其可提供 Amazon ECS 組態，用於服務探索、連線與流量監控。透過 Service Connect，您的應用程式可以使用短名稱與標準連接埠，連線至相同叢集、其他叢集 (包括同一區域內跨 VPC 的叢集) 中的服務。如需詳細資訊，請參閱 [Amazon ECS Service Connect](#)。

當您使用 Service Connect 時，Amazon ECS 將管理服務探索的所有環節：建立可被探索的名稱、在任務啟動與停止時動態管理每個任務的項目、在每個已設定為可探索這些名稱的任務中執行代理程式。應用程式可透過 DNS 名稱的標準功能查詢這些名稱並建立連線。若應用程式已具備此能力，則您無需修改應用程式即可使用 Service Connect。

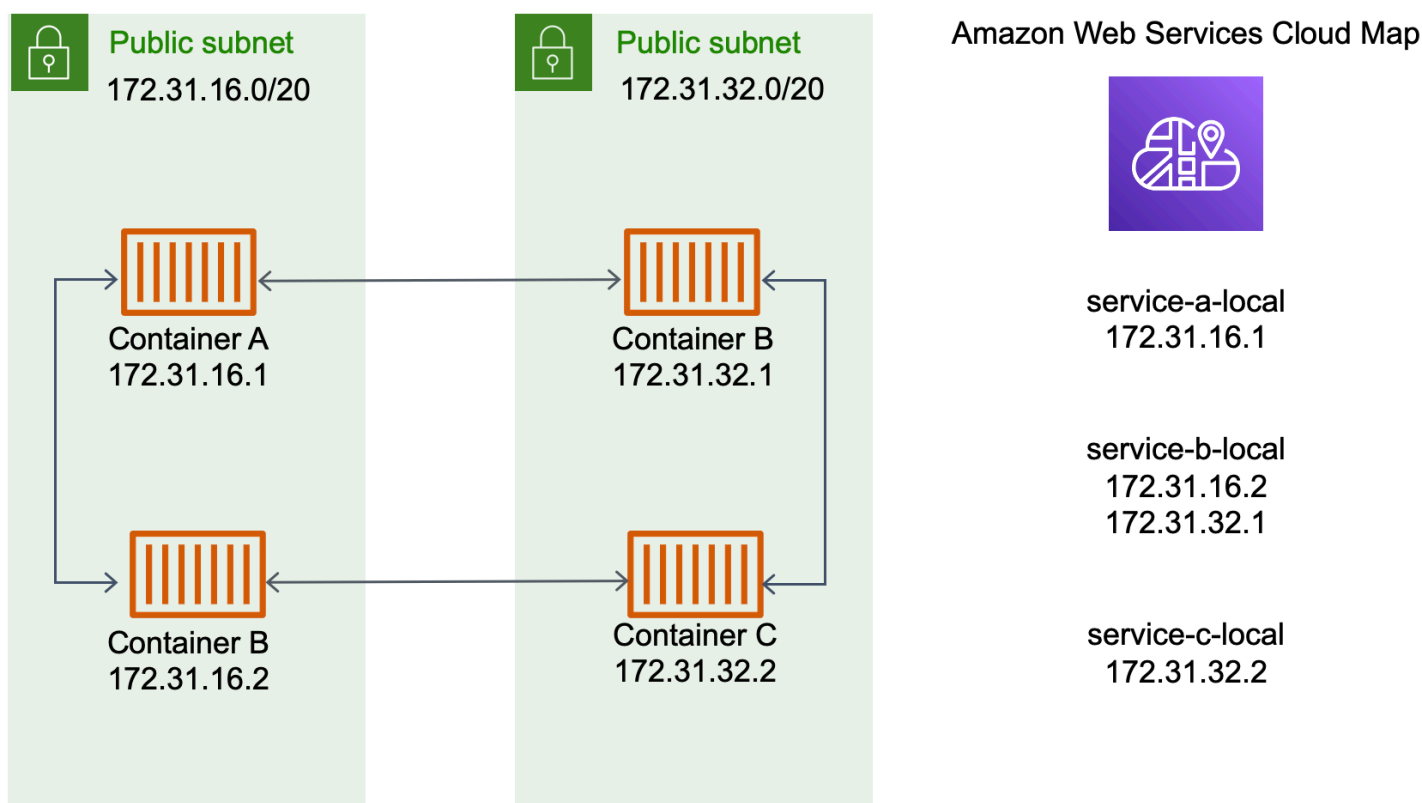


變更僅會在部署期間發生

您需在每個服務與任務定義內提供完整的組態。Amazon ECS 會在每次服務部署時管理此組態的變更，確保統一部署中的所有任務行為一致。例如，使用 DNS 作為服務探索機制時，常見問題在於遷移過程難以控制。若變更 DNS 名稱指向的新替代 IP 位址，可能需要等待最長 TTL 時間，所有用戶端才會開始使用新服務。而透過 Service Connect，用戶端部署會透過取代用戶端任務來更新組態。您可以設定部署斷路器及其他部署組態，使 Service Connect 的變更與任何其他部署一樣受到同等管控。

使用服務探索

服務間通訊的另一種方法是使用服務探索進行直接通訊。在此方法中，您可以使用 AWS Cloud Map 服務探索與 Amazon ECS 的整合。使用服務探索，Amazon ECS 會將已啟動任務的清單同步至 AWS Cloud Map，這會維護 DNS 主機名稱，以解析該特定服務中一或多個任務的內部 IP 地址。Amazon VPC 中的其他服務可使用此 DNS 主機名稱，透過容器的內部 IP 位址直接向另一個容器傳送流量。如需詳細資訊，請參閱[服務探索](#)。



上圖中共有三個服務。service-a-local 包含一個容器，需與包含兩個容器的 service-b-local 通訊。service-b-local 必須與包含一個容器的 service-c-local 通訊。所有三個服務中的每個容器都可以使用來自的內部 DNS 名稱 AWS Cloud Map，從其通訊所需的下游服務尋找容器的內部 IP 地址。

這種服務間通訊方法的延遲較低。乍看之下，這種方式同樣簡單，因為容器之間沒有額外元件。流量會從一個容器直接傳送至另一個容器。

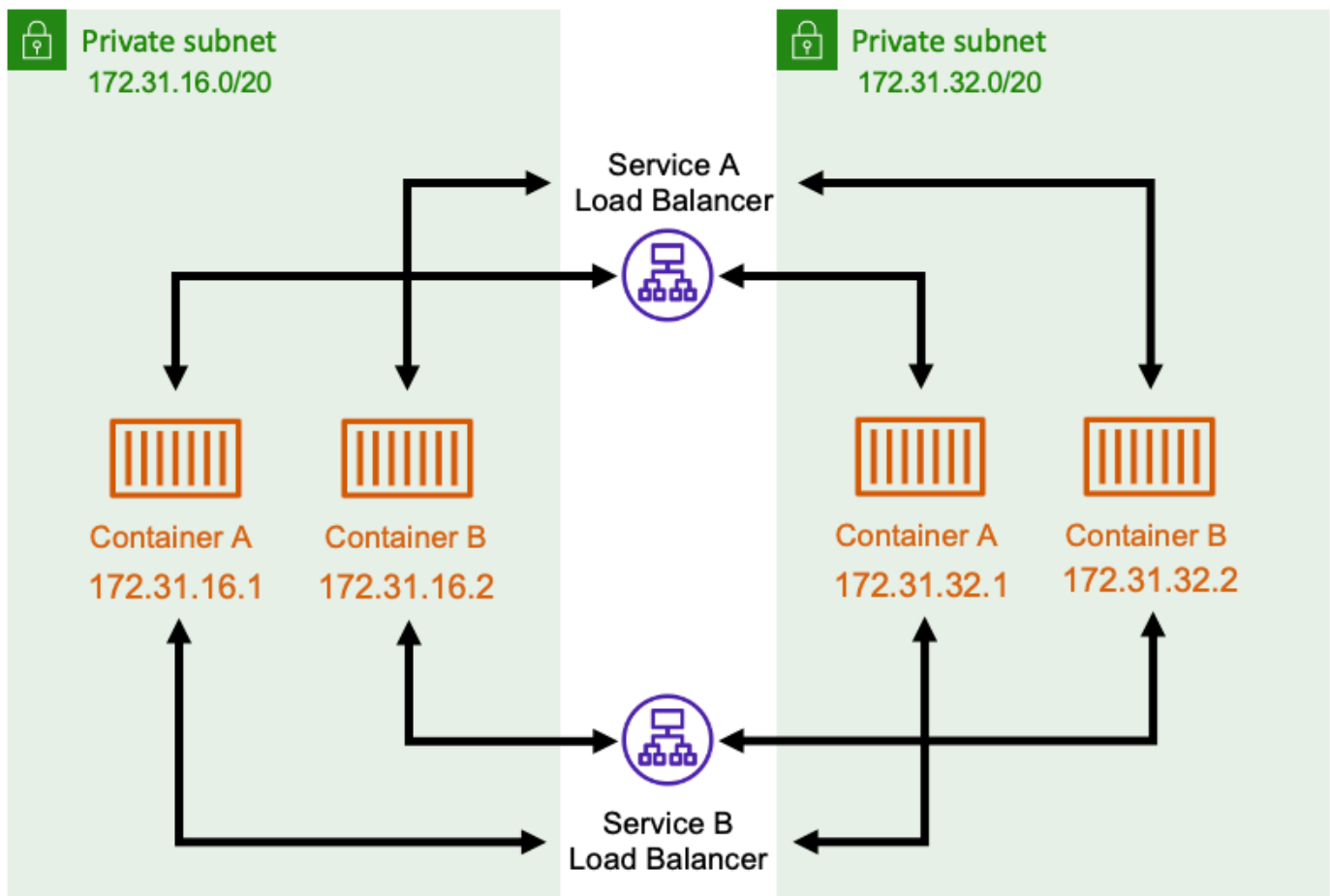
此方法適用於採用 `awsvpc` 網路模式的場景，該模式下每個任務都有專屬的唯一 IP 位址。大多數軟體僅支援使用 DNS A 記錄，這些記錄會直接解析為 IP 位址。使用 `awsvpc` 網路模式時，每個任務的 IP 位址都是 A 記錄。不過，如果您使用 `bridge` 網路模式，多個容器可能會共用同一個 IP 位址。此外，動態連接埠映射會導致容器在該單一 IP 位址上隨機指派連接埠號碼。至此，僅靠 A 記錄已不足以滿足服務探索需求。您必須同時使用 SRV 記錄。這種類型的記錄可以追蹤 IP 位址與連接埠號碼，但要求對應用程式進行相應設定。您使用的某些預先建置的應用程式可能不支援 SRV 記錄。

`awsvpc` 網路模式的另一項優勢，是每個服務都會擁有唯一的安全群組。您可以設定此安全群組，使其僅允許需要與該服務通訊的特定上游服務建立連入連線。

使用服務探索實現服務間直接通訊的主要劣勢在於，您必須實作額外邏輯來處理重試機制與連線失敗問題。DNS 記錄具有存留時間 (TTL) 設定，可控制記錄的快取存留時長。DNS 記錄的更新與快取失效需要一定時間，才能讓應用程式取得最新版本的 DNS 記錄。因此，應用程式最終可能會將 DNS 記錄解析至已不存在的另一個容器。應用程式需要處理重試，並具有忽略不良後端的邏輯。

使用內部負載平衡器

服務間通訊的另一種方法是使用內部負載平衡器。內部負載平衡器完全存在於 VPC 內，且僅供 VPC 內部的服務存取。



負載平衡器透過將備援資源部署至每個子網路來維持高可用性。當 serviceA 中的容器需要與 serviceB 中的容器通訊時，會開啟與負載平衡器的連線。然後，負載平衡器會開啟與 service B 中容器的連線。負載平衡器可作為管理每個服務之間所有連線的集中位置。

如果 serviceB 中的容器停止，則負載平衡器可以從集區中移除該容器。負載平衡器也會針對其集區中的每個下游目標執行運作狀態檢查，並可自動從集區中移除不良目標，直至這些目標恢復正常運作。應用程式不再需要知道有多少個下游容器，只要開啟與負載平衡器的連線。

此方法對所有網路模式都具有優勢。負載平衡器可以在使用 awsvpc 網路模式時追蹤任務 IP 位址，也可以在使用 bridge 網路模式時追蹤更進階的 IP 位址與連接埠組合。其將流量平均分配發至所有 IP 位址與連接埠組合，即使多個容器實際上託管於同一 Amazon EC2 執行個體上，只是使用不同的連接埠。

此方法的一個劣勢是成本較高。若要做到高度可用，負載平衡器需要在每個可用區域中擁有資源。這會增加額外成本，因為需負擔負載平衡器本身的開銷，以及流經負載平衡器的流量所產生的費用。

不過，您可透過讓多個服務共用一個負載平衡器來降低間接成本。這種做法特別適合使用 Application Load Balancer 的 REST 服務。您可以建立路徑型路由規則，將流量路由至不同的服務。例如，`/api/user/*` 可能會路由至屬於 `user` 服務的容器，而 `/api/order/*` 可能會路由至相關聯的 `order` 服務。使用此方法，您只需支付一個 Application Load Balancer 的費用，並為您的 API 準備一個一致的 URL。不過，您仍可將流量分流至後端的各個微服務。

跨 AWS 帳戶 和 VPCs 聯網 Amazon ECS 服務的最佳實務

如果您是擁有多個團隊和部門的組織的一部分，您可能會將服務獨立部署到共用內部 VPCs，AWS 帳戶 或部署到與多個個人相關聯的 VPCs AWS 帳戶。無論部署服務的方式為何，我們都建議補充聯網元件，協助在 VPC 之間路由流量。因此，您可以使用數個 AWS 服務來補充現有的網路元件。

- AWS 傳輸閘道 — 您應該先考慮此聯網服務。此服務可做為中央中樞 AWS 帳戶，用於在 Amazon VPCs 和內部部署網路之間路由連線。如需詳細資訊，請參閱 Amazon VPC Transit Gateways Guide 中的 [What is a transit gateway?](#) 章節。
- Amazon VPC 與 VPN 支援：您可以使用此服務建立站對站 VPN 連線，將內部部署網路連線至 VPC。如需詳細資訊，請參閱 AWS Site-to-Site VPN 《使用者指南》中的 [什麼是 AWS Site-to-Site VPN ?](#)。
- Amazon VPC：您可以使用 Amazon VPC 對等互連，協助自己在相同帳戶中或跨帳戶連接多個 VPC。如需詳細資訊，請參閱《Amazon VPC Peering Guide》中的 [什麼是 VPC 互連 ?](#)。
- 共用 VPC：您可以在多個 AWS 帳戶中使用 VPC 與 VPC 子網路。如需詳細資訊，請參閱 Amazon VPC User Guide 中的 [Working with shared VPCs](#)。

AWS Amazon ECS 網路疑難排解的 服務

下列服務與功能可協助您深入了解網路與服務組態。您可以使用這些資訊對聯網問題進行疑難排解，更有效地最佳化自己的服務。

CloudWatch Container Insights

CloudWatch Container Insights 會從您的容器化應用程式和微型服務收集、彙總及總結指標和日誌。指標包含 CPU、記憶體、磁碟與網路等資源的使用率。這些指標可在 CloudWatch 自動儀表板中取得。如需詳細資訊，請參閱 Amazon CloudWatch User Guide 中的 [Setting up Container Insights on Amazon ECS](#)。

AWS X-Ray

AWS X-Ray 是一種追蹤服務，可用來收集應用程式提出之網路請求的相關資訊。您可以使用 SDK 來檢測您的應用程式，並擷取服務之間，以及服務 AWS 與服務端點之間的流量時間和回應代碼。如需詳細資訊，請參閱 AWS X-Ray 開發人員指南中的[什麼是 AWS X-Ray](#)。

您也可以探索 服務如何彼此聯網的 AWS X-Ray 圖表。或者，利用這些圖表來探索每個服務間連結運作狀況的彙總統計量。最後，您可以深入探查任何特定交易，查看代表網路呼叫的區段如何與該特定交易建立關聯。

您可以使用這些功能來識別是否有聯網瓶頸，或網路中的特定服務是否未如預期般執行。

VPC 流量日誌

您可以使用 Amazon VPC 流程日誌來分析網路效能與偵錯連線問題。啟用 VPC 流程日誌後，您即可擷取 VPC 中所有連線的日誌。這包括與 Elastic Load Balancing、Amazon RDS、NAT 閘道以及您可能正在使用的其他金鑰 AWS 服務相關聯的聯網介面連線。如需詳細資訊，請參閱「Amazon VPC 使用者指南」中的[VPC 流程日誌](#)。

網路調校秘訣

您可以微調部分設定，改善聯網效能。

nofile ulimit

如果預期應用程式具有高流量並處理許多並行連線，您應該考慮到允許的檔案數量的系統配額。當系統中存在大量開啟的網路通訊端時，每個通訊端都必須以檔案描述項表示。如果檔案描述項配額太低，會限制網路通訊端數量。這會導致連線失敗或錯誤。您可以 Amazon ECS 任務定義中，更新容器專屬的檔案數量配額。如果您在 Amazon EC2（而非 AWS Fargate）上執行，則您可能也需要在基礎 Amazon EC2 執行個體上調整這些配額。

sysctl 網路

另一類可調校的設定是 sysctl 網路設定。您應該參考所選 Linux 發行版本的特定設定說明。這類設定多用於調整讀取與寫入緩衝區的大小。在執行規模龐大且其上有大量容器的 Amazon EC2 執行個體時，此調整在某些情況下會有所幫助。

最佳化 Amazon ECS 任務啟動時間

為了加速任務啟動，建議考量下列事項。

- 快取容器映像與 binpack 執行個體

如果使用 EC2，您可以將 Amazon ECS 容器代理程式提取行為設定為 `ECS_IMAGE_PULL_BEHAVIOR: prefer-cached`。如果沒有快取映像，將會從遠端提取映像。否則，將使用執行個體上的快取映像。為了確保快取映像不會被移除，容器會停用自動映像檔清理功能。如此便能縮減後續啟動時的映像檔拉取時間。當容器執行個體具有高任務密度時，快取的效果會更大，您可以使用 binpack 置放策略來設定這種高密度任務。快取容器映像特別適用於通常具有大型 (數十 GB) 容器映像大小的 Windows 型工作負載。使用 binpack 置放策略時，您也可以考慮使用彈性網路介面 (ENI) 主幹功能，在每個容器執行個體上置放採用 awsvpc 網路模式的更多任務。ENI 主幹功能會增加可在 awsvpc 模式下執行的任務數量。例如，一個原本僅能同時執行 2 個任務的 c5.large 執行個體，在啟用 ENI 主幹功能後，可執行的任務數量上限將提升至 10 個。

- 選擇最佳網路模式

雖然在許多情況下 awsvpc 網路模式都是理想的選擇，但此網路模式本質上可以增加任務啟動延遲，因為對於 awsvpc 模式中的每個任務，Amazon ECS 工作流程需要透過調用 Amazon EC2 API 來佈建與連接 ENI，這會為任務啟動增加數秒的額外負荷。相反地，使用 awsvpc 網路模式的主要優勢是每個任務都有安全群組來允許或拒絕流量。這表示您能以更精細的層級，彈性控制任務與服務之間的通訊。如果部署速度是優先考量，您可以考慮使用 bridge 模式來加速任務啟動。如需詳細資訊，請參閱[the section called “AWSVPC 網路模式”](#)。

- 追蹤任務啟動生命週期以尋找最佳化機會

應用程式的啟動時間往往難以掌握。在應用程式啟動過程中，啟動容器映像、執行啟動指令碼及其他組態，都可能耗費大量時間。您可以使用任務中繼資料端點來發布指標，追蹤應用程式的啟動時間 (從容器中繼資料回應中的 `StartedAt`，到任務或服務的 `StartedAt` 時間)。透過這些資料，您能明確掌握應用程式對整體啟動時間的影響，並找出可減少不必要的應用程式專屬額外負荷的環節，進一步最佳化容器映像。如需詳細資訊，請參閱[Amazon ECS 自動擴展與容量管理最佳實務](#)。

- 選擇最佳執行個體類型 (針對 EC2)

根據您在任務上設定的資源保留量 (例如 CPU、記憶體)，選擇正確的執行個體類型。因此，調整執行個體大小時，您能計算可在單一執行個體上置放多少個任務。妥善置放任務的簡單範例：在 m5.large 執行個體 (支援 2 個 vCPU 與 8 GB 記憶體) 中託管 4 個任務，每個任務需要 0.5 個 vCPU 與 2 GB 的記憶體保留量。此任務定義的保留量會充分利用執行個體的資源。

大規模運作 Amazon ECS

當您開始大規模操作 Amazon ECS 時，請考慮 Amazon ECS 的服務配額和 API 調節，以及與 Amazon ECS 整合 AWS 服務的 可能如何影響您。

主題

- [Amazon ECS 服務配額與 API 限流限制](#)
- [因應 Amazon ECS 限流問題](#)

Amazon ECS 服務配額與 API 限流限制

Amazon ECS 與數個 整合 AWS 服務，包括 Elastic Load Balancing AWS Cloud Map 和 Amazon EC2。透過這種緊密整合，Amazon ECS 包含數個功能，例如服務負載平衡、服務探索、任務聯網與叢集自動擴展。Amazon ECS 及其與所有維護服務配額和 API 速率限制整合 AWS 服務 的另一個 ECS，以確保一致的效能和使用率。這些服務配額也可防止意外佈建超過所需的資源，並防範可能導致帳單金額增加的惡意行為。

透過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 服務配額](#) 與 [Request throttling for the Amazon ECS API](#)。

在 Amazon ECS 上擴展工作負載時，請考量下列服務配額。如需如何請求提高服務配額的指示，請參閱 [《》中的管理您的 Amazon ECS AWS Fargate 和服務配額 AWS 管理主控台](#)。

- AWS Fargate 具有配額，可限制每個 中並行執行中的任務數量 AWS 區域。Amazon ECS 上的隨需任務與 Fargate Spot 任務都有配額。每個服務配額的計算範圍，亦涵蓋您在 Fargate 上執行的所有 Amazon EKS Pod。如需有關 Fargate 配額的詳細資訊，請參閱 Amazon Elastic Container Service User Guide for AWS Fargate 中的 [AWS Fargate service quotas](#)。
- 對於在 Amazon EC2 執行個體上執行的任務，每個叢集可註冊的 Amazon EC2 執行個體數量上限為 5,000 個。如果搭配 Auto Scaling 群組容量提供者使用 Amazon ECS 叢集自動擴展，或者自行管理叢集的 Amazon EC2 執行個體，則此配額可能會成為部署瓶頸。如果需要更多容量，您可以建立更多叢集或請求提高服務配額。
- 如果搭配 Auto Scaling 群組容量提供者使用 Amazon ECS Tasks in the PROVISIONING state per cluster 叢集自動擴展，請在擴展服務時將配額納入考量。此配額指每個叢集中處於 PROVISIONING 狀態的任務數上限，容量提供者可針對這些任務增加容量。當同時啟動大量任務時，就很容易達到此配額上限。若同時部署數十個服務，每個服務包含數百個任務，就可能出現這種情況。此時，若叢集容量不足，容量提供者需啟動新的容器執行個體來置放這些任務。在容量提供者啟動額外 Amazon EC2 執行個體的同時，Amazon ECS 服務排程器通常會持續平行啟動任務。但由於叢集容量不足，此操作可能會受到限流。隨著新容器執行個體的啟動，Amazon ECS 服務排程器會採用退避與指數限流策略，重試任務置放操作。因此，您可能會遇到部署或擴展時間延長的情況。若要避免此狀況，可透過以下任一方式規劃服務部署。部署大量無需擴充叢集容量的任務，或者為新任務啟動保留備用叢集容量。

除了在擴展工作負載時考慮 Amazon ECS 服務配額之外，也請考慮與 Amazon ECS AWS 服務 整合之其他的服務配額。下一節將詳細介紹各服務的關鍵速率限制，並提供應對潛在限流問題的建議。

Elastic Load Balancing

您可以設定 Amazon ECS 服務使用 Elastic Load Balancing，將流量平均分發至所有任務。如需有關如何選擇負載平衡器的詳細資訊，請參閱[使用負載平衡分發 Amazon ECS 服務流量](#)。

Elastic Load Balancing 服務配額

在擴展工作負載時，建議考量下列 Elastic Load Balancing 服務配額。大多數 Elastic Load Balancing 服務配額都可調整，您可透過 Service Quotas 主控台申請提高上限。

Application Load Balancer

在使用 Application Load Balancer 時，根據實際使用案例，可能需要申請提高以下項目的配額：

- Targets per Application Load Balancer 配額，即 Application Load Balancer 後端的目標數量。
- Targets per Target Group per Region 配額，即目標群組後端的目標數量。

如需詳細資訊，請參閱 [Quotas for your Application Load Balancers](#)。

Network Load Balancer

Network Load Balancer 的目標註冊數量存在更嚴格的限制。使用 Network Load Balancer 時，通常建議啟用跨可用區域支援功能，但此功能此功能會對 Targets per Availability Zone Per Network Load Balancer 帶來額外的擴展限制，亦即每個 Network Load Balancer 於各可用區域中可容納的目標數量上限將受到規範。如需詳細資訊，請參閱 [Quotas for your Network Load Balancers](#)。

Elastic Load Balancing API 限流

在設定 Amazon ECS 服務使用負載平衡器時，必須先通過目標群組運作狀態檢查，才能將服務視為正常運作。為了執行這些運作狀態檢查，Amazon ECS 會代為調用 Elastic Load Balancing API 操作。如果帳戶中有大量使用負載平衡器設定的服務，可能會因 RegisterTarget、DeregisterTarget 與 DescribeTargetHealth 這幾項 Elastic Load Balancing API 操作遭遇限流，導致服務部署速度減緩。發生限流時，Amazon ECS 服務事件訊息中會發生限流錯誤。

如果您遇到 AWS Cloud Map API 限流，您可以聯絡 支援 以取得有關如何提高 AWS Cloud Map API 限流限制的指引。如需有關監控及疑難排解此類限流錯誤的詳細資訊，請參閱[因應限流問題](#)。

彈性網路介面

隨著任務使用 `awsvpc` 網路模式，Amazon ECS 會為每個任務佈建唯一的彈性網路介面 (ENI)。當 Amazon ECS 服務使用 Elastic Load Balancing 負載平衡器時，這些網路介面也會註冊為該服務中定義的對應目標群組的目標。

彈性網路介面服務配額

當您執行使用 `awsvpc` 網路模式的任務時，每個任務都會連接唯一的彈性網路介面。如果必須透過國際網路到達這些任務，請將公有 IP 位址指派給這些任務的彈性網路介面。在擴展 Amazon ECS 工作負載時，請考量這兩個重要的配額：

- Network interfaces per Region 配額，這是中 AWS 區域您帳戶的最大網路介面數量。
- Elastic IP addresses per Region 配額，即 AWS 區域中彈性 IP 位址的最大數量。

這兩個服務配額皆可調整，您可以直接透過 Service Quotas 主控台提出調增申請。如需詳細資訊，請參閱 [Amazon VPC service quotas](#)。

對於託管在 Amazon EC2 執行個體上的 Amazon ECS 工作負載，當執行使用 `awsvpc` 網路模式的任務時，建議考量 Maximum network interfaces 服務配額，即每個 Amazon EC2 執行個體的網路執行個體數量上限。此配額會限制可在執行個體上置放的任務數量。此配額無法調整，就未於 Service Quotas 主控台中提供。如需詳細資訊，請參閱 Amazon EC2 User Guide 中的 [IP addresses per network interface per instance type](#)。

雖然您無法變更可連接至 Amazon EC2 執行個體的網路介面數量，但可透過啟用彈性網路介面主幹功能來增加可用網路介面的總數。例如，依預設，`c5.large` 執行個體最多可有 3 個網路介面。執行個體的主要網路介面視為一個配額。因此，您可以將額外的兩個網路介面連接至執行個體。因為每個使用 `awsvpc` 網路模式的任務都需要網路介面，所以通常只能對此執行個體類型執行兩個這類任務。這可能會導致叢集容量的使用率偏低。如果啟用彈性網路介面主幹功能，即可提高網路介面密度，從而在每個執行個體上部署更多任務。啟用主幹功能後，`c5.large` 執行個體最多可有 12 個網路介面。該執行個體本身已具備主要網路介面，而 Amazon ECS 會建立並連接「主幹」網路介面至該執行個體。因此，透過此組態，您可在該執行個體上執行 10 個任務，而非預設的 2 個任務。如需詳細資訊，請參閱 [彈性網路介面主幹功能](#)。

彈性網路介面 API 限流

在執行使用 `awsvpc` 網路模式的任務時，Amazon ECS 會調用下列 Amazon EC2 API。每個 API 皆設有不同的 API 限流。如需詳細資訊，請參閱 [Request throttling for the Amazon EC2 API](#)。

- CreateNetworkInterface
- AttachNetworkInterface
- DetachNetworkInterface
- DeleteNetworkInterface
- DescribeNetworkInterfaces
- DescribeVpcs
- DescribeSubnets
- DescribeSecurityGroups
- DescribeInstances

若在彈性網路介面佈建工作流程中遭遇 Amazon EC2 API 呼叫限流，Amazon ECS 服務排程器會自動採用指數退避機制進行重試。這些自動重試有時可能導致任務啟動延遲，進而造成部署速度減緩。當發生 API 限流時，服務事件訊息中會顯示 `Operations are being throttled. Will try again later.`。如果您持續符合 Amazon EC2 API 限流，您可以聯絡支援以取得有關如何提高 API 限流限制的指引。如需有關監控及疑難排解限流錯誤的詳細資訊，請參閱[因應限流問題](#)。

AWS Cloud Map

Amazon ECS 服務探索使用 AWS Cloud Map APIs 來管理 Amazon ECS 服務的命名空間。如果服務有大量任務，建議考量下列事項。如需詳細資訊，請參閱[Amazon ECS 服務探索考量事項](#)。

AWS Cloud Map 服務配額

當 Amazon ECS `Tasks per service` 服務設定為使用服務探索時，服務任務數量上限的配額會受到 AWS Cloud Map `Instances per service` 該服務執行個體數量上限的服務配額影響。特別是，AWS Cloud Map 服務配額會將您可以執行的任務量減少到每個服務最多 1,000 個任務。AWS Cloud Map 配額無法變更。如需更多相關資訊，請參閱[AWS Cloud Map Service Quotas](#)。

AWS Cloud Map API 限流

Amazon ECS 會代表您呼叫 `ListInstances`、`RegisterInstance`、`GetInstancesHealthStatus` 和 `DeregisterInstance` AWS Cloud Map APIs。這些 API 可用於服務探索，會在您啟動任務時執行運作狀態檢查。當同時使用服務探索與大量任務的多個服務同時部署時，這可能會導致超過 AWS Cloud Map API 限流限制。此時，您可能會在 Amazon ECS 服務事件訊息中看到訊息 (`Operations are being throttled. Will try again later`)，且部署與任務啟動速度會減緩。AWS Cloud Map 未記錄這些 API 的限流限制。如果您遇到這些調節，您可以

聯絡 支援 以取得提高 API 調節限制的指引。如需有關監控及疑難排解此類限流錯誤的更多建議，請參閱[因應限流問題](#)。

因應 Amazon ECS 限流問題

限流錯誤分為兩個主要類別：同步限流與非同步限流。

同步限流

發生同步限流時，您會立即收到來自 Amazon ECS 的錯誤回應。如果您在執行任務或建立服務時呼叫 Amazon ECS API，通常會發生此類別錯誤。如需有關所涉限流問題與相關限流限制的詳細資訊，請參閱[Request throttling for the Amazon ECS API](#)。

當您的應用程式啟動 API 請求時，例如使用 AWS CLI 或 AWS SDK，您可以修復 API 限流。具體可透過兩種方式實現：一是將應用程式架構設計為可處理這些錯誤，二是為 API 呼叫視作具有重試邏輯的指數退避與抖動策略。如需詳細資訊，請參閱[Timeouts, retries, and backoff with jitter](#)。

如果您使用 AWS SDK，則自動重試邏輯是內建且可設定的。

非同步限流

非同步限流的發生源於非同步工作流程，在該流程中，Amazon ECS 或 CloudFormation 可能會代表您呼叫 API 來佈建資源。請務必了解 Amazon ECS 代您叫用哪些 AWS APIs。例如，對於使用 awsvpc 網路模式的任務，系統會調用 CreateNetworkInterface API；而對已註冊至負載平衡器的任務執行運作狀態檢查時，則會調用 DescribeTargetHealth API。

當工作負載達到相當大的規模時，這些 API 操作可能會受到限流。也就是說，它們可能會受到足夠限流，違反 Amazon ECS 或正在呼叫 AWS 服務的強制執行的限制。例如，若同時部署數百個服務，每個服務都有數百個使用 awsvpc 網路模式的任務，Amazon ECS 會分別呼叫 Amazon EC2 的 API 操作 (如 CreateNetworkInterface) 以及 Elastic Load Balancing 的 API 操作 (如 RegisterTarget 或 DescribeTargetHealth)，完成彈性網路介面與負載平衡器的註冊工作。這些 API 呼叫可能會超過 API 限制，導致限流錯誤。以下是服務事件訊息中包含的 Elastic Load Balancing 限流錯誤範例。

```
{
  "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，即使其作為服務事件發出，仍可能難以監控。

監控限流

請務必確定受到限流的具體 API 請求以及請求發起來源。您可以使用 AWS CloudTrail 來監控限流，並與 CloudWatch、Amazon Athena 和 Amazon EventBridge 整合。您可以設定 CloudTrail 將特定事件傳送至 CloudWatch Logs。CloudWatch Logs 日誌洞察會剖析並分析事件。這可確定限流事件中的詳細資訊，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 [Monitoring CloudTrail log files with CloudWatch Logs](#)。

如需有關使用 CloudWatch Logs Insights 的詳細資訊以及有關如何查詢日誌檔案的說明，請參閱 [Analyzing log data with CloudWatch Logs Insights](#)。

透過 Amazon Athena，您可以使用標準 SQL 建立查詢並分析資料。例如，您可以建立 Athena 表格來剖析 CloudTrail 事件。如需詳細資訊，請參閱 [Using the CloudTrail console to create an Athena table for CloudTrail logs](#)。

建立 Athena 表格之後，您可以使用 SQL 查詢 (例如下列查詢) 來調查 ThrottlingException 錯誤。

將 *user-input* 取代為實際值。

```
select eventname, errorcode,eventsource,awsregion, useragent,COUNT(*) count
FROM cloudtrail_table-name
where errorcode = 'ThrottlingException'
AND eventtime between '2024-09-24T00:00:08Z' and '2024-09-23T23:15:08Z'
group by errorcode, awsregion, eventsource, useragent, eventname
order by count desc;
```

Amazon ECS 也會將事件通知發送至 Amazon EventBridge。事件主要分為資源狀態變更事件與服務操作事件兩大類。其中包含 API 限流事件，例如 ECS_OPERATION_THROTTLED 與

SERVICE_DISCOVERY_OPERATION_THROTTLED。如需詳細資訊，請參閱[Amazon ECS 服務動作事件](#)。

這些事件可由等服務取用 AWS Lambda，以執行動作來回應。如需詳細資訊，請參閱[因應 Amazon ECS 事件](#)。

如果執行獨立任務，某些 API 操作 (例如 RunTask) 是非同步的，且不會自動執行重試操作。在這種情況下，您可以使用等服務 AWS Step Functions 搭配 EventBridge 整合，重試調節或失敗的操作。如需詳細資訊，請參閱 [Manage a container task \(Amazon ECS, Amazon SNS\)](#)。

使用 CloudWatch 監控限流

CloudWatch 會在依 AWS 資源下的 Usage 命名空間上提供 API 用量監控。這些指標會以類型 API 與指標名稱 CallCount 記錄。您可以建立警示，在這些指標達到特定閾值時啟動。如需詳細資訊，請參閱 [Visualizing your service quotas and setting alarms](#)。

CloudWatch 也提供異常偵測功能。此功能使用機器學習，根據您所啟用指標的特定行為進行分析並建立基準。如有異常的 API 活動，您可以將此功能與 CloudWatch 警示搭配使用。如需詳細資訊，請參閱 [Using CloudWatch anomaly detection](#)。

透過主動監控限流錯誤，您可以聯絡支援來提高相關的限流限制，並收到您唯一應用程式需求的指引。

Amazon ECS 自動擴展與容量管理最佳實務

您可以在 Amazon ECS 上執行規模各異的容器化應用程式工作負載。這涵蓋了從最小化的測試環境到在全球範圍內運作的大規模生產環境。

如同所有 AWS 服務，使用 Amazon ECS 時，您只需支付使用量的費用。當您妥當規劃應用程式架構時，便能透過僅在需要時使用所需資源來節省成本。

下列建議說明如何在滿足服務等級目標的同時，以具成本效益的方式執行 Amazon ECS 工作負載。

主題

- [決定 Amazon ECS 的任務大小](#)
- [最佳化 Amazon ECS 服務自動擴展](#)
- [Amazon ECS 容量與可用性](#)
- [Amazon ECS 叢集容量](#)
- [為 Amazon ECS 選擇 Fargate 任務大小](#)

- [使用 Amazon EC2 容量提供者加速 Amazon ECS 叢集容量佈建](#)

決定 Amazon ECS 的任務大小

在 Amazon ECS 上部署容器時，容器與任務的大小設定是最關鍵的決策之一。容器與任務大小對於擴展和容量規劃至關重要。

Amazon ECS 使用兩種容量資源指標：CPU 與記憶體。Amazon ECS 以完整 vCPU 的 1/1024 為單位計量 CPU (其中 1024 個單位等於 1 個完整 vCPU)。Amazon ECS 以 MB 為單位計量記憶體。

在任務定義中，您可以宣告資源保留量與限制。

宣告保留量時，您要宣告任務所需的資源數量下限。任務至少會收到請求的資源量。應用程式實際使用的 CPU 或記憶體，可能會超出設定的保留量。但這受您同時宣告的所有限制的約束。

使用超過保留數量稱為爆量。爆量表示應用程式使用的資源超過保留的資源量，但會保持在您宣告的限制內。Amazon ECS 會保證保留量。例如，若使用 Amazon EC2 執行個體提供容量，當無法滿足保留量需求時，Amazon ECS 不會將該任務置放於該執行個體上。

限制是容器或任務所能使用的 CPU 單位或記憶體的最大數量。如果容器嘗試使用的 CPU 單位超過此限制，Amazon ECS 會對其進行限流。如果容器嘗試使用的記憶體超過此限制，Amazon ECS 會停止容器。

精準選擇這些值可能具有挑戰性。最適合應用程式的值，在很大程度上取決於應用程式的資源需求。

對應用程式進行負載測試，是成功規劃資源需求的關鍵。負載測試有助進一步了解應用程式的需求。

無狀態應用程式

對於可水平擴展的無狀態應用程式 (如負載平衡器後端的應用程式)，建議先判斷應用程式在處理請求時所耗用的記憶體量。

為此，您可以使用 `ps` 或 `top` 等傳統工具。您也可以使用 CloudWatch Container Insights 等監控解決方案。

在決定 CPU 保留量時，需考量如何擴展應用程式來滿足業務需求。

您可以使用較小的 CPU 保留量，例如 256 個 CPU 單位 (或 1/4 vCPU)，以將成本降至最低的精細方式橫向擴充。但此方式可能無法快速應對需求的大幅激增。

您可以使用較大的 CPU 保留量來更快地進行縮減與橫向擴充。這可協助您更快地應對需求激增情況。但較大的 CPU 保留量會帶來更高的成本。

其他應用程式

對於無法水平擴展的應用程式 (例如單例工作線程或資料庫伺服器)，可用容量與成本是最重要的考量因素。

根據負載測試的結果選擇記憶體與 CPU 的配置量，確保能應對流量並達成服務水準目標。Amazon ECS 可確保應用程式置放在具有足夠容量的主機上。

最佳化 Amazon ECS 服務自動擴展

Amazon ECS 服務是受管任務集合。每個服務都有相關聯的任務定義、所需任務計數以及選用的置放策略。

Amazon ECS 服務自動擴展功能透過應用程式自動擴展服務實現。應用程式自動擴展服務使用 CloudWatch 指標作為擴展指標的來源。該服務也會使用 CloudWatch 警示來設定關於何時縮減或橫向擴充服務的閾值。

您需提供擴展閾值。您可以設定稱為目標追蹤擴展的指標目標。您也可以指定稱為步進擴展的閾值。

設定應用程式自動擴展之後，該服務會持續為應用程式計算最適的任務數量目標值。當需要透過橫向擴充或縮減來調整任務數量時，該服務也會即時通知 Amazon ECS 進行相應變更。

若要有效使用服務自動擴展功能，您必須選擇適當的擴展指標。我們將在以下章節中探討如何選擇指標。

描述應用程式特性

若要適當地擴展應用程式，您必須了解縮減與橫向擴充應用程式的相關條件。

本質上，如果預測需求會超出容量，則應橫向擴充應用程式。相反地，如果資源超過需求，則應縮減應用程式以節省成本。

識別使用率指標

若要有效進行擴展，您必須識別能表示使用率或飽和度的指標。此指標需具備以下特性，才能有效用於擴展操作。

- 該指標必須與需求相關聯。當資源保持穩定但需求變更時，指標值也必須隨之變化。需求增加或減少時，指標應隨之上升或下降。
- 指標數值必須與容量成比例縮減。當需求保持不變時，增加更多資源必須使指標數值產生相應比例的變化。例如，將任務數量加倍，應使指標降低 50%。

識別使用率指標的最佳方法是透過生產前環境 (例如預備環境) 中的負載測試進行識別。市面上有許多商業與開源的負載測試解決方案可供使用。這類解決方案通常既能產生模擬負載，也能模擬真實使用者流量。

開始進行負載測試時，您應先為應用程式的使用率指標建置儀表板。這些指標包括 CPU 使用率、記憶體使用率、I/O 操作、I/O 佇列深度與網路輸送量。您可以透過 CloudWatch Container Insights 等服務收集這些指標。您也可以搭配使用 Amazon Managed Service for Prometheus 與 Amazon Managed Grafana 來收集這些指標。在此過程中，請務必收集並繪製應用程式的回應時間或工作完成率等指標。

進行負載測試時，請從較低的請求量或任務插入率開始。維持此速率數分鐘，讓應用程式完成暖機。接著，緩慢提高速率並穩定維持幾分鐘。重複此循環，逐步提高速率，直至應用程式的回應時間或完成時間過慢，無法達到服務水準目標 (SLO) 為止。

進行負載測試時，請檢查每個使用率指標。隨負載增加而上升的指標，是作為最佳使用率指標的首要候選對象。

然後，識別達到飽和的資源。同時，也請檢查使用率指標，觀察哪一項會率先在高位趨於平緩。或者，觀察哪項指標會最先達到峰值，而後導致應用程式當機。例如，若 CPU 使用率在您新增負載時從 0% 增加至 70-80%，而在您新增更多負載之後，會保持在該層級，則表示 CPU 已飽和。根據 CPU 架構，該使用率可能永遠不會達到 100%。例如，假設記憶體使用率隨著新增負載而增加，然後應用程式在達到任務或 Amazon EC2 執行個體記憶體限制時突然當機。這種情況通常意味著記憶體資源已完全耗盡。應用程式可能會消耗多種資源。因此，請選擇代表先耗盡之資源的指標。

最後，將任務數量或 Amazon EC2 執行個體數量加倍後，再次進行負載測試。此時假設關鍵指標的增長率或下降率為先前的一半。若情況確實如此，則表示該指標與容量成比例。這正是適用於自動擴展的良好使用率指標。

現在請考量下列假設案例。假設對應用程式進行負載測試，發現 CPU 使用率最終在每秒 100 個請求的負載下達到 80%。當您新增更多負載時，CPU 使用率不會再增加。不過，這樣確實會讓應用程式回應速度變慢。然後，再次執行負載測試，將任務數量加倍，但將速率保持在先前的峰值。如果發現平均 CPU 使用率降至約 40%，則平均 CPU 使用率是適合作為擴展指標的良好候選對象。另一方面，如果增加任務數量後，CPU 使用率仍維持在 80%，則平均 CPU 使用率並不適合作為擴展指標。此時便需進一步探尋合適的監控指標。

常見應用模型與擴展屬性

您可以在 AWS 上執行各種軟體。許多工作負載是企業自行開發的，而其他工作負載則是以熱門的開放原始碼軟體為基礎。無論其來源為何。我們觀察到服務存在一些常見的設計模式。如何有效擴展，很大程度上取決於所採用的模式。

高效的 CPU 密集型伺服器

高效的 CPU 密集型伺服器幾乎不使用 CPU 與網路輸送量以外的資源。每個請求均可由應用程式單獨處理。請求不依賴其他服務，例如資料庫。應用程式可以處理數十萬個並行請求，並可高效利用多顆 CPU 來達成此目標。每個請求要麼由記憶體負荷低的專用執行緒處理，要麼由每個 CPU 上執行的非同步事件迴圈處理。應用程式的每個副本都具備同等的請求處理能力。在 CPU 耗盡前，唯一可能耗盡的資源是網路頻寬。在 CPU 密集型服務中，即使處於峰值輸送量，記憶體使用率也僅佔可用資源的一小部分。

對此類應用程式，您可以使用 CPU 型自動擴展。這類應用程式在擴展面向享有最大的彈性。您可以透過提供更大的 Amazon EC2 執行個體或 Fargate vCPU 來進行垂直擴展，也可以透過新增副本來進行水平擴展。無論是新增副本數量，還是將執行個體大小加倍，都會將相對於容量的平均 CPU 使用率減半。

如果為此應用程式使用 Amazon EC2 容量，建議將其置放於運算最佳化執行個體上，例如 c5 或 c6g 系列。

高效率的記憶體密集型伺服器

高效率的記憶體密集型伺服器會為每個請求配置大量記憶體。在最大並行數下，但未必是最大輸送量時，記憶體會先於 CPU 資源耗盡。與請求相關聯的記憶體會在請求結束時釋放。只要有可用記憶體，就能接受更多請求。

對此類應用程式，您可以使用記憶體型自動擴展。這類應用程式在擴展面向享有最大的彈性。您可以透過提供更大的 Amazon EC2 或 Fargate 記憶體資源來進行垂直擴展，也可以透過新增副本來進行水平擴展。無論是新增副本數量，還是將執行個體大小加倍，都會將相對於容量的平均記憶體使用率減半。

如果為此應用程式使用 Amazon EC2 容量，建議將其置放於記憶體最佳化的執行個體上，例如 r5 或 r6g 系列。

有些記憶體密集型應用程式不會在請求結束時釋放與之相關聯的記憶體，因此並行量減少時，已使用的記憶體也不會相應減少。對此，我們不建議使用記憶體型擴展方式。

工作線程型伺服器

工作線程型伺服器會透過個別的工作執行緒逐一處理請求。工作執行緒可以是輕量型執行緒 (例如 POSIX 執行緒)，也可以是重量型執行緒 (例如 UNIX 程序)。無論是哪種類型的執行緒，應用程式所能支援的最大並行數都存在上限。通常，並行上限會依據可用的記憶體資源按比例設定。若達到並行上限，應用程式會將額外請求放入待處理佇列。若待處理佇列溢滿，應用程式會立即拒絕後續傳入的請求。符合此模式的常見應用程式包括 Apache 網頁伺服器與 Unicorn。

請求並行數通常是此類應用程式最適合的擴展指標。由於每個副本都有並行上限，因此在達到平均上限前進行橫向擴充至關重要。

取得請求並行數指標的最佳方式，是讓應用程式將資料回傳至 CloudWatch。應用程式的每個副本都可以高頻率發布並行請求數作為自訂指標。建議頻率至少設定為每分鐘一次。收集多次回傳資料之後，即可將平均並行數作為擴展指標。此指標的計算方式是將總並行數量除以副本數量。例如，若並行總數為 1000，而副本數量為 10，則平均並行數為 100。

如果應用程式位於 Application Load Balancer 後端，您也可以使用負載平衡器的 `ActiveConnectionCount` 指標作為擴展指標的因素。必須將 `ActiveConnectionCount` 指標除以副本數量取得平均值。再使用該平均值進行擴展，而非直接採用原始計數值。

若要使此設計達到最佳效果，回應延遲的標準偏差應該在低請求速率時保持較小。建議在低需求期間，大多數請求能在短時間內得到回應，且不會有大量請求的回應時間顯著超過平均值。平均回應時間應接近第 95 百分位數回應時間。若未達此標準，可能引發佇列溢滿，進而觸發錯誤。建議在必要時提供額外副本，降低溢滿風險。

等候型伺服器

等候型伺服器會為每個請求執行部分處理，但高度依賴一或多個下游服務才能運作。容器應用程式通常會大量使用資料庫與其他 API 服務等下游服務。這些服務可能需要一段時間才能回應，尤其在高容量或高並行場景中更為明顯。這是因為這類應用程式往往使用較少的 CPU 資源，其並行上限主要取決於可用記憶體容量。

等候型服務可能適用於記憶體密集型伺服器模式或工作線程型伺服器模式，具體取決於應用程式的設計方式。如果應用程式的並行數僅受記憶體限制，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數基於工作線程限制，則應使用平均並行數作為擴展指標。

Java 型伺服器

如果 Java 型伺服器是 CPU 密集型伺服器，且能隨 CPU 資源成比例擴展，則可能適用於高效 CPU 密集型伺服器模式。在此情況下，平均 CPU 使用率可能適合作為擴展指標。然而，許多 Java 應用程式並非 CPU 密集型，這使得擴展難以進行。

為取得最佳效能，建議將盡可能多的記憶體配置給 Java 虛擬機器 (JVM) 堆積。JVM 的最新版本 (包括 Java 8 Update 191 或更新版本)，皆會自動將堆積大小設定為容器內可容納的最大值。這表示在 Java 中，記憶體使用率鮮少與應用程式使用率成正比。隨著請求速率和並行數量增加，記憶體使用率仍保持不變。因此，我們不建議根據記憶體使用率擴展 Java 型伺服器。反之，我們通常會建議基於 CPU 使用率進行擴展。

在某些情況下，Java 型伺服器會在 CPU 耗盡前先遭遇堆積記憶體耗盡的問題。如果應用程式在高並行時容易發生堆記憶體耗盡，則平均連接數是最佳擴展指標。如果應用程式在高輸送量時容易發生堆積記憶體耗盡，則平均請求速率是最佳擴展指標。

使用其他具垃圾回收執行時期的伺服器

許多伺服器應用程式基於具備垃圾回收功能的執行時期 (例如 .NET 和 Ruby) 開發。這些伺服器應用程式可能符合前文描述的某種模式。然而，與 Java 類似，我們不建議依據記憶體來擴展這些應用程式，因為觀測到的平均記憶體使用率通常與輸送量或並行量不相關。

對這些應用程式而言，若屬於 CPU 密集型，建議基於 CPU 使用率進行擴展。否則，建議根據負載測試結果，基於平均輸送量或平均並行數進行擴展。

任務處理器

許多工作負載涉及非同步任務處理。其中包括並非即時接收請求，而是透過訂閱工作佇列接收任務的應用程式。對此類應用程式而言，合適的擴展指標幾乎都是佇列深度。佇列成長表明待處理工作超過了處理能力，而空佇列表示處理能力多於實際工作需求。

AWS 訊息服務，例如 Amazon SQS 和 Amazon Kinesis Data Streams，提供可用於擴展的 CloudWatch 指標。對於 Amazon SQS，最佳指標為 `ApproximateNumberOfMessagesVisible`。對於 Kinesis Data Streams，建議使用 Kinesis Client Library (KCL) 發布的 `MillisBehindLatest` 指標。使用此指標進行擴展前，應先計算所有消費者的平均值。

Amazon ECS 容量與可用性

應用程式的可用性至關重要，不僅能提供無錯誤的使用體驗，還有助於減少應用程式的延遲。可用性取決於擁有可存取的資源，以及足夠的容量以滿足需求。AWS 提供數種機制來管理可用性。對於在 Amazon ECS 上託管的應用程式，這些機制包括自動擴展與可用區域 (AZ)。自動擴展會根據您定義的指標來管理任務或執行個體的數量，而可用區域可讓您將應用程式託管在相互隔離但地理位置相近的位置。

與任務大小一樣，容量與可用性之間也存在必須權衡的取捨。在理想情況下，容量應與需求完全一致。系統將一律保有足夠的容量來處理請求與任務，以達成服務水準目標 (SLO) 目標，包括低延遲與錯誤率。容量不會過高而導致成本過高，也不會過低而造成高延遲與高錯誤率。

自動擴展是一個存在延遲的過程。首先，CloudWatch 必須接收即時指標。接著，CloudWatch 需要彙總這些指標進行分析，根據指標的粒度不同，這個過程可能需要長達幾分鐘。隨後，CloudWatch 會將指標與警示閾值進行比對，判斷資源是否短缺或過剩。為避免系統不穩定，您應將警示設定為：需持續超過設定閾值幾分鐘後，警示才會觸發。此外，佈建新任務與終止不再需要的任務也需要時間。

由於系統中存在這些潛在延遲，您應採用過度佈建方式來保留一些緩衝空間。過度佈建有助於因應短期的需求突增。這也有助於應用程式在不達到飽和狀態的前提下處理更多請求。建議將擴展目標設定在使用率的 60-80% 之間。此舉能幫助應用程式在額外容量仍在佈建過程中時，更好地應對突增的額外需求。

我們建議您過度佈建的另一個原因是，以便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，如果某個可用區域發生失敗，在剩餘可用區域中執行的任務仍能持續處理需求。如果應用程式在兩個可用區域中執行，則需要將常規任務數量加倍。如此一來，您就可以在任何潛在的故障期間立即提供足夠容量。如果應用程式在三個可用區域中執行，建議將任務數量設定為常規需求的 1.5 倍。也就是說，每需要 2 個任務來處理日常請求，就應部署 3 個任務。

最大限度提高擴展速度

自動擴展是一種反應式程序，需要一些時間才能生效。不過，有一些方法可協助將橫向擴充所需的時間降至最低。

最小化映像大小。較大的映像需要更長的時間，才能從映像儲存庫下載並解壓縮。因此，保持較小的映像大小可減少容器啟動所需的時間量。若要減小映像大小，您可以遵循下列特定建議：

- 如果您可以建置靜態二進位檔案或使用 Golang，請建置映像 FROM 暫存，並在產生的映像中僅包含二進位應用程式。
- 使用上游發行版廠商提供的精簡基礎映像，例如 Amazon Linux 或 Ubuntu。
- 切勿在最終映像中包含任何建置成品。使用多階段建置有助於實現這一點。
- 盡可能精簡 RUN 階段。每個 RUN 階段都會建立新的映像層，導致額外的層下載往返次數。透過 && 聯結的多個命令的單一 RUN 階段，比多個獨立的 RUN 階段產生的層數更少。
- 若想在最終映像中包含 ML 推論資料等資料，請僅包含啟動與開始處理流量所需的資料。如果隨需從 Amazon S3 或其他儲存擷取資料而不影響服務，請改為將資料儲存於這些位置。

將映像存放於近處。網路延遲越高，下載映像的時間就越長。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 是高效能映像儲存庫，可在提供 Amazon ECS 的每個區域中使用。避免透過網際網路或 VPN 連結下載容器映像。在同一區域中託管映像可改善整體可靠性。此舉可降低不同區域中網路連線問題與可用性問題帶來的風險。此外，您可以實作 Amazon ECR 跨區域複寫來協助解決此問題。

降低負載平衡器運作狀態檢查閾值。負載平衡器會先執行運作狀態檢查，再將流量傳送至應用程式。目標群組的預設運作狀態檢查組態設定，可能需要 90 秒或更長時間。在此期間，負載平衡器會檢查運作狀態並接收請求。縮短運作狀態檢查間隔與降低閾值次數，可讓應用程式更快接收流量，減輕其他任務的負載。

考慮冷啟動效能。某些應用程式會使用執行時期，例如 Java 執行即時 (JIT) 編譯。編譯過程至少在啟動階段會影響應用程式效能表現。解決方法是將工作負載中對延遲敏感的關鍵部分，改用以不會造成冷啟動效能損耗的程式語言重新編寫。

使用步進擴展政策而非目標追蹤擴展政策。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](#)。

如果使用 Amazon EC2 執行個體提供叢集容量，請考量下列建議：

使用更大的 Amazon EC2 執行個體與更快的 Amazon EBS 磁碟區。透過使用更大的 Amazon EC2 執行個體與更快的 Amazon EBS 磁碟區，您可以提升映像下載與準備速度。在特定的 Amazon EC2 執行個體系列中，網路與 Amazon EBS 輸送量上限會隨著執行個體大小的增加而提升 (例如，從 m5.xlarge 到 m5.2xlarge)。此外，您可以自訂 Amazon EBS 磁碟區來提高其輸送量與 IOPS。例如，若使用的是 gp2 磁碟區，可選擇提供更高基準輸送量的較大磁碟區。若使用的是 gp3 磁碟區，可在建立磁碟區時指定輸送量與 IOPS。

為執行於 Amazon EC2 執行個體上的任務使用橋接網路模式。在 Amazon EC2 上使用 bridge 網路模式的任務，啟動速度比使用 awsvpc 網路模式的任務更快。使用 awsvpc 網路模式時，Amazon ECS 會先將彈性網路介面 (ENI) 連接至執行個體，再啟動任務。這會造成額外的延遲。不過，使用橋接聯網存在幾項權衡考量。此類任務不會擁有獨立的安全群組，且在負載平衡面向也會有一些影響。如需詳細資訊，請參閱 Elastic Load Balancing User Guide 中的[Load balancer target groups](#)。

因應需求衝擊

部分應用程式可能遭遇突發性的大量需求衝擊。發生這種情況的原因有很多：新聞事件、大型促銷活動、媒體事件，或者一些其他會迅速傳播並導致流量在非常短的時間內快速大幅增加的事件。如果未加以規劃，這可能會導致需求快速超過可用資源。

因應需求衝擊的最佳方法是預測需求衝擊並相應地進行規劃。由於自動擴展可能需要一些時間，建議您在需求衝擊開始之前橫向擴充應用程式。為達到最佳效果，建議制定一套結合跨團隊協作的業務計畫，其中應包含採用共用行事曆的緊密協作機制。活動規劃團隊應提前與負責應用程式的團隊緊密合作。這能讓負責應用程式的團隊有足夠時間制定明確的排程計畫。他們可以排定容量，在事件之前進行橫向擴充，並在事件之後進行縮減。如需詳細資訊，請參閱《Application Auto Scaling 使用者指南》中的[排程擴展](#)。

如果有企業支援方案，也請務必與您的技術客戶經理 (TAM) 合作。技術客戶經理可以核實您的服務配額，並確保在事件開始之前提高任何必要的配額。如此一來，您就不會意外達到任何服務配額上限。此外，技術客戶經理亦可透過預先啟動負載平衡器等服務的暖機程序，確保活動順利進行。

因應未經排程的需求衝擊是一個更棘手的問題。若幅度夠大，未經排程的衝擊可能會迅速導致需求超過容量，甚至使自動擴展來不及反應。應對未經排程衝擊的最佳方式是過度佈建資源。您必須隨時擁有足夠的資源來因應預期的最大流量需求。

為因應未經排程的需求衝擊而維持最大容量可能耗費頗鉅。若要降低成本影響，可尋找能預測大規模需求衝擊即將發生的領先指標或事件。如果指標或事件能穩定提供足夠的預警時間，則在事件發生時，或指標超過您設定的特定閾值時，立即啟動水平擴展程序。

如果應用程式容易遭遇突發且未經排程的需求衝擊，可考慮為應用程式新增高效能模式。該模式會犧牲非關鍵功能，但保留對客戶而言至關重要的功能。例如，假設應用程式可從產生耗費資源的客製化回應，切換為提供靜態回應頁面。在此案例中，您無需對應用程式進行任何擴展，就能顯著提升輸送量。

最後，您可考慮拆分單體式服務，更高效地因應需求衝擊。如果應用程式是執行成本高昂且擴展速度緩慢的單體式服務，您也許可以擷取或重寫效能關鍵部分，並將其作為個別服務執行。如此，這些新服務便能獨立於非關鍵元件進行擴展。具備單獨橫向擴充應用程式中效能關鍵功能的彈性，不僅可縮短新增容量的時間，也有助於節省成本。

Amazon ECS 叢集容量

您可以透過多種方式為 Amazon ECS 叢集提供容量。例如，您可以啟動 Amazon EC2 執行個體，並在啟動時使用 Amazon ECS 容器代理程式將其註冊至叢集。不過，這種方法可能較具挑戰性，因為您需要自行管理擴展。因此，建議採用 Amazon ECS 容量提供者。容量提供者會自動管理資源擴展。容量提供者有三種：Amazon EC2、Fargate 與 Fargate Spot。如需有關 Fargate 容量提供者的詳細資訊，請參閱 [Fargate 工作負載的 Amazon ECS 叢集](#)；如需有關 EC2 工作負載的詳細資訊，請參閱 [EC2 工作負載的 Amazon ECS 叢集](#)。

Fargate 與 Fargate Spot 容量提供者會為您處理 Fargate 任務的生命週期。Fargate 提供隨需容量，而 Fargate Spot 則提供 Spot 容量。當您啟動任務時，Amazon ECS 會自動佈建 Fargate 資源。此 Fargate 資源隨附的記憶體與 CPU 單位，直接對應您於任務定義中宣告的任務層級限制。每個任務都會收到專用 Fargate 資源，從而在任務與運算資源之間建立 1:1 的關係。

執行於 Fargate Spot 上的任務可能會中斷。中斷會在發出兩分鐘的警告後發生。此類情況通常發生於需求高峰期間。Fargate Spot 最適合能夠容忍中斷的工作負載，例如批次任務、開發或預備環境。其也適用於其他不要求高可用性與低延遲的情況。

您可以將 Fargate Spot 任務與 Fargate 隨需任務一同執行。透過同時使用兩者，您能以較低的成本取得「高載」容量供應。

Amazon ECS 也能為您管理任務的 Amazon EC2 執行個體容量。每個 Amazon EC2 容量提供者都會與您指定的 Amazon EC2 Auto Scaling 群組建立關聯。當您使用 Amazon EC2 容量提供者時，叢集自動擴展會維持 Amazon EC2 Auto Scaling 群組的大小，確保可以置放所有排程任務。

為 Amazon ECS 選擇 Fargate 任務大小

對於 AWS Fargate 任務定義，您必須在任務層級指定 CPU 和記憶體，而且不需要考慮任何額外負荷。您也可以指定 Fargate 任務之容器層級的 CPU 和記憶體。不過，不需要這麼做。資源限制必須大於或等於您宣告的任何保留量。在大多數情況下，您可以將其設定為任務定義中宣告的每個容器的保留量總和。然後將數值向上取整至最接近的 Fargate 大小。如需有關可用大小的詳細資訊，請參閱[任務 CPU 與記憶體](#)。

使用 Amazon EC2 容量提供者加速 Amazon ECS 叢集容量佈建

在 Amazon EC2 上執行 Amazon ECS 的客戶，可以利用 [Amazon ECS 叢集自動擴展 \(CAS\)](#) 來管理 Amazon EC2 Auto Scaling 群組 (ASG) 的擴展。透過 CAS，您可以設定 Amazon ECS 自動擴展 ASG，從而專注於執行任務。Amazon ECS 將確保 ASG 視需求進行擴展與縮減，無需額外人工介入。Amazon ECS 容量提供者可用來管理叢集中的基礎結構，確保有足夠的容器執行個體可滿足應用程式需求。如需了解 Amazon ECS CAS 的運作方式，請參閱 [Deep Dive on Amazon ECS Cluster Auto Scaling](#)。

由於 CAS 依賴與 ASG 的 CloudWatch 型整合來調整叢集容量，其本身存在固有延遲，包括：發布 CloudWatch 指標所需的時間、指標 CapacityProviderReservation 觸發 CloudWatch 高/低閾值警報的處理時間，以及新啟動 Amazon EC2 執行個體完成暖機所需的週期。您可以採取下列動作提高 CAS 的回應速度，加速實現部署：

容量提供者步進擴展大小

Amazon ECS 容量提供者最終將擴增/縮小容器執行個體，以滿足應用程式的需求。將 Amazon ECS 會啟動的執行個體數量下限預設為 1。如果需要多個執行個體來放置待處理任務，可能會增加部署所需的時間。您可以透過 Amazon ECS API 增加 [minimumScalingStepSize](#)，提高 Amazon ECS 每次可縮減或擴展的執行個體數量下限。太低的 [maximumScalingStepSize](#) 會限制每次可縮減或擴展的容器執行個體數量，進而拖慢部署速度。

Note

此組態目前只能透過 [CreateCapacityProvider](#) 或 [UpdateCapacityProvider](#) API 使用。

執行個體暖機期間

執行個體暖機期間是指新啟動的 Amazon EC2 執行個體納入 Auto Scaling 群組 CloudWatch 指標統計前的一段時間。在指定的暖機期間到期後，執行個體才會納入 ASG 的彙總指標計算；而 CAS 會接著進行其下一輪計算，以估算所需的執行個體數量。

[instanceWarmupPeriod](#) 的預設值為 300 秒，您可透過 [CreateCapacityProvider](#) 或 [UpdateCapacityProvider](#) API 將其設定為較低的值，從而實現更即時的擴展。

備用容量

如果容量提供者沒有可用於置放任務的容器執行個體，就必須透過即時啟動 Amazon EC2 執行個體來增加 (橫向擴充) 叢集容量，並等待這些執行個體啟動完成後才能在其上部署容器。這會大幅降低任務啟動速率。對此，您有兩種選擇。

在這種情況下，若事先啟動並準備好可用於執行任務的 Amazon EC2 備用容量，即能提高有效的任務啟動速率。您可以透過 Target Capacity 組態來指定希望叢集維持的備用容量。例如，將 Target Capacity 設定為 80%，即指定叢集需隨時保留 20% 的備用容量。這些備用容量可讓任何獨立任務立即啟動，確保任務啟動不會受到限流。但此方法的權衡在於，維持叢集備用容量可能會增加成本。

另一種替代方法是為服務增加緩衝空間，而非為容量提供者設定。這意味著，您不必透過降低 Target Capacity 組態來啟動備用容量，而是可以藉由修改服務自動擴展的目標追蹤擴展指標或步進擴展閾值，直接增加服務中的副本數量。請注意，此方法僅對突發性工作負載有效；在部署新服務並首次從 0 擴展至 N 個任務時，則無法發揮作用。如需有關相關擴展政策的詳細資訊，請參閱 [目標追蹤擴展政策](#) 或 [步進擴展政策](#)

透過帳戶設定使用 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。
- 根使用者或具有管理員權限的使用者，可為帳戶中的任何特定角色或使用者選擇啟用或停用設定。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或選擇退出每個帳戶設定。

資源名稱	進一步了解
containerInsights	Container Insights
serviceLongArnFormat	Amazon Resource Name (ARN) 和 ID
taskLongArnFormat	
containerInstanceLongArnFormat	
tagResourceAuthorization	標記授權
fargateFIPSMODE	AWS Fargate 聯邦資訊處理標準 (FIPS-140) 合規
fargateTaskRetirementWaitPeriod	AWS Fargate 任務淘汰等待時間
fargateEventWindows	AWS Fargate 使用 EC2 事件時段淘汰任務
guardDutyActivate	執行時期監控 (Amazon GuardDuty 整合)
dualStackIPv6	雙堆疊 IPv6 VPC
awsvpcTrunking	增加 Linux 容器執行個體網路介面數量
defaultLogDriverMode	預設日誌驅動程式模式

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 [stop-task](#) AWS CLI 命令來停止任務，您必須在命令中指定任務 ARN 或 ID。

Amazon ECS 針對 Amazon ECS 服務、任務及容器執行個體，推出了新的 Amazon Resource Name (ARN) 與資源 ID 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。

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。您建立的所有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 帳戶設定](#)。

下表顯示每個資源類型的當前格式和新格式。

Resource Type (資源類型)	ARN
容器執行個體	當前 : <code>arn:aws:ecs: <i>region</i>:<i>aws_account_id</i> :container-instance/ <i>container-instance-id</i></code>

Resource Type (資源類型)	ARN
	新 : <code>arn:aws:ecs: <i>region</i>:<i>aws_account_id</i> :container-instance/<i>cluster-name</i> /<i>container-instance-id</i></code>
Amazon ECS 服務	當前 : <code>arn:aws:ecs: <i>region</i>:<i>aws_account_id</i> :service/<i>service-name</i></code> 新 : <code>arn:aws:ecs: <i>region</i>:<i>aws_account_id</i> :service/<i>cluster-name</i> /<i>service-name</i></code>
Amazon ECS 任務	當前 : <code>arn:aws:ecs: <i>region</i>:<i>aws_account_id</i> :task/<i>task-id</i></code> 新 : <code>arn:aws:ecs: <i>region</i>:<i>aws_account_id</i> :task/<i>cluster-name</i> /<i>task-id</i></code>

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 帳戶設定](#)。

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。

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。

Container Insights

2024 年 12 月 2 日，AWS 發行了 Container Insights 並增強了 Amazon ECS 的可觀測性。此版本的增強可觀測性支援使用 Fargate Amazon ECS 受管執行個體和 EC2 的 Amazon ECS 叢集。在 Amazon ECS 上設定了具有增強可觀測性的 Container Insights 後，Container Insights 會自動收集從叢集層級到容器層級的詳細基礎結構遙測資料，並在顯示各種指標和維度的儀表板中顯示您的資料。然後，您可以在 Container Insights 主控台上使用這些現成儀表板，以更清楚了解容器健全狀況和效能，並通過識別異常來更快減少問題。

建議使用具有增強可觀測性的 Container Insights 而非普通的 Container Insight，因為前者能提供容器環境的詳細可見性，縮短問題解決的平均時間。如需詳細資訊，請參閱 Amazon CloudWatch User Guide 中的 [Amazon ECS Container Insights with enhanced observability metrics](#)。

containerInsights 帳戶設定預設為 disabled。

具有增強可觀測性的 Container Insights

使用以下命令開啟具有增強可觀測性 Container Insights。

將 containerInsights 帳戶設定設為 enhanced。

```
aws ecs put-account-setting --name containerInsights --value enhanced
```

範例輸出

```
{
  "setting": {
    "name": "containerInsights",
    "value": "enhanced",
    "principalArn": "arn:aws:iam::123456789012:johndoe",
    "type": "user"
  }
}
```

設定此帳戶設定後，所有新叢集會自動使用具有增強可觀測性的 Container Insights。使用 `update-cluster-settings` 命令將具有增強可觀測性的 Container Insights 新增至現有叢集，或將叢集從普通的 Container Insights 升級至具有增強可觀測性的 Container Insights。

```
aws ecs update-cluster-settings --cluster cluster-name --settings
name=containerInsights,value=enhanced
```

您也可以使用主控台來設定具有增強可觀測性的 Container Insights。如需詳細資訊，請參閱[修改 Amazon ECS 帳戶設定](#)。

Container Insights

當您將 `containerInsights` 帳戶設定設為 `enabled` 時，所有新叢集都會預設啟用 Container Insights。您可以使用 `update-cluster-settings` 來修改現有叢集。

若要使用 Container Insights，請將 `containerInsights` 帳戶設定設為 `enabled`。使用下列命令來啟用 Container Insights。

```
aws ecs put-account-setting --name containerInsights --value enabled
```

範例輸出

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

當您將 `containerInsights` 帳戶設定設為 `enabled` 時，所有新叢集都會預設啟用 Container Insights。使用 `update-cluster-settings` 命令，將 Container Insights 新增至現有叢集。

```
aws ecs update-cluster-settings --cluster cluster-name --settings
name=containerInsights,value=enabled
```

您也可以使用主控台來設定 Container Insights。如需詳細資訊，請參閱[修改 Amazon ECS 帳戶設定](#)。

AWS Fargate 聯邦資訊處理標準 (FIPS-140) 合規

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。

資源名稱為 `fargateFIPSMODE`。

預設值為 `disabled`。

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 FIPS-140 合規”](#)。

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 帳戶設定](#)。

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》](#)。

- 您可以透過如下命令來啟用 FIPS-140 合規。

```
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。使用者必須擁有建立資源的動作標記許可，例如 `ecsCreateCluster`。當您建立資源並指定該資源的標籤時，AWS 會執行額外的授權，以驗證是否有建立標籤的許可。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 `on`。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。

- 您可以透過如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，您必須設定適當許可，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 – 依預設，所有新帳戶與未受影響的現有帳戶都具有標記授權。您可以啟用或停用 `tagResourceAuthorization` 帳戶設定來驗證 IAM 政策。

AWS 已通知受影響的帳戶。

若要停用此功能，請執行 `put-account-setting-default`，將 `tagResourceAuthorization` 選項設定為 `off`。

- 2024 年 3 月 7 日 – 如果已啟用標記授權，則無法再停用帳戶設定。

建議在此日期之前完成 IAM 政策測試。

- 2024 年 3 月 29 日 – 所有帳戶都會使用標記授權。Amazon ECS 主控台或 AWS CLI 將不再提供帳戶層級設定。

AWS Fargate 任務淘汰等待時間

AWS 當您在標記為淘汰的平台版本修訂版上執行 Fargate 任務時，會傳送通知。如需詳細資訊，請參閱 [Amazon ECS 上 AWS Fargate 的任務淘汰和維護](#)。

AWS 負責修補和維護 AWS Fargate 的基礎基礎設施。當 AWS 判斷在 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來取代它們。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 個日曆日，或等待 14 個日曆日。

此設定位於帳戶層級。

您可以設定 Fargate 開始任務淘汰的時間。預設等待期間為 7 天。對於需要立即套用更新的工作負載，請選擇立即設定 (0)。如果您需要更多時間，請設定 7 或 14 日期選項。

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。

透過以根使用者或管理使用者的身分執行 `put-account-setting-default` 或 `put-account-setting` 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任務。
- 7 - AWS 傳送通知，並等待 7 個日曆天，再開始淘汰受影響的任務。這是預設值。
- 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。

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](#)。

您可以執行如下命令，將等待期間設定為 14 日。

```
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
```

AWS Fargate 使用 EC2 事件時段淘汰任務

AWS 負責修補和維護的基礎設施 AWS Fargate。如果 AWS 判定 Fargate 上託管的某個 Amazon ECS 任務需要進行安全性或基礎設施更新，則需要停止任務並啟動新任務來取代它們。透過啟用此帳戶設定 AWS Fargate，將使用 EC2 事件時段來決定任務淘汰的時間和日期。請注意，這是帳戶的一次性啟用。

若要啟用 Fargate 任務淘汰的 EC2 事件時段，您可以使用下列 CLI 命令：

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

這是帳戶層級設定。您可以使用下列執行個體標籤，將 EC2 事件時段與帳戶、叢集和服務層級的 Fargate 任務建立關聯：

- `aws:ecs:serviceArn`：適用於 服務任務
- `aws:ecs:clusterArn`：適用於屬於叢集的任務
- `aws:ecs:fargateTask`：設定為 `true` 以鎖定帳戶中的所有 Fargate 任務

若要進一步了解 [Amazon EC2 事件視窗](#) 如何針對在 Fargate 上執行的 Amazon ECS 任務運作，請參閱 [步驟 1：設定任務等待時間或使用 Amazon EC2 事件時段](#)

增加 Linux 容器執行個體網路介面數量

每個使用 `awsvpc` 網路模式的 Amazon ECS 任務都會收到專用的彈性網路介面 (ENI)，該介面會連接至託管此任務的容器執行個體。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。因為每個使用 `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。

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。當您使用這些執行個體類型並啟用 `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。

例如，啟用 `awsvpcTrunking` 的 `c5.large` 執行個體，其 ENI 限制會增加至 12 個。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。

執行時期監控 (Amazon GuardDuty 整合)

執行期監控是一種智慧型威脅偵測服務，可透過持續監控 AWS 日誌和聯網活動來識別惡意或未經授權的行為，來保護在 Fargate 和 EC2 容器執行個體上執行的工作負載。

`guardDutyActivate` 參數在 Amazon ECS 中為唯讀，會指明 Amazon ECS 帳戶中安全管理員是否已啟用執行時期監控功能。GuardDuty 會代為控制此帳戶設定。如需詳細資訊，請參閱 [透過執行時期監控保護 Amazon ECS 工作負載](#)。

您可以執行 `list-account-settings` 來檢視目前的 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"
  }
}
```

雙堆疊 IPv6 VPC

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。如需有關其他需求的詳細資訊，請參閱[在雙堆疊模式下使用 VPC](#) (針對 EC2 容量)、[在雙堆疊模式下使用 VPC](#) (針對 Amazon ECS 受管執行個體容量) 以及[在雙堆疊模式下使用 VPC](#) (針對 Fargate 容量)。

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 帳戶設定](#)。

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`。

預設日誌驅動程式模式

Amazon ECS 支援設定從容器到所選日誌驅動程式的日誌訊息預設傳送模式。當從容器到日誌驅動程式的日誌流程中斷時，傳送模式會影響應用程式穩定性。

defaultLogDriverMode 設定支援兩個值：blocking 與 non-blocking。如需有關這些傳送模式的詳細資訊，請參閱 Amazon Elastic Container Service API Reference 中的 [LogConfiguration](#)。

如果未在容器定義的 logConfiguration 中指定傳送模式，則會將您使用此帳戶設定指定的模式用作預設傳送模式。

預設傳送模式為 non-blocking。

Note

自 2025 年 6 月 25 日起，Amazon ECS 將預設日誌驅動程式模式從 blocking 變更為 non-blocking，優先保障任務可用性而非日誌完整性。若要在此變更後繼續使用 blocking 模式，請執行下列任一操作：

- 將容器定義 logConfiguration 中的 mode 選項設定為 blocking。
- 將 defaultLogDriverMode 帳戶設定設為 blocking。

要將預設日誌驅動程式模式設定為 blocking，您可以執行如下命令。

```
aws ecs put-account-setting-default --name defaultLogDriverMode --value "blocking"
```

使用主控台檢視 Amazon ECS 帳戶設定

在主控台中檢視帳戶設定，了解可以使用的具體功能。

Important

dualStackIPv6、fargateFIPSModefargateTaskRetirementWaitPeriod和 fargateEventWindows帳戶設定只能使用 檢視或變更 AWS CLI。

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 帳戶設定

修改帳戶設定，使用 Amazon ECS 功能。

`guardDutyActivate` 參數在 Amazon ECS 中為唯讀，會指明 Amazon ECS 帳戶中安全管理員是否已啟用執行時期監控功能。GuardDuty 會代為控制此帳戶設定。如需詳細資訊，請參閱[透過執行時期監控保護 Amazon ECS 工作負載](#)。

Important

`dualStackIPv6`、`fargateFIPSMODE`、`fargateTaskRetirementWaitPeriod`和 `fargateEventWindows` 帳戶設定只能使用 檢視或變更 AWS CLI。

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. (選用) 若要增加或減少每個 EC2 執行個體可在 `awsvpc` 網路模式中執行的任務數量，請在 `AWSVPC Trunking` 下選取 `AWSVPC Trunking`。
6. (選用) 若要為叢集預設使用或停止使用 CloudWatch Container Insights，請在 CloudWatch Container Insights 可觀測性下，選擇下列任一選項：
 - 若要使用具有增強可觀測性的 Container Insights，請選擇具有增強可觀測性的 Container Insights。
 - 若要使用 Container Insights，請選擇 Container Insights。
 - 若要停止使用 Container Insights，請選擇關閉。
7. (選用) 若要啟用或停用標記授權，請在資源標記授權下，選取或清除資源標記授權。
8. (選用) 若要在容器的 `logConfiguration` 中未定義日誌傳送模式時，設定預設的日誌驅動程式模式，請在預設日誌驅動程式模式下選擇下列任一選項：
 - 如需將預設日誌驅動程式模式設定為 `blocking`，請選擇封鎖。
 - 如需將預設日誌驅動程式模式設定為 `non-blocking`，請選擇非封鎖。
9. 選擇儲存變更。
10. 在確認畫面，選擇確認以儲存選項。

後續步驟

如果已啟用具有增強可觀測性的 Container Insights 或普通的 Container Insights，即可選擇更新現有叢集來使用該功能。如需詳細資訊，請參閱[更新 Amazon ECS 叢集](#)。

還原為預設的 Amazon ECS 帳戶設定

您可以使用 AWS 管理主控台 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 SDKs 來管理帳戶設定。dualStackIPv6、fargateFIPSPModefargateTaskRetirementWaitPeriod和 fargateEventWindows帳戶設定只能使用這些工具檢視或變更。

Note

您可以使用雙堆疊服務端點，透過 IPv4 和 IPv6 AWS CLI從、SDKs 和 Amazon ECS API 與 Amazon ECS 互動。如需詳細資訊，請參閱[使用 Amazon ECS 雙堆疊端點](#)。

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。除非使用者或角色自行明確覆寫這些設定，否則這些變更適用於整個 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。

- [Write-ECSAccountSetting](#) (AWS 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。

- [Write-ECSAccountSetting](#) (AWS 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。

- [Write-ECSAccountSetting](#) (AWS Tools for Windows PowerShell)

```
Write-ECSAccountSetting -Name serviceLongArnFormat -Value enabled -PrincipalArn  
arn:aws:iam::aws_account_id:user/principalName -Region us-east-1 -Force
```

適用於 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，將相關許可授予 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的啟動類型以及您使用的功能。請使用下列表來判斷 Amazon ECS 需要哪些 IAM 角色。

Role	定義	適用時機	其他資訊
任務執行 角色	此角色允許 Amazon ECS 代表您使用其他 AWS 服務。	您的任務託管於 AWS Fargate 或外部執行個體上，且符合下列條件： • 從 Amazon ECR 私有儲存庫提取容器映像。 • 從與執行任務不同帳戶的 Amazon ECR 私有儲存庫提取容器映像。 • 使用 <code>awslogs</code> 日誌驅動程式將容器日誌傳送到 CloudWatch Logs。	Amazon ECS 任務執行 IAM 角色

Role	定義	適用時機	其他資訊
		您的任務託管在 AWS Fargate 或 Amazon EC2 執行個體上，並且： • 使用私有登錄檔身分驗證。 • 使用執行時期監控。 • 任務定義會使用 Secrets Manager 秘密或 AWS Systems Manager 參數存放區參數來參考敏感資料。	
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	此角色允許您的 EC2 執行個體或外部執行個體向叢集完成註冊程序。	您的任務託管於 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 APIs。	您的任務託管於外部執行個體上。	Amazon ECS Anywhere IAM 角色
負載平衡器角色的 Amazon ECS 基礎結構	此角色允許 Amazon ECS 代表您管理叢集中的負載平衡器資源，以實現藍/綠部署。	您想使用 Amazon ECS 藍/綠部署。	負載平衡器的 Amazon ECS 基礎結構 IAM 角色

Role	定義	適用時機	其他資訊
Amazon ECS CodeDeploy 角色	此角色允許 CodeDeploy 更新您的服務。	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 EventBridge 角色	此角色允許 EventBridge 更新您的服務。	您可以透過 EventBridge 規則與目標來排程任務。	Amazon ECS EventBridge IAM 角色
Amazon ECS 基礎結構角色	此角色允許 Amazon ECS 管理叢集中的基礎結構資源。	- 您想使用 Amazon ECS 受管執行個體作為容量。 - 您想將 Amazon EBS 磁碟區連接至 Fargate 或 EC2 啟動類型 Amazon ECS 任務。基礎結構角色允許 Amazon ECS 管理您任務的 Amazon EBS 磁碟區。 - 您想使用 Transport Layer Security (TLS) 來加密 Amazon ECS Service Connect 服務之間的流量。 - 您想建立 VPC Lattice 目標群組。	Amazon ECS 基礎結構 IAM 角色
執行個體設定檔	此角色允許 Amazon ECS 受管執行個體安全地擔任基礎結構角色。	您可以在叢集中使用 Amazon ECS 受管執行個體。	Amazon ECS 受管執行個體執行個體設定檔

Amazon ECS 受管執行個體的架構

Amazon ECS 受管執行個體是 Amazon ECS 的一種全受管運算選項，讓您能在全部類型的 Amazon EC2 執行個體上執行容器化工作負載，同時將基礎結構管理工作交由 AWS 負責。透過 Amazon ECS 受管執行個體，您可以存取特定運算功能，例如 GPU 加速、特定 CPU 架構、高網路效能和特殊執行個體類型，同時 AWS 處理基礎基礎設施的佈建、擴展、修補和維護。

當您使用 Amazon ECS 受管執行個體時，您可以將應用程式封裝在容器中並指定運算需求。AWS 會自動選取最符合工作負載需求的成本最佳化一般用途 Amazon EC2 執行個體類型，或者您可以指定所需的執行個體屬性，包括執行個體類型、CPU 製造商和加速器。Amazon ECS 受管執行個體可完全管理基礎設施的所有層面，包括擴展、修補和成本最佳化，而不會交換對 AWS 功能和 Amazon EC2 整合的存取。

Amazon ECS 受管執行個體支援具有平台特定最佳化和安全性組態的 Linux 容器。依預設，Amazon ECS 受管執行個體透過將多個較小的任務置放在較大的執行個體上來最佳化基礎結構使用率，協助降低成本並改善任務啟動時間。

本主題介紹 Amazon ECS 受管執行個體任務與服務的不同元件，並指出將 Amazon ECS 受管執行個體與 Amazon ECS 搭配使用時的特別考量。

開始使用

若要開始使用 Amazon ECS 受管執行個體，您可以建立所需的 IAM 角色，並在 AWS 帳戶中啟用 Amazon ECS 受管執行個體。然後，您可以建立容量提供者並使用 Amazon ECS 受管執行個體容量提供者啟動任務或服務。

如需有關開始使用的詳細說明，請參閱：

- [了解如何為 Amazon ECS 受管執行個體建立任務](#)
- [了解如何使用 為 Amazon ECS 受管執行個體建立任務 AWS CLI](#)

容量提供者

Amazon ECS 受管執行個體會使用容量提供者來管理工作負載的運算容量。建立容量提供者時，您可以使用預設執行個體選擇或使用指定自訂執行個體需求 `instanceRequirements`。

可用的容量提供者選項如下：

- 預設容量提供者 – 針對工作負載需求自動選取最具成本效益的一般用途執行個體類型。
- 自訂容量提供者 – 可讓您使用屬性型執行個體類型選取來指定執行個體屬性，包括 vCPU 計數、記憶體、CPU 製造商、加速器類型與特定執行個體類型。

容量提供者策略只能包含下列清單中的一個容量提供者類型：

- Amazon ECS 受管執行個體
- Auto Scaling 群組
- Fargate/Fargate_SPOT

執行個體選擇與最佳化

Amazon ECS 可使用下列其中一種方法為 Amazon ECS 受管執行個體工作負載選擇執行個體類型：

- 自動選取 – 當使用預設容量提供者時，Amazon ECS 會自動選取最具成本效益的一般用途執行個體類型，這些類型符合任務定義中指定的 CPU 與記憶體需求。
- 屬性型選取 – 當使用自訂容量提供者時，您可以指定執行個體屬性，例如 vCPU 計數、記憶體大小、CPU 製造商、加速器類型與特定執行個體類型。Amazon ECS 會從符合所指定屬性的所有執行個體類型中選取。

Amazon ECS 受管執行個體透過數種機制來最佳化基礎結構使用率與成本：

- 多任務置放 – 依預設，Amazon ECS 會將多個較小的任務置放到較大的執行個體上，以最大化使用率並降低成本。
- 主動工作負載整合 – Amazon ECS 會識別容器執行個體何時真正處於閒置狀態，同時嘗試避免可能影響應用程式可用性或部署效能的提早終止。系統會遵守為服務設定的任務數量下限和上限、先啟動後停止行為以及任務保護行為。
- 最適化調整 – 隨著工作負載需求的變更，Amazon ECS 會啟動針對目前需求而適當調整了大小的替代執行個體。

Amazon ECS 會使用 Amazon EC2 事件時段，在您偏好的時段內排程維護活動。事件時段可讓您定義 AWS 可以對執行個體執行維護的週期性時段，從而透過使維護與操作排程保持一致，協助您將對工作負載的干擾降至最低。如需詳細資訊，請參閱 Amazon EC2 User Guide 中的 [Scheduled events for your instances](#)。

如果您需要嚴格隔離，則可以將 Amazon ECS 受管執行個體設定為在具有 VM 層級安全隔離界限的個別執行個體上執行各項任務。

任務定義

使用 Amazon ECS 受管執行個體的任務支援大多數 Amazon ECS 任務定義參數。Amazon ECS 受管執行個體與使用平台版本 1.4.0 的現有 Fargate 任務定義相容，讓遷移變得簡單。

若要使用 Amazon ECS 受管執行個體，請將 `requiresCompatibilities` 任務定義參數設定為包含 `MANAGED_INSTANCES`。任務定義可以指定 Fargate 與 Amazon ECS 受管執行個體相容性，以在部署選項中具有彈性。

作業系統與 CPU 架構

支援以下作業系統：

- Bottlerocket

有兩種架構可用於 Amazon ECS 任務定義，即 ARM 和 X86_64。

當您在 Amazon ECS 受管執行個體上執行 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。

主要功能

以下是 Amazon ECS 受管執行個體的主要功能：

- 選取特定 EC2 執行個體類型以符合應用程式的需求，以便存取專用硬體功能，例如 GPU 加速運算、特定 CPU 功能與大型記憶體大小。
- 依預設，透過將多個任務置放到單一執行個體上來最佳化資源使用率與成本。這一點與在自己的隔離環境中執行各項任務的 Fargate 不同。
- 確保安全合規和定期執行個體修補。ECS 受管執行個體會在 14 天後啟動執行個體耗盡，並自動將服務型任務取代為新的執行個體。
- 使用特權 Linux 功能 (包括 `CAP_NET_ADMIN`、`CAP_SYS_ADMIN` 與 `CAP_BPF`)，在容器內啟用進階聯網設定與系統管理功能。

IAM 角色

Amazon ECS 受管執行個體需要兩個 IAM 角色：

- 基礎設施角色：此角色允許 代表您 AWS 管理 Amazon ECS 受管執行個體。
- 執行個體設定檔：執行個體設定檔是將 IAM 角色傳遞至 Amazon ECS 受管執行個體的一種方式。此設定檔用於：
 - 定義執行容器工作負載之 Amazon ECS 受管執行個體的 IAM 許可。
 - 允許 代表您 AWS 管理這些執行個體。
 - 讓執行個體根據設定檔中定義的許可存取 AWS 服務。

安全和合規

Amazon ECS 受管執行個體會實作多層安全性來保護工作負載：

- 安全組態 - Amazon ECS 受管執行個體遵循 AWS 安全最佳實務，包括無 SSH 存取、不可變的根檔案系統，以及透過 SELinux 的核心層級強制性存取控制。
- 自動修補 - 使用最新的安全修補程式 AWS 定期更新 Amazon ECS 受管執行個體，並遵循您設定的維護時段。
- 執行個體生命週期有限 - ECS 會在 14 天後自動啟動執行個體耗盡，確保您的應用程式在具有 up-to-date 安全修補程式的適當設定執行個體上執行。
- 特權功能 – 您可以選擇為需要特權 Linux 功能的工作負載 (例如網路監控與可觀測性解決方案) 啟用這些功能。

Amazon ECS 受管執行個體支援與 Amazon ECS 所支援的相同合規計畫，包括 PCI-DSS、HIPAA 與 FedRAMP。在支援的區域中，Amazon ECS 受管執行個體會遵守您的帳戶層級 FIPS 端點設定，協助達成 FedRAMP 合規。

聯網

Amazon ECS 受管執行個體支援 `awsvpc` 與 `host` 網路模式。`awsvpc` 網路模式為每項任務提供自己的彈性網路介面和 VPC 內的私有 IP 位址。這能夠在任務層級實現精細的安全群組與網路 ACL 控制。在 `host` 網路模式中，任務會共用主機 Amazon ECS 受管執行個體的網路命名空間。如需有關 Amazon ECS 受管執行個體上任務聯網設定的詳細資訊，請參閱 [Amazon ECS 受管執行個體的 Amazon ECS 任務聯網](#)。

執行個體儲存體

Amazon ECS 受管執行個體支援對連接至執行個體的 Amazon EBS 資料磁碟區的大小進行設定。此儲存體會在執行個體上執行的所有任務之間共用，並可用於綁定掛載。磁碟區可以在任務定義中使用 `volumes`、`mountPoint` 與 `volumesFrom` 參數的容器之間共用和掛載。

磁碟區會在執行個體建立期間連接。當您使用 `storageConfiguration` 參數建立 Amazon ECS 受管執行個體容量提供者時，可以指定磁碟區的大小，以 GiB 為單位。

```
{
  ...

  "managedInstancesProvider": {
    "infrastructureRoleArn": "arn:aws:iam::123456789012:role/ecsInfrastructureRole",
    "instanceLaunchTemplate": {
      "ec2InstanceProfileArn": "arn:aws:iam::123456789012:instance-profile/ecsInstanceProfile",
      "networkConfiguration": {
        "subnets": [
          "subnet-abcdef01234567",
          "subnet-bcdefa98765432"
        ],
        "securityGroups": [
          "sg-0123456789abcdef"
        ]
      },
      "storageConfiguration": {
        "storageSizeinGiB" : 100
      }
    }
  }
  ...
}
```

此磁碟區的大小下限為 30 GiB，大小上限為 16,384 GiB。此磁碟區的預設大小為 80 GiB。

任務的提取、壓縮與未壓縮的容器映像都會儲存在此磁碟區中。若要判斷任務必須用作綁定掛載的執行個體儲存總量，則必須從分配給任務的執行個體儲存總量中減去容器映像所使用的儲存量。

連接至 Amazon ECS 受管執行個體的 Amazon EBS 磁碟區效能符合對應的 Amazon EC2 執行個體的效能，如 Amazon EC2 User Guide 中的 [Amazon EBS-optimized instances](#) 文件所述。

您可以建立磁碟區的快照，對安全問題執行鑑識分析或對應用程式偵錯。如需有關建立 Amazon EBS 磁碟區快照的詳細資訊，請參閱 Amazon EBS User Guide 中的 [Amazon EBS snapshots](#)。如果您已預設啟用 Amazon EBS 加密，磁碟區將使用預設指定用於加密的 AWS KMS 金鑰進行加密。如需有關預設加密的詳細資訊，請參閱 Amazon EBS User Guide 中的 [Enable Amazon EBS encryption by default](#)。

除了使用連接至執行個體的資料磁碟區之外，您也可以為在 Amazon ECS 受管執行個體上執行的每項任務設定資料磁碟區。如需有關可用的任務層級儲存選項的詳細資訊，請參閱 [Amazon ECS 任務的儲存選項](#)。

本機儲存體

[Amazon EC2 執行個體存放區](#)為您的 Amazon EC2 執行個體提供臨時區塊層級儲存。Amazon EC2 執行個體存放區提供的儲存體可透過實體連接至主機的磁碟存取。您可以設定 Amazon ECS 受管執行個體，以使用執行個體存放區做為在容器執行個體上執行之任務的資料磁碟區。當您啟用本機儲存體且執行個體具有執行個體存放磁碟區時，Amazon ECS 會使用執行個體存放磁碟區，而不是佈建 Amazon EBS 資料磁碟區。這可以降低儲存成本，並改善延遲敏感工作負載的 I/O 效能。將執行個體存放磁碟區與 Amazon ECS 受管執行個體搭配使用不會產生額外費用。

當執行個體有多個執行個體存放磁碟區時，Amazon ECS 會自動將其合併為單一 RAID 0 磁碟區，並將其顯示為任務的連續儲存體。當執行個體沒有執行個體存放磁碟區，或本機儲存停用時，Amazon ECS 會佈建大小為 `storageSizeGiB` 的 Amazon EBS 資料磁碟區。執行個體存放區磁碟區是暫時性的。當執行個體停止、終止或硬體故障時，執行個體存放磁碟區上的資料會遺失。請勿將執行個體存放區用於必須保留的資料。

若要啟用本機儲存，請使用 `localStorageConfiguration` 參數[建立新的容量提供者](#)，並將 `useLocalStorage` 設定為 `true`。您也可以使用 `instanceRequirements` 參數來確保佈建的執行個體包含特定大小的本機儲存體。

Note

當您使用 建立叢集時 AWS 管理主控台，Amazon ECS 建立的預設受管執行個體容量提供者不包含 `localStorageConfiguration`。

下列範例顯示已啟用本機儲存的容量提供者組態，以及指定至少 50 GiB 之 SSD 型執行個體存放區的執行個體需求。

```
{
  ...
  "managedInstancesProvider": {
    "infrastructureRoleArn": "arn:aws:iam::123456789012:role/ecsInfrastructureRole",
    "instanceLaunchTemplate": {
      "ec2InstanceProfileArn": "arn:aws:iam::123456789012:instance-profile/ecsInstanceProfile",
      "networkConfiguration": {
        "subnets": [
          "subnet-abcdef01234567",
          "subnet-bcdefa98765432"
        ],
        "securityGroups": [
          "sg-0123456789abcdef"
        ]
      },
      "storageConfiguration": {
        "storageSizeGiB": 50
      },
      "localStorageConfiguration": {
        "useLocalStorage": true
      },
      "instanceRequirements": {
        "localStorage": "REQUIRED",
        "totalLocalStorageGB": {
          "Min": 50
        },
        "localStorageTypes": ["ssd"]
      }
    }
  }
}
```

當您省略instanceRequirements組態時，Amazon ECS 預設為 localStorage INCLUDED。在此情況下，Amazon ECS 會考慮使用和不使用執行個體存放區磁碟區的執行個體進行佈建。具有執行個體存放磁碟區的執行個體使用執行個體存放區處理資料磁碟區，而沒有執行個體存放磁碟區的執行個體則使用 Amazon EBS 資料磁碟區。totalLocalStorageGB 未指定時，Amazon ECS 會使用所有可用的執行個體儲存體，不會強制執行大小下限。

中的 `storageSizeGiB` 值只會 `storageConfiguration` 定義 Amazon EBS 資料磁碟區大小，並在執行個體儲存體無法使用時使用。中的 `totalLocalStorageGB` 值 `instanceRequirements` 控制佈建執行個體所需的執行個體儲存體大小下限。為了確保所有執行個體都使用本機儲存體，`localStorage` 請將設定為 `REQUIRED`，並使用指定最小大小 `totalLocalStorageGB`。

若要驗證特定佈建的容器執行個體是否使用本機儲存體，請檢查容器執行個體上的 `ecs.capability.storage.local-storage-enabled` 屬性。

服務負載平衡

使用 Amazon ECS 受管執行個體的 Amazon ECS 服務可以設定為使用 Elastic Load Balancing，將流量平均分配到服務中的各項任務。

Amazon ECS 受管執行個體上的 Amazon ECS 服務支援 Application Load Balancer、Network Load Balancer 與 Gateway Load Balancer 負載平衡器類型。Application Load Balancer 會路由 HTTP/HTTPS (第 7 層) 流量，而 Network Load Balancer 會路由 TCP 或 UDP (第 4 層) 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`。這是因為採用 `awsvpc` 網路模式的任務是與彈性網路介面相關聯，而非直接與 Amazon EC2 執行個體相關聯。

監控與可觀測性

Amazon ECS 受管執行個體透過 CloudWatch 指標提供全面的監控功能，並與可觀測性工具整合：

- CloudWatch 指標 – 在任務與執行個體層級監控 CPU、記憶體、網路以及儲存使用率。
- Container Insights – 取得容器化應用程式的詳細效能指標與日誌。
- 第三方整合 – 啟用特殊權限功能後，您可以執行需要更高 Linux 許可的進階監控與可觀測性解決方案。

定價與成本最佳化

使用 Amazon ECS 受管執行個體時，您需要支付執行任務的整個 Amazon EC2 執行個體的費用。定價取決於為您的工作負載選取的執行個體類型。

Amazon ECS 受管執行個體提供數種成本最佳化功能：

- 多任務最佳化 – 透過在適當大小的執行個體上執行多項任務，將執行個體使用率最大化。

您的 Compute Savings Plans 與 Instance Savings Plans 也適用於 Amazon ECS 受管執行個體工作負載。

Service Quotas

Amazon ECS 受管執行個體工作負載受限於您的 Amazon EC2 隨需執行個體服務配額。使用 Amazon ECS 受管執行個體的 Amazon ECS 服務受限於 Amazon ECS 服務配額。

如需有關服務配額的詳細資訊，請參閱：

- Amazon Web Services 一般參考 中的 [Amazon EC2 endpoints and quotas](#)
- [Amazon ECS 服務配額](#)

遷移考量事項

對於大多數工作負載而言，遷移至 Amazon ECS 受管執行個體非常簡單：

- 從 Fargate 遷移 – 僅需要容量提供者組態變更與重新部署。使用平台版本 1.4.0 的現有任務定義完全相容。
- 從 EC2 遷移 – 與遷移至 Fargate 類似，但保留對 Amazon EC2 功能 (例如特定執行個體類型) 的存取權。

規劃遷移時請考量下列事項：

- 應用程式應容忍 14 天的執行個體生命週期和計劃的維護時段。
- 長時間執行的任務 (超過 14 天) 不適用於 Amazon ECS 受管執行個體。
- 不支援自訂 AMIs - Amazon ECS 受管執行個體使用 AWS 受管、安全最佳化 AMIs。

限制及考量

下列限制適用於 Amazon ECS 受管執行個體：

- 自訂 AMIs - AMI 由 擁有和管理 AWS
- 執行個體生命週期 – 為確保安全修補與合規，每個執行個體的執行時期上限為 14 天。
- SSH 存取 – 出於安全原因而無法使用。使用 Amazon ECS Exec 進行偵錯與疑難排解。管理操作僅透過 Amazon ECS API 進行。

組織控制

有些組織控制可防止 Amazon ECS 受管執行個體正常運作。若是如此，您必須更新這些控制項，以允許 Amazon ECS 擁有代表您管理 EC2 執行個體所需的許可。

Amazon ECS 使用基礎設施角色來啟動支援 Amazon ECS 受管執行個體的 EC2 執行個體。此基礎設施角色是在您的帳戶中建立的 [IAM 角色](#)，可讓 Amazon ECS 代表您管理 Amazon ECS 受管執行個體。[服務控制政策](#) (SCPs) 一律適用於使用基礎設施角色執行的動作。這可讓 SCP 禁止 Amazon ECS 受管執行個體操作。最常見的情況是利用 SCP 來限制可以啟動的 Amazon Machine Image (AMI)。若要允許 Amazon ECS 受管執行個體正常運作，請修改 SCP 以允許從 Amazon ECS 受管執行個體 AMIs 帳戶啟動 AMI。

您也可使用 [EC2 允許的 AMI](#) 功能來限制其他帳戶中 AMI 的可見性。如果您使用此功能，則必須展開映像條件，在感興趣的區域中也包含 Amazon ECS 受管執行個體 AMI 帳戶。

封鎖所有 AMIs SCP 範例，但 Amazon ECS 受管執行個體 AMIs 除外

以下 SCP 可防止呼叫，`ec2:RunInstances` 除非 AMI 屬於 `us-west-2` 或 `us-east-1` 的 Amazon ECS 受管執行個體 AMI 帳戶。

Note

請務必不要使用 `ec2:Owner` 內容索引鍵。Amazon 擁有 Amazon ECS 受管執行個體 AMI 帳戶，此金鑰的值一律為 `amazon`。如果 `ec2:Owner` 為 `amazon` 將允許啟動任何 Amazon ECS 受管執行個體的 AMIs，則建構允許啟動 AMI 的 SCP。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "DenyAMI",
      "Effect": "Deny",
      "Action": "ec2:RunInstances",
      "Resource": "arn:*:ec2:*::image/ami-*",
      "Condition": {
        "StringNotEquals": {
          "aws:ResourceAccount": [
            "187296253231",
            "260073348889"
          ]
        }
      }
    }
  ]
}
```

```

    ]
  }
}
}
]
}

```

Amazon ECS 受管執行個體 AMI 帳戶

AWS 帳戶因區域主機 Amazon ECS 受管執行個體公有 AMIs 而異。

AWS 區域	帳戶
af-south-1	070957084703
ap-east-1	587573215167
ap-northeast-1	679336465495
ap-northeast-2	309903600357
ap-northeast-3	384570461223
ap-south-1	062344138989
ap-south-2	624198668379
ap-southeast-1	832199679391
ap-southeast-2	552073033681
ap-southeast-3	368903466070
ap-southeast-4	696793786439
ap-southeast-5	003457290689
ap-southeast-6	465836752572
ap-southeast-7	622515864387
ca-central-1	853167153192

AWS 區域	帳戶
ca-west-1	899469777611
eu-central-1	832570432258
eu-central-2	041659148495
eu-north-1	851563870067
eu-south-1	766433696616
eu-south-2	003380494496
eu-west-1	986619735082
eu-west-2	591706807364
eu-west-3	108582616801
il-central-1	009537862704
me-central-1	540883425316
me-south-1	181438624895
mx-central-1	210749644920
sa-east-1	591338347621
us-east-1	260073348889
us-east-2	292185169523
us-west-1	187296253231
us-west-2	491085424538

Amazon ECS 受管執行個體的執行個體類型

Amazon ECS 受管執行個體讓您可以為容器化應用程式選取特定的 EC2 執行個體類型。

Amazon ECS 受管執行個體的執行個體系列

支援下列執行個體類型：

一般用途

- m5、m5a、m5ad、m5d、m5dn、m5n、m5zn：平衡運算、記憶體與聯網
- m6a、m6g、m6gd、m6i、m6id、m6idn、m6in：最新一代執行個體，效能得到提升
- m7a、m7g、m7gd、m7i、m7i-flex：下一代一般用途執行個體
- m8g、m8gd：最新一代 ARM 一般用途執行個體
- t3、t3a、t4g：爆量效能執行個體（奈米和微型執行個體大小除外）

運算最佳化

- c5、c5a、c5ad、c5d、c5n：適用於運算密集型應用程式的高效能處理器
- c6a、c6g、c6gd、c6i、c6id、c6in：最新一代運算最佳化執行個體
- c7a、c7g、c7gd、c7gn、c7i、c7i-flex：下一代運算最佳化執行個體
- c8g、c8gd、c8gn：最新一代 ARM 運算最佳化執行個體
- hpc6a、hpc6id、hpc7a：高效能運算執行個體

記憶體最佳化

- r5、r5a、r5ad、r5b、r5d、r5dn、r5n：高記憶體與 vCPU 比率執行個體，適用於記憶體密集型應用程式
- r6a、r6g、r6gd、r6i、r6id、r6idn、r6in：最新一代記憶體最佳化的執行個體
- r7a、r7g、r7gd、r7i、r7iz：下一代記憶體最佳化的執行個體
- r8g、r8gd：最新一代 ARM 記憶體最佳化的執行個體
- u-3tb1、u7i-6tb、u7i-8tb、u7i-12tb、u7in-24tb、u7in-32tb：高記憶體執行個體，RAM 高達 32 TB
- x2gd、x2idn、x2iedn、x2iezn：適用於記憶體資料庫與分析的超大記憶體
- x8g：最新一代超大記憶體執行個體
- z1d：高頻率與 NVMe SSD 儲存體

儲存優化

- d3、d3en：適用於分散式檔案系統的密集 HDD 儲存體
- i4g、i4i：最新一代儲存最佳化執行個體
- i7i、i7ie、i8g：下一代高效能儲存執行個體
- im4gn、is4gen：網路最佳化儲存執行個體

加速運算

- g4dn：搭載 NVIDIA T4 GPU，適用於機器學習推論與圖形
- g5、g5g：搭載 NVIDIA A10G GPU，適用於高效能圖形與 ML
- g6、g6e、g6f：最新一代 GPU 執行個體
- gr6、gr6f：搭載 NVIDIA L4 Tensor 核心 GPU 的 GPU 執行個體，vCPU:RAM 比率為 1:8，適用於圖形工作負載
- p3dn：搭載 NVIDIA V100 GPU，適用於深度學習訓練與 HPC
- p4d：搭載 NVIDIA A100 GPU，可實現最高效能的機器學習訓練
- p5：搭載 NVIDIA H100 GPU 的最新一代執行個體
- p6-b200：搭載 NVIDIA B200 GPU 的下一代執行個體

執行個體選擇方法

Amazon ECS 受管執行個體提供兩種選取執行個體類型的方法：

- 特定執行個體類型選取：您可以明確指定要用於任務的 EC2 執行個體類型。
- 屬性型執行個體類型選取：您可以指定應用程式所需的屬性 (例如 vCPU、記憶體與架構)，Amazon ECS 受管執行個體會選取適當的執行個體類型。

特定執行個體類型選取

透過特定執行個體類型選取，您可以明確指定要用於 Amazon ECS 受管執行個體任務的 EC2 執行個體類型。當您的應用程式需要具有特定硬體特性的特定執行個體類型時，此方法很有用。

屬性型執行個體類型選項

透過屬性型執行個體類型選取，您可以指定應用程式所需的屬性，Amazon ECS 受管執行個體會選取符合這些需求的適當執行個體類型。這可提供更大的彈性，並有助於確保即使特定執行個體類型不可用，也能成功置放任務。

指定多個屬性時，您會取得滿足所有指定屬性的執行個體類型。若您為某個屬性指定多個值，則您將取得滿足任何指定值的執行個體類型。

屬性型執行個體類型選取支援下列屬性：

cpuArchitecture

CPU 架構。

有效值：X86_64 | ARM64

instanceGeneration

指示是否包含目前世代或上一代執行個體類型。

- 針對目前世代的執行個體類型，請指定 `current`。目前世代包含目前建議使用的 EC2 執行個體類型。這通常會包含每個執行個體系列中最新的二至三代。如需詳細資訊，請參閱《Amazon EC2 使用者指南》中的[執行個體類型](#)。
- 針對上一代執行個體類型，請指定 `previous`。
- 若要同時包含目前和上一代執行個體類型，請指定 `all`。

有效值：`current` | `previous` | `all`

預設：任何目前或上一代。

burstablePerformance

指示是否包含、排除或需要爆量效能執行個體類型。如需詳細資訊，請參閱《Amazon EC2 使用者指南》中的[爆量效能執行個體](#)。

有效值：`included` | `excluded` | `required`

預設：`excluded`

cpuManufacturer

請列出要包含的特定 CPU 製造商。

- 針對使用 Intel CPU 的執行個體類型，請指定 `intel`。

- 針對使用 AMD CPU 的執行個體類型，請指定 `amd`。
- 對於具有 AWS CPUs 執行個體類型（例如 AWS Graviton），請指定 `amazon-web-services`。

 Note

請勿混淆 CPU 硬體製造商與 CPU 硬體架構。執行個體會根據您指定的 Amazon Machine Image (AMI)，以相容的 CPU 架構啟動。

有效值：`intel` | `amd` | `amazon-web-services`

預設：任何製造商。

`networkBandwidth`

網路頻寬下限和上限，以每秒 Gb (Gbps) 為單位。

預設：沒有下限或上限。

`networkInterfaceCount`

網路介面數目的下限和上限。

預設：沒有下限或上限。

`localStorage`

指示是否包含、排除或需要執行個體儲存體磁碟區的執行個體類型。如需詳細資訊，請參閱 [《Amazon EC2 使用者指南》中的 Amazon EC2 執行個體存放區](#)。Amazon EC2

有效值：`included` | `excluded` | `required`

預設：`included`

`localStorageType`

指示所需的本機儲存體類型。

- 針對具有硬碟 (HDD) 儲存體的執行個體類型，請指定 `hdd`。
- 針對具有固態硬碟 (SSD) 儲存體的執行個體類型，請指定 `ssd`。

有效值：`hdd` | `ssd`

預設：任何本機儲存類型。

帳單和購買選項

Amazon ECS 受管執行個體支援多種功能，可協助最佳化容器化工作負載的成本：

- Savings Plans (SPs)：Amazon ECS 受管執行個體受益於您為任務使用的執行個體類型購買的 Savings Plans。不需任何其他設定。
- 預留執行個體 (RI)：Amazon ECS 受管執行個體任務可以從您為任務使用的執行個體類型購買的 RI 中受益。不需任何其他設定。
- Spot 執行個體：您可以透過設定 將 Amazon ECS 受管執行個體容量提供者設定為使用 EC2 Spot 執行個體 `capacityOptionType=Spot`
- 容量保留：您可以透過設定 `capacityOptionType=Reserved` 並提供容量保留群組，將 Amazon ECS 受管執行個體容量提供者設定為使用您的 EC2 容量保留。 <https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/create-cr-group.html> 您也可以指定下列保留偏好設定：使用 `reservations-only` 來確保執行個體僅以預留容量啟動，以達到最大可預測性、`reservations-first` 偏好保留，同時在需要時保持恢復隨需容量的彈性，或 `reservations-excluded` 防止您的容量提供者同時使用保留。

Amazon ECS 受管執行個體的執行個體選擇最佳實務

為 Amazon ECS 受管執行個體工作負載選取正確的執行個體組態，對於最佳化效能、成本與資源使用率至關重要。Amazon ECS 提供靈活的執行個體選擇選項，讓您可以平衡應用程式需求與成本效益。下列最佳實務可協助您就容器化工作負載的執行個體選擇作出明智的決策。

1. 使用預設執行個體選取

當您未指定 `instanceRequirements`，Amazon ECS 會選擇符合下列任務定義和服務參數需求的最具成本效益的執行個體：

任務定義

- `operatingSystemFamily`
- `cpuArchitecture`
- `cpu`
- `memory`

服務定義

- `placementConstraints`

- placementStrategy

2. 針對大多數工作負載使用屬性型選取方法，以提供彈性並改善置放成功率

屬性型執行個體選擇方法讓 Amazon ECS 可以從更廣泛的執行個體類型中進行選擇，以符合您指定的需求。這種方法讓 Amazon ECS 可以選取啟動時最具成本效益的執行個體，從而提高成功置放任務的可能性並取得更好的成本最佳化效果。

3. 僅在應用程式有特定硬體需求時，才使用特定的執行個體類型

為需要特定硬體功能 (例如 GPU 加速、高頻率處理器或專用聯網功能) 的工作負載保留特定執行個體類型選取方法。對於一般用途應用程式，屬性型選取通常具有更好的彈性與成本最佳化效果。

4. 選擇平衡的資源，避免過度佈建與不必要的成本

選取最符合應用程式的 CPU 與記憶體需求的執行個體組態。避免大量過度佈建資源，因為這會導致成本增加且效率降低。使用監控資料來了解您的實際資源使用率模式，並相應地調整執行個體選擇。

5. 針對具有不同工作負載的應用程式混合使用執行個體類型，以平衡效能與成本

對於具有不同效能需求或不同工作負載模式的應用程式，請考慮使用具有不同執行個體組態的多個容量提供者。此方法可讓您針對應用程式的不同元件使用適當的執行個體類型來最佳化成本，同時視需要維持效能。

6. 使用以設定的 Amazon ECS 受管執行個體容量提供者時 `capacityOptionType=Reserved`，請注意 ECS 服務使用預設部署組態 `minimumHealthyPercent=100%` 和 `maximumPercent=200%`，這表示 ECS 部署會在停止舊任務之前嘗試啟動新任務，並暫時需要高達 200% 的穩定狀態容量。如果您的服務在穩定狀態下使用 EC2 容量預留中的所有可用容量，部署將會失敗，因為在部署程序期間沒有額外的容量可用於啟動新任務。若要避免這種情況，`minimumHealthyPercent` 請將設定為低於 100% (例如 75%)，並考慮 `maximumPercent` 將設定為 100%，以確保服務在啟動新的任務之前停止任務，允許部署在啟動替代任務之前釋放容量，以成功完成部署。此外，請考慮定期監控容量使用率，以維持保留空間，以適應部署並處理流量激增。

Amazon ECS 受管執行個體容器映像提取行為

容器啟動所需的時間會因底層容器映像而異。例如，比起各自的精簡版本 (Debian-slim、Ubuntu-slim 和 Amazon2-slim) 或較小的基礎映像 (Alpine)，容器中執行的更多服務可能需要更長的時間才能啟動。

當 Amazon ECS 啟動在 Amazon ECS 受管執行個體上執行的任務時，Amazon ECS 只有在相同容器執行個體上先前的任務未提取映像，或是自動映像清除程序移除快取映像時，才會從遠端提取映像。否則，會使用 Amazon ECS 受管執行個體上的快取映像。這可確保不會發生非必要的映像提取。

Amazon ECS 受管執行個體中的修補

在 Amazon ECS 受管執行個體中，修補是一項重要的維護程序，其中 AWS 會自動管理和更新受管容器執行個體上的軟體，以確保安全性和合規性，同時透過受控且可設定的程序維持工作負載可用性。

執行個體生命週期

依預設，Amazon ECS 受管容器執行個體的標準化運作生命週期為 14 至 21 天。Amazon ECS 會在執行個體啟動後第 14 天啟動優雅的工作負載耗盡，並最晚在第 21 天最終終止。Amazon ECS 在特定情況下適應提早耗盡：

- 偵測軟體上的安全漏洞
- 硬體運作狀態降級
- 遵守客戶設定的事件時段

此方法可維持系統合規性，同時符合客戶定義的維護需求。

事件時段與維護排程

AWS 透過監控節點建立時間戳記和維護排程的自動化背景程序來管理受管容器執行個體生命週期。執行個體啟動時，會 AWS 設定預設的 14 天耗盡排程，並評估任何客戶設定的事件時段。

您可以透過設定事件時段來排程 Amazon ECS 受管執行個體的維護活動。您可以使用執行個體 IDs 或執行個體標籤，將一或多個 Amazon ECS 受管執行個體與事件時段建立關聯。使用特定值標記事件時段時，Amazon ECS 會將這些標籤映射至對應叢集的對應 Amazon ECS 受管執行個體，並盡最大努力在定義的期間內排程執行個體維護。

如需事件時段的詳細資訊，請參閱 [《Amazon EC2 使用者指南》中的為影響 Amazon EC2 執行個體的排程事件建立自訂事件時段](#)。Amazon EC2

如果事件時段存在，會 AWS 調整耗盡排程以符合這些時段，這可能會導致比預設的 14 天期間更早耗盡，以符合指定的事件時段。事件時段修改僅會影響新啟動的受管容器執行個體，從而確保預測性維護的排程。

在排程的耗盡時間之前，Amazon ECS 會根據客戶的組態，在受管容器執行個體上繼續正常的任務置放操作。

維護序列

在指定的維護時間，Amazon ECS 會透過調用 `UpdateContainerInstancesState` API 來啟動優雅的工作負載耗盡，開始其維護序列。在優雅終止期間，Amazon ECS 會嘗試停止標記為耗盡的執行個體上的工作負載。

Amazon ECS 會對服務任務採用先啟動後停止的策略 (或根據 Amazon ECS 服務組態)，確保在停止現有任務之前啟動替代任務，將服務中斷降至最低。在此程序中，Amazon ECS 服務會遵守所有服務部署組態，同時繼續進行耗盡嘗試，直到執行個體啟動後第 21 天為止。

如果到第 21 天耗盡仍未完成，Amazon ECS 會執行 `DeregisterContainerInstance` API 來停止受管容器執行個體及其剩餘的工作負載，以維持合規性並使用最新的軟體修補受管執行個體。

Amazon ECS 受管執行個體的安全考量

Amazon ECS 受管執行個體提供全受管容器運算體驗，可讓您在特定 Amazon EC2 執行個體類型上執行工作負載，同時將安全性交由 AWS 負責。本主題介紹了使用 Amazon ECS 受管執行個體時的安全模型、功能與考量。

安全模型

Amazon ECS 受管執行個體會實作全面的安全模型，在彈性與保護之間取得平衡：

- AWS 受管基礎設施 - AWS 控制受管執行個體的生命週期，並處理安全修補，消除人為錯誤和竄改的可能性。
- 無管理存取 – 安全模型已鎖定，並禁止對受管執行個體進行管理存取。
- 多任務置放 – 依預設，Amazon ECS 受管執行個體會將多個任務置放在單一執行個體上，以最佳化成本與使用率。相較於 Fargate，這會放寬工作負載隔離限制。
- 資料隔離 - 雖然 AWS 控制執行個體生命週期和任務置放，但 AWS 無法登入受管執行個體或存取客戶資料。

安全性功能

Amazon ECS 受管執行個體包含數個內建的安全功能，旨在保護工作負載並維持良好的安全狀態。這些功能涵蓋自動化安全修補到在需要時支援特權 Linux 功能。

安全最佳實務

受管執行個體是根據 AWS 安全最佳實務設定，包括：

- 無 SSH 存取 – 遠端 Shell 存取已停用，以防止未經授權的存取。
- 不可變的根檔案系統 – 根檔案系統無法修改，從而可確保系統完整性。
- 核心層級強制性存取控制 – SELinux 在核心層級提供額外的安全性強制執行。

自動安全修補

Amazon ECS 受管執行個體會透過自動化修補，協助改善工作負載的安全狀態：

- 定期安全性更新 - 執行個體會根據您設定的維護時段 AWS，定期使用最新的安全性修補程式進行更新。
- 執行個體生命週期有限 – 執行中的執行個體的生命週期上限限制為 14 天，以確保應用程式在具有適當設定與最新安全性修補程式的執行個體上執行。
- 維護時段控制 – 您可以使用 Amazon EC2 事件時段功能來指定 Amazon ECS 何時應以修補後的執行個體取代原執行個體。

特權 Linux 功能

Amazon ECS 受管執行個體支援需要更高 Linux 權限的軟體，從而啟用進階監控與安全性解決方案：

- 支援的功能 – 您可以選擇使用所有特權 Linux 功能，包括 CAP_NET_ADMIN、CAP_SYS_ADMIN 與 CAP_BPF。
- 熱門解決方案 – 這讓您能夠執行熱門的網路監控與可觀測性解決方案，例如 Wireshark 與 Datadog。
- 需要明確組態 – 您必須明確設定 Amazon ECS 受管執行個體容量提供者，以啟用特權 Linux 功能，因為它可能會對您的應用程式造成額外的安全風險。

Important

啟用特權 Linux 功能可能會讓您的任務面臨額外的安全風險。請僅在應用程式需要時才啟用這些功能，並確保您了解安全性影響。

合規與法規支援

Amazon ECS 受管執行個體會維持與 Amazon ECS 相同的合規狀態：

- 合規計劃 - Amazon ECS 受 AWS 管執行個體的範圍與 Amazon ECS 相同，包括 PCI-DSS、HIPAA 和 FedRAMP。
- FIPS 端點 - Amazon ECS 受管執行個體遵守您的帳戶層級設定，以便在 AWS 區域中使用 FIPS 端點，以協助達成 FedRAMP 合規。
- 客戶自管金鑰 – 它支援實現合規所需的安全功能，例如用於加密的客戶自管金鑰。

Amazon ECS 受管執行個體 FIPS-140 考量

在 Amazon ECS 受管執行個體上使用 FIPS-140 合規時，請考慮下列事項：

- FIPS-140-compliant受管執行個體 AMIs 僅適用於 AWS GovCloud (US) 區域。
- Amazon ECS 受管執行個體支援 FIPS-140-3
- 區域預設會啟用 FIPS-140 AWS GovCloud (US) 合規。如果您需要在沒有 FIPS 合規的情況下執行工作負載，請在受管執行個體容量提供者組態中關閉 FIPS 合規。
- 任務cpuArchitecture的 必須符合 FIPS-140 X86_64 合規。

在 Amazon ECS 受管執行個體上停用 FIPS

根據預設，AWS GovCloud (US) 區域中的 Amazon ECS 受管執行個體容量提供者會啟動符合 FIPS 標準的 AMIs。在建立新的 Amazon ECS 受管執行個體容量提供者時，您可以選擇停用 FIPS-140 合規。請依照下列步驟建立沒有 FIPS 合規的新容量提供者。

1. 在容量提供者上停用 FIPS-140 合規。

```
aws ecs create-capacity-provider \
  --cluster cluster-name \
  --name capacity-provider-name \
  --managed-instances-provider '{
    "infrastructureRoleArn": "infrastructure-role-arn",
    "instanceLaunchTemplate": {
      "ec2InstanceProfileArn": "instance-profile-arn",
      "fipsEnabled": false,
      "networkConfiguration": {
```

```
        "subnets": ["subnet-id"],
        "securityGroups": ["security-group-id"]
    }
}
```

2. 您可以選擇性地使用 ECS Exec 執行下列命令，以驗證容量提供者的 FIPS-140 合規狀態。

將 *cluster-name* 取代為叢集名稱，將 *task-id* 取代為任務的 ID 或 ARN，並將 *container-name* 取代為任務中要執行命令的容器名稱。

傳回值 "1" 表示您正在使用 FIPS。

```
aws ecs execute-command \
  --cluster cluster-name \
  --task task-id \
  --container container-name \
  --interactive \
  --command "cat /proc/sys/crypto/fips_enabled"
```

安全考量

使用 Amazon ECS 受管執行個體時，需要了解和規劃數個重要的安全考量。這些考量可協助您就工作負載架構與安全需求作出明智的決策。

多任務安全模型

Amazon ECS 受管執行個體中的預設多任務置放模型與 Fargate 的單一任務隔離不同：

- 共用執行個體資源 – 多項任務在同一執行個體上執行，可能會讓任務暴露於在同一個執行個體上或同一個 ECS 叢集中執行之其他任務所存在的漏洞。
- 單一任務選項 – 您可以將 Amazon ECS 受管執行個體設定為對需要具有虛擬機器層級安全隔離界限之預設 Fargate 安全模型的客戶使用單一任務模式。
- 成本與安全權衡 – 多任務模式可實現成本最佳化與更快的任務啟動時間，而單一任務模式可提供更強大的隔離。

處理執行個體中斷問題

使用 Amazon ECS 受管執行個體時，請務必將應用程式設計為能夠容忍中斷情況：

- 中斷容忍 – 將 Amazon ECS 受管執行個體與可容忍基礎服務或任務中斷的應用程式搭配使用。
- 服務型工作負載 – 使用 Amazon ECS 服務進行自動任務取代，或在獨立任務上執行受控制且持續時間限制不超過 14 天的工作負載。
- 優雅關閉 – 設定任務關閉寬限期以控制中斷產生的影響。

資料存取與隱私

Amazon ECS 受管執行個體會維持嚴格的資料存取控制：

- 無客戶資料存取 - 雖然 AWS 控制受管執行個體的生命週期和執行個體上的任務置放，AWS 但無法登入受管執行個體或存取客戶資料。
- 僅限指標和日誌 - 僅 AWS 擷取提供 Amazon ECS 受管執行個體功能所需的指標和相關日誌。
- 鎖定的安全模型 – 安全模型禁止管理存取，從而可消除人為錯誤與竄改的可能性。

安全最佳實務

使用 Amazon ECS 受管執行個體時，請遵循下列最佳實務：

- 評估安全模型 – 根據您的安全需求，謹慎地決定採用 Amazon ECS 受管執行個體，特別是在多任務置放模型方面。
- 視需要使用單一任務模式 – 如果您的工作負載需要更強大的隔離，請將 Amazon ECS 受管執行個體設定為使用單一任務模式。
- 將特權功能降至最低 – 請僅在絕對必要且了解相關安全風險時，才啟用特權 Linux 功能。
- 制定中斷計畫 – 將應用程式設計為以優雅的方式處理執行個體取代，尤其是考量 14 天的執行個體生命週期上限。
- 設定維護時段 – 使用 EC2 事件時段來控制執行個體取代的時間，將對工作負載的影響降至最低。
- 監控與稽核 – 定期檢閱您的 Amazon ECS 受管執行個體組態，並監控任何與安全性相關的事件或變更。

啟用 Amazon ECS 受管執行個體的 VPC 加密控制

Amazon ECS 受管執行個體支援 VPC 加密控制，這是一種安全與合規功能，可提供集中式控制，以監控和強制執行傳輸中所有流量的加密，而這些流量會跨區域中 VPCs。當子網路上啟用 VPC 加密控制時，您可以指定支援 Amazon ECS 受管執行個體自訂容量提供者傳輸中加密的執行個體類型，以確保 Amazon ECS 受管執行個體工作負載在傳輸中以加密執行。

先決條件

開始之前，您需要：

- 在子網路上啟用傳輸中加密的 VPC。如需詳細資訊，請參閱 [VPC 加密控制文件](#)。
- Amazon ECS 受管執行個體自訂容量提供者。如需詳細資訊，請參閱 [Amazon ECS 受管執行個體的架構](#)。

識別相容的執行個體類型

Amazon EC2 執行個體類型必須符合兩個要求：

1. 支援傳輸中的 VPC 加密 - 使用以下 AWS CLI 命令列出支援傳輸中加密的 Amazon EC2 執行個體類型：

```
aws ec2 describe-instance-types \
  --filters Name=network-info.encryption-in-transit-supported,Values=true \
  --query "InstanceTypes[*].[InstanceType]" \
  --output text | sort
```

2. Amazon ECS 受管執行個體支援 - Amazon ECS 受管執行個體支援的所有 Amazon EC2 執行個體類型都記錄在 [Amazon ECS 受管執行個體的執行個體類型](#)。

如果您有其他需求（例如特定的 CPU、記憶體或架構需求），請根據您的工作負載需求進一步篩選相容的執行個體類型。

建立支援 VPC 加密的叢集

若要為傳輸中的 VPC 加密設定 Amazon ECS 受管執行個體：

1. 建立新的叢集，並選取基礎設施的 Fargate 和受管執行個體。
2. 選取使用自訂 – 進階以存取其他組態參數。
3. 在允許的執行個體類型中，僅新增支援傳輸中 VPC 加密的特定執行個體類型。

以這種方式設定時，Amazon ECS 受管執行個體只會啟動支援傳輸中 VPC 加密的 Amazon EC2 執行個體類型。

考量事項

- 爆量效能執行個體 - T3, T3a 和 T4g 執行個體類型不支援傳輸中的 VPC 加密，且無法在強制執行模式下啟用加密控制的子網路中使用。
- 模式轉換 - 只有在所有執行中的執行個體都支援傳輸中的 VPC 加密時，您才能將 VPC 子網路從監控模式轉換至強制執行模式。
- 任務啟動失敗 - 在強制執行模式中，如果您指定不支援傳輸中加密的執行個體類型，任務將無法啟動。

疑難排解

強制執行模式中的任務啟動失敗

如果任務無法啟動，請使用上述提供的 AWS CLI 命令，驗證所有指定的執行個體類型是否支援傳輸中的 VPC 加密。

無法轉換為強制執行模式

使用 主控台 或 `GetVpcResourcesBlockingEncryptionEnforcement` 命令來識別未強制執行傳輸中加密的資源。

如需 VPC 加密控制的詳細資訊，請參閱 [VPC 加密控制文件](#)。

Amazon ECS 受管執行個體基礎結構最佳化

Amazon ECS 受管執行個體會根據您的容量提供者組態和目前的工作負載需求，自動佈建大小正確的 EC2 執行個體，確保您的容器化應用程式從部署時就擁有適當的運算資源。隨著應用程式流量模式的演進和工作負載需求隨時間變化，Amazon ECS 受管執行個體會透過智慧調整執行個體大小以符合目前需求、主動取代偏離最佳組態的執行個體，以及動態平衡成本效益、應用程式效能和系統可靠性，來持續監控和最佳化您的基礎設施。此資源管理系統無需任何手動介入即可運作，可降低基礎設施成本，同時維持應用程式的高可用性。

基礎結構最佳化具有下列優點：

- 成本最佳化 – 透過最大化資源使用率並消除閒置容量來降低基礎結構成本
- 效能改善 – 基於資源需求與效能特性來最佳化工作負載置放
- 操作簡單 – 自動作出複雜的資源管理決策，而無需人工介入

- 可靠性增強 – 透過智慧型工作負載分佈與運作狀態監控來維持高可用性

Amazon ECS 受管執行個體會執行兩種類型的基礎設施最佳化，以最大化效率並降低成本：

閒置執行個體偵測

識別和移除沒有執行中任務的 EC2 執行個體，消除未使用的容量中不必要的基礎設施成本。偵測到閒置執行個體時，最佳化程序會將容器執行個體標記為 DEREGISTERING，這會啟動可安全終止基礎 EC2 執行個體的清除序列。

未充分利用的執行個體偵測

分析跨執行個體的任務分佈，以識別更好的資源配置機會。當任務在多個執行個體之間以次最佳化方式執行時，Amazon ECS 受管執行個體會將工作負載合併到較少、更有效率的執行個體，同時降低整體成本，同時維持效能。最佳化程序會將未充分利用的容器執行個體標記為 DRAINING，這會觸發任務取代，將工作負載移至現有或更有效率的新執行個體。安全遷移所有任務後，執行個體會轉換為 DEREGISTERING 狀態並清除。此最佳化適用於執行服務任務的執行個體，並透過遵守服務的最低和最高任務限制、遵守start-before-stop部署行為，以及在整個耗盡過程中維護任何任務保護設定，以確保安全整合。執行獨立任務的執行個體不會考慮進行最佳化，因為 ECS 受管執行個體不會取代獨立任務。

這些最佳化可協同運作，確保您的基礎設施持續適應實際的工作負載需求，自動消除浪費並改善資源使用率，而不會影響應用程式的可用性。這兩種機制都使用事件驅動型監控來回應任務和執行個體生命週期事件，以即時識別最佳化機會。Amazon ECS 受管執行個體會偵測容器執行個體上最後一個任務何時停止，指出成本最佳化的潛在閒置條件。對於未充分利用的執行個體，任何任務停止或新執行個體啟動都會觸發分析，以識別工作負載整合的機會並改善資源效率。

ScaleInAfter

這兩個基礎設施最佳化都尋找終止執行中執行個體的機會，以改善使用率並降低成本。您可以使用 Amazon ECS 受管執行個體容量提供者設定中的 ScaleInAfter 組態來控制這些動作的時間，這同時適用於閒置和未充分利用的執行個體。ScaleInAfter 可讓您指定執行個體閒置或未充分利用與 Amazon ECS 受管執行個體開始最佳化基礎設施之間的延遲，以秒為單位。您可以設定 0 到 3600 秒之間的延遲。您也可以指定 -1 來停用基礎設施最佳化。

閒置執行個體

- ECS 會在最後一個任務停止後等待指定的持續時間，再取消註冊執行個體
- 如果新的任務在等待期間啟動，則執行個體不會再被視為閒置，並取消終止

未充分利用的執行個體

- ECS 會在任務停止事件導致執行個體使用率不足後等待指定的持續時間，再耗盡執行個體
- 如果在等待期間啟動新任務或在特定執行個體上停止現有任務，計時器會從最近的任務停止或新任務建立時間重設，Amazon ECS 受管執行個體會重新評估效率低下，並視需要新的等待期間到期後採取動作

此組態為選擇性。未指定時，ECS 受管執行個體會根據 ECS 受管執行個體預設組態自動決定最佳時間。

從 遷移 AWS Fargate 至 Amazon ECS 受管執行個體

您可以將現有的工作負載從 Fargate 遷移至 Amazon ECS 受管執行個體。此遷移可讓您存取完整的 Amazon EC2 執行個體類型、容量保留和進階功能，同時維護 AWS 受管基礎設施。

遷移考量事項

從 Fargate 遷移至 Amazon ECS 受管執行個體時，請記住下列考量事項：

任務相容性

為 Fargate 設定的現有任務定義主要與 Amazon ECS 受管執行個體相容。如需任務定義差異的詳細資訊，請參閱 [Amazon ECS 受管執行個體的 Amazon ECS 任務定義差異](#)。

安全模型變更

Amazon ECS 受管執行個體預設允許每個執行個體執行多項任務。如果您的工作負載需要更強大的隔離，請考量啟用單一任務模式。

執行個體生命週期

Amazon ECS 受管執行個體的生命週期上限為 14 天。請制定任務替代計畫，並使用 Amazon ECS 服務進行自動任務管理。

定價變更

使用 Amazon ECS 受管執行個體時，您需要支付整個執行個體的費用加上管理費，而不是像 Fargate 一樣按每項任務資源付費。

維護時段

使用 Amazon EC2 事件時段設定維護時段，以控制何時取代 Amazon ECS 受管執行個體進行修補。

先決條件

遷移至 Amazon ECS 受管執行個體之前，請確定您：

- 具有在平台版本 1.4.0 或更新版本上執行的現有 Fargate 任務
- 您有 Amazon ECS 受管執行個體所需的 IAM 角色。其中包含：
 - 基礎設施角色 - 允許 Amazon ECS 代表您呼叫 AWS 服務，以管理 Amazon ECS 受管執行個體基礎設施。

如需詳細資訊，請參閱[Amazon ECS 基礎結構 IAM 角色](#)。

- 執行個體設定檔 – 為在受管執行個體上執行的 Amazon ECS 容器代理程式與 Docker 常駐程式提供許可。

如需詳細資訊，請參閱[Amazon ECS 受管執行個體執行個體設定檔](#)。

- 了解 Fargate 與 Amazon ECS 受管執行個體之間的安全模型差異

Important

Amazon ECS 受管執行個體使用與 Fargate 不同的安全模型。依預設，多個任務可以在同一個執行個體上執行，這可能會讓任務暴露於其他任務所存在的漏洞。遷移之前，請先檢閱您的安全性需求。

步驟 1：更新叢集以使用 Amazon ECS 受管執行個體

建立容量提供者。使用 Amazon ECS 受管執行個體建立的容量提供者只能在指定的叢集內使用。

如需詳細資訊，請參閱[為 Amazon ECS 受管執行個體建立容量提供者](#)。

步驟 2：更新任務定義以取得 Amazon ECS 受管執行個體功能

更新任務定義，使其具有 Amazon ECS 受管執行個體所需的功能。

如需詳細資訊，請參閱[使用主控台更新 Amazon ECS 任務定義](#)。

步驟 3：更新服務以使用 Amazon ECS 受管執行個體容量提供者

更新您現有的 Amazon ECS 服務，以使用 Amazon ECS 受管執行個體容量提供者。

如需詳細資訊，請參閱[更新 Amazon ECS 服務以使用容量提供者](#)。

步驟 4：遷移獨立任務

對於獨立任務，請在執行任務時指定 Amazon ECS 受管執行個體容量提供者。

如需詳細資訊，請參閱[將應用程式作為 Amazon ECS 任務執行](#)。

從 Amazon EC2 遷移至 Amazon ECS 受管執行個體

將現有工作負載從 Amazon EC2 遷移至 Amazon ECS 受管執行個體。此遷移可讓您存取完整的 Amazon EC2 執行個體類型、容量保留和進階功能，同時維護 AWS 受管基礎設施。

遷移考量事項

從 Amazon EC2 遷移至 Amazon ECS 受管執行個體時，請記住下列考量事項：

任務相容性

為 Amazon EC2 設定的現有任務定義主要與 Amazon ECS 受管執行個體相容。如需任務定義差異的清單，請參閱 [Amazon ECS 受管執行個體的 Amazon ECS 任務定義差異](#)。

安全模型變更

Amazon ECS 受管執行個體預設允許每個執行個體執行多項任務。如果您的工作負載需要更強大的隔離，請考量啟用單一任務模式。

執行個體生命週期

Amazon ECS 受管執行個體的生命週期上限為 14 天。請制定任務替代計畫，並使用 Amazon ECS 服務進行自動任務管理。

定價變更

使用 Amazon ECS 受管執行個體時，您需要支付整個執行個體的費用加上管理費用，以及 AWS 處理基礎設施管理開銷。

維護時段

使用 Amazon EC2 事件時段設定維護時段，以控制何時取代 Amazon ECS 受管執行個體進行修補。

先決條件

遷移至 Amazon ECS 受管執行個體之前，請確定您：

- 您有 Amazon ECS 受管執行個體所需的 IAM 角色。其中包含：
 - 基礎設施角色 - 允許 Amazon ECS 代表您呼叫 AWS 服務，以管理 Amazon ECS 受管執行個體基礎設施。

如需詳細資訊，請參閱[Amazon ECS 基礎結構 IAM 角色](#)。

- 執行個體設定檔 – 為在受管執行個體上執行的 Amazon ECS 容器代理程式與 Docker 常駐程式提供許可。

如需詳細資訊，請參閱[Amazon ECS 受管執行個體執行個體設定檔](#)。

- 了解 Amazon EC2 與 Amazon ECS 受管執行個體之間的安全模型差異

步驟 1：更新叢集以使用 Amazon ECS 受管執行個體

建立容量提供者。使用 Amazon ECS 受管執行個體建立的容量提供者只能在指定的叢集內使用。

如需詳細資訊，請參閱[為 Amazon ECS 受管執行個體建立容量提供者](#)。

步驟 2：更新任務定義以取得 Amazon ECS 受管執行個體功能

更新任務定義，使其具有 Amazon ECS 受管執行個體所需的功能。

如需詳細資訊，請參閱[使用主控台更新 Amazon ECS 任務定義](#)。

步驟 3：更新服務以使用 Amazon ECS 受管執行個體容量提供者

更新您現有的 Amazon ECS 服務，以使用 Amazon ECS 受管執行個體容量提供者。

如需詳細資訊，請參閱[更新 Amazon ECS 服務以使用容量提供者](#)。

步驟 4：遷移獨立任務

對於獨立任務，請在執行任務時指定 Amazon ECS 受管執行個體容量提供者。

如需詳細資訊，請參閱[將應用程式作為 Amazon ECS 任務執行](#)。

Amazon ECS 受管協助程式

Amazon ECS Managed Daemons 可讓您在 Amazon ECS 受管執行個體上的容器基礎設施中部署和管理軟體代理程式，例如安全性、可觀測性和聯網代理程式。Managed Daemons 會將協助程式生命週期管理與應用程式操作分離。您可以獨立部署、更新和監控代理程式，而無需重新部署工作負載或協調跨服務的變更。

Managed Daemons 的運作方式

若要使用受管協助程式，請先註冊協助程式任務定義。協助程式任務定義是描述形成協助程式之容器的範本。註冊協助程式任務定義後，請建立協助程式，並將其與叢集和一或多個 Amazon ECS 受管執行個體容量提供者建立關聯。然後，Amazon ECS 可確保在透過這些容量提供者佈建的每個 Amazon EC2 執行個體上執行一個協助程式任務。

協助程式不會獨立啟動執行個體。當您在 Amazon ECS 受管執行個體容量提供者上執行應用程式任務時，Amazon ECS 會佈建 Amazon EC2 執行個體、先啟動協助程式任務，然後才將應用程式任務轉換為 RUNNING。此排序可確保記錄、追蹤和指標集合等跨切割函數在應用程式開始處理請求之前可正常運作。

協助程式對於執行個體運作狀態至關重要。如果協助程式任務停止，Amazon ECS 會自動耗盡並取代該容器執行個體。這種自動修復行為可確保在所有執行個體中可靠的協助程式涵蓋範圍，無需手動介入。

當您將協助程式更新為新的任務定義修訂版時，Amazon ECS 會在關聯容量提供者中的所有執行個體之間執行滾動部署。在部署期間，Amazon ECS 會同時耗盡可設定的執行個體百分比、使用更新的常駐程式佈建取代執行個體，以及自動取代 Amazon ECS 服務任務。Amazon ECS 提供內建斷路器保護。您可以設定製作時間和 CloudWatch 警示，讓 Amazon ECS 在更新所有執行個體後監控部署，並在發生問題時自動轉返。

主要優點

- 解耦生命週期管理 - 獨立於應用程式部署更新協助程式。
- 保證涵蓋範圍 - Amazon ECS 可確保協助程式任務在每個執行個體上的應用程式任務之前啟動，因此交叉修剪函數一律可用。
- 可靠的版本更新 - 當您更新協助程式版本時，Amazon ECS 會將其推展到關聯容量提供者中的所有執行個體，並內建斷路器保護和自動轉返，以確保每個執行個體執行目標修訂。

- 改善資源使用率 - 每個執行個體執行單一協助程式任務可消除sidecar-per-task模型，減少叢集的資源負荷。
- 自動執行個體修復 - 如果協助程式任務停止或運作狀態不佳，Amazon ECS 會自動耗盡並取代該容器執行個體。這可維持可靠的協助程式涵蓋範圍，無需手動介入。

Note

Amazon ECS 為 Amazon EC2 啟動類型的 Amazon EC2 服務提供 DAEMON 排程策略。Managed Daemons 是為 Amazon ECS 受管執行個體建置的新功能，可簡化協助程式部署並提供更強大的協助程式涵蓋範圍保證。

開始使用

若要開始使用，請註冊指定容器映像的協助程式任務定義，然後在叢集中建立具有相關聯容量提供者的協助程式。您可以使用 AWS 管理主控台、AWS CLI CloudFormation、或 AWS SDKs。如需逐步說明，請參閱 [建立和管理協助程式](#)。

常駐程式任務定義

協助程式任務定義是協助程式的藍圖。Amazon ECS 受管協助程式使用專用協助程式任務定義資源，與標準 Amazon ECS 任務定義不同。您可以註冊協助程式任務定義，以指定協助程式的容器映像、資源需求和組態。

建立協助程式任務定義

您可以使用 AWS 管理主控台 或 建立協助程式任務定義 AWS CLI。

使用 AWS 管理主控台

1. 開啟 Amazon ECS 主控台。在左側導覽中，選擇協助程式任務定義。
2. 選擇建立新的協助程式任務定義。
3. 針對協助程式任務定義系列，輸入唯一的名稱。名稱最多可包含 255 個英數字元、連字號和底線。
4. (選用) 針對任務角色，選擇將許可授予容器中執行之應用程式的 IAM 角色。如果您的容器未呼叫 AWS APIs請將此保留空白。
5. (選用) 針對任務執行角色，選取 `ecsTaskExecutionRole`。Amazon ECS 需要此角色才能提取容器映像並發佈日誌。

6. (選用) 針對任務大小，指定要預留給協助程式的 CPU 和記憶體。例如，0.25 vCPU和 512 MB 記憶體。
7. 設定您的容器詳細資訊：
 - 容器名稱 - 輸入容器的名稱。名稱最多可包含 255 個英數字元、連字號和底線。
 - 必要容器 - 選取是。每個協助程式任務定義至少需要一個基本容器。如果基本容器失敗，Amazon ECS 會停止整個任務。
 - 映像 URI - 輸入 Docker 映像 URI。您可以瀏覽 Amazon ECR 映像，或使用來自 Docker Hub 或其他登錄檔的公有映像。
8. (選用) 視需要設定資源配置、運作狀態檢查、環境變數、日誌收集和標籤。
9. 檢閱您的組態，然後選擇建立。

使用 AWS CLI

建立 JSON 檔案並使用 `register-daemon-task-definition` 命令註冊協助程式任務定義。

以下是範例 JSON 檔案：

```
{
  "family": "my-daemon-task",
  "containerDefinitions": [
    {
      "name": "daemon-container",
      "image": "public.ecr.aws/docker/library/busybox:latest",
      "essential": true,
      "command": ["sh", "-c", "while true; do echo 'Daemon running'; sleep 30;
done"],
      "memoryReservation": 512
    }
  ]
}
```

執行下列命令來註冊協助程式任務定義：

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

協助程式任務定義參數

您可以在註冊協助程式任務定義時使用下列參數。

必要參數

- `family` - 任務定義系列名稱。這會將相同協助程式任務定義的多個修訂分組。
- `containerDefinitions` - 描述協助程式任務中容器的 `DaemonContainerDefinition` 物件陣列。

選用參數

- `cpu` - 做為字串的任務層級 CPU 單位 (例如 "256")。
- `memory` - MiB 中的任務層級記憶體做為字串 (例如, "512")。
- `taskRoleArn` - IAM 角色的 ARN, 將許可授予協助程式任務中的容器。
- `executionRoleArn` - Amazon ECS 容器代理程式用來代表您進行 AWS API 呼叫的 IAM 角色 ARN (例如, 從 Amazon ECR 提取映像)。
- `volumes` - `DaemonVolume` 物件陣列。協助程式磁碟區僅支援使用 `host` 和 繫結掛載 `sourcePath`。協助程式任務定義不支援 Amazon EBS、Amazon EFS、FSx for Windows File Server、Docker 磁碟區或暫時性儲存。
- `tags` - 標記協助程式任務定義的鍵值對陣列。

網路模式

協助程式使用 Amazon ECS 自動設定的特殊 `daemon_bridge` 網路模式。您無法在協助程式任務定義中指定網路模式。執行個體上的所有協助程式共用單一網路命名空間, 且可透過靜態協助程式橋接 IP 地址 (169.254.172.2 適用於 IPv4 或 `fd00:ec2::172:2` 適用於 IPv6) 在本機存取。應用程式任務 (非協助程式) 可以與此地址的協助程式通訊, 無需額外的網路組態。

由於協助程式會在與應用程式任務不同的網路命名空間中執行, 因此非協助程式任務不需要擔心連接埠與協助程式的衝突。不過, 由於所有協助程式任務共用相同的命名空間, 因此同一執行個體上的協助程式無法繫結至相同的連接埠。部署多個協助程式時, 請確保每個協助程式使用唯一的連接埠。

特殊權限功能

在 Amazon ECS 受管執行個體上, 協助程式支援系統層級操作的特權 Linux 功能。安全代理程式、網路監控工具和可觀測性代理程式通常需要核心層級存取權才能正常運作。

您可以透過兩種方式授予 功能：

完整特殊權限模式會將所有 Linux 功能授予容器。當您的代理程式需要廣泛的系統存取時，請使用此選項：

```
{
  "containerDefinitions": [{
    "name": "security-daemon",
    "image": "my-security-agent:latest",
    "privileged": true
  }]
}
```

個別功能僅授予協助程式所需的許可，並遵循最低權限原則。使用 `linuxParameters.capabilities` 欄位來新增個別功能，例如 `SYS_ADMIN`、`SYS_PTRACE`、`NET_ADMIN`和 `BPF`：

```
{
  "containerDefinitions": [{
    "name": "monitoring-daemon",
    "image": "my-monitoring-agent:latest",
    "linuxParameters": {
      "capabilities": {
        "add": ["NET_ADMIN", "SYS_PTRACE"]
      }
    }
  }]
}
```

磁碟區

Managed Daemons 支援使用具有 `sourcePath` 規格的主機磁碟區來繫結掛載。這可讓協助程式容器掛載主機目錄，以存取基礎 Amazon EC2 執行個體上的日誌、指標和系統資訊。日誌收集代理程式、指標匯出程式和安全掃描器通常會使用此功能來取得主機層級資料的可見性。

協助程式任務定義支援使用的主機磁碟區規格 `sourcePath`。協助程式磁碟區會在基礎容器執行個體的生命週期內持續存在。

Note

協助程式任務定義不支援 Amazon EBS 磁碟區、Amazon EFS 磁碟區、FSx for Windows File Server 磁碟區、Docker 磁碟區和暫時性儲存。

 Note

協助程式不支援彼此之間的磁碟區共用。每個協助程式都會使用自己的獨立磁碟區組態運作。

下列範例顯示具有綁定掛載的日誌收集器協助程式：

```
{
  "containerDefinitions": [{
    "name": "log-collector",
    "image": "fluent/fluentd:latest",
    "mountPoints": [{
      "sourceVolume": "var-log",
      "containerPath": "/var/log"
    }]
  }],
  "volumes": [{
    "name": "var-log",
    "host": {
      "sourcePath": "/var/log"
    }
  }]
}
```

支援的容器參數

協助程式容器定義支援下列參數：

- `image` (必要) - 容器映像 URI
- `name` (必要) - 容器名稱
- `cpu` - 為容器預留的 CPU 單位
- `memory` / `memoryReservation` - 硬記憶體和軟記憶體限制
- `essential` - 容器是否為必要 (預設值：true)
- `command` / `entryPoint` - 覆寫容器命令或進入點
- `environment` / `environmentFiles` / `secrets` - 環境組態
- `privileged` - 在特權模式下執行容器
- `user` - 以身分執行容器的使用者

- `workingDirectory` - 容器內的工作目錄
- `readonlyRootFilesystem` - 將根檔案系統掛載為唯讀
- `mountPoints` - 磁碟區掛載點 (僅限具有主機和 `sourcePath` 的繫結掛載)
- `logConfiguration` - 記錄組態 (支援 `awslogs`、`splunk`、`awsfirelens`)
- `healthCheck` - 容器運作狀態檢查組態
- `dependsOn` - 容器啟動相依性
- `ulimits` - 資源限制
- `systemControls` - 核心參數
- `linuxParameters.capabilities` - 要新增或捨棄的 Linux 功能
- `linuxParameters.initProcessEnabled` - 在容器中執行初始化程序
- `repositoryCredentials` - 私有登錄檔的登入資料
- `restartPolicy` - 容器重新啟動政策

建立和管理協助程式

註冊協助程式任務定義後，請建立協助程式，將它部署到 Amazon ECS 受管執行個體容量提供者。Amazon ECS 會自動在關聯容量提供者的每個 Amazon EC2 執行個體上放置一個協助程式任務，並管理協助程式生命週期。

先決條件

建立協助程式之前，請確認您具有下列項目：

- Amazon ECS 叢集。
- 與叢集相關聯的一或多個 Amazon ECS 受管執行個體容量提供者。
- 已註冊的協助程式任務定義。

建立協助程式

AWS 管理主控台

1. 開啟 Amazon ECS 主控台。在左側導覽中，選擇叢集，然後選取您的叢集。
2. 選擇精靈索引標籤，然後選擇建立。

3. 對於精靈任務定義系列，請從下拉式清單中選取您的精靈任務定義。
4. 對於精靈任務定義修訂，選取要使用的修訂。保留空白以使用最新的修訂版。
5. 針對協助程式名稱，輸入唯一的名稱。名稱最多可包含 255 個英數字元、連字號和底線。
6. 針對容量提供者，選取一或多個 Amazon ECS 受管執行個體容量提供者。這些會決定哪些執行個體執行您的協助程式任務。
7. (選用) 設定部署設定：
 - 耗盡百分比 - 在更新期間同時耗盡的執行個體百分比。預設：25。
 - 使用 CloudWatch alarm(s) - 開啟 以監控部署運作狀態，並在警示觸發時自動轉返。
 - 製作時間 - Amazon ECS 在將所有執行個體更新為新的常駐程式修訂版後，在完成部署之前等待的分鐘數。在此期間，Amazon ECS 會監控 CloudWatch 警示，並在觸發任何警示時自動復原部署。預設：0。
8. (選用) 新增標籤。
9. (選用) 開啟啟用 ECS Exec 以在協助程式容器中執行互動式命令以進行故障診斷。
10. 檢閱您的組態，然後選擇建立。

AWS CLI

使用您的協助程式組態建立 JSON 檔案，並執行 `create-daemon` 命令。

以下是範例 JSON 檔案：

```
{
  "clusterArn": "arn:aws:ecs:us-east-1:123456789012:cluster/my-daemon-cluster",
  "daemonName": "my-monitoring-daemon",
  "daemonTaskDefinitionArn": "arn:aws:ecs:us-east-1:123456789012:daemon-task-definition/my-daemon-task:1",
  "capacityProviderArns": [
    "arn:aws:ecs:us-east-1:123456789012:capacity-provider/my-daemon-capacity-provider"
  ]
}
```

執行下列命令來建立協助程式：

```
aws ecs create-daemon --cli-input-json file://create-daemon.json
```

必要欄位

- `daemonName` - 協助程式的唯一名稱。
- `clusterArn` - 叢集的 ARN。
- `daemonTaskDefinitionArn` - 協助程式任務定義的 ARN。
- `capacityProviderArns` - Amazon ECS 受管執行個體容量提供者 ARNs 的陣列。

選用欄位

- `deploymentConfiguration` - 自訂部署行為的 `DaemonDeploymentConfiguration` 物件。
- `tags` - 用於標記的鍵/值對。
- `propagateTags` - 標籤傳播設定。
- `clientToken` - 等冪符記。

驗證協助程式部署

建立協助程式之後，請使用 AWS 管理主控台 或 來驗證其狀態 AWS CLI。

AWS 管理主控台

1. 開啟 Amazon ECS 主控台。在左側導覽中，選擇叢集，然後選取您的叢集。
2. 選擇 Daemons 索引標籤。
3. 確認您的協助程式顯示作用中狀態。
4. 選擇任務索引標籤，以確認每個容器執行個體上執行一個協助程式任務。

AWS CLI

執行下列命令來驗證協助程式狀態：

```
aws ecs list-daemons \
  --cluster-arn arn:aws:ecs:us-east-1:123456789012:cluster/my-daemon-cluster
```

```
aws ecs describe-daemons \
  --daemon-arn arn:aws:ecs:us-east-1:123456789012:daemon/my-daemon-cluster/my-
  monitoring-daemon
```

更新協助程式

若要使用新的任務定義修訂版更新協助程式，或新增容量提供者，請使用 AWS 管理主控台 或 AWS CLI。這會觸發所有執行個體的滾動部署。

AWS 管理主控台

1. 開啟 Amazon ECS 主控台。在左側導覽中，選擇叢集，然後選取您的叢集。
2. 選擇協助程式索引標籤，然後選取您要更新的協助程式。
3. 選擇更新。
4. 對於精靈任務定義修訂，選取新的修訂。
5. (選用) 更新容量提供者或部署設定。
6. 選擇更新以開始滾動部署。

AWS CLI

執行 `update-daemon` 命令：

```
aws ecs update-daemon \
  --daemon-arn arn:aws:ecs:us-east-1:123456789012:daemon/my-daemon-cluster/my-
monitoring-daemon \
  --daemon-task-definition-arn arn:aws:ecs:us-east-1:123456789012:daemon-task-
definition/my-daemon-task:2 \
  --capacity-provider-arns arn:aws:ecs:us-east-1:123456789012:capacity-provider/my-
daemon-capacity-provider
```

Important

當您在 UpdateDaemon 請求中提供協助程式組態設定時，Amazon ECS 會使用您指定的設定，而不是預設值。更新之間不會保留協助程式組態設定，包括標籤和啟用執行命令旗標。對的每次呼叫 UpdateDaemon 都必須包含您要套用的完整組態設定集。從請求中省略的任何設定都會還原為其預設值。

刪除協助程式

若要刪除協助程式，請使用 AWS 管理主控台 或 AWS CLI。刪除容量提供者或叢集之前，請等待所有協助程式任務停止。

AWS 管理主控台

1. 開啟 Amazon ECS 主控台。在左側導覽中，選擇叢集，然後選取您的叢集。
2. 選擇協助程式索引標籤，然後選取您要刪除的協助程式。
3. 選擇 刪除。
4. 在確認對話方塊中，選擇刪除以確認。

AWS CLI

執行 delete-daemon 命令：

```
aws ecs delete-daemon \  
    --daemon-arn arn:aws:ecs:us-east-1:123456789012:daemon/my-daemon-cluster/my-  
monitoring-daemon
```

協助程式部署

當您建立或更新協助程式時，Amazon ECS 會執行滾動部署，以在相關聯容量提供者中的所有執行個體上執行協助程式任務。本節說明協助程式部署的運作方式，以及如何設定部署行為。

協助程式部署的運作方式

協助程式部署遵循 drain-provision-replace 模式。當您在具有現有執行個體的容量提供者上建立協助程式，或將協助程式更新為新的任務定義修訂版時，Amazon ECS 會耗盡一批執行個體、使用更新的協助程式佈建替換執行個體，並自動取代應用程式任務。此程序會重複執行，直到所有執行個體執行新的協助程式修訂版為止。

部署生命週期會轉換到下列狀態：

1. PENDING - Amazon ECS 已建立部署並準備開始。
2. IN_PROGRESS - Amazon ECS 正積極耗盡執行個體和佈建替換。
3. SUCCESSFUL - 所有執行個體都在執行目標協助程式修訂。
4. STOPPED - Amazon ECS 已停止部署，因為它失敗或被新部署取代。協助程式可能處於混合狀態，其中某些執行個體執行舊修訂版，而其他執行個體則執行新修訂版。

如果部署斷路器偵測到故障或在部署期間觸發 CloudWatch 警示，Amazon ECS 會自動轉返至先前的協助程式修訂版。

部署組態參數

您可以在建立或更新協助程式時使用 `deploymentConfiguration` 參數自訂部署行為。

- `drainPercent` (1.0–100.0) - 部署期間要同時耗盡的執行個體百分比。較高的值可加速部署，但可能會暫時減少可用容量。例如，的值一次20.0耗盡 20% 的執行個體。若沒有指定，則預設為 25.0。
- `alarms` (`DaemonAlarmConfiguration`) - 在部署期間要監控的 CloudWatch 警示。Amazon ECS 會在部署期間評估指定的警示，並在任何警示進入 ALARM 狀態時自動轉返。部署開始時，Amazon ECS 會忽略已處於 ALARM 狀態的警示。
- `bakeTimeInMinutes` (0–1440) - Amazon ECS 在將所有執行個體更新為新的常駐程式修訂版後，在完成部署之前等待的分鐘數。在此期間，Amazon ECS 會監控 CloudWatch 警示，並在觸發任何警示時自動復原部署。若沒有指定，則預設為 0。

部署組態的範例：

```
aws ecs create-daemon \
  --cluster-arn arn:aws:ecs:us-east-1:123456789012:cluster/my-daemon-cluster \
  --daemon-name my-monitoring-daemon \
  --daemon-task-definition-arn arn:aws:ecs:us-east-1:123456789012:daemon-task-definition/my-daemon-task:1 \
  --capacity-provider-arns arn:aws:ecs:us-east-1:123456789012:capacity-provider/my-daemon-capacity-provider \
  --deployment-configuration '{"drainPercent":20.0,"bakeTimeInMinutes":5}'
```

監控部署

使用下列命令追蹤部署進度：

```
aws ecs list-daemon-deployments \
  --daemon-arn arn:aws:ecs:us-east-1:123456789012:daemon/my-daemon-cluster/my-monitoring-daemon
```

```
aws ecs describe-daemon-deployments \
  --daemon-deployment-arns arn:aws:ecs:us-east-1:123456789012:daemon-deployment/my-daemon-cluster/abc123
```

describe-daemon-deployments 回應包含部署狀態、目標協助程式修訂版，以及指出有多少執行個體執行目標協助程式的 totalRunningInstanceCount 欄位。

協助程式部署案例

本節會逐步解說 Managed Daemons 的常見部署案例。

- [在空容量提供者上部署協助程式](#)
- [使用現有執行個體在容量提供者上部署協助程式](#)
- [將容量提供者新增至現有的協助程式](#)
- [將協助程式更新為新的修訂版](#)

在空容量提供者上部署協助程式

在此案例中，您會在沒有現有執行個體的容量提供者上部署協助程式。執行個體會在您排程應用程式任務時啟動。

先決條件：叢集和沒有執行中執行個體的 Amazon ECS 受管執行個體容量提供者。

1. 註冊您的協助程式任務定義。
2. 建立協助程式。即使還沒有執行個體，Amazon ECS 也會立即完成建立。

```
aws ecs create-daemon \
  --cluster-arn arn:aws:ecs:us-east-1:123456789012:cluster/my-daemon-cluster \
  --daemon-name my-monitoring-daemon \
  --daemon-task-definition-arn arn:aws:ecs:us-east-1:123456789012:daemon-task-definition/my-daemon-task:1 \
  --capacity-provider-arns arn:aws:ecs:us-east-1:123456789012:capacity-provider/my-daemon-capacity-provider
```

3. 建立使用相同容量提供者的服務。當 Amazon ECS 為服務佈建執行個體時，它會先啟動協助程式任務，然後將應用程式任務轉換為 RUNNING。

```
aws ecs create-service \
  --cluster my-daemon-cluster \
  --service-name my-app-service \
  --task-definition my-app-task \
  --desired-count 2 \
  --capacity-provider-strategy capacityProvider=my-daemon-capacity-provider,weight=1
```


4. 使用 `describe-daemons` 命令或 主控台 中的 `Daemons` 索引標籤，確認協助程式任務是否在每個執行個體上執行。

Note

當 Amazon ECS 佈建執行個體並啟動協助程式任務時，協助程式部署可能會短暫保持 `PENDING` 狀態。等待部署到達，`SUCCESSFUL` 再驗證協助程式任務置放。

使用現有執行個體在容量提供者上部署協助程式

在此案例中，您會在已執行執行個體和應用程式任務的容量提供者上部署協助程式。

先決條件：具有執行中執行個體之 Amazon ECS 受管執行個體容量提供者的叢集。

1. 註冊您的協助程式任務定義。
2. 建立協助程式。Amazon ECS 會啟動滾動部署，在所有現有的執行個體上放置協助程式任務。

```
aws ecs create-daemon \
  --cluster-arn arn:aws:ecs:us-east-1:123456789012:cluster/my-daemon-cluster \
  --daemon-name my-monitoring-daemon \
  --daemon-task-definition-arn arn:aws:ecs:us-east-1:123456789012:daemon-task-definition/my-daemon-task:1 \
  --capacity-provider-arns arn:aws:ecs:us-east-1:123456789012:capacity-provider/my-daemon-capacity-provider \
  --deployment-configuration '{"drainPercent":25.0,"bakeTimeInMinutes":3}'
```

Amazon ECS 會耗盡一批現有的執行個體（根據 `drainPercent`）、使用協助程式佈建取代執行個體，以及取代應用程式任務。此程序會重複執行，直到所有執行個體執行協助程式為止。

3. 監控部署進度：

```
aws ecs list-daemon-deployments \
  --daemon-arn arn:aws:ecs:us-east-1:123456789012:daemon/my-daemon-cluster/my-monitoring-daemon
```

將容量提供者新增至現有的協助程式

在此案例中，您會將第二個容量提供者新增至現有的協助程式。協助程式會自動部署到新容量提供者中的執行個體。

先決條件：在第一個容量提供者上執行的協助程式，以及建立與叢集相關聯的第二個容量提供者。

1. 更新協助程式以包含兩個容量提供者：

```
aws ecs update-daemon \
  --daemon-arn arn:aws:ecs:us-east-1:123456789012:daemon/my-daemon-cluster/my-
monitoring-daemon \
  --daemon-task-definition-arn arn:aws:ecs:us-east-1:123456789012:daemon-task-
definition/my-daemon-task:1 \
  --capacity-provider-arns \
    arn:aws:ecs:us-east-1:123456789012:capacity-provider/my-daemon-capacity-
provider \
    arn:aws:ecs:us-east-1:123456789012:capacity-provider/my-daemon-capacity-
provider-2
```

2. 監控部署，並確認協助程式任務在兩個容量提供者的執行個體上執行。

將協助程式更新為新的修訂版

在此案例中，您會更新現有的協助程式，以使用新的任務定義修訂。

1. 使用更新的容器映像或組態註冊協助程式任務定義的新修訂版。
2. 更新協助程式以使用新的修訂：

```
aws ecs update-daemon \
  --daemon-arn arn:aws:ecs:us-east-1:123456789012:daemon/my-daemon-cluster/my-
monitoring-daemon \
  --daemon-task-definition-arn arn:aws:ecs:us-east-1:123456789012:daemon-task-
definition/my-daemon-task:2
```

Amazon ECS 會執行滾動部署。它會耗盡執行舊修訂版的執行個體、使用新修訂版佈建取代執行個體，以及自動取代應用程式任務。如果斷路器偵測到故障，Amazon ECS 會回復到先前的修訂版。

3. 監控部署並確認所有執行個體都執行新的修訂版：

```
aws ecs describe-daemon-deployments \
```

```
--daemon-deployment-arns arn:aws:ecs:us-east-1:123456789012:daemon-deployment/my-  
daemon-cluster/deployment-id
```

協助程式事件

Amazon ECS 會將協助程式事件傳送至 `Amazon CloudWatch Events`，以提供協助程式任務管理和部署生命週期變更的可見性。您可以使用這些事件來監控協助程式運作狀態、追蹤部署進度，以及觸發自動化工作流程。

Amazon ECS 支援下列協助程式事件類型：

- [協助程式服務動作事件](#) - 與協助程式任務置放和啟動問題相關的事件。
- [協助程式部署狀態變更事件](#) - 與協助程式部署生命週期轉換相關的事件。

若要了解如何為這些事件建立規則，請參閱 [建立協助程式事件的規則](#)。

協助程式服務動作事件

Amazon ECS 會傳送詳細資訊類型為 `ECS Daemon Service Action` 的協助程式服務動作事件。當 Amazon ECS 在容器執行個體上啟動協助程式任務時，這些事件會通知您。

協助程式_TASK_START_IMPAIRED

當 Amazon ECS 無法在容器執行個體上成功啟動協助程式任務時，會傳送 `DAEMON_TASK_START_IMPAIRED` 事件。事件包含指出失敗原因 `failureType` 的欄位：

- `TASK_FAILED_TO_RUN` - 協助程式任務已建立，但無法達到 `RUNNING` 狀態。常見原因包括容器映像提取失敗、容器運作狀態檢查失敗或基本容器結束。 `taskArn` 欄位出現在事件中。
- `TASK_CREATION_FAILED` - 無法在容器執行個體上建立協助程式任務。常見原因包括執行個體上的 CPU、記憶體或其他資源不足。 `taskArn` 欄位不存在於事件中，因為未建立任務。

範例：TASK_FAILED_TO_RUN 事件

下列事件顯示 協助程式任務已建立，但由於容器映像提取失敗而無法達到 `RUNNING` 狀態。

```
{  
  "version": "0",  
  "id": "12345678-1234-1234-1234-123456789012",  
  "detail-type": "ECS Daemon Service Action",
```

```

"source": "aws.ecs",
"account": "123456789012",
"time": "2026-03-24T12:00:00Z",
"region": "us-west-2",
"resources": [
  "arn:aws:ecs:us-west-2:123456789012:task/my-cluster/a1b2c3d4e5f6"
],
"detail": {
  "eventType": "WARNING",
  "eventName": "DAEMON_TASK_START_IMPAIRED",
  "clusterArn": "arn:aws:ecs:us-west-2:123456789012:cluster/my-cluster",
  "containerInstanceArn": "arn:aws:ecs:us-west-2:123456789012:container-
instance/my-cluster/a1b2c3d4e5f6",
  "taskArn": "arn:aws:ecs:us-west-2:123456789012:task/my-cluster/a1b2c3d4e5f6",
  "daemonRevisionArn": "arn:aws:ecs:us-west-2:123456789012:daemon-revision/my-
cluster/my-daemon/a1b2c3d4-e5f6-7890-abcd-ef1234567890",
  "capacityProviderName": "my-capacity-provider",
  "daemonArn": "arn:aws:ecs:us-west-2:123456789012:daemon/my-cluster/my-daemon",
  "daemonTaskDefinitionArn": "arn:aws:ecs:us-west-2:123456789012:daemon-task-
definition/my-daemon-td:1",
  "failureType": "TASK_FAILED_TO_RUN",
  "createdAt": "2026-03-24T12:00:00.000Z",
  "reason": "Task failed to reach RUNNING status: CannotPullContainerError: pull
image manifest has been retried 5 time(s)"
}
}

```

範例：TASK_CREATION_FAILED 事件

下列事件顯示由於 CPU 資源不足而無法在容器執行個體上建立的協助程式任務。taskArn 欄位不存在，因為未建立任務。

```

{
  "version": "0",
  "id": "87654321-4321-4321-4321-210987654321",
  "detail-type": "ECS Daemon Service Action",
  "source": "aws.ecs",
  "account": "123456789012",
  "time": "2026-03-24T12:01:00Z",
  "region": "us-west-2",
  "resources": [],
  "detail": {
    "eventType": "WARNING",

```

```

    "eventName": "DAEMON_TASK_START_IMPAIRED",
    "clusterArn": "arn:aws:ecs:us-west-2:123456789012:cluster/my-cluster",
    "containerInstanceArn": "arn:aws:ecs:us-west-2:123456789012:container-instance/my-cluster/b2c3d4e5f6a7",
    "daemonRevisionArn": "arn:aws:ecs:us-west-2:123456789012:daemon-revision/my-cluster/my-daemon/a1b2c3d4-e5f6-7890-abcd-ef1234567890",
    "capacityProviderName": "my-capacity-provider",
    "daemonArn": "arn:aws:ecs:us-west-2:123456789012:daemon/my-cluster/my-daemon",
    "daemonTaskDefinitionArn": "arn:aws:ecs:us-west-2:123456789012:daemon-task-definition/my-daemon-td:1",
    "failureType": "TASK_CREATION_FAILED",
    "createdAt": "2026-03-24T12:01:00.000Z",
    "reason": "RESOURCE:CPU - Unable to place daemon task on container instance: insufficient CPU"
  }
}

```

協助程式部署狀態變更事件

Amazon ECS 會傳送詳細資訊類型為 的協助程式部署狀態變更事件 ECS Daemon Deployment State Change。每次協助程式部署的狀態變更時，Amazon ECS 都會發出這些事件。

部署狀態變更事件類型

Amazon ECS 依下列事件類型分類部署狀態變更事件：

INFO 事件

- DAEMON_DEPLOYMENT_PENDING - Amazon ECS 已啟動協助程式部署。
- DAEMON_DEPLOYMENT_IN_PROGRESS - Amazon ECS 已開始部署，並積極取代協助程式任務。
- DAEMON_DEPLOYMENT_SUCCESSFUL - Amazon ECS 已成功完成部署，所有協助程式任務皆執行正常。
- DAEMON_DEPLOYMENT_STOPPED - Amazon ECS 已停止部署。當部署失敗或已由新的部署取代時，就會發生這種情況。
- DAEMON_DEPLOYMENT_STOP_REQUESTED - Amazon ECS 已停止部署繼續進行，現在將開始復原。
- DAEMON_DEPLOYMENT_ROLLBACK_IN_PROGRESS - 由於部署失敗或斷路器觸發，Amazon ECS 已啟動轉返。
- DAEMON_DEPLOYMENT_ROLLBACK_SUCCESSFUL - Amazon ECS 已成功完成部署轉返。

錯誤事件

- `DAEMON_DEPLOYMENT_ROLLBACK_FAILED` - Amazon ECS 無法完成部署轉返。

範例：部署擱置事件

下列事件顯示 Amazon ECS 已啟動的協助程式部署。

```
{
  "version": "0",
  "id": "3329f79b-3dca-07f8-b1c2-5fe99f0b5e87",
  "detail-type": "ECS Daemon Deployment State Change",
  "source": "aws.ecs",
  "account": "111122223333",
  "time": "2026-03-05T15:54:41Z",
  "region": "us-west-2",
  "resources": [
    "arn:aws:ecs:us-west-2:111122223333:daemon/my-cluster/my-daemon"
  ],
  "detail": {
    "eventType": "INFO",
    "eventName": "DAEMON_DEPLOYMENT_PENDING",
    "clusterArn": "arn:aws:ecs:us-west-2:111122223333:cluster/my-cluster",
    "daemonArn": "arn:aws:ecs:us-west-2:111122223333:daemon/my-cluster/my-daemon",
    "daemonDeploymentArn": "arn:aws:ecs:us-west-2:111122223333:daemon-deployment/my-cluster/my-daemon/0EYSiB0qap8xf0N76FsbE",
    "targetDaemonRevisionArn": "arn:aws:ecs:us-west-2:111122223333:daemon-revision/my-cluster/my-daemon/85707969-3732-4b6a-a37d-5cefddd7d7dd",
    "updatedAt": "2026-03-05T15:54:41.618059641Z"
  }
}
```

建立協助程式事件的規則

您可以建立規則，以在協助程式事件發生時接收通知。如需建立規則的詳細資訊，請參閱《Amazon EventBridge 使用者指南》中的[建立規則](#)。

下列範例事件模式示範如何篩選協助程式事件。

範例：比對所有協助程式服務動作事件

```
{
```

```
"source": ["aws.ecs"],
"detail-type": ["ECS Daemon Service Action"]
}
```

範例：符合特定失敗類型

```
{
  "source": ["aws.ecs"],
  "detail-type": ["ECS Daemon Service Action"],
  "detail": {
    "eventName": ["DAEMON_TASK_START_IMPAIRED"],
    "failureType": ["TASK_CREATION_FAILED"]
  }
}
```

範例：符合特定叢集的事件

```
{
  "source": ["aws.ecs"],
  "detail-type": ["ECS Daemon Service Action"],
  "detail": {
    "eventName": ["DAEMON_TASK_START_IMPAIRED"],
    "clusterArn": ["arn:aws:ecs:us-west-2:123456789012:cluster/my-cluster"]
  }
}
```

範例：符合所有部署狀態變更事件

```
{
  "source": ["aws.ecs"],
  "detail-type": ["ECS Daemon Deployment State Change"]
}
```

協助程式指標

Amazon ECS 發佈 Managed Daemons 的 CloudWatch 指標。您可以使用這些指標來追蹤叢集中協助程式任務的 CPU 和記憶體使用率。

協助程式指標使用與服務指標相同的AWS/ECS命名空間。Amazon ECS 會以ServiceName維度值區分協助程式指標與服務指標，該值使用字首，daemon:後面接著協助程式名稱（例如，daemon:*my-daemon*）。

協助程式使用率指標

Amazon ECS 會將常駐程式 CPU 和記憶體使用率測量為與常駐程式任務定義中指定的資源相比，常駐程式任務使用的資源百分比。

CloudWatch 提供精靈使用率指標的下列統計資料：

- 平均 - 叢集中所有協助程式任務的平均使用率。
- 最低 - 資源用量最低的協助程式任務使用率。
- 上限 - 資源用量最高的協助程式任務使用率。

使用率公式

Amazon ECS 使用以下公式計算平均統計資料：

$$\text{Daemon CPU utilization} = \frac{(\text{Total CPU units used by daemon tasks}) \times 100}{(\text{Total CPU units specified in daemon task definition}) \times (\text{Number of daemon tasks})}$$

$$\text{Daemon memory utilization} = \frac{(\text{Total MiB of memory used by daemon tasks}) \times 100}{(\text{Total MiB of memory specified in daemon task definition}) \times (\text{Number of daemon tasks})}$$

這些公式僅適用於平均統計資料。最小和最大統計資料分別代表資源使用率最低和最高的個別協助程式任務。

Amazon ECS 容器代理程式會每 20 秒針對每個執行中的協助程式任務取樣 CPU 和記憶體用量。代理程式每分鐘向 Amazon ECS 報告三個範例。Amazon ECS 會彙總叢集中所有協助程式任務的範例，並將平均值、最小值和最大值以協助程式任務定義中指定總資源的百分比發佈至 CloudWatch。

範例：使用率計算

協助程式任務定義會指定 256 個 CPU 單位和 512 MiB 的記憶體。協助程式在 2 個容器執行個體中有 2 個執行中的任務。一個協助程式任務使用 200 個 CPU 單位和 400 MiB 的記憶體。另一個使用 100 個 CPU 單位和 256 MiB 的記憶體。Amazon ECS 會報告下列指標：

- 平均 CPU 使用率： $((200 + 100) \times 100) / (256 \times 2) = 58.6\%$
- 平均記憶體使用率： $((400 + 256) \times 100) / (512 \times 2) = 64.1\%$

檢視協助程式指標

您可以使用 AWS 管理主控台 或 檢視協助程式指標 AWS CLI。

AWS 管理主控台

1. 開啟 CloudWatch 主控台。
2. 在左側導覽中，選擇指標，然後選擇所有指標。
3. 選擇 AWS/ECS 命名空間。
4. 選擇 ClusterName、ServiceName 維度。
5. 篩選以 開頭ServiceName的值daemon:，以尋找您的協助程式指標。

AWS CLI

執行 `get-metric-statistics` 命令，從 CloudWatch 擷取協助程式指標。使用 `daemon:` 字首指定 `ServiceName` 維度。

範例：檢視協助程式 CPU 使用率

下列命令會以 5 分鐘的間隔擷取 12 小時期間內協助程式的平均 CPU 使用率。

```
aws cloudwatch get-metric-statistics \
  --namespace AWS/ECS \
  --metric-name CPUUtilization \
  --dimensions Name=ClusterName,Value=my-cluster Name=ServiceName,Value=daemon:my-daemon \
  --start-time 2026-03-24T00:00:00Z \
  --end-time 2026-03-24T12:00:00Z \
  --period 300 \
```

```
--statistics Average
```

範例：檢視協助程式記憶體使用率

下列命令會以 5 分鐘的間隔擷取 12 小時期間內協助程式的平均記憶體使用率。

```
aws cloudwatch get-metric-statistics \  
  --namespace AWS/ECS \  
  --metric-name MemoryUtilization \  
  --dimensions Name=ClusterName,Value=my-cluster Name=ServiceName,Value=daemon:my-daemon \  
  --start-time 2026-03-24T00:00:00Z \  
  --end-time 2026-03-24T12:00:00Z \  
  --period 300 \  
  --statistics Average
```

協助程式自動修復

Amazon ECS 將所有協助程式視為對執行個體運作狀態至關重要。如果任何協助程式任務停止或運作狀態不佳，Amazon ECS 會將執行個體視為受損，並自動耗盡和取代它。協助程式自動修復動作如下所示：

1. Amazon ECS 會偵測協助程式任務何時停止或運作狀態不佳。
2. Amazon ECS 會將執行個體標記為耗盡，以防止其接受新的應用程式任務。
3. Amazon ECS 會佈建替換執行個體，並在其上啟動協助程式任務。
4. 在協助程式任務達到運作狀態後，Amazon ECS 會將應用程式任務從耗盡執行個體排程到取代。
5. Amazon ECS 會終止原始執行個體。

Important

協助程式運作狀態檢查是選用的，但強烈建議您使用。如果沒有運作狀態檢查，Amazon ECS 只能在協助程式任務停止時偵測失敗。

您可以使用 DescribeContainerInstances API 或 DescribeTasks API 監控協助程式運作狀態。

為 Amazon ECS 規劃 AWS Fargate 架構

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 執行任務與服務時，將會在容器中封裝應用程式、指定 CPU 與記憶體需求、定義聯網與 IAM 政策，以及啟動應用程式。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[Capacity](#)。

Fargate 提供多種平台版本，包括 Amazon Linux 2 (平台版本 1.3.0)、Bottlerocket 作業系統 (平台版本 1.4.0) 以及 Microsoft Windows 2019 Server Full 與 Core 版本。除非另有說明，否則相關資訊適用於所有 Fargate 平台。

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 Windows 容器”](#)。

逐步解說

如需有關如何開始使用主控台的資訊，請參閱：

- [了解如何為 Fargate 建立 Amazon ECS Linux 任務](#)
- [了解如何為 Fargate 建立 Amazon ECS Windows 任務](#)

如需如何開始使用 的資訊 AWS CLI，請參閱：

- [使用 為 Fargate 建立 Amazon ECS Linux 任務 AWS CLI](#)
- [使用 為 Fargate 建立 Amazon ECS Windows 任務 AWS CLI](#)

容量提供者

可用的容量提供者如下：

- Fargate
- Fargate Spot – 能以折扣價 (與 AWS Fargate 價格相比) 執行可容忍中斷的 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，您的任務會受到兩分鐘警告而中斷。如需詳細資訊，請參閱[Fargate 的 Amazon ECS 叢集](#)。

任務定義

Fargate 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 平台版本用於參考 Fargate 任務基礎設施的特定執行期環境。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有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問題，會 AWS 建立新的平台版本修補修訂，並淘汰在易受攻擊修訂上執行的任務。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[Amazon ECS 上 AWS Fargate 的任務淘汰和維護](#)。

如需更多資訊，請參閱[適用於 Amazon ECS 的 Fargate 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、Network Load Balancer 與 Gateway 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[使用負載平衡分佈 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[使用負載平衡分佈 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 CloudWatch 用量指標來提供您帳戶的資源用量可見度。使用這些指標，以 CloudWatch 圖表和儀表板視覺化目前的服務使用狀況。

AWS Fargate 用量指標對應至 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。如需有關 AWS Fargate 服務配額的詳細資訊，請參閱 Amazon Web Services 一般參考 中的[Amazon ECS endpoints and quotas](#)。

如需 AWS Fargate 用量指標的詳細資訊，請參閱[AWS Fargate 用量指標](#)。

Amazon ECS 中 Fargate 適用時機安全考量

建議需要為任務實現嚴格隔離的客戶採用 Fargate。Fargate 會在硬體虛擬化環境中執行每個任務。此舉可確保這些容器化工作負載，不會與其他任務共用網路介面、Fargate 暫時性儲存、CPU 或記憶體。如需詳細資訊，請參閱[的安全概觀 AWS Fargate](#)。

Amazon ECS 中的 Fargate 安全最佳實務

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[的安全概觀 AWS Fargate](#)。

使用 AWS KMS 加密 Fargate 的暫時性儲存

您應該讓 AWS KMS 或您自己的客戶受管金鑰加密暫時性儲存。對於使用平台版本 1.4.0 或更新版本，且託管於 Fargate 上的任務，每個任務都會配置至少 20 GiB 的暫時性儲存。如需詳細資訊，請參閱[客戶自管金鑰 \(CMK\)](#)。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 日當日或之後啟動的此類任務，其暫時性儲存空間均採用 AES-256 加密演算法，並透過 Fargate 託管的加密金鑰進行加密。

如需詳細資訊，請參閱[Amazon ECS 任務的儲存選項](#)。

範例：在 Fargate 平台版本 1.4.0 上啟動具有暫時性儲存加密功能的任務

以下命令將在 Fargate 平台版本 1.4 上啟動任務。由於此任務是作為叢集的組成部分啟動，因此任務會使用自動加密的 20 GiB 暫時性儲存。

```
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
```

Fargate 核心系統呼叫追蹤的 SYS_PTRACE 功能

由 Docker 提供從容器中新增或刪除的 Linux 功能預設組態。

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

下列影片說明如何透過 Sysdig [Falco](#) 專案使用此功能。

[#ContainersFromTheCouch：使用 SYS_PTRACE 功能對 Fargate 任務進行疑難排解](#)

在上一個影片中討論的程式碼可以在 GitHub [這裡](#)找到。

搭配 Fargate 執行時期監控使用 Amazon GuardDuty

Amazon GuardDuty 是一種威脅偵測服務，可協助保護您 AWS 環境中的帳戶、容器、工作負載和資料。GuardDuty 使用機器學習 (ML) 模型，以及異常和威脅偵測功能，持續監控不同的日誌來源和執行時期活動，以識別環境中的潛在安全風險和惡意活動和排定其優先順序。

GuardDuty 中的執行期監控透過持續監控 AWS 日誌和聯網活動來識別惡意或未經授權的行為，來保護在 Fargate 上執行的工作負載。執行時期監控使用輕量且全受管的 GuardDuty 安全代理程式，可分析主機上的行為，例如檔案存取、程序執行與網路連線。這涵蓋的問題包括：提升權限、使用暴露的憑證、與惡意 IP 位址或網域通訊，以及 Amazon EC2 執行個體與容器工作負載上出現惡意軟體。如需詳細資訊，請參閱 GuardDuty User Guide 中的 [GuardDuty Runtime Monitoring](#)。

為 Amazon ECS 選用 Fargate 的安全考量

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，因為這些容器將一律在相同的主機上執行並共用運算資源。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 運算子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 也可防止容器存取底層主機的資源，例如檔案系統、裝置、聯網與容器執行期。
- 聯網 - 您可以使用安全群組與網路 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 平台版本

AWS Fargate 平台版本用於參考 Fargate 任務基礎設施的特定執行期環境。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有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問題，會 AWS 建立新的平台版本修補修訂，並淘汰在易受攻擊修訂上執行的任務。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[Amazon ECS 上 AWS Fargate 的任務淘汰和維護](#)。

您可於執行任務或部署服務時指定平台版本。

指定平台版本時，應考慮以下項目：

- 您可以指定特定版本編號，例如 1.4.0 或 LATEST。

LATEST Linux 平台版本為 1.4.0。

LATEST Windows 平台版本為 1.0.0。

- 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。這可確保任務一律位於安全且經修補的基礎結構上。
- 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。
- 以下內容適用於 Fargate Windows 平台版本。

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](#)。

在 Amazon ECS 上遷移至 Linux 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。最佳實務是，在遷移任務之前確認任務在平台版本 1.4.0 上正常運作。

- 進出任務的網路流量行為已更新。自平台版本 1.4.0 起，所有在 Fargate 上執行的 Amazon ECS 任務會配置單一彈性網路介面 (也稱為任務 ENI)，且所有網路流量均透過該 ENI 在 VPC 內傳輸。您可以透過 VPC 流程日誌查看流量詳情。如需更多資訊，請參閱 [Fargate 的 Amazon ECS 任務聯網選項](#)。

- 如果使用的是介面 VPC 端點，建議考量以下事項。
 - 若容器映像透過 Amazon ECR 託管，您需要下列端點。如需詳細資訊，請參閱 Amazon Elastic Container Registry User Guide 中的 [Amazon ECR interface VPC endpoints \(AWS PrivateLink\)](#)。
 - com.amazonaws.**region**.ecr.dkr Amazon ECR VPC 端點
 - com.amazonaws.**region**.ecr.api Amazon ECR VPC 端點
 - Amazon S3 閘道端點
 - 當任務定義引用 Secrets Manager 秘密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](#)。
 - 當任務定義引用 Systems Manager Parameter Store 參數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 User Guide 中的 [Improve the security of EC2 instances by using VPC endpoints for Systems Manager](#)。
 - 與任務相關聯的彈性網路介面 (ENI) 的安全群組需要具有安全群組規則，以允許任務與 VPC 端點之間的流量。

Fargate Linux 平台版本變更日誌

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[Fargate 的 Amazon ECS 任務聯網選項](#)。
 - 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[任務中繼資料端點第 4 版](#)。

- 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[使用負載平衡分佈 Amazon ECS 服務流量](#)。
- 從 2020 年 5 月 28 日開始，在 Fargate 上使用平台版本啟動的任何新 Amazon ECS 任務 1.4.0 都會使用 AWS 擁有的加密金鑰，以 AES-256 加密演算法加密其暫時性儲存。如需詳細資訊，請參閱[Amazon ECS 的 Fargate 任務暫時性儲存](#)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[將 Amazon EFS 磁碟區與 Amazon ECS 搭配使用](#)。
- 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[Amazon ECS 的 Fargate 任務暫時性儲存](#)。
- 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[EC2 的 Amazon ECS 任務聯網選項](#)。如需有關 Fargate 聯網的詳細資訊，請參閱[Fargate 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 Container Insights 將包含 Fargate 任務的網路效能指標。如需詳細資訊，請參閱[使用具有增強可觀測性的 Container Insights 監控 Amazon ECS](#)。
- 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 任務中繼資料端點第 4 版](#)及[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[Amazon ECS 已停止任務錯誤訊息](#)。
- 以 Amazon Linux 2 為基礎。

AWS Fargate Linux 平台版本棄用

Important

我們將在 2026 年 6 月 30 日結束對 Fargate 中 PV 1.3.0 的支援。從 2026 年 6 月 15 日開始，我們會將平台版本 1.3.0 設為已淘汰。屆時，您將無法啟動新任務或建立使用平台版本 1.3.0 設定的新服務，但現有任務仍可持續執行。在 2026 年 6 月 30 日，我們將終止使用平台 1.3.0 版設定的所有剩餘執行中任務。

如需有關如何遷移至平台版本 1.4 的資訊，請參閱[在 Amazon ECS 上遷移至 Linux 平台版本 1.4.0](#)。

下表列出 AWS Fargate 已棄用或已排程棄用的 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。

如需有關如何遷移至最新平台版本的資訊，請參閱[在 Amazon ECS 上遷移至 Linux 平台版本 1.4.0](#)。

平台版本達到棄用日期後，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
1.3.0		2026 年 6 月 15 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 平台版本](#)。

已棄用的 Fargate Linux 版本變更日誌

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 FireLens for Amazon ECS 設定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 日誌傳送至 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[Amazon ECS 上 AWS Fargate 的任務淘汰和維護](#)。
- 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，啟動的任何新 Fargate 任務都支援將敏感資料注入您的容器，方法是將敏感資料存放在 AWS Secrets Manager 秘密或 AWS Systems Manager 參數存放區參數中，然後在您的容器定義中參考它們。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，啟動的任何新 Fargate 任務都支援 CloudWatch Container Insights。如需詳細資訊，請參閱[使用具有增強可觀測性的 Container Insights 監控 Amazon ECS](#)。
- 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[Fargate 的 Amazon ECS 叢集](#)。
- 以 Amazon Linux 2 為基礎。

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](#)。

- 新增使用的私有登錄檔身分驗證支援 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[Fargate 上任務可用的 Amazon ECS 任務中繼資料](#)。
- 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[使用服務探索以利用 DNS 名稱連接 Amazon ECS 服務](#)。

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。

Fargate Windows 平台版本變更日誌

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

為 Amazon ECS 選用 Fargate Windows 容器的安全考量

以下是您在 AWS Fargate 上執行 Windows 容器時需要知道的差異和考量事項。

如需在 Linux 與 Windows 容器上執行任務，必須為每個作業系統建立獨立的任務定義。

AWS 會處理作業系統授權管理，因此您不需要任何其他 Microsoft Windows Server 授權。

AWS Fargate 上的 Windows 容器支援下列作業系統：

- Windows Server 2019 Full
- Windows Server 2019 Core
- Windows Server 2022 Full
- Windows Server 2022 Core

AWS Fargate 上的 Windows 容器支援 awslogs 驅動程式。如需詳細資訊，請參閱[the section called “將日誌傳送至 CloudWatch”](#)。

Fargate 上的 Windows 容器不支援下列功能：

- Amazon FSx
- ENI 中繼
- Windows 容器的 gMSA
- 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
- EBS 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 容器會忽略任務層級的 CPU 與記憶體參數。我們建議為 Windows 容器指定容器層級的資源。
- 任務的記憶體
- 容器上的記憶體保留量
- 容器上的重新啟動政策
- 您無法更新服務的平台系列。

Amazon ECS 的 Fargate 容器映像上 Linux 容器的提取行為

每個 Fargate 任務均執行於專屬的一次性、單租用戶執行個體上。在 Fargate 上執行 Linux 容器時，系統不會快取執行個體上的容器映像或容器映像層。因此，對於任務中定義的每個容器映像，每個 Fargate 任務皆需從容器映像登錄檔中完整提取整個映像。提取映像所需的時間與啟動 Fargate 任務所需的時間直接相關。

請考量下列事項，以最佳化映像提取時間。

容器映像就近存取

若要縮短下載容器映像所需的時間，應將資料存放位置盡可能靠近運算資源。透過網際網路或跨提取容器映像 AWS 區域可能會影響下載時間。建議將容器映像存放在將執行任務的同一區域中。如果將容器映像儲存於 Amazon ECR 中，可使用 VPC 介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 User Guide 中的 [Amazon ECR interface VPC endpoints \(AWS PrivateLink\)](#)。

容器映像大小縮減

容器映像的大小會直接影響下載時間。縮減容器映像的大小或容器映像層的數量，可以縮短下載映像所需的時間。輕量型基礎映像 (例如，精簡版 Amazon Linux 2023 容器映像) 的體積，可能遠小於基於傳統作業系統的基礎映像。如需有關精簡映像的詳細資訊，請參閱 Amazon Linux 2023 User Guide 中的 [AL2023 Minimal container image](#)。

替代壓縮演算法

容器映像層在推送至容器映像登錄檔時，通常會經過壓縮處理。壓縮容器映像層可減少需透過網路傳輸的資料量，以及儲存在容器映像檔登錄中的資料量。當容器映像層經由容器執行時期下載至執行個體後，該層會進行解壓縮。所使用的壓縮演算法以及執行時期可用的 vCPU 數量，皆會影響容器映像解壓縮所需的時間。在 Fargate 上，您可以透過增加任務大小或採用效能更佳的 zstd 壓縮演算法來縮短解壓縮時間。如需詳細資訊，請參閱 GitHub 上的 [zstd](#)。如需如何實作 Fargate 映像的資訊，請參閱[使用 zstd 壓縮容器映像縮短 AWS Fargate 啟動時間](#)。

延遲載入容器映像

對於大型容器映像 (> 250 MB)，最好是延遲載入容器映像，而不是下載所有容器映像。在 Fargate 上，您可以使用 Seekable OCI (SOCI) 從容器映像登錄檔延遲載入容器映像。如需詳細資訊，請參閱 GitHub 上的 [soci-snapshotter](#)，以及[使用 Seekable OCI \(SOC\) 延遲載入容器映像](#)。

Amazon ECS 的 Fargate 容器映像上 Windows 容器的提取行為

Fargate Windows 會快取 Microsoft 提供的最新一個月與上一個月的伺服器核心基礎映像。這些映像對應每個修補程式星期二更新的 KB/建置編號。例如，Microsoft 於 2024 年 4 月 9 日針對 Windows Server 2019 發布了 KB5036896 (17763.5696)。2024 年 3 月 12 日發布的前月 KB 修補程式為 KB5035849 (17763.5576)。因此，針對平台 WINDOWS_SERVER_2019_CORE 與 WINDOWS_SERVER_2019_FULL，下列容器映像已快取：

- `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 於 2024 年 4 月 9 日針對 Windows Server 2022 發布了 KB5036909 (20348.2402)。2024 年 3 月 12 日發布的前月 KB 修補程式為 KB5035857 (20348.2340)。因此，針對平台 WINDOWS_SERVER_2022_CORE 與 WINDOWS_SERVER_2022_FULL，下列容器映像已快取：

- `mcr.microsoft.com/windows/servercore:ltsc2022`

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

Amazon ECS 的 Fargate 任務暫時性儲存

佈建時，在上託管在 Linux 容器上的每個 Amazon ECS 任務都會 AWS Fargate 收到下列暫時性儲存體，用於綁定掛載。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。上的 Windows 容器不支援此項目 AWS Fargate。

Fargate Linux 容器平台版本

1.4.0 版或更新版本

依預設，使用平台版本 1.4.0 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 GiB 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、壓縮和未壓縮容器映像會存放在暫時性儲存體中。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受管金鑰。如需詳細資訊，請參閱[AWS Fargate 暫時性儲存的客戶受管金鑰](#)。

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 Amazon CloudWatch Container Insights，暫時性儲存預留大小和使用量會傳送至 Container Insights。

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。

1.3.0 版或更早版本

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

1.0.0 版或更新版本

依預設，使用平台版本 1.0.0 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 GiB 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、壓縮和未壓縮容器映像會存放在暫時性儲存體中。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](#)。

Amazon ECS AWS Fargate 暫時性儲存的客戶受管金鑰

AWS Fargate 支援客戶受管金鑰來加密存放在暫時性儲存體中的 Amazon ECS 任務的資料，以協助對法規敏感的客戶滿足其內部安全政策。客戶在享有 Fargate 無伺服器優勢的同時，仍能向合規稽核人員展現對自行管理儲存加密的增強可見性。儘管 Fargate 預設提供由 Fargate 管理的暫時性儲存加密，但客戶在加密財務或健康相關資訊等敏感資料時，也可使用自行管理金鑰。

您可以將自己的金鑰匯入，AWS KMS 或在其中建立金鑰 AWS KMS。這些自我管理金鑰存放在 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 日誌中稽核金鑰的存取與使用情況。

依預設，每個 KMS 金鑰可支援 50,000 個授權。Fargate 會針對每個客戶受管金鑰任務使用單一 AWS KMS 授權，因此支援金鑰最多 50,000 個並行任務。若想提高此數量，您可以申請提高限額，此類申請將依具體情況加以核准。

Fargate 不會對使用客戶自管金鑰而收取任何額外費用。您只需支付使用儲存和 API 請求 AWS KMS 金鑰的標準價格。

主題

- [為 Amazon ECS 建立 Fargate 暫時性儲存的加密金鑰](#)
- [管理 Amazon ECS Fargate 暫時性儲存的 AWS KMS 金鑰](#)

為 Amazon ECS 建立 Fargate 暫時性儲存的加密金鑰

建立客戶自管金鑰，加密儲存於 Fargate 暫時性儲存中的資料。

Note

使用客戶自管金鑰的 Fargate 暫時性儲存加密，不適用於 Windows 任務叢集。

使用客戶自管金鑰的 Fargate 暫時性儲存加密，不適用於早於 1.4.0 的 platformVersions。

Fargate 會在僅供 Fargate 使用的暫時性儲存上保留空間，而且不會向您收取該空間的費用。資源配置能與非客戶自管金鑰的任務不同，但總空間保持不變。您可以在 df 等工具中檢視此變更。

Fargate 暫時性儲存不支援多區域金鑰。

Fargate 暫時性儲存不支援 KMS 金鑰別名。

若要建立客戶受管金鑰 (CMK) 來加密 Fargate 中的暫時性儲存 AWS KMS，請遵循下列步驟。

1. 導覽至 <https://console.aws.amazon.com/kms>。
2. 請遵循 [AWS Key Management Service Developer Guide](#) 中的 [Creating Keys](#) 的說明操作。
3. 建立 AWS KMS 金鑰時，請務必在金鑰政策中提供 Fargate 服務相關的 AWS KMS 操作許可。若要將客戶自管金鑰與 Amazon ECS 叢集資源搭配使用，必須在金鑰政策中允許下列 API 操作。
 - kms:GenerateDataKeyWithoutPlainText - 呼叫 從提供的金鑰GenerateDataKeyWithoutPlainText產生加密的資料 AWS KMS 金鑰。

- `kms:CreateGrant`：新增客戶自管金鑰的授權。授予控制對指定 AWS KMS 金鑰的存取，允許存取 Amazon ECS Fargate 所需的授予操作。如需詳細資訊，請參閱 [AWS Key Management Service Developer Guide](#) 中的 [Using Grants](#)。此授權使 Amazon ECS Fargate 能執行以下操作：
- 呼叫 `Decrypt` AWS KMS 以取得加密金鑰，以解密暫時性儲存資料。
- 設定淘汰主體，允許服務 `RetireGrant`。
- `kms:DescribeKey` - 提供客戶自管金鑰詳細資訊，使 Amazon ECS 能驗證金鑰是否對稱且已啟用。

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政策。若要使用下列政策範例，請將#####取代為您自己的資訊。一如往常，只設定您需要的許可，但您需要為至少一個使用者提供 AWS KMS 許可，以避免錯誤。

```
{
  "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/customer-chosen-role" },
  "Action": [
    "kms:DescribeKey"
  ],
  "Resource": "*"
}

```

Fargate 任務使用 `aws:ecs:clusterAccount` 與 `aws:ecs:clusterName` 加密內容金鑰搭配該金鑰進行密碼編譯操作。客戶應新增這些許可，限制對特定帳戶和/或叢集的存取。在指定叢集時，請使用叢集名稱而非 ARN。

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`。此條件索引鍵讓客戶可以更精細地控制 IAM 政策。`fargateEphemeralStorageKmsKeyId` 組態的更新僅在新服務部署上生效。

以下是允許客戶僅將許可授予一組特定核准 AWS KMS 金鑰的範例。

JSON

```
{
  "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建立關聯的 AWS KMS 金鑰的範例。

JSON

```
{
  "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 `describe-cluster`、或 AWS CLI `describe-tasks``describe-services` 命令，查看其未受管任務或服務任務是否使用 金鑰加密。

如需詳細資訊，請參閱 [AWS KMS Developer Guide](#) 中的 [Condition keys for AWS KMS](#)。

AWS 管理主控台

1. 開啟主控台，網址為 <https://console.aws.amazon.com/ecs/v2>。
2. 請於左側導覽中選擇叢集，然後點選右上角的建立叢集，或選取現有叢集。若採用現有叢集，則選擇右上角的更新叢集。
3. 在工作流程的加密區段下，您可以選擇在受管儲存和 Fargate 暫時性儲存下選取 AWS KMS 金鑰。您也可以從這裡選擇建立 AWS KMS 金鑰。
4. 完成新叢集的建立後，請選擇建立；如果要更新現有叢集，請選擇更新。

AWS CLI

以下是使用 建立叢集和設定 Fargate 暫時性儲存的範例 AWS CLI（使用您自己的值取代##值）：

```

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
}
```

CloudFormation

以下是使用 建立叢集和設定 Fargate 暫時性儲存的範例範本 CloudFormation（使用您自己的值取代##值）：

```

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 Amazon ECS Fargate 暫時性儲存的 AWS KMS 金鑰

在建立或匯入您的 AWS KMS 金鑰以加密 Fargate 暫時性儲存體之後，您可以像管理任何其他 AWS KMS 金鑰一樣進行管理。

自動輪換 AWS KMS 金鑰

您可以啟用自動金鑰輪換功能，也可以手動輪換金鑰。自動金鑰輪換會每年為您輪換金鑰，方法是為金鑰產生新的密碼編譯材料。AWS KMS 也會儲存所有舊版本的密碼編譯材料，因此您可以解密使用舊金鑰版本的任何資料。在您刪除金鑰 AWS KMS 之前，不會刪除任何輪換的資料。

自動金鑰輪換是選用功能，可以隨時啟用或停用。

停用或撤銷 AWS KMS 金鑰

如果您在 中停用客戶受管金鑰 AWS KMS，對執行中的任務沒有任何影響，而且它們會在其生命週期內繼續運作。如果新任務使用已停用或已撤銷的金鑰，任務會因無法存取金鑰而失敗。您應該設定 CloudWatch 警示或類似警示，確保不再需要已停用的金鑰來解密已加密的資料。

刪除 AWS KMS 金鑰

刪除金鑰應一律是最後手段，且僅在確定不再需要刪除的金鑰時才應執行該動作。嘗試使用已刪除金鑰的新任務將會失敗，因為他們無法存取該金鑰。AWS KMS 建議停用金鑰，而不是將其刪除。若確有必要刪除某個金鑰，建議先停用金鑰並設定 CloudWatch 警示，確保該金鑰不再被使用。如果您刪除金鑰，AWS KMS 至少提供七天來改變主意。

稽核 AWS KMS 金鑰存取

您可以使用 CloudTrail 日誌來稽核對 AWS KMS 金鑰的存取。您可以檢查 AWS KMS 操作 CreateGrant、GenerateDataKeyWithoutPlaintext 和 Decrypt。這些操作也會在 CloudTrail 記錄中顯示作為 EncryptionContext 組成部分的 aws:ecs:clusterAccount 與 aws:ecs:clusterName。

以下是 GenerateDataKeyWithoutPlaintext、GenerateDataKeyWithoutPlaintext (DryRun)、CreateGrant、CreateGrant (DryRun) 與 RetireGrant 的 CloudTrail 事件範例 (將 ## 值取代為實際參數)。

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-EXAMPLEEaaaaa",
    "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-EXAMPLEEaaaaa",
  "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":
        "e3b0c44298fc1c149afb4c8996fb92427ae41e4649b934ca495991b7852b855",
    "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-EXAMPLEEaaaaa",
"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-EXAMPLEEaaaaa",
  "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": "e3b0c44298fc1c149afb4c8996fb92427ae41e4649b934ca495991b7852b855"
  },
  "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-EXAMPLEEaaaaa",
  "eventCategory": "Management"
}
```

Amazon ECS 上 AWS Fargate 的任務淘汰和維護

AWS 負責維護 AWS Fargate 的基礎基礎設施。AWS 決定何時需要將平台版本修訂取代為基礎設施的新修訂。這稱為任務淘汰。在平台版本修訂淘汰時 AWS 傳送任務淘汰通知。我們會定期更新我們支援的平台版本，推出新的修訂版，內含 Fargate 執行時期軟體更新，以及作業系統與容器執行時期等底層相依性更新。推出較新的修訂版後，我們會淘汰較舊的修訂版，確保所有客戶工作負載在 Fargate 平台版本的最新修訂版上執行。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。服務任務部署為服務的組成部分，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 啟動的任務，無論是直接或外部排程器啟動，例如排程任務（由 Amazon EventBridge 啟動）AWS Batch，或 AWS Step Functions。您無需採取任何動作來因應服務任務的任務淘汰，因為 Amazon ECS 排程器會自動取代這些任務。

對於獨立任務，您可能需要執行額外的處理以因應任務淘汰。如需詳細資訊，請參閱[Amazon ECS 是否可以自動處理獨立任務？](#)。

對於服務任務，您不需要採取任何動作來淘汰任務，除非您想要在 AWS 之前取代這些任務。Amazon ECS 排程器會在停止任務時使用 `maximumPercent` 並啟動新任務，嘗試維持服務的預期執行數量。遵循最佳實務，盡力降低任務淘汰傳送的影響。使用 REPLICAS 服務排程器的服務，其預設 `maximumPercent` 值為 200%。因此，當 AWS Fargate 開始淘汰任務時，Amazon ECS 會先排程新任務，然後等待它執行，然後再淘汰舊任務。在將 `maximumPercent` 值設定為 100% 後，Amazon ECS 會先停止任務，再取代任務。

對於獨立任務淘汰，會在任務淘汰日期或之後 AWS 停止任務。Amazon ECS 不會在任務停止時啟動替代任務。若需讓這些任務持續執行，您必須在通知指定的時間前，停止執行中任務並啟動替代任務。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。

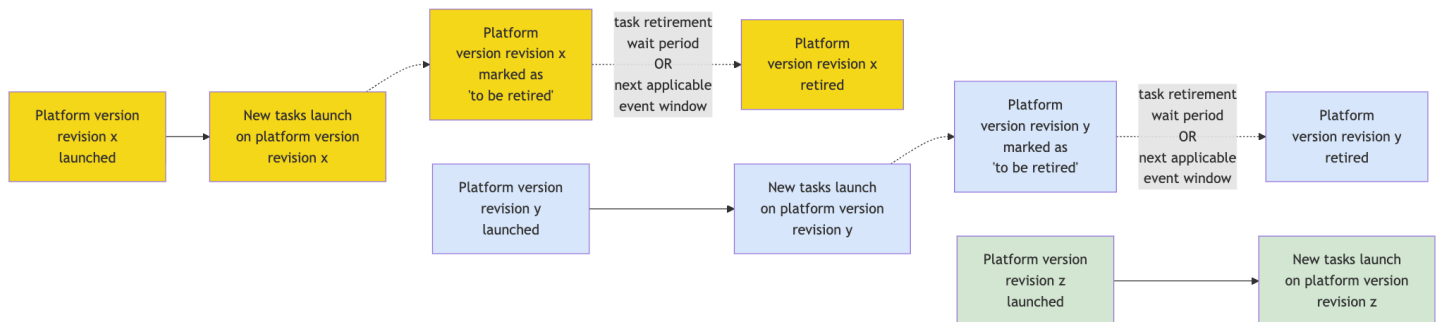
當任務在任何場景下停止時，您都可以執行 `describe-tasks`。回應中的 `stoppedReason` 是 `ECS is performing maintenance on the underlying infrastructure hosting the task`。

如需將新的平台版本修訂版取代為新的修訂版，即會套用任務維護機制。如果底層 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。

任務淘汰通知概觀

當將平台版本修訂 AWS 標記為需要淘汰時，我們會識別所有區域中在該平台版本修訂上執行的所有任務。

下圖呈現了 Fargate 平台版本修訂版的生命週期，涵蓋從新修訂版發布到平台修訂版淘汰的全過程。



下面提供相關詳細資訊。

- 新平台版本修訂版發布後，所有新任務都將排程至此修訂版上執行。
- 已排程和執行中的現有任務，皆會保留在最初於任務期間置放的修訂版上，而不會遷移至新的修訂版。
- 新任務，例如作為服務或 Fargate 任務淘汰更新的組成部分，皆會置放在啟動時可用的最新平台版本修訂版上。

任務淘汰通知會透過 AWS Health Dashboard 和電子郵件傳送至已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 AWS 區域中的每項服務與獨立任務傳送一則通知。然而在某些情況下，您可能會針對每種任務類型收到多則事件通知，例如當待淘汰任務數量過多，超出我們通知機制的承載上限時。

您可以透過以下方法確定已排程要淘汰的任務：

- 的 Health 儀板表

AWS Health 通知可以透過 Amazon EventBridge 傳送至封存儲存，例如 Amazon Simple Storage Service、執行 AWS Lambda 函數等自動動作，或其他通知系統，例如 Amazon Simple Notification Service。如需詳細資訊，請參閱[使用 Amazon EventBridge 監控 AWS Health 事件](#)。如需將通知傳送至 Amazon Chime、Slack 或 Microsoft Teams 的組態範例，請參閱 GitHub 上的[AWS Health Aware](#) 儲存庫。

以下是 EventBridge 事件範例。

```
{
  "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": "arn:aws:ecs:eu-west-1:111222333444:task/
examplecluster/00805ce1d81940b5a37398e5a2c23333"
      }
    ]
  }

```

```
    },
    {
      "entityValue": "arn:aws:ecs:eu-west-1:111222333444:task/
examplecluster/00805ce1d81940b5a37398e5a2c25555"
    }
  ]
}
```

- Email

系統會將電子郵件傳送至 AWS 帳戶 ID 的註冊電子郵件。

如需有關如何為任務淘汰做準備的資訊，請參閱[準備在 Amazon ECS 上淘汰 AWS Fargate 任務](#)。

我可以選擇退出任務淘汰機制嗎？

否。作為 AWS 共同責任模型的一部分，AWS 負責管理和維護其基礎基礎設施 AWS Fargate。這包括執行定期平台更新，確保安全性與穩定性。這些更新會由自動套用 AWS，而且客戶無法選擇退出。與在 EC2 執行個體上執行工作負載 AWS Fargate 相比，這是使用的主要優點，維護基礎平台的責任由處理 AWS。此模式可讓您專注於應用程式，而不是基礎結構維護。透過自動套用這些平台更新，AWS 能夠讓 Fargate 環境 up-to-date 狀態和安全，而無需您作為客戶採取任何動作。這有助於提供可靠且安全的容器化環境，以利在 Fargate 上執行工作負載。

我可以透過其他服務取得任務淘汰通知 AWS 嗎？

AWS 會將任務淘汰通知傳送至 `Health` 儀板表並傳送給上的主要電子郵件聯絡人 AWS 帳戶。`Health` 儀板表提供多種與其他 AWS 服務的整合，包括 EventBridge。您可以使用 EventBridge 自動化的可見性 (例如轉送訊息至 ChatOps 工具)。如需詳細資訊，請參閱[Solution overview: Capturing task retirement notifications](#)。

我可以在任務排定後變更任務淘汰嗎？

否。排程是根據任務淘汰等待時間，預設值為 7 天。如果需要更多時間，可選擇將等待期間設定為 14 天。如需詳細資訊，請參閱[步驟 2：擷取任務淘汰通知，提醒團隊採取動作](#)。

自 12/18/2025 起，Amazon ECS 可讓您為 Fargate 任務設定 [Amazon EC2 事件時段](#)。如果您需要精確控制任務淘汰的確切時間，例如，將它們安排在週末以避免上班時間中斷，您可以為您的任務、服務或叢集設定 Amazon EC2 事件時段，請參閱 [步驟 1：設定任務等待時間或使用 Amazon EC2 事件時](#)

段。請注意，此組態的變更適用於未來將排程的淘汰。目前已排定的淘汰不會受到影響。此外，當您為 Fargate 任務設定 Amazon EC2 事件時段時，其優先順序會高於您的任務淘汰等待時間組態。如果您有任何其他疑慮，請聯絡 支援。

Amazon ECS 如何處理屬於服務組成部分的任務？

對於服務任務，您不需要採取任何動作來回應任務淘汰，除非您想要在 AWS 之前取代這些任務。Amazon ECS 排程器會在停止任務時依據運作狀態百分比下限啟動新任務，嘗試維持服務的預期執行數量。為盡力降低 Fargate 任務淘汰產生的影響，工作負載應按照 Amazon ECS 最佳實務進行部署。例如，將無狀態應用程式部署為 Amazon ECS 服務 (例如 Web 或 API 伺服器) 時，客戶應部署多個任務副本，並將 minimumHealthyPercent 設定為 100%。依預設，服務的運作狀態百分比下限為 100%。因此，當 Fargate 開始淘汰任務時，Amazon ECS 會先排程新任務，待其進入執行狀態後，再淘汰舊任務。當您擴展服務規模、部署組態變更或部署任務定義修訂版時，在任務淘汰過程中，系統會以相同的方式定期取代服務任務。若要為任務淘汰程序做準備，請參閱[準備在 Amazon ECS 上淘汰 AWS Fargate 任務](#)。

Amazon ECS 是否可以自動處理獨立任務？

否。AWS 無法為由 RunTask、排程任務 (例如透過 EventBridge 排程器) AWS Batch 或啟動的獨立任務建立替代任務 AWS Step Functions。Amazon ECS 僅管理屬於服務組成部分的任務。

對任務淘汰期間的服務可用性進行疑難排解

如果 Amazon ECS 在任務淘汰期間無法啟動替代任務，服務可用性可能會受到影響。此情況可能由錯誤的客戶設定引發，例如：

- IAM 角色遺失或設定不正確
- 目標子網路中的容量不足
- 安全群組設定錯誤
- 任務定義錯誤

當 Amazon ECS 無法啟動替代任務時，淘汰任務將在無替換的情況下停止，導致服務可用容量降低，甚至中斷服務。請監控服務的任務數量與 Amazon CloudWatch 指標，確保在淘汰事件期間，替代任務能順利啟動。

準備在 Amazon ECS 上淘汰 AWS Fargate 任務

若要為任務淘汰做準備，請執行下列操作：

1. 設定任務淘汰等待期間或使用 Amazon EC2 事件時段。
2. 擷取任務淘汰通知，以便通知團隊成員。
3. 您可以使用強制部署選項更新服務，以確保所有服務的任務在最新的平台修訂版上執行。此為選擇性步驟。

步驟 1：設定任務等待時間或使用 Amazon EC2 事件時段

您有兩個帳戶設定選項來設定 Fargate 開始任務淘汰的時間：
`fargateTaskRetirementWaitPeriod` 和 `fargateEventWindows`。

使用 `fargateTaskRetirementWaitPeriod` 帳戶設定

您可以設定 Fargate 開始任務淘汰的時間。預設等待期間為 7 天。對於需要立即套用更新的工作負載，請選擇立即設定 (0)。如果您需要更多時間，請設定 7 或 14 日期選項。

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。

透過以根使用者或管理使用者的身分執行 `put-account-setting-default` 或 `put-account-setting` 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任務。
- 7 - AWS 傳送通知，並等待 7 個日曆天，再開始淘汰受影響的任務。這是預設值。
- 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。

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](#)。

使用 `fargateEventWindows` 帳戶設定

自 12/18/2025 起，Amazon ECS 可讓您為 Fargate 任務設定 [Amazon EC2 事件時段](#)。如果您需要精確控制任務淘汰的確切時間，例如，將它們安排在週末以避免上班時間中斷，您可以為您的任務、服務或叢集設定 Amazon EC2 事件時段。

當您使用事件時段時，Fargate 會確保您的任務在下一個可用時段內淘汰前至少執行 3 天，除非使用者啟動的動作或重大運作狀態事件停止，例如基礎硬體降級。

將 `fargateEventWindows` 帳戶設定設為 `enabled`。您可以使用下列其中一個 APIs：`put-account-setting-default` 或 `put-account-setting` 做為根使用者或管理使用者。

每個 Amazon EC2 事件時段每週必須開放至少 4 小時，而且每個時間範圍必須至少 2 小時。對於大型叢集和服務，我們建議設定事件時段，其持續時間較長 (8 小時或以上) 或更頻繁，至少每 3 天發生一次。您可以進一步檢閱[使用者指南](#) AWS Fargate 中 Amazon EC2 事件時段的考量，確保您的任務在淘汰前執行至少 3 天，除非使用者啟動的動作或重大運作狀態事件停止，例如基礎硬體降級。

Important

最好在事件時段內取代任務。如果您注意到任務在事件時段之外遭到淘汰，請考慮延長持續時間 (8 小時或以上) 或增加頻率 (至少每 3 天一次)。

若要將 Amazon EC2 事件時段套用至 Fargate 任務淘汰：

- 將 fargateEventWindows 帳戶設定設為 enabled。您可以使用下列其中一個 APIs：put-account-setting-default 或 put-account-setting 做為根使用者或管理使用者。請注意，這是針對 Fargate 任務使用 Amazon EC2 事件時段功能的一次性啟用。
- 透過 AWS 主控台或 AWS CLI 建立 Amazon EC2 事件視窗。若要使用 CLI 建立事件時段，請使用 EC2 create-instance-event-window API 搭配時間範圍或 Cron 表達式。記下 InstanceEventWindowId 回應中的。

```
aws ec2 create-instance-event-window \
    --time-range
    StartWeekDay=monday,StartHour=2,EndWeekDay=wednesday,EndHour=8 \
    --tag-specifications "ResourceType=instance-event-
window,Tags=[{Key=K1,Value=V1" \
    --name myEventWindowName
```

或者，您可以在建立 EC2 事件時段時使用 cron 表達式。

```
aws ec2 create-instance-event-window \
    --cron-expression "* 21-23 * * 2,3" \
    --tag-specifications "ResourceType=instance-event-
window,Tags=[{Key=K1,Value=V1" \
    --name myEventWindowName
```

- 然後，您可以使用 EC2 associate-instance-event-window API 將事件視窗與帳戶中的特定服務、叢集或所有任務建立關聯。
- 對於 ECS 服務任務

```
aws ec2 associate-instance-event-window \  
--instance-event-window-id iew-0abcdef1234567890 \  
--association-target "InstanceTags=[{Key=aws:ecs:serviceArn,Value=your-service-arn}]"
```

- 對於 ECS 叢集

```
aws ec2 associate-instance-event-window \  
--instance-event-window-id iew-0abcdef1234567890 \  
--association-target "InstanceTags=[{Key=aws:ecs:clusterArn,Value=your-cluster-arn}]"
```

- 將事件時段與帳戶中的所有任務建立關聯

```
aws ec2 associate-instance-event-window \  
--instance-event-window-id iew-0abcdef1234567890 \  
--association-target "InstanceTags=[{Key=aws:ecs:fargateTask,Value=true}]"
```

您可以使用多個鍵值對，將事件時段與多個服務或叢集建立關聯。

Fargate 會依下列順序為每個任務選擇事件時段：

- 如果有與任務服務相關聯的事件時段，則會使用該時段。這不適用於獨立或未受管任務。
- 如果有與任務叢集相關聯的事件時段，則會使用該時段。
- 如果為所有 Fargate 任務設定了事件時段，則會使用該時段。
- 如果沒有事件時段與任務相符，則會使用 `fargateTaskRetirementWaitPeriod` 設定。

設定 Fargate 任務維護的事件時段

當您在具有不同可用性需求的 Fargate 上執行多個 ECS 服務時，請考慮一個案例。您想要精確控制任務淘汰。您可以設定多個事件時段，如下所示：

- 所有 Fargate 任務的預設維護：在離峰時間（每天 12AM 到 4AM）建立例行維護的事件時段，並使用 `aws:ecs:fargateTask` 標籤將其與所有 Fargate 任務建立關聯。

- 開發叢集的週末限定維護：對於具有可容忍週末中斷服務的開發叢集，請建立 24 小時週末時段（週六和週日，全天），並使用 `aws:ecs:clusterArn` 標籤與叢集 ARN 將其與叢集建立關聯。
- 關鍵任務服務的限制時段：對於在工作日需要高正常執行時間的任務關鍵付款處理服務，請將維護限制為週末凌晨時間（週六和週日，12AM 點至上午 4AM），並將其與您的服務 ARN 使用 `aws:ecs:serviceArn` 標籤與特定服務建立關聯。

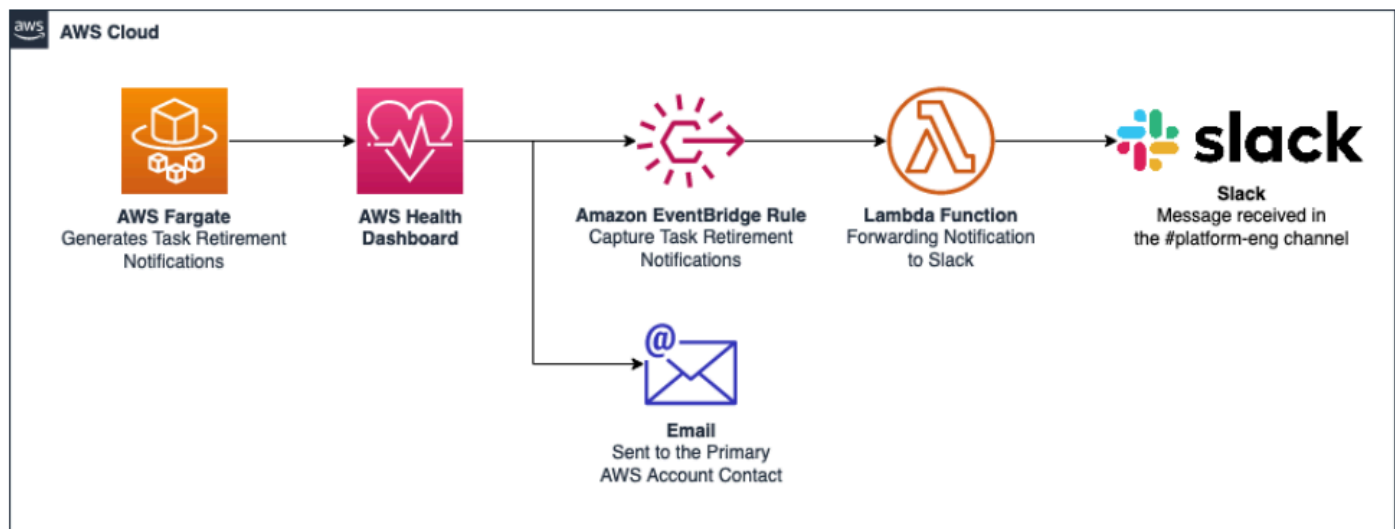
透過此組態，付款服務會使用其特定的僅限週末時段，開發叢集服務和任務會使用週末 24 小時時段，而所有其他 Fargate 任務會使用預設的每日維護時段。

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](#)。

步驟 2：擷取任務淘汰通知，提醒團隊採取動作

當即將淘汰任務時，AWS 會將任務淘汰通知傳送到 AWS Health 儀表板，並傳送給上的主要電子郵件聯絡人 AWS 帳戶。AWS Health 儀表板提供多種與其他 AWS 服務的整合，包括 Amazon EventBridge。您可以使用 EventBridge 從任務淘汰通知建立自動化，例如透過將訊息轉送到 ChatOps 工具來提高即將淘汰的可見性。AWS Health Aware 是一種資源，可顯示 AWS Health 儀表板的強大功能，以及如何將通知分佈到整個組織中。您可以將任務淘汰通知轉送至 Slack 等聊天應用程式。

下圖說明了解決方案概觀。



下面提供相關詳細資訊。

- Fargate 會將任務淘汰通知傳送至 AWS Health 儀表板。

- AWS Health 儀表板會將郵件傳送至 上的主要電子郵件聯絡人 AWS 帳戶，並通知 EventBridge。
- EventBridge 具備可擷取淘汰通知的規則。

此規則用於篩選「事件詳細類型」如下的事件："AWS Health Event" and the Event Detail Type Code: "AWS_ECS_TASK_PATCHING_RETIREMENT"

- 此規則會觸發 Lambda 函式，該函式會透過 Slack 傳入 Webhook 將資訊轉送至 Slack。如需詳細資訊，請參閱 [Incoming Webhooks](#)。

如需程式碼範例，請參閱在 Github [上擷取 AWS Fargate 任務淘汰通知](#)。

步驟 3：控制任務替代時機

您無法控制任務淘汰的確切時間，但可以定義等待時間。若需按照自訂排程控制取代任務，您可以擷取任務淘汰通知，先了解任務淘汰日期。然後，您可以重新部署服務來啟動替代任務，並同步取代所有獨立任務。對於使用滾動部署的服務，您可以在淘汰開始時間前，搭配 force-deployment 選項使用 update-service 來更新服務。

下列 update-service 範例使用 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 AWS CodeDeploy 中建立新的部署。如需有關如何建立部署的資訊，請參閱 AWS Command Line Interface Reference 中的 [create-deployment](#)。

AWS Fargate 上 Amazon ECS 支援的 區域

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

AWS Fargate 上的 Linux 容器

下列 AWS Fargate 支援 上的 Amazon ECS Linux 容器 AWS 區域。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)
美國西部 (奧勒岡)	us-west-2
加拿大 (中部)	ca-central-1
加拿大西部 (卡加利)	ca-west-1
墨西哥 (中部)	mx-central-1
Africa (Cape Town)	af-south-1
亞太地區 (香港)	ap-east-1
亞太區域 (孟買)	ap-south-1
亞太地區 (東京)	ap-northeast-1 (僅 apne1-az1 、 apne1-az2 和 apne1-az4)
亞太區域 (首爾)	ap-northeast-2
亞太地區 (大阪)	ap-northeast-3
亞太地區 (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 (泰國)	ap-southeast-7
亞太地區 (雅加達)	ap-southeast-3
亞太地區 (墨爾本)	ap-southeast-4
亞太地區 (馬來西亞)	ap-southeast-5

區域名稱	區域
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
歐洲 (巴黎)	eu-west-3
歐洲 (米蘭)	eu-south-1
歐洲 (西班牙)	eu-south-2
Europe (Stockholm)	eu-north-1
南美洲 (聖保羅)	sa-east-1
以色列 (特拉維夫)	il-central-1
Middle East (Bahrain)	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 Windows 容器

下列 AWS Fargate 支援 上的 Amazon ECS Windows 容器 AWS 區域。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
Africa (Cape Town)	af-south-1
亞太地區 (香港)	ap-east-1
亞太區域 (孟買)	ap-south-1
亞太地區 (海德拉巴)	ap-south-2
亞太地區 (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
亞太地區 (墨爾本)	ap-southeast-4
亞太地區 (馬來西亞)	ap-southeast-5
亞太地區 (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)

區域名稱	區域
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
Europe (Stockholm)	eu-north-1
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
Middle East (Bahrain)	me-south-1

為 Amazon ECS 規劃 EC2 容量架構

對於必須進行成本最佳化的大型工作負載，可使用 EC2 容量。

在考慮如何透過 EC2 建置工作定義與服務的模型時，建議思考哪些程序必須一起執行，以及如何針對各元件進行擴展。

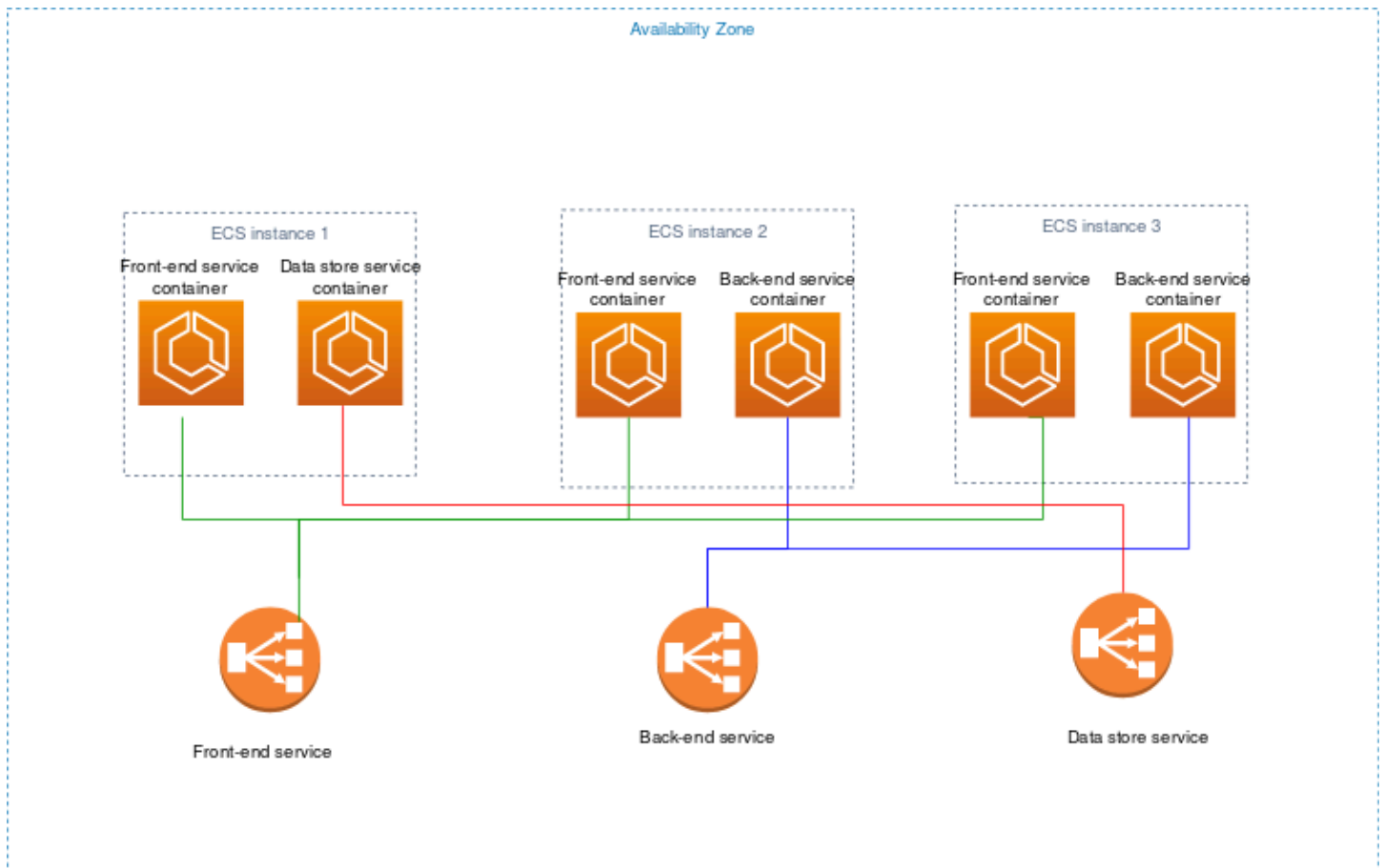
舉例來說，假設由下列元件組成的應用程式：

- 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。將不同的元件拆分為多個任務定義。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[建立 Amazon ECS 滾動更新部署](#)。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[使用負載平衡分佈 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[更新 Amazon ECS 服務](#)。



Amazon ECS 上 EC2 與外部執行個體的容器映像提取行為

容器啟動所需的時間會因底層容器映像而異。例如，比起各自的精簡版本 (Debian-slim、Ubuntu-slim 與 Amazon-slim) 或較小型的基礎映像 (Alpine)，較大型的映像 (Debian、Ubuntu 與 Amazon1/2 的完整版本) 可能需要更長的啟動時間，因為容器中執行的服務數量更多。

當 Amazon ECS 代理程式啟動任務時，會從其遠端登錄檔中提取 Docker 映像檔，然後快取一份本機副本。若為應用程式的每個版本使用新的映像標籤，此快取行為便不再必要。

ECS_IMAGE_PULL_BEHAVIOR 代理程式參數會決定映像提取行為。下列選項可供使用：

- ECS_IMAGE_PULL_BEHAVIOR: default

系統會從遠端提取映像。若提取失敗，則使用執行個體中的快取映像。

- ECS_IMAGE_PULL_BEHAVIOR: always

系統會從遠端提取映像。若提取失敗，則任務會失敗。

若要加速部署，請將 Amazon ECS 代理程式參數設定為下列任一值：

- `ECS_IMAGE_PULL_BEHAVIOR: once`

只有當相同容器執行個體上的前一個任務未提取過映像，或快取映像已由自動化映像清理程序移除時，才會從遠端提取該映像。否則，將使用執行個體上的快取映像。這可確保不會發生非必要的映像提取。

- `ECS_IMAGE_PULL_BEHAVIOR: prefer-cached`

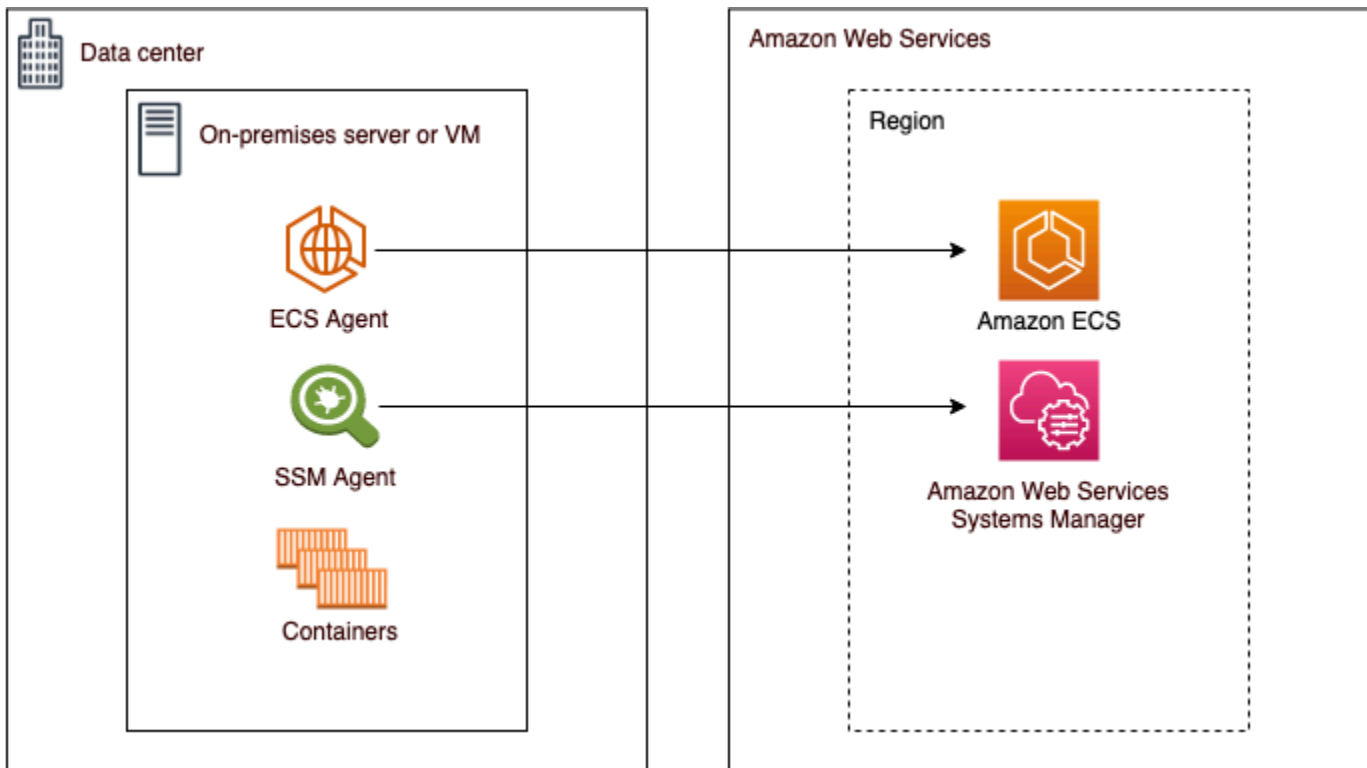
如果沒有快取映像，將會從遠端提取映像。否則，將使用執行個體上的快取映像。為了確保快取映像不會被移除，容器會停用自動映像檔清理功能。

將 `ECS_IMAGE_PULL_BEHAVIOR` 參數設定為上述任一值可節省時間，因為 Amazon ECS 代理程式會直接使用已下載的現有映像。對於較大型的 Docker 映像檔，透過網路提取可能需要 10 至 20 秒的下載時間。

Amazon ECS 的外部執行個體 (Amazon ECS Anywhere)

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。您的內部部署伺服器已安裝 Amazon ECS 代理程式與 SSM 代理程式。



如需詳細資訊，請參閱[外部執行個體的 Amazon ECS 叢集](#)。

Amazon ECS Express 模式

Amazon ECS Express Mode 服務透過提供合理的預設值和自動化支援服務的組態，降低部署容器化應用程式的複雜性 AWS。快速模式服務不需要跨多個服務管理組態參數，只需三件事即可開始：

- 容器映像
- 任務執行角色
- 基礎設施角色

Amazon ECS Express Mode 會協調和設定所有必要的基礎設施：具有唯一可存取 URL 的 Fargate 型 ECS 服務、具有 SSL/TLS 的負載平衡器、自動擴展政策、監控和聯網元件。

在下列案例中使用 Amazon ECS Express Mode 服務：

- Web 應用程式和 APIs - 提供 HTTP 請求的無狀態容器化應用程式
- 快速原型設計 - 快速部署和測試應用程式，無需基礎設施設定額外負荷
- 開發人員生產力 - 讓應用程式團隊無需深入了解即可獨立部署 AWS
- 平台團隊效率 - 透過提供自助式部署功能來降低維護開銷

Amazon ECS Express Mode 服務支援公有或私有 HTTPS 應用程式，並根據使用率或流量自動擴展。

Amazon ECS Express Mode 服務的優點

Amazon ECS Express Mode 服務提供部署和管理 上容器化應用程式的一些主要優勢 AWS。

- 簡化部署 - 在多個 AWS 服務中部署具有生產就緒預設值的容器化應用程式。
- 功能不受影響 - 當您需要精細控制或進階功能時，所有基礎 AWS 資源仍可供直接管理。
- 成本最佳化 - 使用相同的聯網組態跨多個快速模式服務共用 Application Load Balancer，以降低成本。
- 透明基礎設施 - 所有資源都會在 AWS 您的帳戶中建立，並透過 AWS 主控台和 APIs 提供完整的可見性和可存取性。

定價

使用 Amazon ECS Express Mode 服務無需額外費用。您只需為為了執行應用程式而建立的基礎 AWS 資源付費，包括：

- Fargate 運算資源
- Application Load Balancer
- CloudWatch 日誌和指標
- 資料傳輸費用

可用性

Amazon ECS Express Mode 可用於支援 Amazon ECS 和 Fargate 的所有 AWS 區域。您可以使用下列任一項來建立和管理快速模式服務：

- Amazon ECS 主控台
- AWS CLI
- AWS SDKs
- CloudFormation
- Terraform
- AWS 實驗室 MCP Server for Amazon ECS

建立 Amazon ECS Express Mode 服務

Amazon ECS Express Mode 服務透過提供合理的預設值和自動化支援服務的組態，降低部署容器化應用程式的複雜性 AWS。快速模式服務不需要跨多個服務管理組態參數，只需容器映像、任務執行角色和基礎設施角色即可開始使用。

部署完成後，您會收到應用程式的唯一 URL。您可以在叢集中找到您剛部署的服務，如果您未指定叢集，則會在Default叢集中。

先決條件

建立快速模式服務之前，請確定您有下列其中一項：

- 存放在 Amazon ECR 中的容器映像。

- 或存放在私有登錄檔中的容器映像。若要使用私有登錄檔，請設定 Secrets Manager 秘密。如需詳細資訊，請參閱AWS Secrets Manager 《使用者指南》中的[建立 AWS Secrets Manager 秘密](#)。

Note

IAM 角色也是必要的，但分別在[主控台](#)和 [CLI](#) 指南中涵蓋。Amazon ECS 主控台具有自動流程以建立角色。

逐步解說

- 如需使用 主控台的簡化初次執行體驗，請參閱 [the section called “在主控台中建立您的第一個快速模式服務”](#)。
- 若要使用 建立快速模式服務 AWS CLI，請參閱 [the section called “使用 建立您的第一個快速模式服務 AWS CLI”](#)。
- 若要進一步了解快速模式建立的內容及其運作方式，請參閱 [the section called “Express 模式服務建立的資源”](#)。

自訂

建立快速模式服務時，Amazon ECS 會為您的 Amazon ECS 和其他 AWS 資源選取預設值。視需要使用選用參數自訂應用程式。利用快速模式服務並自訂下列項目：

- 服務名稱
- Amazon ECS 叢集
- 容器設定 - 連接埠和運作狀態檢查路徑
- 環境變數和機密
- 自訂命令
- 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。

當您的應用程式存取 Amazon S3 等其他服務時 AWS，您需要此角色。

- 運算資源 - CPU 和記憶體配置
- 網路組態 - 子網路和服務安全群組
- 應用程式自動擴展政策 - 指標、目標值和任務限制
- CloudWatch Logs 組態

- 資源標籤

在主控台中建立您的第一個 Amazon ECS Express Mode 服務

快速模式服務的主控制台體驗提供簡化的方式，以最少的組態部署容器化應用程式。閱讀有關 中先決條件的詳細資訊[Amazon ECS Express 模式](#)。若要進一步了解快速模式建立的內容及其運作方式，請參閱 [the section called “Express 模式服務建立的資源”](#)。

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，選擇快速模式。
3. 在讓我們設定您的應用程式下：
 - a. 指定要用於應用程式的映像。針對映像 URI，輸入映像的 URI。若要瀏覽 Amazon ECR 映像，請選擇瀏覽 ECR 映像，然後執行下列動作：
 - i. 針對私有儲存庫，選擇 Amazon ECR 私有儲存庫。
 - ii. 針對映像，選擇您的映像。
 - iii. 選擇如何識別映像。針對選取影像依據，選擇下列其中一個選項：
 - AWS 建議您選擇映像摘要。
 - 若要使用標籤，請選擇映像標籤，然後選擇標籤。
 - b. 如果使用私有登錄檔，請選取私有登錄檔。然後，針對 Secrets Manager ARN 或名稱，輸入您在先決條件中建立的 Secrets Manager ARN。
 - c. 針對任務執行角色和基礎設施角色，選擇角色。如果您之前從未在 Amazon ECS 中建立這些角色，您會在下拉式清單中看到建立新角色的選項。當您選取此選項時，當您按一下建立時，系統會自動為您建立角色。對於任務執行角色，許可詳細資訊會顯示連結，以取得 [AmazonECSTaskExecutionRolePolicy](#) 在 AWS 受管政策參考指南中的其他詳細資訊。對於基礎設施角色，許可詳細資訊會顯示 [AmazonECSInfrastructureRoleforExpressGatewayServices](#) 受管政策在 AWS 受管政策參考指南中的其他詳細資訊連結。

如果您偏好自訂這些角色，建立新角色按鈕會引導您前往 IAM 主控台，您可以在其中建立角色、檢視和編輯連接到角色的許可。當您返回 Amazon ECS Express 模式主控台時，請重新整理以檢視您建立的角色。

Note

第一次在 Amazon ECS 主控台中建立服務時，有些使用者可能會看到 `Invalid Parameter Exception: Unable to assume the service linked role.`。如果發生這種情況，請等待幾秒鐘，然後再試一次。

4. 選擇建立。您剛建立第一個 Amazon ECS Express Mode 服務！

後續步驟

建立第一個快速模式服務之後：

- 部署完成後，請使用提供的應用程式 URL 存取您的應用程式。
- 在 主控台的資源索引標籤中監控部署進度。部署完成時，您的服務已準備好接收流量。如需詳細資訊，請參閱[the section called “檢視快速模式服務的詳細資訊”](#)。
- 若要導覽回您的服務，請從導覽叢集中選取，如果您未在建立期間指定叢集，請選取default叢集。Amazon ECS 將服務組織為稱為叢集的邏輯分組。這是在您帳戶中組織應用程式的簡單方法。
- 如果您在部署期間遇到任何問題，請參閱 [the section called “疑難排解”](#)。

自訂

您也可以選擇自訂快速模式服務的組態。開啟標題為其他組態 - 選用的區段。

1. 如果有您要部署服務的特定 Amazon ECS 叢集，請從下拉式清單中選取叢集。如果未指定叢集，則快速模式將使用Default叢集。如果您離開快速模式主控台來建立叢集，請使用重新整理按鈕來填入下拉式清單。
2. 輸入服務的名稱。如果未指定名稱，快速模式會從容器映像名稱產生名稱。此服務名稱將用於多個資源，包括應用程式 URL、Amazon ECS Service、Amazon ECS 任務定義、憑證、擴展目標和擴展政策。
3. 選擇性地指定容器的詳細資訊：
 - a. 針對容器連接埠，輸入應用程式接聽的連接埠（預設為 80）。
 - b. 針對運作狀態檢查路徑，輸入運作狀態檢查的路徑（例如 `"/health"`）。預設為 `"/".`

- c. 在環境變數下，為您的應用程式需要的環境變數新增索引鍵/值對。針對金鑰，輸入環境變數名稱。針對值類型，選擇環境變數或秘密。針對值或值來源，輸入值或參考。選擇新增環境變數，視需要新增更多變數。未提供預設值。
 - d. 在命令中，選擇性地輸入自訂命令來覆寫 Docker CMD 指令。未提供預設值。
 - e. 針對任務角色，選擇將許可授予應用程式的 IAM 角色。未提供預設值。
4. 選擇性地指定有關運算和自動擴展的詳細資訊：
- a. 針對 CPU，選擇任務的 vCPU 配置（預設為 1 個 vCPU）。
 - b. 針對記憶體，選擇任務的記憶體配置（預設值為 2 GB）。
 - c. 對於 ECS 服務指標，選擇要擴展的指標（預設為平均 CPU 使用率）。
 - d. 針對目標值，選擇擴展的目標百分比（預設值為 60。）
 - e. 針對任務數量下限和任務數量上限，設定擴展限制。（預設值為 1 和 20）
5. 勾選方塊以自訂聯網組態。如果您不自訂這些組態，快速模式將使用預設 VPC。
- a. 選取 VPC 以協助快速導覽至您的服務將在其中執行的子網路。或者，離開 Amazon ECS 主控台以建立新的 VPC，然後返回重新整理並在下拉式清單中找到該 VPC。
 - b. 針對安全群組，選擇或建立安全群組，以允許對您的服務進行其他傳入網路存取。
6. 在日誌下：
- a. 針對 Amazon CloudWatch 日誌群組，輸入應用程式日誌的偏好日誌群組名稱。預設日誌群組會根據您的叢集和服務名稱命名。
 - b. 對於 Amazon CloudWatch 日誌串流字首，輸入日誌串流的偏好字首。預設串流字首為 `ecs/Main/`。
7. 在標籤下，新增索引鍵/值對來標記您的資源。對於 Key (鍵)，輸入標籤鍵名稱。對於 Value (值)，輸入標籤值。選擇新增項目，視需要新增更多標籤。標籤只能在建立時新增。

使用 建立您的第一個快速模式服務 AWS CLI

本教學課程說明如何使用 建立快速模式服務應用程式 AWS CLI。您將部署簡單的 Web 應用程式，以示範快速模式服務的核心功能。

先決條件

開始前，請確保您具備以下條件：

- 具有適當許可 AWS 的帳戶
- AWS CLI 已安裝和設定的
- 存放在 Amazon ECR 或私有登錄檔中的容器映像

步驟 1：建立 IAM 角色

快速模式服務需要兩個 IAM 角色。任務執行角色允許 Amazon ECS 代您提取容器映像和寫入日誌。使用下列政策建立任務執行角色和基礎設施角色：

```
#Create the roles with ECS trust policies
aws iam create-role --role-name ecsTaskExecutionRole \
  --assume-role-policy-document '{
    "Version": "2012-10-17",
    "Statement": [
      {
        "Effect": "Allow",
        "Principal": {
          "Service": "ecs-tasks.amazonaws.com"
        },
        "Action": "sts:AssumeRole",
      }
    ]
  }'

aws iam create-role --role-name ecsInfrastructureRoleForExpressServices \
  --assume-role-policy-document '{
    "Version": "2012-10-17",
    "Statement": [
      {
        "Sid": "AllowAccessInfrastructureForECSExpressServices",
        "Effect": "Allow",
        "Principal": {
          "Service": "ecs.amazonaws.com"
        },
        "Action": "sts:AssumeRole"
      }
    ]
  }'
```

```
#Attach the AWS managed policies
aws iam attach-role-policy --role-name ecsTaskExecutionRole \
```



```
--policy-arn arn:aws:iam::aws:policy/service-role/AmazonECSTaskExecutionRolePolicy
aws iam attach-role-policy --role-name ecsInfrastructureRoleForExpressServices \
--policy-arn arn:aws:iam::aws:policy/service-role/
AmazonECSInfrastructureRoleforExpressGatewayServices
```

如需詳細資訊，請參閱[the section called “任務執行 IAM 角色”](#)。

步驟 2：建立您的第一個快速模式服務應用程式

使用最低必要參數建立快速模式服務應用程式：

```
aws ecs create-express-gateway-service \
--primary-container "image"="public.ecr.aws/nginx/nginx:latest" \
--execution-role-arn arn:aws:iam::123456789012:role/ecsTaskExecutionRole \
--infrastructure-role-arn arn:aws:iam::123456789012:role/
ecsInfrastructureRoleForExpressServices \
--monitor-resources
```

此命令會使用下列項目建立快速模式服務應用程式：

- 從映像名稱產生的唯一服務名稱
- 1 個 vCPU 和 2 GB 記憶體
- HTTPS 流量的連接埠 80
- 根據 CPU 使用率自動擴展
- 預設 VPC 和公有子網路中的面向網際網路的 Application Load Balancer
- 此服務唯一的 URL，格式為 `servicename.ecs.region.on.aws`

命令會持續監控佈建的資源，並傳回狀態。服務部署完成後，應用程式 URL 已準備好接收流量。當您中斷監控時，會傳回所建立服務的相關資訊，包括服務 ARN 和 URL：

```
{
  "service": {
    "cluster": "arn:aws:ecs:region:123456789012:cluster/default",
    "serviceName": "nginx-1234",
    "serviceArn": "arn:aws:ecs:region:123456789012:service/default/nginx-1234",
    "infrastructureRoleArn": "arn:aws:iam::123456789012:role/
ecsInfrastructureRoleForExpressServices",
    "status": {
      "statusCode": "ACTIVE"
    }
  }
}
```

```
    },  
    "createdAt": "UNIXTIMESTAMP"  
  }  
}
```

步驟 3：使用自訂設定建立快速模式服務應用程式

您可以指定其他參數來自訂快速模式服務應用程式：

```
aws ecs create-express-gateway-service \  
  --execution-role-arn arn:aws:iam::123456789012:role/ecsTaskExecutionRole \  
  --infrastructure-role-arn arn:aws:iam::123456789012:role/  
ecsInfrastructureRoleForExpressServices \  
  --primary-container '{"image":"123456789012.dkr.ecr.region.amazonaws.com/my-  
app:latest","containerPort":8080,"environment":[{"name":"ENV","value":"production"},  
{"name":"DEBUG","value":"false"}]}' \  
  --service-name "my-web-app" \  
  --cpu 2 \  
  --memory 4 \  
  --health-check-path "/health" \  
  --scaling-target '{"minTaskCount":3,"maxTaskCount":100}' \  
  --monitor-resources
```

這會建立具有下列項目的應用程式：

- 自訂名稱 "my-web-app"
- 2 個 vCPU 和 4 GB 記憶體
- 應用程式流量的連接埠 8080
- 自訂運作狀態檢查端點
- 環境變數
- 最少 3 個任務，最多 100 個任務用於自動擴展

步驟 4：監控您的部署

`--monitor-resources` 旗標適用於對快速模式服務的任何建立、更新或刪除呼叫。但是，您也可以隨時監控服務中的資源，與變動動作分開。部署時間可能會因需要佈建的資源而有所不同。狀態變更為後ACTIVE，您的應用程式就已準備好接收流量。

```
aws ecs monitor-express-gateway-service --service-arn
arn:aws:ecs:region:123456789012:service/app-23d97h88
```

您也可以找到快速模式服務應用程式的目前組態和狀態：

```
aws ecs describe-express-gateway-service --service-arn
arn:aws:ecs:region:123456789012:service/app-23d97h88
```

步驟 5：存取您的應用程式

使用提供的 URL 在啟用時存取快速模式服務應用程式。URL 格式為：

```
https://<service-name>.ecs.<region>.on.aws/
```

例如：

```
https://app-23d97h88.ecs.us-west-2.on.aws/
```

您的應用程式現在正在執行：

- 自動 SSL/TLS 終止
- 跨多個可用區域的負載平衡
- 根據 CPU 使用率自動擴展
- CloudWatch 記錄和監控
- 用於未來更新的 5XX 轉返警示和 Canary 部署

Amazon ECS Express Mode 服務建立的資源

快速模式服務透過提供合理的預設值和自動化支援服務的組態，降低部署容器化應用程式的複雜性 AWS。快速模式服務不需要跨多個服務管理數百個組態參數，只需容器映像、任務執行角色和基礎設施角色即可開始使用。

快速模式服務已簡化佈建和設定 Amazon ECS 服務的程序，將偏好的基礎設施綁定在一起，以讓生產 Web 應用程式執行。快速模式服務會自動整合操作和組態最佳實務。系統會建立下列資源：

- 具有 Fargate 容量提供者的 ECS 預設叢集（如果尚未存在）
- 具有容器、記錄和聯網組態的任務定義

- 具有 Canary 部署和自動擴展的服務
- Application Load Balancer 搭配 HTTPS 接聽程式、接聽程式規則和目標群組
- 具有最少必要輸入的安全群組
- 用於自動擴展和負載平衡的服務連結角色
- Application Auto Scaling 可擴展的目標和目標追蹤擴展政策。

擴展政策會視需要新增或移除服務任務，讓指標保持在或接近指定的目標值。除了將指標保持接近目標值之外，目標追蹤擴展政策也會根據由於負載模式波動而導致的指標波動進行調整，並將服務中執行的任務數量快速波動降至最低。

- 服務特定的日誌群組
- 偵測故障部署的指標警示
- ACM 憑證

使用預設值開始使用快速模式服務，或使用一組最少的參數來設定您的應用程式。

叢集預設值

下列預設值可從叢集的快速模式內設定：

- `clusterName`：使用default叢集。

您可以使用 主控台 中的其他組態選項，或指定 來自訂此選項 `create-express-gateway-service --cluster`。

下列預設值是由快速模式設定，但可以直接在叢集中更新。

- `capacityProviders`：【「FARGATE」】 - 使用 Fargate

任務定義預設值

下列預設值可從任務定義的快速模式內設定：

- `cpu`：配置給任務的 1024 - 1 個 vCPU 單位

您可以使用 主控台 中的其他組態選項，或執行 來自訂此選項 `create-express-gateway-service --cpu`。

- 記憶體：2048 - 配置給任務的 2 GB 記憶體

您可以使用 主控台 中的其他組態選項，或執行 來自訂此選項 `create-express-gateway-service --memory`。

下列預設值是由快速模式設定，但可以直接在任務定義中更新。

- `networkMode`："awsvpc" - 使用awsvpc網路模式
- `operatingSystemFamily`："LINUX" - 在 Linux 作業系統上執行
- `cpuArchitecture`："X86_64" - 使用 x86_64 處理器架構
- `requiresCompatibilities`：【「FARGATE」】 - 相容於 Fargate 啟動類型
- `platformVersion`："LATEST" - 使用最新的 Fargate 平台版本

容器定義預設值

下列預設值可從容器的快速模式內設定：

- 連接埠：80 - 容器流量的預設連接埠

下列預設值是由快速模式設定，但可以直接在任務定義 - 容器定義中更新：

- 基本：true - 容器標記為對任務運作狀態至關重要
- 通訊協定："tcp" - 負載平衡器通訊的預設通訊協定
- 名稱：「主要」 - 主要容器的預設名稱

變更預設容器的名稱可能會影響快速模式對服務進行後續更新的能力。如果您想要繼續使用快速模式 主控台 或 APIs，則不建議這麼做

- `versionConsistency`：已啟用 - 解析影像標籤以摘要 IDs 以進行一致部署
- `initProcessEnabled`：true - 啟用 Docker init 程序支援。
- `stopTimeout`：30 秒 - SIGTERM 和 SIGKILL 訊號之間的時間。

記錄預設值

下列預設值可從 CloudWatch Logs 的快速模式內設定：

- `awslogs-group`："/aws/ecs/<cluster>/<name>-#####" - 從叢集和服務名稱衍生的日誌群組名稱

您可以使用 主控台 中的其他組態選項，或執行 來自訂此選項 `create-express-gateway-service --primary-container "awsLogsConfiguration"=[{"logGroup"="/my/logGroup/"}]`。

- `awslogs-stream-prefix` : "ecs" - 日誌串流名稱的字首

您可以使用 主控台 中的其他組態選項，或執行 來自訂此選項 `create-express-gateway-service --primary-container "awsLogsConfiguration"=[{"logStreamPrefix"="myprefix"}]`。

下列預設值是由快速模式設定，但可以直接在任務定義 - 日誌組態中更新：

- `logDriver` : "awslogs" - 使用 CloudWatch Logs
- `awslogs-region` : 目前區域 - 使用與快速模式服務相同的區域
- 模式 : 「非封鎖」 - 記錄模式設定為非封鎖
- `max-buffer-size` : "25m" - 日誌寫入的緩衝區大小上限

服務預設值

下列預設值可從 Amazon ECS Service 的快速模式內設定：

- `serviceName` : 快速模式服務 → 名稱 - 服務的名稱 (客戶提供或衍生自映像名稱)

名稱只能在建立時設定，無法在更新時設定。

- 叢集 : 快速模式服務 → 叢集 - 此服務的 ECS 叢集 (客戶提供或預設)

叢集只能在建立時設定，無法在更新時設定。

- `desiredCount` : 快速模式服務 → `MinTasks` - 服務所需的計數 (預設為 1)

您可以使用 主控台 中的其他組態選項，或執行 來自訂此選項 `create-express-gateway-service --scaling-target '{"minTaskCount":3}'`，

- 標籤 : 用於 AWS 標記的快速模式服務標籤 (客戶提供)

您可以使用 主控台 中的其他組態選項，或執行 來自訂此選項 `create-express-gateway-service --tags`。標籤只能在建立新資源時新增。

下列預設值是由快速模式設定，但可以直接在 Amazon ECS Service 中更新：

- `availabilityZoneRebalancing` : `true` - 啟用自動 AZ 重新平衡
- `CapacityProviderStrategy` : `{"base" : 1, "capacityProvider" : "FARGATE", "weight" : 1}` - 將 Fargate 用於運算容量
- `deploymentConfiguration` : Canary 預設 - Express Mode 服務使用 Canary 部署

請注意，無法在快速模式服務上更新部署策略。

- `enableECSTags` : `true` - 啟用 ECS 受管資源標記
- `healthCheckGracePeriodSeconds` : `300` - 排程器查看 ELB 或 Lattice 運作狀態檢查之前的寬限期 (符合 ELB 運作狀態檢查寬限期預設值)
- `launchType` : `<Not set>` - 使用容量提供者
- `loadBalancers` : ECS 處理的負載平衡器組態

請注意，無法在快速模式服務上更新負載平衡器組態。

- `placementStrategy` : 未設定 - 啟動時使用 Fargate (應一律為 AZ 分散)
- `platformVersion` : `LATEST` - Fargate 平台版本 (硬式編碼為 `LATEST`)
- `propagateTags` : `"SERVICE"` - 將標籤從服務傳播到任務
- `schedulingStrategy` : `"REPLICA"` - 快速模式服務是 REPLICA 服務
- `taskDefinition` : 已建立快速模式服務 - 要使用的任務定義 (由快速模式服務提供)

網路組態預設值

下列預設值可從 Amazon ECS Service - 網路組態的快速模式內設定：

- `networkConfiguration.Subnets`: 如果未提供，則快速模式將使用預設 VPC 中的預設公有子網路。

預設 VPC 必須具有至少兩個公有子網路，在至少兩個可用區域中，每個子網路每個指派的 CIDR 區塊至少有 8 個可用的 IPs。

如果您提供自訂公有子網路，快速模式會為您的任務佈建面向網際網路的 ALB，並開啟 `assignPublicIP`。如果您提供私有子網路 (路由表中沒有網際網路閘道的子網路)，快速模式會佈建內部 ALB。

如果您指定子網路，VPC 的第一個快速模式服務會定義與該 VPC 面向網際網路或內部負載平衡器相關聯的子網路。在相同 VPC 中啟動的後續快速模式服務必須具有符合負載平衡器所支援可用區域的子網路。因此，我們建議您使用來自所有可用區域的子網路建立快速模式服務。

- `networkConfiguration.SecurityGroups`: 如果未提供，則快速模式會建立服務安全群組和 Load Balancer 安全群組

服務安全群組允許公有網際網路的傳出流量，但視您網路的其他層面而定，例如子網路組態，這可能會受到限制。

Load Balancer 安全群組 Load Balancer 安全群組允許 HTTPS 解析的傳入流量，以及在您指定的容器連接埠上傳出流量至服務安全群組（預設 80）。且會自動更新，以符合快速模式建立或更新呼叫中提供的容器連接埠。

您可以使用主控台的其他組態選項，或執行來自訂此選項 `create-express-gateway-service --network-configuration '{"securityGroup": ["sg-xxxxxxx"]}'`。當您提供安全群組時，您會為服務提供額外的輸入路徑。

下列預設值是由快速模式設定，但可以直接在 Amazon ECS 服務 - 網路組態中更新：

- `assignPublicIp`：根據子網路類型 - 針對公有子網路啟用，以確保預設 VPC 的使用者能夠存取公有網際網路，我們預設會為每個任務啟用公有 IPs。如果您提供私有子網路，且負責在任務需要網際網路存取時設定 NAT 閘道，則會停用此功能。

IAM 角色預設值

下列 IAM 角色可針對 Express Mode Service 設定，並適時自動建立服務連結角色。

- `executionRoleArn`：任務執行角色（必要參數）

當所需的許可位於受管政策中時，可以連接其他政策和許可。《AWS 受管政策參考指南》中的 [AmazonECSTaskExecutionRolePolicy](#) 的其他詳細資訊

- `infrastructureRoleArn`：Express Gateway Services 的基礎設施角色（必要參數）

當所需的許可位於受管政策中時，可以連接其他政策和許可。《受管政策參考指南》中的 [AmazonECSInfrastructureRoleforExpressGatewayServices](#) AWS 受管政策的其他詳細資訊

- `taskRoleArn`：從容器應用程式碼呼叫的任務角色（選用）

您可以使用主控台的其他組態選項，或執行來自訂此選項 `create-express-gateway-service --task-role-arn`。

您必須先建立新的 IAM 角色，才能建立或更新快速模式服務。

下列 IAM 角色是由快速模式自動設定。

- `ecsServiceRoleForECS`：允許 Amazon ECS 管理您的叢集
- `AWSServiceRoleForElasticLoadBalancing`：代表您呼叫其他 AWS 服務 - 由 Elastic Load Balancing 自動建立
- `AWSServiceRoleForApplicationAutoScaling_ECSService`：代表您呼叫其他 AWS 服務 - 由 Application Auto Scaling 自動建立

Application Auto Scaling 預設值

下列 Application Auto Scaling 設定可在快速模式中設定：

- `autoScalingTargetValue`：60 - 擴展的目標 CPU 使用率百分比
- `autoScalingMetric`：「CPUUtilization」 - 用於自動擴展決策的指標
- `desiredMinTaskCount`：1 - 要維護的最低任務數量
- `desiredMaxTaskCount`：20 - 要擴展到的任務數量上限（可以增加或減少）

下列設定可以直接在 Application Auto Scaling 中混淆：

- `policyType`：「TargetTrackingScaling」 - 使用目標追蹤擴展政策
- `disableScaleIn`：false - 允許在需求減少時縮減規模

應用程式擴展目標預設值

Application Auto Scaling 目標定義自動擴展操作的可擴展資源和容量限制。您可以在快速模式中修改下列項目：

- `MaxCapacity`：快速模式服務 → `MaxTasks` - 服務擴展到的任務上限 (`CreateLoadBalancedService` 或客戶提供的預設值)
- `MinCapacity`：快速模式服務 → `MinTasks` - 服務將縮減至的最低任務 (`CreateLoadBalancedService` 或客戶提供的預設值)
- `ScalableDimension`：`ecs:service:DesiredCount` - 要擴展的屬性
- 標籤：衍生自 `CreateLoadBalancedService` 中傳遞的標籤 - 此資源的資源標籤

您可以在 Application Auto Scaling 可擴展目標中修改下列預設值：

- `DynamicScalingInSuspended` : `false` - 指出是否已暫停向內擴展
- `DynamicScalingOutSuspended` : `false` - 指出是否暫停向外擴展
- `ScheduledScalingSuspended` : `false` - 指出排程擴展是否暫停

應用程式擴展政策預設值

擴展政策實作目標追蹤演算法，監控 CPU 使用率並調整任務計數，以維持最佳效能。您可以從快速模式修改下列項目：

- `TargetValue` : `60%` - 追蹤政策的目標值（快速模式服務 → `scaling-target-value`）
- `PredefinedMetricType` : `ECSServiceAverageCPUUtilization` - 要追蹤的指標名稱類型（快速模式服務 → 擴展指標）

快速模式提供平均 `CPUUtilization`、平均記憶體使用率和每個目標請求計數的擴展指標。每個目標的請求計數來自 Application Load Balancer，每秒每個目標最多可設定 65536 個請求，這是 Application Load Balancer 目標群組中的 Amazon ECS Service。

下列預設值是由快速模式設定，但可以在 Application Auto Scaling 政策中變更：

- `PolicyName` : `<ServiceName><"TargetMetric">` - 此擴展政策的名稱（衍生自 ECS 服務名稱和目標指標）
- `PolicyType` : `TargetTrackingScaling` - 正在完成的擴展類型（預先定義的預設值）
- `ScalableDimension` : `"ecs : service : DesiredCount"` - 要擴展的屬性（預先定義的預設值）
- `DisableScaleIn` : `false` - 指出縮減是否已停用

Application Load Balancer 預設

下列 Application Load Balancer 預設值可在快速模式中設定：

- 機制：取決於子網路 - 在提供公有子網路時，使用公有 IP 地址建立面向網際網路的負載平衡器。在提供私有子網路時，使用私有 IP 地址建立內部負載平衡器。
- `ip-address-type`：取決於子網路 - 當地址為 IPv4 時，建立僅限 IPv4 的 Application Load Balancer。有 IPv6-enabled 的子網路時，將會建立雙堆疊 Application Load Balancer。請注意，如果您有 IPv6-enabled 的子網路，VPC 中的第一個快速模式服務會定義該 VPC 的內部或面向網際網路的負載平衡器。因此，我們建議您先建立 IPv6-only 的子網路，或在新的 VPC 中建立。

下列預設值設定為快速模式，並且可以在 Application Load Balancer 中設定：

- `desync-mitigation-mode`：關閉 - HTTP 取消同步緩解已停用
- `access-logs.enabled`：false - 停用存取記錄
- `listener-configurations.protocol`：https - 使用 HTTPS 通訊協定進行安全通訊
- `listener-configurations.port`：443 - 在標準 HTTPS 連接埠上接聽
- `listener-configurations.rule-type`：host-header - 根據主機標頭規則路由流量

目標群組預設值

下列 Amazon EC2 目標群組預設值可在快速模式中設定：

- `health-check-path`：(預設 "/") Express Mode 服務運作狀態檢查路徑 - 運作狀態檢查請求的 URL 路徑
- `連接埠`：(預設 80) - 目標接收流量的連接埠
- `health-check-port`：(預設值 80) - 運作狀態檢查請求的連接埠

下列預設值是由快速模式設定，並且可以在 Amazon EC2 目標群組中設定：

- `通訊協定`：HTTP - 將流量路由到目標的通訊協定
- `protocol-version`：HTTP1 - 用於通訊的 HTTP 通訊協定版本
- `vpc-id`：預設是目標群組的預設 VPC - Virtual Private Cloud 識別符，但會從提供的子網路衍生
- `health-check-protocol`：與通訊協定相同 - 運作狀態檢查請求的通訊協定
- `health-check-enabled`：一律啟用 - 運作狀態檢查會自動啟用
- `health-check-interval-seconds`：30 - 個別目標運作狀態檢查之間的時間
- `health-check-timeout-seconds`：5 - 運作狀態檢查回應的逾時持續時間
- `healthy-threshold-count`：5 - 持續成功檢查運作狀態良好所需的運作狀態
- `unhealthy-threshold-count`：2 - 持續檢查運作狀態不佳所需的運作狀態失敗
- `target-type`：ip - 依 IP 地址註冊目標
- `ip-address-type`：ipv4 - 使用 IPv4 地址進行目標註冊

資源共用和成本最佳化

快速模式服務會盡可能自動共用資源，以最佳化成本：

- 負載平衡器共用 - 相同 VPC 中最多 25 個快速模式服務可以共用 Application Load Balancer。快速模式會根據您佈建的快速模式服務數量，視需要佈建額外的 Application Load Balancer。此外，當您減少 VPC 中的快速模式服務數量時，快速模式也會取消佈建未使用的 Application Load Balancer。當您部署更多快速模式服務時，此共用會降低每個應用程式的有效成本。
- 叢集共用 - 快速模式服務可以在 Amazon ECS 叢集中分組在一起。快速模式服務也可以與非由快速模式管理的 Amazon ECS 服務共用 Amazon ECS 叢集。

檢視 Amazon ECS Express Mode 服務的詳細資訊

您可以檢視快速模式服務的詳細資訊，以監控其狀態、組態和效能指標。服務詳細資訊頁面提供有關服務目前狀態、執行中的任務、部署歷史記錄和操作指標的完整資訊。

檢視快速模式服務 (Amazon ECS 主控台)

1. 開啟主控台，網址為 <https://console.aws.amazon.com/ecs/v2>。
2. 在導覽窗格中，選擇叢集。
3. 在叢集頁面上，選擇包含快速模式服務的叢集名稱。
4. 在叢集詳細資訊頁面上，選擇服務索引標籤。
5. 設定篩選條件以檢視您的快速模式服務。針對篩選資源管理類型，選擇 ECS。

Express Mode 服務的名稱旁有一個 Express 徽章。

6. 選取快速模式服務的名稱，以導覽至快速模式服務的服務詳細資訊頁面。
7. 檢閱服務詳細資訊頁面上顯示的服務資訊：
 - a. 快速服務概觀 - 檢視基本服務資訊，包括應用程式 URL、服務狀態、擱置和執行中的任務，以及作用中部署。
 - b. 可觀測性 - 檢視 CPU 使用率（最小值、最大值、平均值）記憶體使用率（最小值、最大值、平均值）、目標回應時間 (p50、p90、p99)、RequestCount、4xx 狀態和 5xx 狀態的 CloudWatch 指標。
 - c. 日誌 - 檢視任務中容器的日誌。
 - d. 資源 - 為快速模式服務建立 AWS 的資源。時間軸檢視會在建立或更新後自動更新，並顯示狀態和事件。資料表檢視提供可搜尋的資料表，其中包含每個資源的狀態和完整 ARNs。
 - e. 部署 - 追蹤部署歷史記錄和目前的部署狀態，包括在更新期間提供寶貴資訊的服務修訂和任務。事件面板可讓您知道快速模式服務何時達到穩定狀態。

- f. 組態 - 檢視服務組態，包括叢集、角色、容器組態、運算、自動擴展和聯網組態。
- g. 標籤 - 管理您的標籤。

更新 Amazon ECS Express Mode 服務

Express Mode 服務透過提供聯網、負載平衡和 Application Auto Scaling 的可設定選項，以及協調這些服務的變更，來簡化服務更新程序。

您可以更新快速模式服務來修改容器映像、調整資源配置或變更組態設定。使用 Canary 部署搭配警示型復原警示來部署更新，以維持可用性。

部署行為

根據預設，快速模式服務會使用 Canary 部署，以確保安全更新和快速復原：

- 建立新的環境，並隨您的變更部署任務
- 5% 的流量會轉移到新的環境
- 如果 3 分鐘內 2 個資料點的 4xx 和 5xx 錯誤百分比總和 > 1，則會觸發警示型轉返
- 運作狀態檢查會驗證新任務是否正常運作
- 3 分鐘後，100% 的流量會轉移到新環境
- 在 3 分鐘的製作時間進行監控後，舊任務會逐漸停止並取代

如需詳細資訊，請參閱 [Amazon ECS Canary 部署](#)

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，選擇包含快速模式服務的叢集名稱。
4. 在叢集詳細資訊頁面上，選擇服務索引標籤。
5. 設定篩選條件以檢視您的快速模式服務。針對篩選資源管理類型，選擇 ECS。

快速模式服務的名稱旁有一個快速徽章。

6. 在組態下：

- a. 指定要用於應用程式的映像。針對映像 URI，輸入映像的 URI。若要瀏覽 Amazon ECR 映像，請選擇瀏覽 ECR 映像，然後執行下列動作：
 - i. 針對私有儲存庫，選擇 Amazon ECR 私有儲存庫。
 - ii. 針對映像，選擇您的映像。
 - iii. 選擇如何識別影像。針對選取影像依據，選擇下列其中一個選項：
 - AWS 建議您選擇映像摘要。
 - 若要使用標籤，請選擇映像標籤，然後選擇標籤。
 - b. 若要使用私有登錄檔，請選取私有登錄檔。然後，針對 Secrets Manager ARN 或名稱，輸入您在先決條件中建立的 Secrets Manager ARN。
 - c. 針對任務執行角色，選擇角色或建立新的角色並重新整理。當您需要新增其他許可時，可以更新任務執行角色。
7. 在其他組態下，自訂您的服務。
- a. 在容器下：
 - i. 針對容器連接埠，更新應用程式接聽的連接埠（預設為 80）。
 - ii. 對於運作狀態檢查路徑，請更新運作狀態檢查的路徑（例如 /health）。
 - b. 在環境變數下，為您的應用程式需要的環境變數新增索引鍵/值對：
 - i. 針對金鑰，輸入環境變數名稱。
 - ii. 針對值類型，選擇環境變數或秘密。
 - iii. 針對 的值或值，輸入值或參考。
 - iv. 選擇新增環境變數，視需要新增更多變數。
 - c. 在命令中，選擇性地輸入自訂命令來覆寫 Docker CMD 指令。
 - d. 針對任務角色，新增 IAM 角色，將許可授予在容器中執行的應用程式。這可讓您的應用程式對 AWS 服務進行 API 呼叫。
 - e. 在運算下：
 - i. 對於 CPU，請更新任務的 vCPU 配置（例如 1 個 vCPU）。
 - ii. 對於記憶體，請更新任務的記憶體配置（例如 2 GB）。
 - f. 在 Auto Scaling 下：

- i. 對於 ECS 服務指標，選擇要擴展的指標（例如，ECS 服務平均記憶體使用率或每個目標的請求計數）。
 - ii. 針對目標值，輸入擴展的目標（例如 60 或 1000）。
 - iii. 對於任務數量下限和任務數量上限，請更新擴展限制。
 - g. 在日誌下：
 - i. 對於 Amazon CloudWatch 日誌群組，請更新應用程式日誌的日誌群組名稱。請注意，這不會移動現有的日誌，而是開始從新的服務修訂版寫入日誌。
 - ii. 針對 Amazon CloudWatch 日誌串流字首，輸入日誌串流的新字首。
8. 選擇更新以更新您的快速模式服務。

刪除 Amazon ECS Express Mode 服務

當您刪除快速模式服務時，您可以移除服務並刪除其相關聯的資源，包括叢集、負載平衡器和其他基礎設施元件。

Important

刪除快速模式服務將停止所有執行中的任務，並使您的應用程式無法使用。這個操作無法復原。

根據您的選擇，可能會刪除下列資源：

- Amazon ECS 叢集（如果沒有其他服務正在執行）
- Amazon ECS 服務、任務定義和任何執行中的任務
- 服務安全群組
- CloudWatch 日誌群組
- 指標警示
- ACM 憑證
- Application Load Balancer（如果未設定其他服務）、目標群組、安全群組、接聽程式和接聽程式規則
- Auto Scaling 政策，可擴展的目標

請考慮下列事項：

- 在刪除之前，請確定您已備份任何重要資料
- 在刪除之前考慮將服務擴展到零任務，以正常停止流量
- 檢閱可能受刪除影響的相依性

刪除快速模式服務和 服務之間的差異

當您刪除服務時，如果沒有執行中的任務，Amazon ECS 會刪除服務。Amazon ECS 不會刪除與服務或預設叢集相關聯的目標群組（如果已使用）。當您刪除快速模式服務時，Amazon ECS 會刪除該服務以及任何與快速模式服務不同的資源。例如，如果快速模式服務建立負載平衡器，並且是唯一使用負載平衡器的服務，則會刪除相關聯的負載平衡器。如果正在共用負載平衡器，即使資源不是由快速模式管理，也不會將其刪除。

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. 選擇 Services (服務) 索引標籤。
5. 選取要刪除的服務。
6. 選擇 刪除。
7. 在刪除確認對話方塊中：
 - 輸入 **delete** 以確認刪除。
8. 選擇刪除以確認刪除。
9. 停留在頁面上以監控刪除進度，並檢視已移除的資源清單，以及已保留且可能需要手動清除的資源清單。

Amazon ECS Express Mode 服務的最佳實務

了解在生產環境中有效使用快速模式服務的最佳實務和建議。

安全最佳實務

秘密管理

- 將 Secrets Manager 用於秘密 - 將敏感資料儲存在 Secrets Manager 中（例如，私有儲存庫或資料庫登入資料）。

如需 Secrets Manager 最佳實務的詳細資訊，請參閱 [《Secrets Manager 使用者指南》中的 Secrets Manager 最佳實務](#)

- 啟用靜態加密 - 確保秘密在存放在 AWS 服務中時加密。

使用 Secrets Manager 等服務可讓您使用 AWS 受管金鑰或客戶提供的金鑰進行加密。

- 實作秘密輪換 - 對資料庫密碼和 API 金鑰使用自動輪換。

使用 Secrets Manager 之類的服務可以管理 Amazon Aurora 和 Amazon RDS 等服務的 secret 輪換

在快速模式服務中使用秘密的範例：

```
aws ecs update-express-gateway-service \
  --primary-container \
    '{ "environment"=[{ "name"="DB_PASSWORD", "value"="arn:aws:secretsmanager:us-
west-2:123456789012:secret:prod/db/password"}, \
    { "name"="API_KEY", "value"="arn:aws:ssm:us-west-2:123456789012:parameter/prod/
api-key"}]}' \
```

網路安全

- 針對敏感應用程式使用私有子網路 - 在私有子網路中部署不需要直接網際網路存取的應用程式。

如需建議架構的詳細資訊，請參閱 [將 Amazon ECS 應用程式連線至網際網路](#)。

- 將安全群組設定為最低寬鬆 - 將傳入和傳出流量限制為僅限必要的連接埠和來源。

若要限制快速模式服務安全群組的傳出流量，您可以修改傳出規則，或使用下列命令，直接在 Amazon EC2 安全群組主控台中進行編輯：

```
aws ec2 authorize-security-group-egress
```

```
--group-id sg-xxxxxxx \
--protocol tcp \
--port 443 \
--cidr 0.0.0.0/0

aws ec2 revoke-security-group-egress
--group-id sg-xxxxxxx \
--protocol tcp \
--port 443 \
--cidr 0.0.0.0/0
```

- 啟用 Amazon VPC 流程日誌 - 監控網路流量以進行安全性分析和疑難排解。

您可以在 VPC 子網路主控台中您的快速模式應用程式所使用的每個子網路中啟用此功能，或使用 `aws ec2 create-flow-logs --resource-ids subnet-xxx`

- AWS WAF 用於 Web 應用程式 - 防止常見的 Web 入侵和攻擊。

您可以透過建立 Web ACL 並將其與快速模式服務所使用的 Application Load Balancer 建立關聯來啟用此功能。在主控台中，在 WAF & Shield Service 中建立 Web ACL，並與您的 Application Load Balancer 建立關聯。或者，使用 `aws wafv2 create-web-acl` 和 `aws wafv2 associate-web-acl --resource-arn <alb>`。

效能和運算最佳化

資源大小

- 適當大小的 CPU 和記憶體 - 監控應用程式效能，並根據實際使用模式調整 CPU 和記憶體配置。

AWS Compute Optimizer 會針對 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 ?](#)。

- 效能測試您的應用程式 - 為了確保您的應用程式以指定的擴展閾值和資源配置大規模運作，請執行負載測試。

自動擴展組態

- 設定適當的擴展閾值 - 設定在效能降低之前觸發擴展的 CPU 或記憶體閾值。

您可以在快速模式服務主控台中修改服務指標的目標值。

請考慮新增預測擴展政策，特別是如果您的流量遵循以時間為基礎的模式時。如需詳細資訊，請參閱[預測 Auto Scaling](#)。

- 使用多個擴展指標 - 考慮同時使用 CPU 或記憶體和以請求為基礎的擴展，以實現更靈敏的擴展。

您可以將多個政策新增至服務。根據預設，快速模式會新增一個政策，但您可以直接將附加政策連接到您的服務。

- 設定任務的下限和上限 - 設定合理的界限來控制成本並確保可用性。

對於生產工作負載，一旦初始測試完成，我們建議在三個可用區域中執行，以遵循可用性最佳實務。您可以在快速模式主控台中或使用 `更新最低任務數量` `update-express-gateway-service --scaling-target '{"minTaskCount":3}'`。

運作狀態檢查

- 實作有意義的運作狀態檢查 - 建立運作狀態檢查端點，以驗證關鍵應用程式相依性。

您可以在快速模式主控台中更新運作狀態檢查路徑。或使用 `update-express-gateway-service --health-check-path "/health"`。

如需為您的應用程式建立運作狀態檢查的詳細資訊，請參閱[實作運作狀態檢查](#)

- 保持輕量型運作狀態檢查 - 避免在運作狀態檢查端點中進行昂貴的操作。

範例可能包括外部 API 呼叫、CPU 或記憶體密集型選項，或使用 增強功能長時間執行操作以逾時。

- 使用適當的逾時 - 設定允許正常回應時間的運作狀態檢查逾時，同時快速偵測失敗。

您可以在 Application Load Balancer 目標群組上設定快速模式的運作狀態檢查逾時。在 Amazon EC2 主控台中，導覽至目標群組區段，然後選取您的快速模式目標群組。選取運作狀態檢查索引標籤，然後按一下進階運作狀態檢查設定下的編輯，即可調整逾時。或者，使用 `aws elbv2 modify-target-group --target-group-arn <targetgroup> --health-check-timeout`。

- 傳回適當的 HTTP 狀態碼 - 使用 200 表示狀態良好，使用 4xx/5xx 表示狀態不良。

操作最佳實務

監控和記錄

- 啟用增強型容器洞見 - 使用 CloudWatch；增強型容器洞見來全面監控您的快速模式服務應用程式。

如需詳細資訊，請參閱在 [Amazon ECS 上設定 Container Insights](#)。

- 設定自訂指標 - 將應用程式特定的指標發佈至 CloudWatch；以進行業務邏輯監控。

如需詳細資訊，請參閱 CloudWatch 使用者指南中的 [公有自訂指標](#)。

- 設定日誌保留 - 設定適當的日誌保留期間，以平衡成本和合規要求。

快速模式建立的 CloudWatch Log Groups 設定為永不過期，並在刪除快速模式服務時保留。您可以在 CloudWatch Log Group 中調整此設定。

- 建立儀表板和警示 - 設定 CloudWatch；儀表板和警示以進行主動監控。

部署策略

- 實作製作時間 - Express Mode 實作 Canary 製作時間，以確保部署有時間穩定，同時減少有問題部署的爆量半徑。如果您的應用程式需要更多時間來穩定，可以在快速模式服務的 Amazon ECS 服務定義中設定。如需詳細資訊，[請參閱建立 Amazon ECS Canary 部署](#)。
- 實作轉返程序 - 制定計畫，以便在發生問題時快速還原至先前的版本。

有意義的運作狀態檢查和警示型復原都有助於復原。快速模式的 Canary 部署策略結合 4xx 和 5xx 流量上的警示型轉返，可在應用程式程式碼或組態故障時設定您的部署以進行快速轉返。

對 Amazon ECS Express Mode 服務進行故障診斷

本節可協助您識別並解決部署和管理快速模式服務時的常見問題。

部署問題

服務卡在 ACTIVE 或 DRAINING 狀態

徵狀：DescribeServiceRevisions 顯示資源仍在佈建或取消佈建。DescribeServices 顯示部署未穩定

可能的原因和解決方案：

- IAM 許可不足 - 確認任務執行角色和基礎設施角色具有必要的許可，如其各自的受管政策所示。

```
# Check if the role has the required managed policy
aws iam list-attached-role-policies --role-name ecsTaskExecutionRole
```

- 映像提取失敗 - 確保容器映像存在且可存取。

```
# Test image pull manually
docker pull 123456789012.dkr.ecr.us-west-2.amazonaws.com/my-app:latest
```

- 網路連線問題 - 檢查子網路是否具有 AWS 服務的網際網路存取或 Amazon VPC 端點。
- 資源限制 - 確認您的帳戶有足夠的 Fargate 容量，且尚未達到服務配額。

診斷步驟：

1. 使用 DescribeExpressGatewayService 取得 DescribeServiceRevisions 追蹤的目前 ServiceRevision 修訂，以取得佈建或取消佈建的狀態
2. 檢查 Amazon ECS 主控台中的服務事件，以取得詳細的錯誤訊息。
3. 檢查容器連接埠是否已正確設定
4. 檢查 Amazon ECS 和 Fargate AWS 的服務配額。

任務啟動失敗

症狀：任務無法啟動或在啟動後立即停止。

常見原因：

- 應用程式錯誤 - 容器應用程式因組態或執行時間錯誤而結束。
- 運作狀態檢查失敗 - 應用程式不會回應預期連接埠或路徑上的運作狀態檢查。
- 資源限制 - 應用程式的 CPU 或記憶體配置不足。
- 缺少環境變數或秘密 - 應用程式無法使用必要的組態。

解決步驟：

1. 檢查 CloudWatch Logs 中的應用程式日誌，從 DescribeServiceRevisions 取得日誌群組名稱：

```
aws logs describe-log-streams --log-group-name /ecs/express-service-my-app
```

```
aws logs get-log-events --log-group-name /ecs/express-service-my-app --log-stream-name stream-name
```

2. 確認運作狀態檢查路徑傳回 HTTP 200 狀態。
3. 在本機測試容器映像，以確保正確啟動。
4. 視需要檢閱和調整 CPU 和記憶體配置。

連線問題

應用程式無法透過負載平衡器連線

徵狀：應用程式 URL 會傳回逾時或連線錯誤。

故障診斷步驟：

1. 驗證您的資源已完成佈建
2. 確認任務正在執行且運作狀態良好：

```
aws ecs describe-services --cluster my-cluster --services my-express-service
```

3. 檢查 Application Load Balancer 目標群組運作狀態：

```
aws elbv2 describe-target-health --target-group-arn  
arn:aws:elasticloadbalancing:region:account:targetgroup/name/id
```

4. 確保應用程式正在接聽容器中的正確連接埠。

效能問題

回應時間緩慢

症狀：應用程式回應比預期慢。

診斷方法：

1. 監控 CPU 和記憶體使用率：

```
# Check CloudWatch metrics for the service
aws cloudwatch get-metric-statistics \
  --namespace AWS/ECS \
  --metric-name CPUUtilization \
  --dimensions Name=ServiceName,Value=my-express-service Name=ClusterName,Value=my-cluster \
  --start-time 2024-01-01T00:00:00Z \
  --end-time 2024-01-01T01:00:00Z \
  --period 300 \
  --statistics Average
```

2. 檢閱應用程式日誌是否有錯誤或效能警告。
3. 檢查自動擴展是否適當地回應載入。
4. 分析請求分佈的負載平衡器指標。

最佳化策略：

- 如果資源受限，請增加 CPU 或記憶體配置。
- 調整自動擴展閾值以提早擴展。
- 最佳化應用程式程式碼和資料庫查詢。

自動擴展未如預期般運作

症狀：服務在高負載期間不會縱向擴展，或在低負載期間不會縱向擴展。

故障診斷步驟：

1. 檢查自動擴展政策及其組態：

```
aws application-autoscaling describe-scaling-policies \
  --service-namespace ecs \
  --resource-id service/my-cluster/my-express-service
```

2. 檢閱 CloudWatch 指標，以確保符合擴展觸發條件。

3. 確認服務具有擴展的許可（檢查 IAM 角色）。
4. 檢查任何擴展活動及其結果。

監控和偵錯工具

使用 CloudWatch Container Insights

啟用 Container Insights 進行全面監控：

```
aws ecs put-account-setting --name containerInsights --value enabled
```

Container Insights 提供：

- CPU、記憶體、磁碟和網路指標
- 效能監控儀表板
- 日誌相互關聯和分析
- 異常偵測

在快速模式之外更新資源

快速模式透過代表您自動佈建和管理 AWS 資源來簡化部署，但請務必了解責任模型。雖然快速模式會建立和設定 Amazon ECS 服務、負載平衡器和自動擴展政策等資源，但所有資源都會保留在您的 AWS 帳戶中，並可供直接管理完整存取。您可以使用 AWS 主控台 AWS CLI 或 APIs 在快速模式之外修改這些資源，但這樣做可能會影響快速模式管理服務的能力。本節說明快速模式管理哪些資源、如何在快速模式之外安全地進行變更，以及當您執行時會發生什麼情況。

了解共同責任模型

快速模式中的責任界限是由 API 用量定義。快速模式 APIs 會使用跨多個 AWS 服務的協調更新來代表您管理資源。標準 AWS 服務 APIs 可讓您直接控制個別資源。這可讓您超越快速模式為組態提供的參數集，並讓您控制每個資源的每個參數。兩組 APIs 都會在帳戶中的相同基礎設施上運作，兩者都會執行您請求的操作。AWS 負責執行您請求的操作。快速模式不會驗證使用直接 APIs 進行的資源修改是否會與快速模式組態衝突，也不會防止可能中斷服務功能的變更。這也表示您能夠進行變更，而且這些變更會持續存在。除非在快速模式更新中要求，否則快速模式不會覆寫變更。您有責任了解使用直接 APIs 修改如何與快速模式的組態互動，以及解決任何產生的衝突或服務問題。

如需快速模式設定的預設值及其協調的資源的詳細資訊，請參閱 [Amazon ECS Express Mode 服務建立的資源](#)。

在快速模式服務中利用 Amazon ECS 功能之前，是否需要執行任何動作？

否，快速模式沒有漸層路徑或破碎玻璃。整個 Amazon ECS 功能集一律可供您的快速模式服務使用。如果您想要確保服務不會發生進一步的快速模式更新，有幾個選項。透過移除資源上的受管標籤，Amazon ECS 將無法再將其識別和操作為快速模式服務。或者，您也可以使用 IAM 將使用者許可限制為快速模式 APIs。

快速模式外的範例更新

將附屬項目新增至快速模式服務

您可以將任何附屬容器新增至您的服務。此範例說明如何新增 Fluent Bit 記錄附屬項目。

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. 選取您的快速模式服務所在的叢集，然後在服務面板中，選取它。
4. 前往資源索引標籤，然後選取任務定義。
5. 選取 JSON 索引標籤，然後選取頂端按鈕中標示為建立新修訂的向下箭頭。在這裡，您會找到名為使用 JSON 建立新修訂的動作。
6. 在主要容器定義之前或之後新增附屬容器。如需詳細資訊，請參閱 [範例 Amazon ECS 任務定義：將日誌路由至 FireLens](#)。
7. 如果您有記錄附屬項目，例如在此範例中，您會想要變更主要容器中logDriver的：

```
"logConfiguration": {  
  "logDriver": "awsfirelens"  
}
```

8. 假設快速模式服務的預設設定，請將主要容器memoryReservation設定為 1998，並將 Fluent Bit 容器memoryReservation設定為 50。這可確保兩個容器的總記憶體保留不超過您在 Fargate 任務上執行的數量（預設快速模式服務為 2048）。
9. 建立新的修訂版。
10. 按一下顯示部署的下拉式清單，然後選取更新服務。

11. 選取您要更新的叢集和服務，然後選取強制新部署。應該自動選取您剛建立的任務定義修訂。選取 Update (更新)。
12. 觀看您的部署發生。此服務中的任務現在將有兩個容器。透過快速模式主控台或 APIs 進行的其他更新不會覆寫任務定義的變更，除非它們發生直接衝突 - 例如，在這種情況下，如果您為快速模式提供了新的日誌群組和日誌串流名稱，則 logDriver 會更新回 awslogs。

將自訂網域新增至您的服務

先決條件：本指南假設您的網域是由 Route 53 使用 ACM 憑證管理的託管區域。如果您的網域託管在其他地方，您將需要 CNAME 來指向 Application Load Balancer DNS 記錄。

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. 選取您的快速模式服務所在的叢集，然後在服務面板中，選取它。
4. 前往資源索引標籤，然後選取接聽程式規則。
5. 選取動作和編輯規則。
6. 複製目前的主機標頭值（您的應用程式 URL），然後移除規則，因為每種類型只能有一個。
7. 新增主機標頭類型的條件，將您的應用程式 URL 貼回 做為主機標頭條件值。
8. 按一下新增 OR 條件值。
9. 在此輸入您的自訂網域。
10. 按一下下一步並儲存變更。
11. 現在選取 Application Load Balancer 接聽程式的憑證索引標籤。
12. 按一下新增憑證。在這裡，您可以將憑證匯入接聽程式。
13. 在 Route 53 託管區域中，按一下建立記錄以建立路由記錄至您的服務。
14. 選取簡單的路由記錄。
15. 新增記錄名稱。
16. 選取記錄類型。
17. 將流量路由到別名到 Application 和 Classic Load Balancer。
18. 選擇您的區域。
19. 選取您的快速模式服務使用的負載平衡器。

傳播幾分鐘後，您的服務將開始接收自訂網域上的流量。如果您要將許多憑證新增至 Application Load Balancer 或從 Application Load Balancer 提供許多快速模式服務，建議您考慮提高 Application Load Balancer 上憑證的服務限制。

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：

- 所要使用的容量，會決定託管任務的基礎結構
- 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
- 記憶體與 CPU 需求
- 任務執行所在容器的作業系統
- 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 [Fargate 的 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 Amazon Q Developer 在 Amazon ECS 主控台中提供任務定義建議](#)

- [使用主控台更新 Amazon ECS 任務定義](#)
- [使用主控台取消註冊 Amazon ECS 任務定義修訂版](#)
- [使用主控台刪除 Amazon ECS 任務定義修訂版](#)
- [Amazon ECS 任務定義使用案例](#)
- [Amazon ECS 受管執行個體的 Amazon ECS 任務定義參數](#)
- [Fargate 的 Amazon ECS 任務定義參數](#)
- [Amazon EC2 的 Amazon ECS 任務定義參數](#)
- [Amazon ECS 任務定義範本](#)
- [Amazon ECS 任務定義範例](#)

Amazon ECS 任務定義狀態

建立、取消註冊或刪除任務定義時，任務定義狀態會隨之變更。您可以透過主控台或 `DescribeTaskDefinition` 檢視任務定義狀態。

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。您隨時可以提交要刪除的任務定義，而不會對現有任務與服務造成影響。

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處於 DELETE_IN_PROGRESS 狀態，則任務定義名稱將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 來管理您的任務定義，會收取所有任務定義註冊 AWS Config 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

如果存在依賴任務定義修訂版的任何 Amazon ECS 資源，任務定義刪除請求將無法完成。以下資源可能會阻止任務定義遭刪除：

- Amazon ECS 獨立任務 – 需要任務定義才能讓任務保持良好的運作狀態。
- Amazon ECS 服務任務 – 需要任務定義才能讓任務保持良好的運作狀態。
- Amazon ECS 服務部署與任務集 – 當為 Amazon ECS 部署或任務集啟動擴展事件時，需要任務定義。

如果您的任務定義仍處於 DELETE_IN_PROGRESS 狀態，您可以使用 主控台或 AWS CLI 來識別，然後停止封鎖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 服務部署與任務集 – 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

您可以透過為應用程式建立任務定義來架構自己的應用程式。任務定義包含用於定義應用程式相關資訊的參數，包括：

- 所要使用的容量，會決定託管任務的基礎結構。

在使用 EC2 容量提供者時，您也需選擇執行個體類型。在使用 Amazon ECS 受管執行個體容量提供者時，您可為 Amazon ECS 提供執行個體需求來管理運算容量。對於部分執行個體類型 (例如 GPU)，您需要設定特定參數。如需詳細資訊，請參閱[Amazon ECS 任務定義使用案例](#)。

- 容器映像，會包含應用程式程式碼，以及應用程式程式碼執行所需的所有相依性。
- 任務中容器所要使用的聯網模式。

聯網模式決定了任務如何透過網路進行通訊。

對於在 EC2 執行個體與 Amazon ECS 受管執行個體上執行的任務，雖有多種選項可供選擇，但建議使用 awsvpc 網路模式。awsvpc 網路模式可簡化容器聯網設定，賦予您更高的控制權來控管應用程式彼此之間以及應用程式與 VPC 內其他服務之間的通訊方式。

對於在 Fargate 上執行的任務，必須使用 awsvpc 網路模式。

- 任務所要使用的記錄組態。
- 任務中容器所使用的任何資料磁碟區。

如需有關任務定義參數的完整清單，請參閱 [Fargate 的 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，網站與 API 通常需要不同的擴展模式。隨著流量增加，所需的 Web 容器數量可能與 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 容器與 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，每個任務定義修訂版皆代表特定容器映像設定的時間點快照。這類似於容器是執行特定版本應用程式程式碼所需所有元件的快照。

建立應用程式程式碼版本、容器映像標籤與任務定義修訂版之間的一對一映射關係。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，更新 Amazon ECS 服務以部署這個新的任務定義修訂版。

- 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。除了實作此實務，還需為每個業務元件提供專屬的任務定義系列。透過實作這兩個最佳實務，您可以限制每個服務對您 AWS 帳戶中資源的存取量。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 任務大小的最佳實務

容器與任務大小對於擴展與容量規劃都至關重要。在 Amazon ECS 中，CPU 與記憶體是用於容量的兩個資源指標。CPU 以完整 vCPU 的 1/1024 為單位進行計量 (其中 1024 個單位等於 1 個完整 vCPU)。記憶體以 MiB 為單位進行計量。在任務定義中，您可以設定資源保留量與限制。

設定保留量時，您要設定任務所需的資源數量下限。任務至少會收到請求的資源量。應用程式實際使用的 CPU 或記憶體，可能會超出設定的保留量。但這受您同時設定的所有限制的約束。使用超過保留數量稱為爆量。在 Amazon ECS 中，保留量會得到保證。例如，若使用 Amazon EC2 執行個體提供容量，當無法滿足保留需求時，Amazon ECS 不會將該任務置放於該執行個體上。

限制是容器或任務所能使用的 CPU 單位或記憶體的最大數量。任何嘗試使用超過此限制之 CPU 的行為，都會導致限流。任何嘗試使用更多記憶體的行為，都會導致容器停止。

精準選擇這些值可能具有挑戰性。這是因為最適合應用程式的值，在很大程度上取決於應用程式的資源需求。對應用程式進行負載測試，是成功規劃資源需求以及更清楚地了解應用程式需求的關鍵。

無狀態應用程式

對於可水平擴展的無狀態應用程式 (如負載平衡器後端的應用程式)，建議先決定應用程式在處理請求時所耗用的記憶體量。若要做到這一點，您可以使用 `ps` 或 `top` 等傳統工具，也可以使用 CloudWatch Container Insights 這類監控解決方案。

在決定 CPU 保留量時，需考量如何擴展應用程式來滿足業務需求。您可以使用較小的 CPU 保留量，例如 256 個 CPU 單位 (或 1/4 vCPU)，以將成本降至最低的精細方式橫向擴充。但它們的擴展速度可能不足以應對需求的大幅激增。您可以使用較大的 CPU 保留量，更快速地進行縮減與橫向擴充，從而更快匹配需求激增。但較大的 CPU 保留量會帶來更高的成本。

其他應用程式

對於無法水平擴展的應用程式 (例如單例工作線程或資料庫伺服器)，可用容量與成本是最重要的考量因素。您應根據負載測試的結果選擇記憶體與 CPU 的配置量，確保能應對流量並達成服務水準目標。Amazon ECS 可確保應用程式置放在具有足夠容量的主機上。

Amazon ECS 受管執行個體的 Amazon ECS 任務聯網

在 Amazon ECS 受管執行個體上執行之 Amazon ECS 任務的聯網行為，由任務定義中指定的網路模式決定。您必須在任務定義中指定網路模式。若使用未指定網路模式的任務定義，將無法在 Amazon ECS 受管執行個體上執行任務。Amazon ECS 受管執行個體支援以下聯網模式，可確保從 Fargate 或 Amazon EC2 上的 Amazon ECS 遷移工作負載時具備向後相容性：

網路模式	Description
awsvpc	每項任務皆會取得專屬的彈性網路介面 (ENI) 與私有 IPv4 位址。這能提供與 Amazon EC2 執行個體相同的聯網屬性，且與傳統 Fargate 任務相容。此模式透過 ENI 主幹功能實現高任務密度。
host	任務會直接共用主機網路命名空間。容器聯網與底層主機執行個體緊密綁定。

在純 IPv6 模式下使用 VPC

在純 IPv6 組態中，Amazon ECS 任務只會透過 IPv6 進行通訊。若要為純 IPv6 組態設定 VPC 與子網路，您必須將 IPv6 CIDR 區塊新增至 VPC，並建立僅包含 IPv6 CIDR 區塊的子網路。如需詳細資訊，請參閱 Amazon VPC User Guide 中的 [Add IPv6 support for your VPC](#) 與 [Create a subnet](#)。您還必須更新路由表，加入 IPv6 目標，並在安全群組中設定 IPv6 規則。如需詳細資訊，請參閱 Amazon VPC User Guide 中的 [Configure route tables](#) 與 [Configure security group rules](#)。

適用下列注意事項：

- 您可透過以下兩種方式，將純 IPv4 或雙堆疊的 Amazon ECS 服務升級為純 IPv6 組態：一是直接將服務更新為使用純 IPv6 子網路；二是建立平行的純 IPv6 服務，再透過 Amazon ECS 藍綠部署將流量遷移至新服務。如需有關 Amazon ECS 藍/綠部署的詳細資訊，請參閱 [Amazon ECS 藍/綠部署](#)。
- 純 IPv6 的 Amazon ECS 服務必須使用具 IPv6 目標群組的雙堆疊負載平衡器。若要遷移的現有 Amazon ECS 服務後端掛載了 Application Load Balancer 或 Network Load Balancer，可建立新的雙堆疊負載平衡器並將流量從舊負載平衡器遷移過去，或直接更新現有負載平衡器的 IP 位址類型。

如需有關 Network Load Balancer 的詳細資訊，請參閱 User Guide for Network Load Balancers 中的 [Create a Network Load Balancer](#) 與 [Update the IP address types for your Network Load Balancer](#)。如需有關 Application Load Balancer 的詳細資訊，請參閱 User Guide for Application Load Balancers 中的 [Create an Application Load Balancer](#) 與 [Update the IP address types for your Application Load Balancer](#)。

- 若要讓純 IPv6 組態中的 Amazon ECS 任務與純 IPv4 端點通訊，您可以設定 DNS64 與 NAT64，實現從 IPv6 到 IPv4 的網路位址轉換。如需更多詳細資訊，請參閱《Amazon VPC 使用者指南》中的「[DNS64 和 NAT64](#)」。
- 在純 IPv6 組態中，Amazon ECS 工作負載從 Amazon ECR 拉取映像時，必須使用 Amazon ECR 雙堆疊映像 URI 端點。如需詳細資訊，請參閱 Amazon Elastic Container Registry User Guide 中的 [Getting started with making requests over IPv6](#)。

Note

Amazon ECR 不支援可供純 IPv6 組態中任務使用的雙堆疊介面 VPC 端點。如需詳細資訊，請參閱 Amazon Elastic Container Registry User Guide 中的 [Getting started with making requests over IPv6](#)。

- 純 IPv6 組態不支援 Amazon ECS Exec。

為 Amazon ECS 受管執行個體上的任務配置網路介面

使用 Amazon ECS 受管執行個體中的 `awsvpc` 網路模式可簡化容器聯網設定，因為您有更高的控制權來控管應用程式彼此之間以及應用程式與 VPC 內其他服務之間的通訊方式。`awsvpc` 網路模式也讓您可以您在任務中更精細地使用安全群組與網路監控工具，為您的容器提供更高的安全性。

依預設，若執行個體類型支援主幹功能，每個 Amazon ECS 受管執行個體在啟動時皆會連接一個主幹彈性網路介面 (ENI)，並將其作為主要 ENI。如需有關支援 ENI 主幹功能之執行個體類型的詳細資訊，請參閱[適用於更多 Amazon ECS 容器網路介面的支援執行個體](#)。

 Note

若選擇的執行個體類型不支援中繼 ENI，該執行個體將會搭配一般 ENI 啟動。

在執行個體上執行的每項任務，皆會取得連接至中繼 ENI 的專屬 ENI，並配有一個主要私有 IP 位址。如果 VPC 已設定為雙堆疊模式，且使用的子網路帶有 IPv6 CIDR 區塊，則該 ENI 還會取得一個 IPv6 位址。使用公有子網路時，您可透過為子網路啟用 IPv4 公有定址功能，選擇是否將公有 IP 位址指派給 Amazon ECS 受管執行個體的主要 ENI。如需詳細資訊，請參閱 Amazon VPC User Guide 中的[Modify the IP addressing attributes of your subnet](#)。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 VPC 與子網路的詳細資訊，請參閱 Amazon VPC User Guide 中的[How Amazon VPC works](#)

下列操作使用連接至執行個體的主要 ENI：

- 映像下載 – 透過主要 ENI 從 Amazon ECR 下載容器映像。
- 秘密擷取 – 透過主要 ENI 擷取 Secrets Manager 秘密與其他憑證。
- 日誌上傳 – 日誌透過主要 ENI 上傳至 CloudWatch。
- 環境檔案下載 – 環境檔案透過主要 ENI 下載。

應用程式流量會流經任務 ENI。

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 流程日誌](#)。

您也可以利用 AWS PrivateLink。您可以設定 VPC 介面端點，以便透過私有 IP 地址存取 Amazon ECS API。AWS PrivateLink 會將您 VPC 與 Amazon ECS 之間的所有網路流量限制於 Amazon 網路中。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 介面 VPC 端點 \(AWS PrivateLink\)](#)。

awsvpc 網路模式也可讓您在未連接中繼 ENIs 的執行個體類型時，利用 Amazon VPC 流量鏡射進行安全性和網路流量監控。如需詳細資訊，請參閱 Amazon VPC Traffic Mirroring Guide 中的[What is Traffic Mirroring?](#)。

awsvpc 模式考量

- 任務需要 Amazon ECS 服務連結角色，才能進行 ENI 管理。此角色會在您建立叢集或服務時自動建立。
- 任務 ENI 由 Amazon ECS 管理，無法手動卸離或修改。
- 不支援在執行獨立任務 (RunTask) 或建立/更新服務 (CreateService/UpdateService) 時，使用 assignPublicIp 將公有 IP 位址指派給任務 ENI。
- 在任務層級設定 awsvpc 聯網時，必須使用在 Amazon ECS 受管執行個體容量提供者啟動範本中指定的相同 VPC。您可以使用與啟動範本中指定之子網路和安全群組不同的子網路與安全群組。
- 對於 awsvpc 網路模式任務，請在設定負載平衡器目標群組時使用 ip 目標類型。Amazon ECS 會針對支援的聯網模式，自動管理目標群組註冊。

在雙堆疊模式下使用 VPC

在雙堆疊模式中使用 VPC 時，任務可透過 IPv4 或 IPv6 或兩者進行通訊。IPv4 與 IPv6 位址彼此獨立。因此，您必須在 VPC 中為 IPv4 與 IPv6 分別配置路由與安全設定。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 ECS 受管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[透過帳戶設定使用 Amazon ECS 功能](#)。
- 您的任務是使用 awsvpc 網路模式。
- 您的 VPC 與子網路已針對 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](#)。

主機網路模式

在 host 模式下，任務會直接共用主機的網路命名空間。容器的聯網組態會與底層的 Amazon ECS 受管執行個體主機執行個體綁定，而該主機執行個體是您建立 Amazon ECS 受管執行個體容量提供者時透過 `networkConfiguration` 參數所指定。

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。

僅在需要直接存取主機聯網，或遷移需具備主機層級網路存取權的應用程式時，才使用主機模式。

EC2 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	Description
awsvpc	是	是	任務會配置專屬的彈性網路介面 (ENI) 與主要私有 IPv4 位址或 IPv6 位址。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

網路模式	EC2 上的 Linux 容器	EC2 上的 Windows 容器	Description
			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。
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 Linux 上 Docker 聯網的詳細資訊，請參閱 Docker Documentation 中的 [Networking overview](#)。

如需有關 Windows 上 Docker 聯網的詳細資訊，請參閱 Microsoft Containers on Windows Documentation 中的 [Windows container networking](#)。

在純 IPv6 模式下使用 VPC

在純 IPv6 組態中，Amazon ECS 任務只會透過 IPv6 進行通訊。若要為純 IPv6 組態設定 VPC 與子網路，您必須將 IPv6 CIDR 區塊新增至 VPC，並建立僅包含 IPv6 CIDR 區塊的新子網路。如需詳細資訊，請參閱 Amazon VPC User Guide 中的 [Add IPv6 support for your VPC](#) 與 [Create a subnet](#)。

您還必須更新路由表，加入 IPv6 目標，並在安全群組中設定 IPv6 規則。如需詳細資訊，請參閱 Amazon VPC User Guide 中的 [Configure route tables](#) 與 [Configure security group rules](#)。

適用下列注意事項：

- 您可透過以下兩種方式，將純 IPv4 或雙堆疊的 Amazon ECS 服務升級為純 IPv6 組態：一是直接將服務更新為使用純 IPv6 子網路；二是建立平行的純 IPv6 服務，再透過 Amazon ECS 藍綠部署將流量遷移至新服務。如需有關 Amazon ECS 藍/綠部署的詳細資訊，請參閱 [Amazon ECS 藍/綠部署](#)。
- 純 IPv6 的 Amazon ECS 服務必須使用具 IPv6 目標群組的雙堆疊負載平衡器。若要遷移的現有 Amazon ECS 服務後端掛載了 Application Load Balancer 或 Network Load Balancer，可建立新的雙堆疊負載平衡器並將流量從舊負載平衡器遷移過去，或直接更新現有負載平衡器的 IP 位址類型。

如需有關 Network Load Balancer 的詳細資訊，請參閱 User Guide for Network Load Balancers 中的 [Create a Network Load Balancer](#) 與 [Update the IP address types for your Network Load Balancer](#)。如需有關 Application Load Balancer 的詳細資訊，請參閱 User Guide for Application Load Balancers 中的 [Create an Application Load Balancer](#) 與 [Update the IP address types for your Application Load Balancer](#)。

- Windows 不支援純 IPv6 組態。若要在純 IPv6 組態中執行任務，必須使用 Amazon ECS 最佳化 Linux AMI。如需有關 Amazon ECS 最佳化 AMI 的詳細資訊，請參閱 [Amazon ECS 最佳化 Linux AMI](#)。
- 在啟動容器執行個體以在純 IPv6 組態中執行任務時，必須透過 `--enable-primary-ipv6` EC2 參數為該執行個體設定一個主要 IPv6 位址。

Note

如果沒有主要 IPv6 位址，在容器執行個體上以主機或橋接網路模式執行的任務，將無法向負載平衡器或 AWS Cloud Map 註冊。

如需有關用於執行 Amazon EC2 執行個體的 `--enable-primary-ipv6` 的詳細資訊，請參閱 AWS CLI Command Reference 中的 [run-instances](#)。

如需使用 啟動容器執行個體的詳細資訊 AWS 管理主控台，請參閱 [啟動 Amazon ECS Linux 容器執行個體](#)。

- 依預設，Amazon ECS 容器代理程式會透過偵測執行個體的預設 IPv4 與 IPv6 路由，判斷容器執行個體是否相容純 IPv6 組態。若要覆寫此行為，您可以在執行個體的 `/etc/ecs/ecs.config` 檔案中將 `ECS_INSTANCE_IP_COMPATIBILITY` 參數設定為 `ipv4` 或 `ipv6`。
- 任務必須使用版本 1.99.1 或更新版本的容器代理程式。如需有關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資訊，請參閱 [更新 Amazon ECS 容器代理程式](#)。

- 若要讓純 IPv6 組態中的 Amazon ECS 任務與純 IPv4 端點通訊，您可以設定 DNS64 與 NAT64，實現從 IPv6 到 IPv4 的網路位址轉換。如需更多詳細資訊，請參閱《Amazon VPC 使用者指南》中的「[DNS64 和 NAT64](#)」。
- 在純 IPv6 組態中，Amazon ECS 工作負載從 Amazon ECR 拉取映像時，必須使用 Amazon ECR 雙堆疊映像 URI 端點。如需詳細資訊，請參閱 Amazon Elastic Container Registry User Guide 中的 [Getting started with making requests over IPv6](#)。

 Note

Amazon ECR 不支援可供純 IPv6 組態中任務使用的雙堆疊介面 VPC 端點。如需詳細資訊，請參閱 Amazon Elastic Container Registry User Guide 中的 [Getting started with making requests over IPv6](#)。

- 純 IPv6 組態不支援 Amazon ECS Exec。

AWS 區域 支援 IPv6-only 模式

您可以在 Amazon ECS 可用的下列 AWS 區域中，在 IPv6-only 的組態中執行任務：

- 美國東部 (俄亥俄)
- 美國東部 (維吉尼亞北部)
- 美國西部 (加利佛尼亞北部)
- 美國西部 (奧勒岡)
- 非洲 (開普敦)
- 亞太地區 (香港)
- 亞太區域 (海德拉巴)
- 亞太地區 (雅加達)
- 亞太地區 (墨爾本)
- 亞太地區 (孟買)
- 亞太區域 (大阪)
- 亞太區域 (首爾)
- 亞太區域 (新加坡)
- 亞太地區 (雪梨)
- 亞太區域 (東京)

- 加拿大 (中部)
- 加拿大西部 (卡加利)
- 中國 (北京)
- 中國 (寧夏)
- 歐洲 (法蘭克福)
- 歐洲 (倫敦)
- 歐洲 (米蘭)
- Europe (Paris)
- 歐洲 (西班牙)
- 以色列 (特拉維夫)
- Middle East (Bahrain)
- 中東 (阿拉伯聯合大公國)
- 南美洲 (聖保羅)
- AWS GovCloud (美國東部)
- AWS GovCloud (美國西部)

為 Amazon ECS 任務配置網路介面

awsipc 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 awsipc 網路模式可簡化容器聯網設定，因為您有更高的控制權來控管應用程式彼此之間以及應用程式與 VPC 內其他服務之間的通訊方式。awsipc 網路模式也讓您可以您在任務中更精細地使用安全群組與網路監控工具，為您的容器提供更高的安全性。您也可以利用其他 Amazon EC2 聯網功能，例如 VPC 流程日誌，以便您監控任務的進出流量。此外，屬於同一個任務的容器可以透過 localhost 介面進行通訊。

任務彈性網路介面 (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 會顯示在帳戶的 Amazon EC2 主控台中。帳戶無法卸離或修改這些 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 主控台中或使用 [DescribeTasks](#) API 操作來檢視任務的 ENI 連接資訊。當任務停止或服務縮小規模時，任務 ENI 即予以分離和刪除。

如需增加 ENI 密度，請使用 `awsvpcTrunking` 帳戶設定。Amazon ECS 也會為容器執行個體建立並連接「中繼」網路介面。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` 帳戶設定的詳細資訊，請參閱[先決條件](#)。

您可以在任務定義的 `networkMode` 參數中指定 `awsvpc`。如需詳細資訊，請參閱[網路模式](#)。

接著，當您執行任務或建立服務時，請使用 `networkConfiguration` 參數。該參數包含一或多個用於置放任務的子網路，以及一個或多個要連接至 ENI 的安全群組。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。

- 如果在 `awsvpc` 模式下使用 `p5.48xlarge` 執行個體，則無法在執行個體上執行超過 1 項任務。
- 使用 `awsvpc` 網路模式的任務和服務需要 Amazon ECS 服務連結角色為 Amazon ECS 提供代表您呼叫其他 AWS 服務的許可。這個角色會在您建立叢集，或在 AWS 管理主控台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。如需有關每個執行個體類型的預設 ENI 限制的詳細資訊，請參閱 Amazon EC2 User Guide 中的[IP addresses per network interface per instance type](#)。

- 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 [透過帳戶設定使用 Amazon ECS 功能](#)。如需 ENI 中繼的詳細資訊，請參閱 [增加 Amazon ECS Linux 容器執行個體網路介面數量](#)。
- 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 Elastic Container Service API 參考中的 [AwsVpcConfiguration](#)。
- 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。接著，它會執行 `pauseamazon-ecs-cni-plugins` [CNI 外掛程式，來設定](#) 容器的網路命名空間。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 [使用負載平衡分佈 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，以使用採用 awsvpc 網路模式的 Windows 容器 CloudWatch 指標。
- 使用 awsvpc 網路模式的任務和服務需要 Amazon ECS 服務連結角色為 Amazon ECS 提供代表您呼叫其他 AWS 服務的許可。這個角色會在您建立叢集，或在 AWS 管理主控台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 “引導容器執行個體”](#)。
- 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。如需有關每個執行個體類型的預設 ENI 限制的詳細資訊，請參閱 Amazon EC2 User Guide 中的[IP addresses per network interface per instance type](#)。
- 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 Elastic Container Service API 參考中的 [AwsVpcConfiguration](#)。
- 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。接著，它會執行 `pauseamazon-ecs-cni-plugins` [CNI 外掛程式，來設定](#) 容器的網路命名空間。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[使用負載平衡分佈 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

在雙堆疊模式中使用 VPC 時，任務可透過 IPv4 或 IPv6 或兩者進行通訊。IPv4 與 IPv6 位址彼此獨立。因此，您必須在 VPC 中為 IPv4 與 IPv6 分別配置路由與安全設定。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](#)。

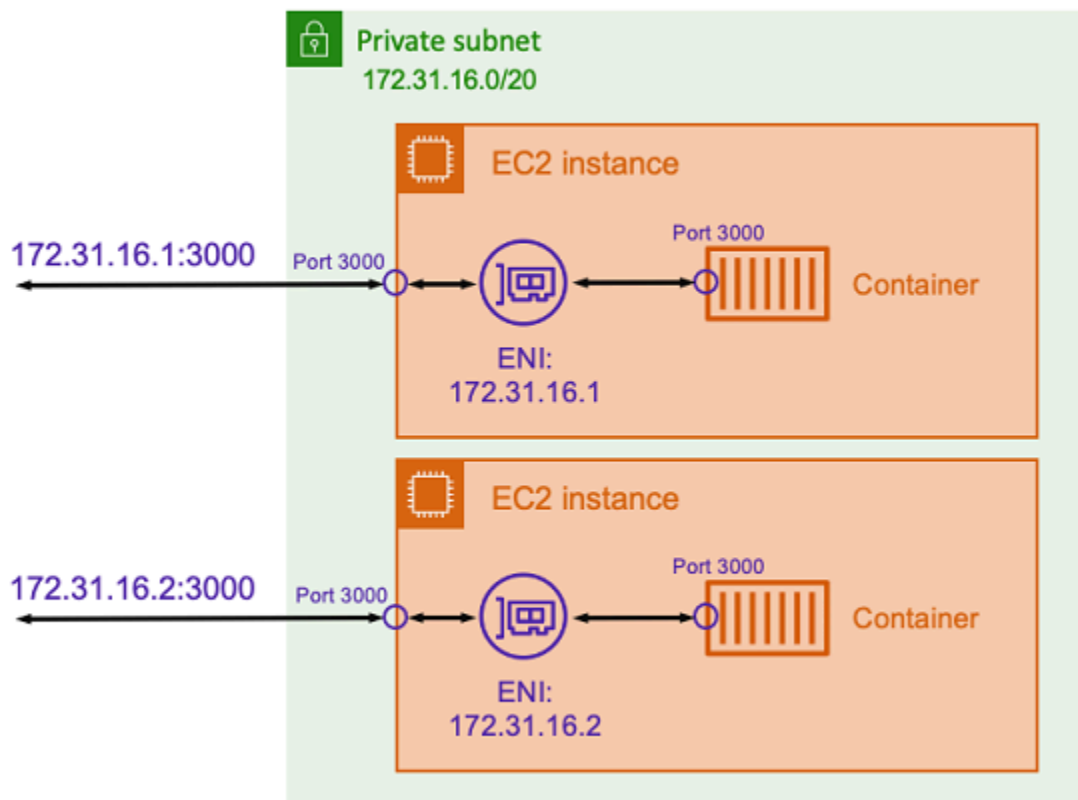
如果符合下列條件，將會為 Amazon ECS 任務指派 IPv6 地址：

- 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[透過帳戶設定使用 Amazon ECS 功能](#)。
- 您的任務是使用 awsvpc 網路模式。
- 您的 VPC 與子網路已針對 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 容器連接埠映射至 EC2 執行個體網路介面

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。

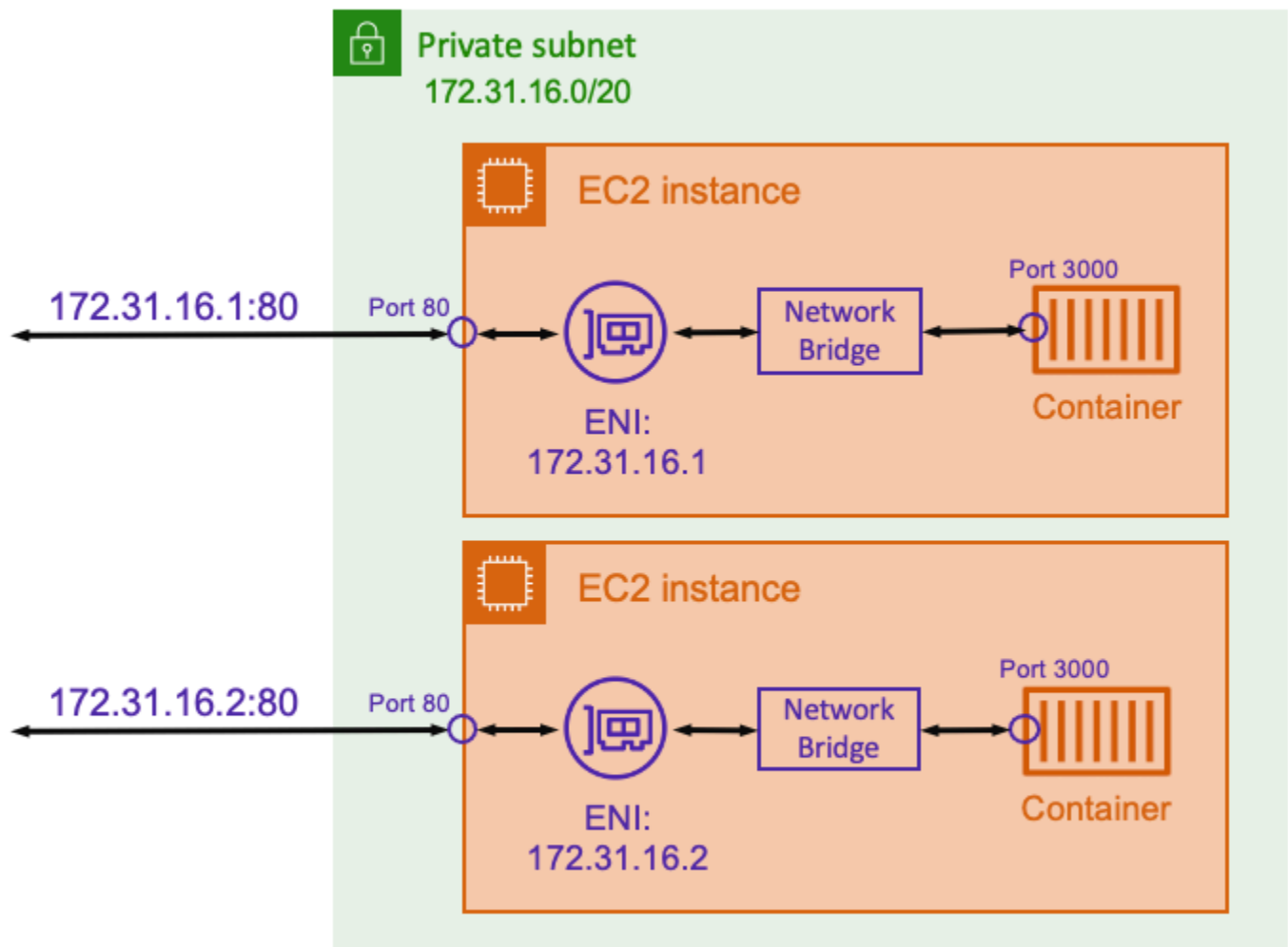
使用此網路模式存在重大的缺點。您不能在每台主機上執行多個任務的執行個體化。這是因為只有第一個任務可以繫結至 Amazon EC2 執行個體上所需的連接埠。使用 host 網路模式時，也無法重新對應容器連接埠。例如，如果應用程式需要接聽特定的連接埠號碼，則無法直接重新對應連接埠號碼。相反地，您必須透過變更應用程式組態來管理任何連接埠衝突。

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。

為 Amazon ECS Linux 任務使用 Docker 的虛擬網路

僅 Amazon EC2 執行個體上託管的 Amazon ECS 任務支援 bridge 網路模式。

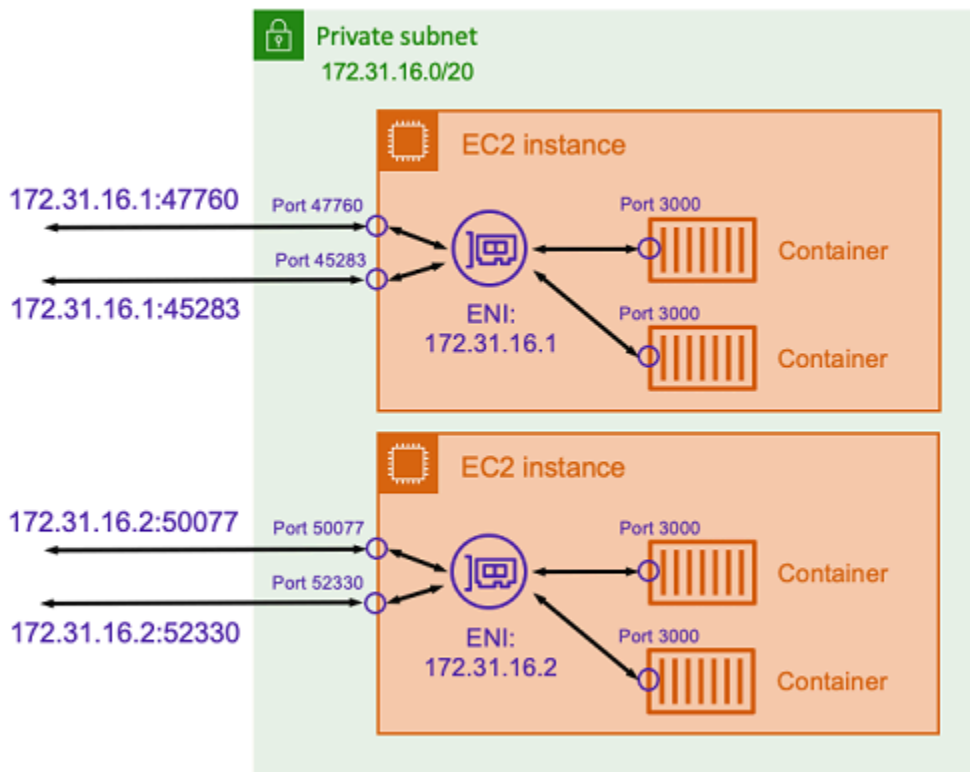
在 bridge 模式下，您正在使用虛擬網路橋接器，在主機和容器的網路之間建立一個層。如此一來，您就可以建立將主機連接埠重新對應至容器連接埠的連接埠映射。映射可以是靜態或動態模式。



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 從暫時性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。其做法是自動更新負載平衡器目標群組 AWS Cloud Map 和服務探索，以取得任務 IP 地址和連接埠的清單。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。

為純 IPv6 的工作負載設定橋接聯網模式

若要設定透過 IPv6 通訊的 bridge 模式，您必須更新 Docker 常駐程式設定。請按照以下設定更新 `/etc/docker/daemon.json`：

```
{
  "ipv6": true,
  "fixed-cidr-v6": "2001:db8:1::/64",
  "ip6tables": true,
  "experimental": true
}
```

更新 Docker 常駐程式設定後，您需要重新啟動常駐程式。

Note

在更新並重新啟動常駐程式時，Docker 會在執行個體上啟用 IPv6 轉送功能，這可能會導致使用 Amazon Linux 2 AMI 的執行個體丟失預設路由。若要避免此問題，請使用下列命令，透過子網路的 IPv6 閘道新增預設路由。

```
ip route add default via FE80:EC2::1 dev eth0 metric 100
```

Fargate 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 VPC 已設定為雙堆疊模式，且使用的子網路帶有 IPv6 CIDR 區塊，則該任務的 ENI 還會取得 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 VPC 與子網路的詳細資訊，請參閱 Amazon VPC User Guide 中的 [How Amazon VPC works](#)。

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 User Guide 中的 [Amazon ECR interface VPC endpoints \(AWS PrivateLink\)](#)。

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 流程日誌](#)。

您也可以利用 AWS PrivateLink。您可以設定 VPC 介面端點，以便透過私有 IP 地址存取 Amazon ECS APIs。AWS PrivateLink 會限制 VPC 和 Amazon ECS 之間的所有網路流量到 Amazon 網路。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 介面 VPC 端點 \(AWS PrivateLink\)](#)。

如需如何使用 NetworkConfiguration 資源的範例 CloudFormation，請參閱 [the section called “範本範例 CloudFormation”](#)。

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。

Action	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
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 for 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

Action	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
Amazon EFS 檔案系統流量	不適用	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 提供代表您呼叫其他 AWS 服務的許可。這個角色會在您建立叢集，或在 AWS 管理主控台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 Elastic Container Service API 參考中的 [AwsVpcConfiguration](#)。
- 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 子網路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)。
- 下列規則適用於在特定 Fargate 平台版本上執行的任務：Linux 系統對應 1.4.0 或更新版本，Windows 系統對應 1.0.0。在雙堆疊子網路中啟動的任務會取得 IPv4 位址與 IPv6 位址。在純 IPv6 子網路中啟動的任務只會取得 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 之任務的服務僅支援 Application Load Balancer 與 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[使用負載平衡分佈 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：

- 在您啟動任務的區域中，執行任務啟動操作之 IAM 主體的 Amazon ECS dualStackIPv6 帳戶設定必須處於開啟 (enabled) 狀態。此設定只能使用 API 或進行修改 AWS CLI。您可選擇為帳戶中特定的 IAM 主體開啟此設定，或透過設定帳戶預設值為整個帳戶開啟此設定。如需詳細資訊，請參閱[透過帳戶設定使用 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。

對於以雙堆疊模式在 VPC 中執行的 Fargate 上的 Amazon ECS 任務，若要與任務啟動程序中使用的相依性服務通訊，例如 ECR、SSM 和 SecretManager，公有子網路的路由表需要 IPv4 (0.0.0.0/0) 路由至網際網路閘道，而私有子網路的路由表需要 IPv4 (0.0.0.0/0) 路由至 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[NAT 閘道](#)。

如需如何設定雙堆疊 VPC 的範例，請參閱[雙堆疊 VPC 組態範例](#)。

在純 IPv6 模式下使用 VPC

在純 IPv6 組態中，Amazon ECS 任務只會透過 IPv6 進行通訊。若要為純 IPv6 組態設定 VPC 與子網路，您必須將 IPv6 CIDR 區塊新增至 VPC，並建立僅包含 IPv6 CIDR 區塊的子網路。如需詳細資訊，請參閱 Amazon VPC User Guide 中的[Add IPv6 support for your VPC](#) 與 [Create a subnet](#)。您還必須

更新路由表，加入 IPv6 目標，並在安全群組中設定 IPv6 規則。如需詳細資訊，請參閱 Amazon VPC User Guide 中的 [Configure route tables](#) 與 [Configure security group rules](#)。

適用下列注意事項：

- 您可透過以下兩種方式，將純 IPv4 或雙堆疊的 Amazon ECS 服務升級為純 IPv6 組態：一是直接將服務更新為使用純 IPv6 子網路；二是建立平行的純 IPv6 服務，再透過 Amazon ECS 藍綠部署將流量遷移至新服務。如需有關 Amazon ECS 藍/綠部署的詳細資訊，請參閱 [Amazon ECS 藍/綠部署](#)。
- 純 IPv6 的 Amazon ECS 服務必須使用具 IPv6 目標群組的雙堆疊負載平衡器。若要遷移的現有 Amazon ECS 服務後端掛載了 Application Load Balancer 或 Network Load Balancer，可建立新的雙堆疊負載平衡器並將流量從舊負載平衡器遷移過去，或直接更新現有負載平衡器的 IP 位址類型。

如需有關 Network Load Balancer 的詳細資訊，請參閱 User Guide for Network Load Balancers 中的 [Create a Network Load Balancer](#) 與 [Update the IP address types for your Network Load Balancer](#)。如需有關 Application Load Balancer 的詳細資訊，請參閱 User Guide for Application Load Balancers 中的 [Create an Application Load Balancer](#) 與 [Update the IP address types for your Application Load Balancer](#)。

- Windows 不支援純 IPv6 組態。
- 若要讓純 IPv6 組態中的 Amazon ECS 任務與純 IPv4 端點通訊，您可以設定 DNS64 與 NAT64，實現從 IPv6 到 IPv4 的網路位址轉換。如需更多詳細資訊，請參閱《Amazon VPC 使用者指南》中的「[DNS64 和 NAT64](#)」。
- Fargate 平台版本 1.4.0 或更新版本支援純 IPv6 組態。
- 在純 IPv6 組態中，Amazon ECS 工作負載從 Amazon ECR 拉取映像時，必須使用 Amazon ECR 雙堆疊映像 URI 端點。如需詳細資訊，請參閱 Amazon Elastic Container Registry User Guide 中的 [Getting started with making requests over IPv6](#)。

 Note

Amazon ECR 不支援可供純 IPv6 組態中任務使用的雙堆疊介面 VPC 端點。如需詳細資訊，請參閱 Amazon Elastic Container Registry User Guide 中的 [Getting started with making requests over IPv6](#)。

- 純 IPv6 組態不支援 Amazon ECS Exec。
- Amazon CloudWatch 不支援雙堆疊 FIPS 端點，因此無法監控採用純 IPv6 組態且需符合 FIPS-140 規範的 Amazon ECS 任務。如需有關 FIPS-140 的詳細資訊，請參閱 [AWS Fargate 聯邦資訊處理標準 \(FIPS-140\)](#)。

AWS 區域 支援 IPv6-only 模式

您可以在下列 Amazon ECS 可用的 中 AWS 區域 ，以IPv6-only 的組態執行任務：

- 美國東部 (俄亥俄)
- 美國東部 (維吉尼亞北部)
- 美國西部 (加利佛尼亞北部)
- 美國西部 (奧勒岡)
- 非洲 (開普敦)
- 亞太地區 (香港)
- 亞太區域 (海德拉巴)
- 亞太地區 (雅加達)
- 亞太地區 (墨爾本)
- 亞太地區 (孟買)
- 亞太區域 (大阪)
- 亞太區域 (首爾)
- 亞太區域 (新加坡)
- 亞太地區 (雪梨)
- 亞太區域 (東京)
- 加拿大 (中部)
- 加拿大西部 (卡加利)
- 中國 (北京)
- 中國 (寧夏)
- 歐洲 (法蘭克福)
- 歐洲 (倫敦)
- 歐洲 (米蘭)
- Europe (Paris)
- 歐洲 (西班牙)
- 以色列 (特拉維夫)
- Middle East (Bahrain)

- 中東 (阿拉伯聯合大公國)
- 南美洲 (聖保羅)
- AWS GovCloud (美國東部)
- AWS GovCloud (美國西部)

Amazon ECS 任務的儲存選項

Amazon ECS 會依據您的需求，提供彈性、具成本效益且易於使用的資料儲存選項。Amazon ECS 支援容器的下列資料磁碟區選項：

資料量	支援的容量	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、Amazon ECS 受管執行個體	Linux、Windows (僅限於 Amazon EC2)	連接至獨立任務時，可為持續性儲存。連接至由服務維護的任務時，則為暫時性儲存。	Amazon EBS 磁碟區為資料密集型容器化工作負載，提供具成本效益、持久且高效能的區塊儲存服務。常見使用案例包括交易型工作負載 (如資料庫、虛擬桌面與根磁碟區)，以及輸送量密集型工作負載 (如日誌處理與 ETL 工作負載)。如需詳細資訊，請參閱 將 Amazon EBS 磁碟區與 Amazon ECS 搭配使用 。
Amazon Elastic File System (Amazon EFS)	Fargate、Amazon EC2、Amazon	Linux	持續	Amazon EFS 磁碟區提供可用於 Amazon ECS 任務的簡單、可

資料量	支援的容量	支援的作業系統	儲存持續性	使用案例
	ECS 受管執行個體			擴展且持續性的共用檔案儲存服務，該服務會隨著您新增或刪除檔案自動擴容與縮容。Amazon EFS 磁碟區支援並行操作，適用於需水平擴展且需要低延遲、高輸送量與寫後讀一致性等儲存功能的容器化應用程式。常見使用案例包括資料分析、媒體處理、內容管理與 Web 服務等工作負載。如需詳細資訊，請參閱 將 Amazon EFS 磁碟區與 Amazon ECS 搭配使用 。

資料量	支援的容量	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 磁碟區提供全受管 Windows 檔案伺服器，可用於對需要持續性、分散式、共用且靜態檔案儲存的 Windows 任務進行佈建。常見使用案例包括 .NET 應用程式，這類應用程式可能需要將本機資料夾作為持續性儲存空間來儲存應用程式輸出內容。Amazon FSx for Windows File Server 會在容器中提供一個本機資料夾，支援多個容器在由「SMB 共用」提供支援的同一檔案系統中進行讀寫操作。如需詳細資訊，請參閱 搭配 Amazon ECS 使用 FSx for Windows File Server 磁碟區 。

資料量	支援的容量	支援的作業系統	儲存持續性	使用案例
Amazon FSx for NetApp ONTAP	Amazon EC2	Linux	持續	Amazon FSx for NetApp ONTAP 磁碟區提供全受管 NetApp ONTAP 檔案系統，可用於對需要持久性、高效能且功能豐富的共用檔案儲存的 Linux 任務進行佈建。Amazon FSx for NetApp ONTAP 支援 NFS 與 SMB 通訊協定，並提供快照、複製及重複資料刪除等企業級功能。 常見使用案例包括高效能運算工作負載、內容儲存庫，以及需要符合 POSIX 標準之共用儲存的應用程式。如需詳細資訊，請參閱 Mounting Amazon FSx for NetApp ONTAP file systems from Amazon ECS containers。

資料量	支援的容量	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 容器執行時期的一項功能，允許容器透過掛載主機檔案系統中的目錄來實現資料持久化。Docker 磁碟區驅動程式 (也稱為外掛程式) 用於整合容器磁碟區與外部儲存系統。Docker 磁碟區可由第三方驅動程式管理，也可由內建的 local 驅動程式管理。Docker 磁碟區的常見使用案例包括提供持續性資料磁碟區，或在同一容器執行個體上不同容器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 Docker 磁碟區 。

資料量	支援的容量	支援的作業系統	儲存持續性	使用案例
綁定掛載	Fargate、Amazon EC2、Amazon ECS 受管執行個體	Windows , Linux	暫時	綁定掛載由主機上的檔案或目錄組成 AWS Fargate，例如掛載到容器的 Amazon EC2 執行個體或。綁定掛載的常見使用案例包括：在同一任務中，將來源容器的磁碟區與其他容器公用；或在一個或多個容器中，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 。

將 Amazon EBS 磁碟區與 Amazon ECS 搭配使用

Amazon Elastic Block Store (Amazon EBS) 磁碟區為資料密集型工作負載提供高可用性、具成本效益、持久且高效能的區塊儲存服務。Amazon EBS 磁碟區可與 Amazon ECS 任務搭配使用，適用於高輸送量及交易密集型的應用程式。如需有關 Amazon EBS 磁碟區的詳細資訊，請參閱 Amazon EBS User Guide 中的 [Amazon EBS volumes](#)。

連接至 Amazon ECS 任務的 Amazon EBS 磁碟區由 Amazon ECS 代表您管理。在獨立任務啟動期間，您可以提供相應組態，用於將一個 EBS 磁碟區連接至任務。在服務建立或更新期間，您可以提供相應組態，用於為 Amazon ECS 服務管理的每項任務分別連接一個 EBS 磁碟區。您可以設定新的空白磁碟區進行連接，也可以使用快照從現有磁碟區載入資料。

Note

在使用快照設定磁碟區時，您可以指定 `volumeInitializationRate` (單位為 MiB/s)，從而以該速率從快照擷取資料，來建立在可預測的時間內完整初始化的磁碟區。如需有關磁碟區初始化的詳細資訊，請參閱 Amazon EBS User Guide 中的 [Initialize Amazon EBS volumes](#)。如需有關設定 Amazon EBS 磁碟區的詳細資訊，請參閱[在 Amazon ECS 任務定義中將磁碟區組態延至啟動時進行與在部署 Amazon ECS 時指定 Amazon EBS 磁碟區組態](#)。

磁碟區組態會透過任務定義中的 `configuredAtLaunch` 參數，延至啟動時進行。透過在啟動時而非在任務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 磁碟區設定限制的任務定義。然後，您可以在不同的執行時期環境中重複使用任務定義。例如，您可以在部署期間為生產環境工作負載提供比預生產環境更高的輸送量。

連接至任務的 Amazon EBS 磁碟區可以使用 AWS Key Management Service (AWS KMS) 金鑰加密，以保護您的資料。如需詳細資訊，請參閱 [加密儲存於連接至 Amazon ECS 任務之 Amazon EBS 磁碟區中的資料](#)。

若要監控磁碟區效能，您也可以使用 Amazon CloudWatch 指標。如需有關 Amazon EBS 磁碟區 Amazon ECS 指標的詳細資訊，請參閱 [Amazon ECS CloudWatch 指標](#)與 [Amazon ECS Container Insights metrics](#)。

所有支援 Amazon ECS 的商業與中國 [AWS 區域](#)，皆支援將 Amazon EBS 磁碟區連接至任務。

支援的作業系統與容量

下表提供支援的作業系統與容量組態。

Capacity	Linux	Windows
Fargate	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 平台版本 。	不支援
EC2	Amazon EBS 磁碟區支援用於託管在使用 Amazon ECS 最	使用 Amazon ECS 最佳化 Amazon Machine Image (AMI)

Capacity	Linux	Windows
	佳化 Amazon Machine Image (AMI) 的 Nitro 型執行個體上的任務。如需有關執行個體類型的詳細資訊，請參閱 Amazon EC2 User Guide 中的 Instance types。 ECS 最佳化 AMI 20231219 及更新版本支援 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。	的 Nitro 型執行個體上託管的任務。如需有關執行個體類型的詳細資訊，請參閱 Amazon EC2 User Guide 中的 Instance types。 ECS 最佳化 AMI 20241017 及更新版本支援 Amazon EBS 磁碟區。如需詳細資訊，請參閱擷取 Amazon ECS 最佳化 Windows AMI 中繼資料。
Amazon ECS 受管執行個體	Amazon EBS 磁碟區支援用於託管在 Amazon ECS Linux 受管執行個體上的任務。	不支援

考量事項

使用 Amazon EBS 磁碟區時，請考量下列事項：

- 在 use1-az3 可用區域中，您無法設定要連接至 Fargate Amazon ECS 任務的 Amazon EBS 磁碟區。
- 磁性 (standard) Amazon EBS 磁碟區類型不支援用於 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 User Guide 中的 [Amazon EBS volumes](#)。
- 建立在部署時設定磁碟區的服務或獨立任務時，需用到 Amazon ECS 基礎結構 IAM 角色。您可以將 AWS 託管的 AmazonECSInfrastructureRolePolicyForVolumes IAM 政策連接至該角色，也可參照該托管政策建立並連接自己的政策，該政策需具有符合特定需求的許可。如需詳細資訊，請參閱[Amazon ECS 基礎結構 IAM 角色](#)。
- 每項 Amazon ECS 任務最多僅能連接一個 Amazon EBS 磁碟區，且必須是全新磁碟區。現有 Amazon EBS 磁碟區無法連接至任務。但在部署時，可透過現有磁碟區的快照來設定新的 Amazon EBS 磁碟區。
- 若要將 Amazon EBS 磁碟區與 Amazon ECS 服務搭配使用，部署控制器必須是 ECS。使用此部署控制器時，滾動部署與藍/綠部署策略均受支援。

- 若要讓任務中的容器寫入掛載的 Amazon EBS 磁碟區，容器必須具有適當的檔案系統許可。當您在容器定義中指定非根使用者時，Amazon ECS 會自動使用群組型許可來設定磁碟區，以允許指定的使用者讀取和寫入磁碟區。如果未指定使用者，容器會以根身分執行，並具有磁碟區的完整存取權。
- Amazon ECS 會自動將 AmazonECSCreated 與 AmazonECSManaged 保留標籤新增至連接的磁碟區。如果從磁碟區中移除這些標籤，Amazon ECS 將無法代表您管理磁碟區。如需有關標記 Amazon EBS 磁碟區的詳細資訊，請參閱[標記 Amazon EBS 磁碟區](#)。如需有關標記 Amazon ECS 資源的詳細資訊，請參閱[標記 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。
- 連接至由服務管理之任務的磁碟區不會保留下來，一律會在任務終止時被刪除。
- 您無法設定要連接至在 AWS Outposts 上執行之 Amazon ECS 任務的 Amazon EBS 磁碟區。

非根使用者行為

當您在容器定義中指定非根使用者時，Amazon ECS 會自動使用群組型許可來設定 Amazon EBS 磁碟區，以允許指定的使用者讀取和寫入磁碟區。磁碟區是以下列特性掛載：

- 磁碟區由根使用者和根群組擁有。
- 群組許可設定為允許讀取和寫入存取。
- 非根使用者會新增至適當的群組以存取磁碟區。

搭配非根容器使用 Amazon EBS 磁碟區時，請遵循下列最佳實務：

- 在容器映像中使用一致的使用者 IDs (UIDs) 和群組 IDs (GIDs)，以確保一致的許可。
- 在容器映像中預先建立掛載點目錄，並設定適當的擁有權和許可。
- 在開發環境中使用 Amazon EBS 磁碟區測試您的容器，以確認檔案系統許可如預期般運作。
- 如果相同任務中的多個容器共用磁碟區，請確保它們使用相容的 UIDs/GIDs 或掛載具有一致存取期望的磁碟區。

在 Amazon ECS 任務定義中將磁碟區組態延至啟動時進行

若要設定 Amazon EBS 磁碟區以連接至任務，您必須在任務定義中指定掛載點組態，並為該磁碟區命名。您也必須將 `configuredAtLaunch` 設定為 `true`，因為無法在任務定義中設定 Amazon EBS 磁碟區進行連接。但可在部署期間設定 Amazon EBS 磁碟區進行連接。

若要使用 AWS Command Line Interface (AWS CLI) 註冊任務定義，請將範本儲存為 JSON 檔案，然後將檔案傳遞為 [register-task-definition](#) 命令的輸入。

若要使用 建立和註冊任務定義 AWS 管理主控台，請參閱 [使用主控台建立 Amazon ECS 任務定義](#)。

以下任務定義展示了 mountPoints 與 volumes 物件在任務定義中的語法。如需有關任務定義參數的詳細資訊，請參閱 [Fargate 的 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*。

Linux

```
{
  "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"
}

```

Windows

```

{
  "family": "mytaskdef",
  "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"]
  "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"
        }
      ],
      "mountPoints": [
        {
          "sourceVolume": "myEBSVolume",
          "containerPath": "drive:\\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載點。此參數會映射至 Docker API 中 create-container 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，且掛載點不能跨磁碟機使用。您必須指定掛載點，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)

掛載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一讀存取權。如果此值為 `false`，則容器可寫入磁碟區。預設值為 `false`。

對於在執行 Windows 作業系統之 EC2 執行個體上執行的任務，將該值保留為預設值 `false`。

`name`

類型：字串

必要：否

磁碟區名稱。最多可輸入 255 個字母 (大小寫)、數字、連字號 (-) 與底線 (_)。在容器定義 `mountPoints` 物件的 `sourceVolume` 參數中參考此名稱。

`configuredAtLaunch`

類型：布林值

必要：是，當您想要直接將 EBS 磁碟區連接至任務時。

指定磁碟區可否在啟動時設定。若設定為 `true`，您可以在執行獨立任務或建立/更新服務時設定磁碟區。若設定為 `false`，您無法在任務定義中提供另一個磁碟區組態。必須提供此參數並將其設定為 `true`，以設定要連接至任務的 Amazon EBS 磁碟區。

加密儲存於連接至 Amazon ECS 任務之 Amazon EBS 磁碟區中的資料

您可以使用 AWS Key Management Service (AWS KMS) 來建立和管理保護資料的密碼編譯金鑰。Amazon EBS 磁碟區是使用進行靜態加密 AWS KMS keys。以下類型的資料會經過加密：

- 儲存在磁碟區上的靜態資料
- 磁碟 I/O 資料
- 從磁碟區建立的快照
- 從加密快照建立的新磁碟區

連接至任務的 Amazon EBS 磁碟區可透過以下兩種金鑰加密：一是別名為 `alias/aws/ebs` 的預設 AWS 受管金鑰，二是磁碟區組態中指定的對稱客戶自管金鑰。AWS 帳戶 每個 的預設值 AWS 受管金鑰 都是唯一的 AWS 區域，且會自動建立。若要建立對稱客戶自管金鑰，請遵循 AWS KMS Developer Guide 中的 [Creating symmetric encryption KMS keys](#) 章節所述步驟進行。

您可以預設設定 Amazon EBS 加密，以便使用您為帳戶指定的 KMS 金鑰來 AWS 區域 加密在特定中建立並連接至任務的所有新磁碟區。如需有關 Amazon EBS 加密與預設加密的詳細資訊，請參閱 Developer Guide 中的 [Amazon EBS encryption](#)。

Amazon ECS 受管執行個體行為

您可以透過啟用加密功能來為 Amazon EBS 磁碟區加密，一種方式是啟用預設加密功能，另一種方式是在建立要加密的磁碟區時直接啟用加密功能。如需有關如何啟用預設加密功能 (帳戶層級) 的資訊，請參閱 Amazon EBS User Guide 中的 [Encryption by default](#)。

您可以設定這些金鑰的任意組合。KMS 金鑰的優先順序如下：

1. 磁碟區組態中指定的 KMS 金鑰。在磁碟區組態中指定 KMS 金鑰時，其會覆寫 Amazon EBS 預設值以及在帳戶層級指定的任何 KMS 金鑰。
2. 在帳戶層級指定的 KMS 金鑰。在為 Amazon ECS 受管儲存體指定叢集層級的加密 KMS 金鑰時，其會覆寫 Amazon EBS 預設加密，但不會覆寫磁碟區組態中指定的任何 KMS 金鑰。
3. Amazon EBS 預設加密。如果您未指定帳戶層級 KMS 金鑰，也未在磁碟區組態中指定金鑰，則會套用預設加密。如果您啟用 Amazon EBS 預設加密，則預設使用的是您為預設加密指定的 KMS 金鑰。否則，預設使用別名為 `alias/aws/ebs` 的 AWS 受管金鑰。

Note

若在磁碟區組態中將 `encrypted` 設定為 `false`，不指定帳戶層級 KMS 金鑰，但啟用 Amazon EBS 預設加密，則該磁碟區仍會使用為 Amazon EBS 預設加密指定的金鑰進行加密。

非 Amazon ECS 受管執行個體行為

您也可以在建立或更新叢集時，設定 Amazon ECS 受管儲存體的 Amazon ECS 叢集層級加密。叢集層級加密會在任務層級生效，可用於透過指定的 KMS 金鑰，對連接至特定叢集中執行之每項任務的 Amazon EBS 磁碟區進行加密。如需有關為每項任務設定叢集層級加密的詳細資訊，請參閱 Amazon ECS API reference 中的 [ManagedStorageConfiguration](#)。

您可以設定這些金鑰的任意組合。KMS 金鑰的優先順序如下：

1. 磁碟區組態中指定的 KMS 金鑰。在磁碟區組態中指定 KMS 金鑰時，其會覆寫 Amazon EBS 預設值以及在叢集層級指定的任何 KMS 金鑰。
2. 在叢集層級指定的 KMS 金鑰。在為 Amazon ECS 受管儲存體指定叢集層級的加密 KMS 金鑰時，其會覆寫 Amazon EBS 預設加密，但不會覆寫磁碟區組態中指定的任何 KMS 金鑰。
3. Amazon EBS 預設加密。如果您未指定叢集層級 KMS 金鑰，也未在磁碟區組態中指定金鑰，則會套用預設加密。如果您啟用 Amazon EBS 預設加密，則預設使用的是您為預設加密指定的 KMS 金鑰。否則，預設值為 AWS 受管金鑰 具有別名的 `alias/aws/ebs`。

Note

若在磁碟區組態中將 `encrypted` 設定為 `false`，不指定叢集層級 KMS 金鑰，但啟用 Amazon EBS 預設加密，則該磁碟區仍會使用為 Amazon EBS 預設加密指定的金鑰進行加密。

客戶受管 KMS 金鑰政策

若要使用客戶自管金鑰加密連接至任務的 EBS 磁碟區，您必須設定 KMS 金鑰政策，確保用於磁碟區組態的 IAM 角色具有使用金鑰的必要許可。金鑰政策必須同時包含 `kms:CreateGrant` 與 `kms:GenerateDataKey*` 許可。加密使用快照建立的磁碟區時，需用到 `kms:ReEncryptTo` 與 `kms:ReEncryptFrom` 許可。如果只想為要連接的新空白磁碟區進行設定與加密，您可以排除 `kms:ReEncryptTo` 與 `kms:ReEncryptFrom` 許可。

下列 JSON 程式碼片段展示了可連接至 KMS 金鑰政策的金鑰政策陳述式。使用這些陳述式將提供 Amazon ECS 使用該金鑰加密 EBS 磁碟區的存取權。若要使用這些範例政策陳述式，請以自己的資訊取代 *user input placeholders*。與往常一樣，只設定需要的許可即可。

```
{
  "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
    }
  }
}
}

```

如需有關金鑰政策與許可的詳細資訊，請參閱 AWS KMS Developer Guide 中的 [Key policies in AWS KMS](#) 與 [AWS KMS permissions](#)。如需對與金鑰許可相關的 EBS 磁碟區連接問題進行疑難排解，請參閱 [對 Amazon ECS 任務的 Amazon EBS 磁碟區連接問題進行疑難排解](#)。

在部署 Amazon ECS 時指定 Amazon EBS 磁碟區組態

在註冊任務定義 (configuredAtLaunch 參數設定為 true) 之後，即可在部署時 (執行獨立任務或建立/更新服務時) 設定 Amazon EBS 磁碟區。如需有關使用 configuredAtLaunch 參數將磁碟區組態延至啟動時進行的詳細資訊，請參閱 [在 Amazon ECS 任務定義中將磁碟區組態延至啟動時進行](#)。

若要設定磁碟區，您可以使用 Amazon ECS APIs，也可以傳遞 JSON 檔案做為下列 AWS CLI 命令的輸入：

- [run-task](#)：用於執行獨立的 ECS 任務。
- [start-task](#)：用在特定容器執行個體中執行獨立的 ECS 任務。此命令不適用於 Fargate 任務。
- [create-service](#)：用於建立新的 ECS 服務。
- [update-service](#)：用於更新現有服務。

Note

若要讓任務中的容器寫入掛載的 Amazon EBS 磁碟區，容器必須具有適當的檔案系統許可。當您在容器定義中指定非根使用者時，Amazon ECS 會自動使用群組型許可來設定磁碟區，以允許指定的使用者讀取和寫入磁碟區。如果未指定使用者，容器會以根身分執行，並具有磁碟區的完整存取權。

您也可以使用 AWS 管理主控台來設定 Amazon EBS 磁碟區。如需詳細資訊，請參閱[將應用程式作為 Amazon ECS 任務執行](#)、[建立 Amazon ECS 滾動更新部署](#)及[更新 Amazon ECS 服務](#)。

下列 JSON 程式碼片段展示了可在部署時設定的 Amazon EBS 磁碟區的所有參數。若要將這些參數用於磁碟區組態，請以自己的資訊取代 *user input placeholders*。如需有關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",  
      "volumeInitializationRate":100,  
      "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

請確保您在組態中指定的 `volumeName` 與在任務定義中指定的 `volumeName` 相同。

如需有關檢查磁碟區連接狀態的詳細資訊，請參閱[對 Amazon ECS 任務的 Amazon EBS 磁碟區連接問題進行疑難排解](#)。如需 EBS 磁碟區連接所需的 Amazon ECS 基礎設施 AWS Identity and Access Management (IAM) 角色資訊，請參閱[Amazon ECS 基礎結構 IAM 角色](#)。

以下 JSON 程式碼片段範例展示了 Amazon EBS 磁碟區組態。這些範例可以透過將程式碼片段儲存在 JSON 檔案中，並將檔案做為 AWS CLI 命令的參數（使用 `--cli-input-json file://filename` 參數）傳遞來使用。以您自己的資訊取代 *user input placeholders*。

為獨立任務設定磁碟區

下列程式碼片段展示了用於設定 Amazon EBS 磁碟區以連接至獨立任務的語法。下列 JSON 程式碼片段展示了用於設定 `volumeType`、`sizeInGiB`、`encrypted` 與 `kmsKeyId` 的語法。JSON 檔案中指定的組態將用於建立 EBS 磁碟區並將其連接至獨立任務。

```

{
  "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:11112223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
    }
  }
]
}

```

在建立服務時設定磁碟區

下列程式碼片段展示了用於設定 Amazon EBS 磁碟區以連接至由服務管理之任務的語法。這些磁碟區來源於透過 `snapshotId` 參數指定的快照，傳輸速率為 200 MiB/s。JSON 檔案中指定的組態將用於建立 EBS 磁碟區並將其連接至由服務管理的每項任務。

```

{
  "cluster": "mycluster",
  "taskDefinition": "mytaskdef",
  "serviceName": "mysvc",
  "desiredCount": 2,
  "volumeConfigurations": [
    {
      "name": "myEbsVolume",
      "managedEBSVolume": {
        "roleArn": "arn:aws:iam::1111222333:role/ecsInfrastructureRole",
        "snapshotId": "snap-12345",
        "volumeInitializationRate": 200
      }
    }
  ]
}

```

在更新服務時設定磁碟區

下列 JSON 程式碼片段展示了用於更新先前未設定 Amazon EBS 磁碟區以連接至任務之服務的語法。您必須提供任務定義修訂版的 ARN，並將 `configuredAtLaunch` 設定為 `true`。下列 JSON 程式碼

片段展示了用於設定 volumeType、sizeInGiB、throughput 與 iops 以及 filesystemType 的語法。此組態用於建立 EBS 磁碟區，並將其連接至由服務管理的每項任務。

```
{
  "cluster": "mycluster",
  "taskDefinition": "mytaskdef",
  "service": "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 程式碼片段展示了用於更新服務以不再使用 Amazon EBS 磁碟區的語法。您必須提供一個將 configuredAtLaunch 設定為 false 的任務定義 ARN，或提供一個不包含 configuredAtLaunch 參數的任務定義。您還必須提供空 volumeConfigurations 物件。

```
{
  "cluster": "mycluster",
  "taskDefinition": "mytaskdef",
  "service": "mysvc",
  "desiredCount": 2,
  "volumeConfigurations": []
}
```

Amazon EBS 磁碟區的終止政策

當 Amazon ECS 任務終止時，Amazon ECS 會依據 deleteOnTermination 值，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，在任務終止後，會刪除連接至任務的 EBS 磁碟區。對於獨立任務，您可以變更此設定，在任務終止時保留磁碟區。

 Note

連接至由服務管理之任務的磁碟區不會保留下來，一律會在任務終止時被刪除。

標記 Amazon EBS 磁碟區

您可以使用 `tagSpecifications` 物件來標記 Amazon EBS 磁碟區。透過該物件，您可以提供自訂標籤，也可以根據磁碟區是連接至獨立任務還是服務中的任務，設定標籤是從任務定義還是從服務進行傳播。一個磁碟區最多可連接 50 個標籤。

 Important

Amazon ECS 會自動將 `AmazonECSCreated` 和 `AmazonECSManaged` 保留標籤附加至 Amazon EBS 磁碟區。這表示您最多可以控制為一個磁碟區連接 48 個額外標籤。這些額外標籤可為使用者自訂標籤、ECS 受管標籤或傳播標籤。

若想為磁碟區新增 Amazon ECS 受管標籤，必須在呼叫 `UpdateService`、`CreateService`、`RunTask` 或 `StartTask` 時，將 `enableECSManagedTags` 設定為 `true`。開啟 Amazon ECS 受管標籤功能後，Amazon ECS 會自動為磁碟區新增包含叢集與服務資訊的標籤 (即 `aws:ecs:clusterName` 與 `aws:ecs:serviceName`)。如需有關標記 Amazon ECS 資源的詳細資訊，請參閱[標記 Amazon ECS 資源](#)。

下列 JSON 程式碼片段展示了用於為連接至服務中每項任務的每個 Amazon EBS 磁碟區新增使用者自訂標籤的語法。若要使用此範例建立服務，請以自己的資訊取代 *user input placeholders*。

```
{
  "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:11112223333: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。

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 與輸送量，取決於您為任務請求的 CPU 總單位數。如果您為 Fargate 任務請求 0.25、0.5 或 1 個虛擬 CPU 單位 (vCPU)，建議您設定一般用途 SSD 磁碟區 (gp2 或 gp3) 或硬碟磁碟機 (HDD) 磁碟區 (st1 或 sc1)。如果為 Fargate 任務請求超過 1 個 vCPU，則下列基準效能限制適用於連接至任務的 Amazon EBS 磁碟區。您取得的 EBS 效能可能會暫時高於下列限制，但建議您根據這些限制來規劃工作負載。

請求的 CPU 單位數 (vCPU)	基準 Amazon EBS IOPS (16 KiB I/O)	基準 Amazon EBS 輸送量 (MiBps、128 KiB I/O)	基準頻寬 (Mbps)
2	3,000	75	360
4	5,000	120	1,150

請求的 CPU 單位數 (vCPU)	基準 Amazon EBS IOPS (16 KiB I/O)	基準 Amazon EBS 輸送量 (MiBps、128 KiB I/O)	基準頻寬 (Mbps)
8	10,000	250	2,300
16	15,000	500	4 , 500

Note

在設定要連接至 Fargate 任務的 Amazon EBS 磁碟區時，Fargate 任務的 Amazon EBS 效能限制會在任務的暫時性儲存體與連接的磁碟區之間共用。

EC2 任務的 Amazon EBS 磁碟區效能

Amazon EBS 提供有不同效能特性及價格的磁碟區類型，可讓您量身打造符合應用程式需求的儲存效能與成本。如需有關效能的資訊，包括每個磁碟區的 IOPS 與每個磁碟區的輸送量，請參閱 Amazon Elastic Block Store User Guide 中的 [Amazon EBS volume types](#)。

Amazon ECS 受管執行個體任務的 Amazon EBS 磁碟區效能

Amazon EBS 提供有不同效能特性及價格的磁碟區類型，可讓您量身打造符合應用程式需求的儲存效能與成本。如需有關效能的資訊，包括每個磁碟區的 IOPS 與每個磁碟區的輸送量，請參閱 Amazon Elastic Block Store User Guide 中的 [Amazon EBS volume types](#)。

對 Amazon ECS 任務的 Amazon EBS 磁碟區連接問題進行疑難排解

您可能需要對 Amazon EBS 磁碟區連接至 Amazon ECS 任務的相關情況進行疑難排解或驗證。

檢查磁碟區連接狀態

您可以使用 AWS 管理主控台 檢視 Amazon EBS 磁碟區連接至 Amazon ECS 任務的狀態。若任務啟動但連接失敗，您也會看到用於疑難排解的狀態原因。此時，建立的磁碟區將被刪除，任務也會停止。如需有關狀態原因的詳細資訊，請參閱[Amazon EBS 磁碟區連接至 Amazon ECS 任務的狀態原因](#)。

使用主控台檢視磁碟區的連接狀態與狀態原因

1. 開啟主控台，網址為 <https://console.aws.amazon.com/ecs/v2>。
2. 在叢集頁面上，選擇任務執行所在的叢集。叢集詳細資訊頁面隨即出現。

3. 在叢集詳細資訊頁面中，選擇任務索引標籤。
4. 選擇要檢視磁碟區連接狀態的任務。若要檢查的任務已停止，可能需要使用篩選預期狀態並選擇已停止。
5. 在任務詳細資訊頁面上，選擇磁碟區索引標籤。您將能夠在連接狀態下方檢視 Amazon EBS 磁碟區的連接狀態。若磁碟區連接至任務失敗，可選擇連接狀態下方的狀態資訊，顯示失敗原因。

您也可透過 [DescribeTasks](#) API 檢視任務的磁碟區連接狀態及對應的狀態原因。

服務與任務失敗

您可能會遇到非 Amazon EBS 磁碟區特有的服務或任務失敗問題，這類問題仍可能影響磁碟區的連接。如需詳細資訊，請參閱

- [服務事件訊息](#)
- [已停止任務的錯誤代碼](#)
- [API 失敗原因](#)

容器無法寫入 Amazon EBS 磁碟區

沒有適當許可的非根使用者

當您在容器定義中指定非根使用者時，Amazon ECS 會自動設定具有群組型許可的磁碟區，以允許寫入存取。不過，如果您仍然遇到許可問題：

- 使用格式 `uid:gid` (例如)，確認您的容器定義中已正確指定 `user` 參數 `1001:1001`。
- 確保掛載磁碟區後，您的容器映像不會覆寫使用者許可。
- 透過檢查容器日誌或使用 Amazon ECS Exec 檢查執行中的容器，檢查您的應用程式是否使用預期的使用者 ID 執行。

具有許可問題的根使用者

如果您的容器定義中未指定使用者，則容器會以根身分執行，且應具有磁碟區的完整存取權。如果您遇到問題：

- 檢查容器內的掛載點，確認磁碟區已正確掛載。
- 請確定掛載點組態中的磁碟區未設定為唯讀。

具有不同使用者的多容器任務

在具有多個容器以不同使用者身分執行的任務中，Amazon ECS 會自動管理群組許可，以允許所有指定的使用者寫入磁碟區。如果容器無法寫入：

- 確認所有需要寫入存取權的容器都已正確設定 `user` 參數。
- 檢查磁碟區是否掛載在需要存取磁碟區的所有容器中。

如需在容器定義中設定使用者的詳細資訊，請參閱 [Fargate 的 Amazon ECS 任務定義參數](#)。

Amazon EBS 磁碟區連接至 Amazon ECS 任務的狀態原因

使用下列參考來修正您在設定 Amazon EBS 磁碟區以連接至 Amazon ECS 任務 AWS 管理主控台時，在 中以狀態原因形式可能遇到的問題。如需有關在主控台中尋找這些狀態原因的詳細資訊，請參閱 [檢查磁碟區連接狀態](#)。

ECS was unable to assume the configured ECS Infrastructure Role
'arn:aws:iam::**111122223333**:role/*ecsInfrastructureRole*'. Please verify that the role being passed has the proper trust relationship with Amazon ECS

此狀態原因會出現在下列案例中。

- 您提供的 IAM 角色未連接必要的信任政策。若角色缺少必要的信任政策，Amazon ECS 將無法存取您提供的 Amazon ECS 基礎結構 IAM 角色。任務可能會卡在 DEPROVISIONING 狀態。如需有關必要信任政策的詳細資訊，請參閱 [Amazon ECS 基礎結構 IAM 角色](#)。
- 您的 IAM 使用者沒有將 Amazon ECS 基礎結構角色傳遞給 Amazon ECS 的許可。任務可能會卡在 DEPROVISIONING 狀態。若要避免此問題，您可以將 `PassRole` 許可連接至使用者。如需詳細資訊，請參閱 [Amazon ECS 基礎結構 IAM 角色](#)。
- 您的 IAM 角色沒有 Amazon EBS 磁碟區連接所需的必要許可。任務可能會卡在 DEPROVISIONING 狀態。如需有關將 Amazon EBS 磁碟區連接至任務所需特定許可的詳細資訊，請參閱 [Amazon ECS 基礎結構 IAM 角色](#)。

Note

您也可能因角色傳播延遲而看到此錯誤訊息。若等待幾分鐘後重新嘗試使用該角色仍無法修復問題，有可能是您對該角色的信任政策設定錯誤。

ECS failed to set up the EBS volume. Encountered IdempotentParameterMismatch"; "The client token you have provided is associated with a resource that is already deleted. Please use a different client token."

下列 AWS KMS 關鍵案例可能會導致 `IdempotentParameterMismatch` 訊息出現：

- 您指定的 KMS 金鑰 ARN、ID 或別名無效。在此案例中，任務可能看起來已成功啟動，但任務最終會失敗，因為會以非同步方式 AWS 驗證 KMS 金鑰。如需詳細資訊，請參閱 Amazon EC2 User Guide 中的 [Amazon EBS encryption](#)。
- 您提供的客戶自管金鑰缺少必要許可，導致 Amazon ECS 基礎結構 IAM 角色無法使用該金鑰進行加密。若要避免金鑰政策許可問題，請參閱 [Amazon EBS 磁碟區資料加密](#) 中的範例 AWS KMS 金鑰政策。

您可以設定 Amazon EventBridge，將 Amazon EBS 磁碟區事件與 Amazon ECS 任務狀態變更事件傳送至目標，例如 Amazon CloudWatch 群組。然後，您可透過這些事件識別影響磁碟區連接的特定客戶自管金鑰相關問題。如需詳細資訊，請參閱

- [如何建立 CloudWatch 日誌群組，做為 re : Post 上 EventBridge 規則的目標？](#)。AWS
- [任務狀態變更事件](#)。
- Amazon EBS User Guide 中的 [Amazon EventBridge events for Amazon EBS](#)。

ECS timed out while configuring the EBS volume attachment to your Task.

下列檔案系統格式案例會導致出現此訊息。

- 您在設定期間指定的檔案系統格式與[任務的作業系統](#)不相容。
- 您設定要從快照建立的 Amazon EBS 磁碟區，但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。

您可利用 Amazon ECS 容器代理程式日誌，就此訊息對 EC2 任務進行疑難排解。如需詳細資訊，請參閱 [Amazon ECS 日誌檔案位置](#)與 [Amazon ECS 日誌收集器](#)。

將 Amazon EFS 磁碟區與 Amazon ECS 搭配使用

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：

- 對於在 EC2 上執行的任務，Amazon EFS 檔案系統支援作為公開預覽版，隨帶有容器代理程式版本 1.35.0 的 Amazon ECS 最佳化 AMI 版本 20191212 一同推出。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 最佳化 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 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與 CPU。查詢任務中繼資料第 4 版端點時，可看見監督容器。此外，其在 CloudWatch Container Insights 中顯示為容器名稱 aws-fargate-supervisor。如需有關使用 EC2 的詳細資訊，請參閱[Amazon ECS 任務中繼資料端點第 4 版](#)。如需有關使用 Fargate 的詳細資訊，請參閱[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。
- 執行於 Amazon ECS 受管執行個體上的任務，支援使用 Amazon EFS 磁碟區。
- 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發出的所有檔案系統請求，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

在搭配 Amazon ECS 使用 Amazon EFS 時，請注意下列最佳實務建議。

Amazon EFS 磁碟區的安全與存取控制

Amazon EFS 提供存取控制功能，您可透過這些功能確保儲存在 Amazon EFS 檔案系統中的資料安全，且僅供有需求的應用程式存取。您可透過啟用靜態加密與傳輸中加密來保障資料安全。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](#)。

除資料加密外，您也可透過 Amazon EFS 限制對檔案系統的存取。EFS 提供三種實作存取控制的方式。

- **安全群組：**透過 Amazon EFS 掛載目標，您可以設定用於允許與拒絕網路流量的安全群組。您可對連接至 Amazon EFS 的安全群組進行設定，使其允許來自連接至 Amazon ECS 執行個體之安全群組的 NFS 流量 (連接埠 2049)；若使用 awsvpc 網路模式，則允許來自連接至 Amazon ECS 任務之安全群組的流量。

- IAM：您可透過 IAM 限制對 Amazon EFS 檔案系統的存取。設定後，Amazon ECS 任務若需掛載 EFS 檔案系統，必須具備用於檔案系統存取的 IAM 角色。如需詳細資訊，請參閱 Amazon Elastic File System User Guide 中的 [Using IAM to control file system data access](#)。

IAM 政策也能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 Elastic File System User Guide 中的 [Amazon EFS condition keys for clients](#)。

- Amazon EFS 存取點：Amazon EFS 存取點是應用程式專用於存取 Amazon EFS 檔案系統的進入點。您可以使用存取點為所有透過該存取點發出的檔案系統請求，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。

IAM 政策

您可透過 IAM 政策控制對 Amazon EFS 檔案系統的存取。

您可以使用檔案系統政策，為存取檔案系統的用戶端指定下列動作。

Action	Description
<code>elasticfilesystem:ClientMount</code>	提供檔案系統的唯一讀存取權。
<code>elasticfilesystem:ClientWrite</code>	在檔案系統上提供具有寫入權限。
<code>elasticfilesystem:ClientRootAccess</code>	存取檔案系統時，提供使用根使用者的功能。

您需要在政策中指定每個動作。這些政策可透過下列方式定義：

- 基於用戶端 – 將政策連接至任務角色
在建立任務定義時設定 IAM 授權選項。
- 基於資源 – 將政策連接至 Amazon EFS 檔案系統

若不存在基於資源的政策，預設會在檔案系統建立時將存取權授予所有主體 (*)。

在您設定 IAM 授權選項時，我們會合併與任務角色相關聯的政策，以及基於 Amazon EFS 資源的政策。IAM 授權選項會將任務身分 (任務角色) 與政策一同傳遞給 Amazon EFS。這讓基於 Amazon EFS

資源的政策能夠識別政策中指定的 IAM 使用者或角色背景資訊。如果您未設定該選項，Amazon EFS 資源層級政策會將 IAM 使用者識別為「匿名」。

為實現最高級別的安全性，建議在 Amazon EFS 檔案系統上同時實作三種存取控制方式。例如，您可以設定連接至 Amazon EFS 掛載點的安全群組，僅允許來自與容器執行個體或 Amazon ECS 任務相關聯的安全群組的 NFS 輸入流量。此外，即使連線來自已允許的安全群組，您仍可以設定 Amazon EFS，要求必須透過 IAM 角色才能存取檔案系統。最後，您可使用 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展示了如何透過存取點掛載 Amazon EFS 檔案系統。

```
"volumes": [
  {
    "efsVolumeConfiguration": {
      "fileSystemId": "fs-1234",
      "authorizationConfig": {
        "accessPointId": "fsap-1234",
        "iam": "ENABLED"
      },
      "transitEncryption": "ENABLED",
      "rootDirectory": ""
    },
    "name": "my-filesystem"
  }
]
```

Amazon EFS 磁碟區效能

Amazon EFS 提供兩種效能模式：一般用途與最大 I/O。一般用途模式適用於對延遲敏感的應用程式，例如內容管理系統與 CI/CD 工具。相對地，最大 I/O 檔案系統則適用於資料分析、媒體處理與機器學習等工作負載。這類工作負載需要透過數百甚至數千個容器執行平行操作，並且需要盡可能高的彙總輸送量與 IOPS。如需詳細資訊，請參閱 Amazon Elastic File System User Guide 中的 [Amazon EFS performance modes](#)。

部分對延遲敏感的工作負載，既需要最大 I/O 效能模式提供的較高 I/O 等級，也需要一般用途效能模式提供的較低延遲。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。只要工作負載與應用程式本身支援，您就能透過這種方式，將應用程式工作負載分散至所有這些檔案系統中。

Amazon EFS 磁碟區輸送量

所有 Amazon EFS 檔案系統皆有對應的計量輸送量，其數值取決於採用的輸送量模式：若是使用佈建輸送量的檔案系統，則取決於佈建輸送量的數量；若是使用爆量輸送量的檔案系統，則取決於儲

存在 EFS Standard 或 One Zone 儲存類別中的資料量。如需詳細資訊，請參閱 Amazon Elastic File System User Guide 中的 [Understanding metered throughput](#)。

Amazon EFS 檔案系統預設的輸送量模式為爆量模式。使用爆量模式時，隨著檔案系統的成長，檔案系統可用的輸送量會隨之橫向擴充或縮減。由於檔案型工作負載通常會遽增，即在特定時間段需要高輸送量，其餘時間則只需低輸送量，因此 Amazon EFS 設計了爆量機制，可在特定時間段提供高輸送量。此外，因許多工作負載以讀取為主，故讀取操作的計量比例不同於其他 NFS 操作 (如寫入操作)，兩者計量比例為 1:3。

所有 Amazon EFS 檔案系統，針對每 TB 的 Amazon EFS Standard 或 Amazon EFS One Zone 儲存空間，皆能提供 50 MB/s 的穩定基準效能。無論容量大小，所有檔案系統皆可爆量至 100 MB/s 的輸送量。而超過 1 TB EFS Standard 或 EFS One Zone 儲存空間的檔案系統，每 TB 儲存空間可爆量至 100 MB/s 輸送量。由於讀取操作採用 1:3 的計量比例，每 TiB 儲存空間可支援高達 300 MiB/s 的讀取輸送量。在將資料新增至檔案系統時，檔案系統可使用的最大輸送量會隨著 Amazon EFS Standard 儲存類別中的儲存容量，自動呈線性擴展。若需要的輸送量超過以所儲存資料量能提供的輸送量，您可將佈建輸送量設定為工作負載所需的特定量。

檔案系統輸送量會由連線至同一檔案系統的所有 Amazon EC2 執行個體共用。例如，一個可爆量至 100 MB/s 輸送量的 1 TB 檔案系統，既可以由單個 Amazon EC2 執行個體獨占 100 MB/s 輸送量，也可以由 10 個 Amazon EC2 執行個體分用 (即每個執行個體各占用 10 MB/s)。如需詳細資訊，請參閱 Amazon Elastic File System User Guide 中的 [Amazon EFS performance](#)。

最佳化 Amazon EFS 磁碟區的成本

Amazon EFS 可為您簡化儲存空間的擴展程序。在新增更多資料時，Amazon EFS 檔案系統會自動擴增。特別是在 Amazon EFS 爆量輸送量模式下，Amazon EFS 的輸送量會隨著標準儲存類別中檔案系統的容量增長而擴展。若想在額額外支付 EFS 檔案系統佈建輸送量費用的前提下提升輸送量，可將一個 Amazon EFS 檔案系統與多個應用程式共用。透過 Amazon EFS 存取點，您能在共用的 Amazon EFS 檔案系統中實作儲存隔離。如此一來，即使多個應用程式仍然共用同一檔案系統，未經授權的應用程式也無法存取相關資料。

隨著資料量增加，Amazon EFS 會協助您自動將不常存取的檔案移至成本更低的儲存類別。Amazon EFS Standard-Infrequent Access (IA) 儲存類別能降低非每日存取檔案的儲存成本。此過程不會犧牲 Amazon EFS 提供的高可用性、高持久性、彈性，以及 POSIX 檔案系統存取權。如需詳細資訊，請參閱 Amazon Elastic File System User Guide 中的 [EFS storage classes](#)。

建議使用 Amazon EFS 生命週期政策，透過自動將不常存取的檔案移至 Amazon EFS IA 儲存類別來節省成本。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，您可選擇讓 Amazon EFS 將資料跨多個可用區域複寫 (對應 Standard 儲存類別)，或在單一可用區域內以備援方式儲存資料。相較於 Amazon EFS Standard 儲存類別，Amazon EFS One Zone 儲存類別能大幅降低儲存成本。若工作負載不需要多可用區域的復原能力，建議使用 Amazon EFS One Zone 儲存類別。您可以將不常存取的檔案移至 Amazon EFS One Zone-Infrequent Access，進一步降低 Amazon EFS One Zone 儲存的成本。如需詳細資訊，請參閱 [Amazon EFS 不頻繁存取](#)。

Amazon EFS 磁碟區資料保護

使用 Standard 儲存類別的 Amazon EFS 檔案系統，會以備援方式跨多個可用區域儲存資料。如果選取 Amazon EFS One Zone 儲存類別，則會以備援方式將資料儲存於單一可用區域內。此外，Amazon EFS 在設計上可提供每年 99.999999999% (11 個 9) 的資料持久性。

如同任何環境，建立備份並建置防誤刪防護措施是最佳實務。對於 Amazon EFS 資料，該最佳實務包含正常運作、定期測試的備份 AWS Backup。依預設，建立使用 Amazon EFS One Zone 儲存類別的檔案系統時，系統會將其設定為自動備份檔案，除非您選擇停用此功能。如需詳細資訊，請參閱 Amazon Elastic File System User Guide 中的 [Backing up EFS file systems](#)。

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": "public.ecr.aws/amazonlinux/amazonlinux:latest",
      "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

了解如何搭配 Amazon ECS 使用 Amazon Elastic File System (Amazon EFS) 檔案系統。

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 叢集範例”](#)。

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。

如需有關如何建立安全群組的資訊，請參閱 Amazon EC2 User Guide 中的 [Create a security group for your Amazon EC2 instance](#)。

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

本教學課程假設您的 Amazon EFS 檔案系統、Amazon ECS 叢集、容器執行個體與任務皆位於同一 VPC 中。如需有關從不同 VPC 掛載檔案系統的詳細資訊，請參閱 Amazon EFS User Guide 中的 [Walkthrough: Mount a file system from a different VPC](#)。

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. 前往 <https://console.aws.amazon.com/ec2/> 開啟 Amazon EC2 主控台。
2. 選擇啟動執行個體。
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 User Guide 中的 [Connect to your Linux instance using SSH](#)。
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": "public.ecr.aws/docker/library/nginx:latest"
    }
  ],
  "volumes": [
    {
      "name": "efs-html",
      "efsVolumeConfiguration": {
        "fileSystemId": "fs-1324abcd",
        "transitEncryption": "ENABLED"
      }
    }
  ],
  "family": "efs-tutorial",
  "executionRoleArn": "arn:aws:iam::111122223333:role/ecsTaskExecutionRole"
}
```

Note

Amazon ECS 任務執行 IAM 角色不需要任何特定的 Amazon EFS 相關許可，即可掛載 Amazon EFS 檔案系統。依預設，如果不存在基於 Amazon EFS 資源的政策，則會在建立檔案系統時授予所有主體 (*) 存取權。

僅當在 Amazon ECS 任務定義中啟用「EFS IAM 授權」時，才需要 Amazon ECS 任務角色。啟用該功能後，必須在基於 Amazon EFS 資源的政策中允許該任務角色的身分存取 Amazon EFS 檔案系統，並應停用匿名存取。

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。

啟動服務的資源	步驟	
	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)。	

- (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。	

- 針對 Application type (應用程式類型)，選擇 Task (任務)。
- 在任務定義中，選擇您先前建立的 `efs-tutorial` 任務定義。
- 在所需任務中，輸入 1。
- 選擇建立。
- 在叢集頁面上，選擇基礎設施。
- 在容器執行個體下，選擇要連線的容器執行個體。
- 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。
- 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。

搭配 Amazon ECS 使用 FSx for Windows File Server 磁碟區

FSx for Windows File Server 提供全受管 Windows 檔案伺服器，這些伺服器由 Windows 檔案系統提供支援。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 組態上的 Windows 容器中使用 FSx for Windows File Server 磁碟區。但您可以[修改容器，使其在啟動時完成掛載](#)。

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。您可以將一個或多個 FSx for Windows File Server 檔案系統磁碟區掛載至 Amazon ECS Windows 執行個體上執行的 Amazon ECS 容器。您可於單一 Amazon ECS 任務內，在多個 Amazon ECS 容器之間共用 FSx for Windows File Server 檔案系統磁碟區。

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。該網域可以透過 [AWS Directory Service for Microsoft Active Directory](#)、內部部署 Active Directory 或 Amazon EC2 上自我託管的 Active Directory 進行託管。
- AWS Secrets Manager 秘密或 Systems Manager 參數，其中包含用於加入 Active Directory 網域和連接 FSx for Windows File Server 檔案系統的登入資料。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[了解如何為 Amazon ECS 設定 FSx for Windows File Server 檔案系統](#)。

考量事項

在使用 FSx for Windows File Server 磁碟區時，請考量下列事項：

- Windows Amazon EC2 執行個體上的 Amazon ECS 原生支援 FSx for Windows File Server 磁碟區 — Amazon ECS 透過任務定義組態自動管理掛載。

在 Linux Amazon EC2 執行個體上，Amazon ECS 無法透過任務定義自動掛載 FSx for Windows File Server 磁碟區。不過，您可以在主機層級的 Linux EC2 執行個體上手動掛載 FSx for Windows File Server 檔案共享，然後將該路徑繫結掛載到 Amazon ECS 容器中。如需詳細資訊，請參閱[從 Linux 掛載 Amazon FSx 檔案共用](#)。

⚠ Important

這是自我管理的組態。如需在 Linux 上掛載和維護 FSx for Windows File Server 檔案共享的指引，請參閱 [FSx for Windows File Server 文件](#)。

⚠ Important

在 Linux EC2 執行個體上使用手動掛載的 FSx for Windows File Server 共用時，Amazon ECS 和 FSx for Windows File Server 會獨立運作 — Amazon ECS 不會監控 Amazon FSx 掛載，而 FSx for Windows File Server 不會追蹤 Amazon ECS 任務置放或生命週期事件。您有責任確保 Amazon ECS 容器執行個體與 Amazon FSx 檔案系統之間的網路連線能力、實作掛載運作狀態檢查，以及處理重新連線邏輯，以容忍容錯移轉事件。

- FSx for Windows File Server 與 Amazon ECS 不支援 AWS Fargate。
- 在 Amazon ECS 受管執行個體上，不支援將 FSx for Windows File Server 與 Amazon ECS 搭配使用。
- 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。只有在加入 Active Directory 網域且已正確設定 Active Directory 安全群組的 EC2 執行個體上託管的任

務，才能存取 FSx for Windows File Server 檔案共用。如果安全群組設定錯誤，Amazon ECS 會使任務啟動失敗，並顯示下列錯誤訊息：unable to mount file system *fs-id*。

- FSx for Windows File Server 已與 AWS Identity and Access Management (IAM) 整合，以控制 IAM 使用者和群組可對特定 FSx for Windows File Server 資源採取的動作。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](#)。

搭配 Amazon ECS 使用 FSx for Windows File Server 的最佳實務

在搭配 Amazon ECS 使用 FSx for Windows File Server 時，請注意下列最佳實務建議。

FSx for Windows File Server 的安全與存取控制

FSx for Windows File Server 提供下列存取控制功能，您可透過這些功能確保儲存在 FSx for Windows File Server 檔案系統中的資料安全，且僅供有需求的應用程式存取。

FSx for Windows File Server 磁碟區的資料加密

FSx for Windows File Server 支援兩種形式的檔案系統加密：「傳輸中資料加密」與「靜態資料加密」。當檔案共用映射至支援 SMB 通訊協定 3.0 或更新版本的容器執行個體時，即會啟用傳輸中資料加密。靜態資料加密則會在建立 Amazon FSx 檔案系統時自動啟用。當您存取檔案系統時，Amazon FSx 會透過 SMB 加密技術自動對傳輸中的資料進行加密，無需修改應用程式即可實現。如需詳細資訊，請參閱 Amazon FSx for Windows File Server User Guide 中的[Data encryption in Amazon FSx](#)。

使用 Windows ACL 進行資料夾層級存取控制

Windows Amazon EC2 執行個體會透過 Active Directory 憑證存取 Amazon FSx 檔案共用。此類執行個體也會使用標準 Windows 存取控制清單 (ACL)，實現精細的檔案層級與資料夾層級存取控制。您可以建立多個憑證，每個憑證分別對應共用區內的特定資料夾，並映射至特定任務。

在下列範例中，任務可以使用儲存在 Secrets Manager 中的憑證存取 App01 資料夾。Amazon Resource Name (ARN) 為 1234。

```
"rootDirectory": "\\path\\to\\my\\data\\App01",  
"credentialsParameter": "arn-1234",  
"domain": "corp.fullyqualified.com",
```

在另一個範例中，任務可以使用儲存在 Secrets Manager 中的憑證存取 App02 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

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 EC2 Active Directory 託管的完整網域名稱。

儲存 FSx for Windows File Server 磁碟區憑證的方法

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秘密類別，在 AWS Secrets Manager 主控台中建立此登入資料。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](#)。

了解如何為 Amazon ECS 設定 FSx for Windows File Server 檔案系統

了解如何啟動 Amazon ECS 最佳化 Windows 執行個體，該執行個體同時託管 FSx for Windows File Server 檔案系統，以及可存取該檔案系統的容器。若要這樣做，請先建立 Directory Service AWS Managed Microsoft Active Directory。接著，您可以建立 Amazon FSx for Windows File Server 檔案系統，以及具有 Amazon EC2 執行個體與任務定義的 Amazon ECS 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。如需有關如何建立金鑰對的資訊，請參閱 Amazon EC2 User Guide 中的 [Amazon EC2 key pairs and Amazon EC2 instances](#)。

- 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。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 VPCs 和作用目錄的詳細資訊，請參閱[建立 VPC](#) 和 [建立 AWS Managed Microsoft AD 的先決條件](#)。
- IAM `ecsInstanceRole` 和 `ecsTaskExecutionRole` 許可與您的帳戶建立關聯。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 管理主控台建立叢集。

1. 請參閱 [Amazon ECS 容器執行個體 IAM 角色](#) 來檢查您是否擁有 `ecsInstanceRole`，並查看如何在沒有的情況下建立一個。
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，請確認下列 AWS 受管政策已連接至您的 `ecsInstanceRole`。連接政策 (如果尚未連接)。
 - AmazonEC2ContainerServiceforEC2Role
 - AmazonSSMMangedInstanceCore
 - AmazonSSMDirectoryServiceAccess

連接 AWS 受管政策。

- 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. 請參閱 [Amazon ECS 任務執行 IAM 角色](#) 來檢查您是否擁有 `ecsTaskExecutionRole`，並查看如何在沒有的情況下建立一個。

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，請確認下列 AWS 受管政策已連接至您的 `ecsTaskExecutionRole`。連接政策 (如果尚未連接)。使用上一節中提供的程序來連接 AWS 受管政策。

- SecretsManagerReadWrite
- AmazonFSxReadOnlyAccess
- AmazonSSMReadOnlyAccess
- AmazonECSTaskExecutionRolePolicy

步驟 2：建立 Windows Active Directory (AD)

1. 請遵循 Directory Service 管理指南中[建立 AWS Managed Microsoft AD](#) AWS 中所述的步驟。使用您在本教學課程中指定的 VPC。在建立 AWS Managed Microsoft AD 的步驟 3 中，儲存使用者名稱和管理員密碼以用於下列步驟。另外，請記下完整目錄 DNS 名稱，以供後續步驟中使用。您可以在建立 Active Directory 期間完成以下步驟。
2. 建立要在下列步驟中使用的 AWS Secrets Manager 秘密。如需詳細資訊，請參閱 [《Secrets Manager 使用者指南》](#) 中的 [Secrets Manager 入門](#)。AWS
 - 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**。對於值，輸入您在建立 AWS 受管 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	<i>sg-securitygroup</i>
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	目標
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 位址編輯傳入規則 RDP 來源，詳情請參閱 Amazon EC2 User Guide 中的 [Connect to your Windows instance using RDP](#)。

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. 在網路與安全區段中，選擇您為 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 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 Advanced details (進階詳細資料) 下的 IAM instance profile (IAM 執行個體設定檔) 欄位中，選擇 ecsInstanceRole。
- 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 [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 windows-fsx-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 Docker 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 Docker 文件中的 [Volumes](#)。

Docker 磁碟區的一些常用案例如下：

- 提供搭配容器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，並將其掛載到相同任務內的多個容器上
- 將資料磁碟區提供給由第三方驅動程式管理的任務

使用 Docker 磁碟區時的考量

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
```

如需有關如何在任務定義中指定 Docker 磁碟區的資訊，請參閱[在 Amazon ECS 任務定義中指定 Docker 磁碟區](#)。

在 Amazon ECS 任務定義中指定 Docker 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。最多可輸入 255 個字母 (大小寫)、數字、連字號 (-) 與底線 (_)。在容器定義 `mountPoints` 物件的 `sourceVolume` 參數中參考此名稱。

dockerVolumeConfiguration

類型：[DockerVolumeConfiguration](#) 物件

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 容器只支援使用 `local` 驅動程式。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)。此欄位只有在 `scope` 的值為 `shared` 才會使用。如果 `scope` 的值為 `task`，則此參數必須省略。

driver

類型：字串

必要：否

要使用的 Docker 磁碟區驅動程式。驅動程式值必須符合 Docker 提供的驅動程式名稱，因為此名稱會用於任務置放。如果使用 Docker 外掛程式 CLI 安裝驅動程式，則使用 `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，則使用 Docker 外掛程式探索來擷取驅動程式名稱。

`driverOpts`

類型：字串

必要：否

一組需傳遞的、專用於 Docker 驅動程式的選項映射。此參數會映射至 Docker 中「Create a volume」區段的 `DriverOpts`。

`labels`

類型：字串

必要：否

自訂中繼資料，新增到您的 Docker 磁碟區。

`mountPoints`

類型：物件陣列

必要：否

容器中資料磁碟區的掛載點。此參數會映射至 Docker API 中 `create-container` 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，且掛載點不能跨磁碟機使用。您必須指定掛載點，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)

掛載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`。

對於在執行 Windows 作業系統之 EC2 執行個體上執行的任務，將該值保留為預設值 `false`。

Amazon ECS 的 Docker 磁碟區範例

下列範例示範如何為容器提供暫時性儲存，以及如何為多個容器提供共用磁碟區，以及如何為容器提供 NFS 持久性儲存。

使用 Docker 磁碟區為單個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 磁碟區為多個容器提供持續性儲存

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

透過綁定掛載，主機上的檔案或目錄 (例如 Amazon EC2 執行個體) 會掛載至容器中。在 Fargate 和 Amazon EC2 執行個體上託管的任務支援綁定掛載。綁定掛載與使用它們的容器的生命週期相關聯。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 Amazon EC2 執行個體上託管的任務，可以將資料連接至主機 Amazon EC2 執行個體的生命週期，方法是在任務定義中指定 `host` 與選用的 `sourcePath` 值。如需詳細資訊，請參閱 Docker 文件中的 [Bind mounts](#)。

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。

- 在預設情況下，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) 在上託管的任務，會接收至少 20 GiB 的暫時性儲存，以進行綁定掛載。您可以透過在任務定義中指定 `ephemeralStorage` 參數來增加暫時性儲存的總量，最多可達 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。
- 對於託管在 Amazon ECS 受管執行個體上的任務，根檔案系統的部分是唯讀的。讀取/寫入繫結掛載必須使用可寫入目錄，例如 `/var` 持久性資料或 `/tmp` 暫存資料。嘗試建立其他目錄的讀取/寫入繫結掛載會導致任務無法啟動，並出現類似以下的錯誤：

```
error creating empty volume: error while creating volume path '/path': mkdir /path:
read-only file system
```

唯讀繫結掛載（在 `mountPoints` 參數 `"readOnly": true` 中以設定）可以指向主機上任何可存取的目錄。

若要檢視可寫入路徑的完整清單，您可以在 Amazon ECS 受管執行個體上執行任務，並使用檢查執行個體的掛載資料表。使用下列設定建立任務定義，以存取主機檔案系統：

```
{
  "pidMode": "host",
  "containerDefinitions": [{
    "privileged": true,
    ...
  }]
}
```

然後從容器內執行下列命令：

```
# List writable mounts
cat /proc/1/root/proc/1/mounts | awk '$4 ~ /^rw,/ || $4 == "rw" {print $2}' | sort

# List read-only mounts
cat /proc/1/root/proc/1/mounts | awk '$4 ~ /^ro,/ || $4 == "ro" {print $2}' | sort
```

Important

`privileged` 設定會授予主機上的容器延伸功能，相當於根存取。在此範例中，它用於檢查主機的掛載資料表以進行診斷。如需詳細資訊，請參閱[避免以特權方式執行容器 \(Amazon EC2\)](#)。

如需在容器中以互動方式執行命令的詳細資訊，請參閱[使用 ECS Exec 監控 Amazon ECS 容器](#)。

在 Amazon ECS 任務定義中指定綁定掛載

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 程式碼片段展示了任務定義的 `volumes`、`mountPoints` 以及 `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。最多可輸入 255 個字母 (大小寫)、數字、連字號 (-) 與底線 (_)。在容器定義 `mountPoints` 物件的 `sourceVolume` 參數中參考此名稱。

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

只有在使用託管在 Amazon EC2 執行個體或 Amazon ECS 受管執行個體上的任務時，才支援 `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載點。此參數會映射至 Docker API 中 `create-container` 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，且掛載點不能跨磁碟機使用。您必須指定掛載點，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)

掛載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`。

對於在執行 Windows 作業系統之 EC2 執行個體上執行的任務，將該值保留為預設值 `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 (1.0.0 Windows) 在上託管的任務，此參數用於擴展可用的暫時性儲存總量，超過預設數量。

您可以使用 Copilot CLI、CloudFormation、AWS SDK 或 CLI 來指定繫結掛載的暫時性儲存。

Amazon ECS 的綁定掛載範例

下列範例涵蓋對容器使用綁定掛載的常見使用案例。

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。

- 在任務定義 volumes 區段中，以名稱 database_scratch 定義綁定掛載。

```
"volumes": [  
  {  
    "name": "database_scratch"  
  }  
]
```

- 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久性儲存，請務必在任務啟動時間使用 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[將 Amazon EFS 磁碟區與 Amazon ECS 搭配使用](#)。

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": "public.ecr.aws/docker/library/nginx:latest",  
    "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範例 my-repo/httpd_dockerfile_volume 映像是使用前述的 Dockerfile 所建立。

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。

在 Amazon ECS 上管理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。相對的，具有高暫時性記憶體需求的容器可以開啟交換，以降低容器在處於低負載時發生記憶體不足錯誤的機會。

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。此參數會對應 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 User Guide 中的 [Instance Store Swap Volumes](#)，或[如何透過使用交換檔案配置記憶體，以用作 Amazon EC2 執行個體中的交換空間？](#)
- 只有指定 EC2 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，交換總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 受管執行個體的 Amazon ECS 任務定義差異

若要使用 Amazon ECS 受管執行個體，必須將任務定義設定為使用 Amazon ECS 受管執行個體啟動類型。使用 Amazon ECS 受管執行個體時需注意額外考量。

任務定義參數

使用 Amazon ECS 受管執行個體的任務，支援大多數可用的 Amazon ECS 任務定義參數。不過，部分參數在與 Amazon ECS 受管執行個體任務搭配使用時有特定的行為或限制。

下列任務定義參數在 Amazon ECS 受管執行個體任務中無效：

- disableNetworking
- dnsSearchDomains

- `dnsServers`
- `dockerLabels`
- `dockerSecurityOptions`
- `dockerVolumeConfiguration`
- `ephemeralStorage`
- `extraHosts`
- `fsxWindowsFileServerVolumeConfiguration`
- `hostname`
- `inferenceAccelerator`
- `ipcMode`
- `links`
- `maxSwap`
- `proxyConfiguration`
- `sharedMemorySize`
- `sourcepath` 磁碟區
- `swappiness`
- `tmpfs`

下列任務定義參數在 Amazon ECS 受管執行個體任務中有效，但應注意其限制：

- `networkConfiguration` – Amazon ECS 受管執行個體任務使用 `awsvpc` 或 `host` 網路模式。
- `placementConstraints` – 支援下列條件約束屬性。
 - `ecs.subnet-id`
 - `ecs.availability-zone`
 - `ecs.instance-type`
 - `ecs.cpu-architecture`
- `requiresCompatibilities` – 必須包含 `MANAGED_INSTANCES`，才能確保任務定義與 Amazon ECS 受管執行個體相容。
- `resourceRequirement` – `InferenceAccelerator` 不受支援。
- `operatingSystemFamily` – Amazon ECS 受管執行個體使用 `LINUX`。

- `volumes` - 搭配使用繫結掛載時 `sourcePath`，路徑必須指向主機上的可寫入目錄。Amazon ECS 受管執行個體檔案系統的部分為唯讀。可寫入目錄包括 `/var` 和 `/tmp`。如需詳細資訊，請參閱[搭配 Amazon ECS 使用綁定掛載](#)。

為確保任務定義通過驗證，可與 Amazon ECS 受管執行個體搭配使用，您可以在註冊任務定義時指定下列項目：

- 在 AWS 管理主控台，對於需要相容性欄位中，指定 `MANAGED_INSTANCES`。
- 在 AWS CLI，指定 `--requires-compatibilities` 選項。
- 在 Amazon ECS API 中，指定 `requiresCompatibilities` 標記。

Fargate 的 Amazon ECS 任務定義差異

若要使用 Fargate，必須將任務定義設定為使用 Fargate 啟動類型。使用 Fargate 時需注意額外考量。

任務定義參數

使用 Fargate 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。
- `networkConfiguration` - Fargate 任務一律使用 `awsvpc` 網路模式。

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 管理主控台，對於需要相容性欄位中，指定 `FARGATE`。
- 在 `中` AWS CLI，指定 `--requires-compatibilities` 選項。
- 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 容器僅使用主機作業系統中的核心與核心組態。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 工作負載的任務定義”](#)。

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。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。您可以在任務定義中將記憶體值指定為以 MiB 或 GB 為單位的字串。例如，可指定 3072 MiB 作為記憶體值，也可指定 3 GB 作為記憶體值。您可以在 JSON 檔案中將 CPU 值指定為以 CPU 為單位或虛擬 CPU (vCPU) 為單位的字串。例如，可指定 1024 個 CPU 單位作為 CPU 值，也可指定 1 vCPU 個 vCPU 作為 CPU 值。

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
 Note 此選項需要 Linux 平台 1.4.0 或更新版本。		
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路介面。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介面，以及指定一或多個安全群組來套用至網路介面。

如果使用公有子網路，請決定是否提供網路介面的公有 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介面。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。如需有關 Amazon ECR 介面端點的詳細資訊，請參閱 Amazon Elastic Container Registry User Guide 中的 [Amazon ECR interface VPC endpoints \(AWS PrivateLink\)](#)。

以下是適用於 Fargate 服務之 `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 65535，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](#)。

記錄

事件記錄

Amazon ECS 記錄 EventBridge 採取的動作。您可以使用適用於 EventBridge 的 Amazon ECS 事件，接收有關您的 Amazon ECS 叢集、服務和任務目前狀態的近乎即時通知。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 EventBridge 自動化因應 Amazon ECS 錯誤](#)。

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 和 中的任務詳細資訊中查看時間戳記，AWS 管理主控台 方法是在 AWS CLI 和 SDKs 中描述任務。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，將日誌資訊傳送給 Amazon CloudWatch Logs。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"
}
```

如需在任務定義中使用 awslogs 日誌驅動程式將您的容器日誌傳送到 CloudWatch Logs 的詳細資訊，請參閱[將 Amazon ECS 日誌傳送至 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 日誌傳送至 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 日誌驅動程式](#)。

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，提供具成本效益、持久且高效能的區塊儲存服務。如需詳細資訊，請參閱[將 Amazon EBS 磁碟區與 Amazon ECS 搭配使用](#)。
- 適用於持久性儲存的 Amazon EFS 磁碟區。如需詳細資訊，請參閱[將 Amazon EFS 磁碟區與 Amazon ECS 搭配使用](#)。
- 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 (SOCI) 是一種由開發的開放原始碼技術 AWS，可透過延遲載入容器映像來更快速地啟動容器。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，請參閱 [引入 Seekable OCI 用於延遲載入容器映像](#)。

希望使用 SOCI 的客戶，只能使用 SOCI 索引資訊清單 v2。先前在 Fargate 上使用 SOCI 的現有客戶可以繼續使用 SOCI 索引資訊清單 v1，但強烈建議這些客戶遷移至 SOCI 索引資訊清單 v2。SOCI 索引資訊清單 v2 會在容器映像與其 SOCI 索引之間建立明確關係，確保部署的一致性。

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。
- 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。
- 對於 SOCI 索引資訊清單 v2，產生 SOCI 索引資訊清單會修改容器映像資訊清單，因為我們會為 SOCI 索引新增註釋。此舉會產生新的容器映像摘要，但容器映像檔案系統層的內容會保持不變。
- 對於 SOCI 索引資訊清單 v2，在容器映像已儲存於容器映像儲存庫後，若再產生 SOCI 索引，則需重新推送該容器映像。重新推送容器映像不會因為重複檔案系統層而增加儲存成本，因為此過程僅會上傳新的資訊清單檔案。
- 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。
- 若想防止容器映像延遲載入，需在不連接 SOCI 索引的情況下，重新推送該容器映像。

建立 Seekable OCI 索引

若要讓容器映像延遲載入，需建立 SOCI 索引 (中繼資料檔案)，並將其與容器映像一起儲存於容器映像儲存庫中。若要建立並推送 SOCI 索引，您可以使用 GitHub 上的開放原始碼 [soci-snapshotter CLI 工具](#)。或者，您可以部署 CloudFormation AWS SOCI 索引建置器。這是無伺服器解決方案，可在容器映像推送至 Amazon ECR 時自動建立並推送 SOCI 索引。如需有關解決方案與安裝步驟的詳細資訊，請參閱 GitHub 上的 [CloudFormation AWS SOCI Index Builder](#)。CloudFormation AWS SOCI 索引建置器是一種自動化 SOCI 入門的方法，而開放原始碼社交工具在索引產生方面具有更大的靈活性，並且能夠在您的持續整合和持續交付 (CI/CD) 管道中整合索引產生。

 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，則此欄位會設定為 `soci`。若要驗證容器映像是否已連接 SOCI 索引資訊清單 v2，您可以透過 AWS CLI 從 Amazon ECR 擷取映像索引。

```
IMAGE_REPOSITORY=r
IMAGE_TAG=latest

aws ecr batch-get-image \
  --repository-name=$IMAGE_REPOSITORY \
  --image-ids imageTag=$IMAGE_TAG \
  --query 'images[0].imageManifest' --output text | jq -r '.manifests[] |
select(.artifactType=="application/vnd.amazon.soci.index.v2+json")'
```

若要驗證容器映像是否已連接 SOCI 索引資訊清單 v1，您可以使用 OCI Referrers API。

```
ACCOUNT_ID=111222333444
AWS_REGION=us-east-1
IMAGE_REPOSITORY=nginx-demo
IMAGE_TAG=latest
IMAGE_DIGEST=$(aws ecr describe-images --repository-name $IMAGE_REPOSITORY --image-ids
  imageTag=$IMAGE_TAG --query 'imageDetails[0].imageDigest' --output text)
ECR_PASSWORD=$(aws ecr get-login-password)

curl \
  --silent \
  --user AWS:$ECR_PASSWORD \
  https://$ACCOUNT_ID.dkr.ecr.$AWS_REGION.amazonaws.com/v2/$IMAGE_REPOSITORY/
  referencers/$IMAGE_DIGEST?artifactType=application%2Fvnd.amazon.soci.index.v1%2Bjson | jq
  -r '.'
```

EC2 Windows 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，並不支援所有可用的 Amazon ECS 任務定義參數。有些參數完全不受支援，而有些參數的行為會有所不同。

Amazon EC2 Windows 任務定義不支援下列任務定義參數：

- 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 執行個體上任務的 IAM 角色功能需要額外的設定，但其中大部分設定步驟與 Linux 容器執行個體上任務的 IAM 角色設定類似。如需更多資訊，請參閱[the section called “Amazon EC2 Windows 執行個體的額外組態”](#)。

使用主控台建立 Amazon ECS 任務定義

建立任務定義，以便您定義以任務或服務方式執行的應用程式。

在為外部啟動類型建立任務定義時，您需要使用 JSON 編輯器建立任務定義，並將 `requireCapabilities` 參數設定為 `EXTERNAL`。

您可以使用控制台建立任務定義，也可以透過指定 JSON 檔案建立任務定義。使用 JSON 編輯器時，您可以讓 Amazon Q 提供建議。如需詳細資訊，請參閱[使用 Amazon Q Developer 在 Amazon ECS 主控台中提供任務定義建議](#)

JSON 驗證

Amazon ECS 主控台 JSON 編輯器會對 JSON 檔案的以下方面進行驗證：

- 該檔案為有效的 JSON 檔案。
- 該檔案不包含任何無關的金鑰。
- 該檔案包含 `familyName` 參數。
- `containerDefinitions` 下至少會有一個項目。

CloudFormation 堆疊

下列行為適用於 2023 年 1 月 12 日之前在新版 Amazon ECS 主控台中建立的任務定義。

建立任務定義後，Amazon ECS 主控台會自動建立名稱開頭為 `ECS-Console-V2-TaskDefinition-` 的 CloudFormation 堆疊。如果您使用 AWS CLI 或 AWS 開發套件取消註冊任務定義，則必須手動刪除任務定義堆疊。如需詳細資訊，請參閱 CloudFormation User Guide 中的[Deleting a stack](#)。

在 2023 年 1 月 12 日之後建立的任務定義不會自動建立 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，選擇建立新任務定義。
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。主控台預設為 AWS Fargate (即無伺服器模式)。Amazon ECS 會使用此值來執行驗證，確保任務定義參數對基礎結構類型有效。
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 架構上執行任務，請選擇 Linux/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
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

 **Note**
 此選項需要 Linux 平台 1.4.0 或更新版本。

CPU 數值	記憶體數值	AWS Fargate 支援的作業系統
16384 (16vCPU)	介於 32 GB 與 120 GB 之間，以 8 GB 為單位遞增	Linux

 Note

此選項需要 Linux 平台 1.4.0 或更新版本。

對於使用 EC2 執行個體或外部執行個體的任務，支援的任務 CPU 數值範圍介於 128 個 CPU 單位 (0.125 個 vCPU) 至 196608 個 CPU 單位 (192 個 vCPU) 之間。

若要指定以 GB 為單位的記憶體值，請在值之後輸入 GB。例如，若要將記憶體值設定為 3GB，請輸入 3GB。

 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](#)。

若選擇橋接模式，請在連接埠映射下方的主機連接埠欄位中，輸入要為容器預留的容器執行個體連接埠號碼。

9. (選用) 展開任務角色區段，以設定任務的 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呼叫 AWS API 操作的許可。
 - 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”](#)。如果您不需要該角色，請選擇無。

10. (選用) 展開任務置放區段，新增置放限制條件。任務置放限制條件可讓您使用內建或自訂屬性，篩選用於置放任務的容器執行個體。

11. (選用) 展開故障注入區段，啟用故障注入功能。您可以透過故障注入功能，測試應用程式在特定故障案例中的應對表現。
12. 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。僅可使用 Amazon ECR 公共映像庫登錄檔名稱來指定 Amazon ECR 公共映像庫登錄檔中的映像。例如，若指定 `public.ecr.aws/ecs/amazon-ecs-agent:latest`，則會使用在 Amazon ECR 公共映像庫上託管的 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` 格式來指定儲存庫。
 - 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. 對於必要容器，如果任務定義已定義兩個或更多容器，則可指定容器是否應被視為必要容器。如果某個容器被標記為必要，則若該容器停止，任務也將停止。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一讀存取權限，請在唯一讀根檔案系統中選取唯一讀。
- g. (選用) 若要定義與資源配置限制下的任務層級值不同的容器層級 CPU、GPU 和記憶體限制，請執行下列動作：
 - 在 CPU 欄位中，輸入 Amazon ECS 容器代理程式為容器預留的 CPU 單位數量。
 - 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 工作負載的任務定義”](#)。

- 在記憶體硬性限制欄位中，輸入要分配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。
- 由於 Docker 20.10.0 或更新版本常駐程式會為容器預留最低 6 MiB 的記憶體，因此為容器指定的記憶體不可少於 6 MiB。

由於 Docker 19.03.13-ce 或更早版本常駐程式會為容器預留最低 4 MiB 的記憶體，因此為容器指定的記憶體不可少於 4 MiB。

- 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。

Windows 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。您可以使用鍵值對單獨指定環境變數，也可以透過指定 Amazon S3 儲存貯體中託管的環境變數檔案來批量指定環境變數。如需有關如何設定環境變數檔案格式的資訊，請參閱 [將個別環境變數傳遞至 Amazon ECS 容器](#)。

在指定秘密儲存的環境變數時，請在金鑰欄位中輸入秘密名稱。然後，在 ValueFrom 欄位中輸入 Systems Manager Parameter Store 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式都有要指定的日誌驅動程式選項。預設選項會將容器日誌傳送至 Amazon CloudWatch Logs。其他日誌驅動程式選項是透過使用設定 AWS FireLens。如需詳細資訊，請參閱[將 Amazon ECS 日誌傳送至 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 – 將任務設定為將容器日誌傳送至 CloudWatch Logs。系統會提供預設日誌驅動程式選項，用來代表您建立 CloudWatch 日誌群組。若要指定不同的日誌群組名稱，請變更驅動程式選項值。
- 將日誌匯出至 Splunk：設定任務以將容器日誌傳送至 Splunk 驅動程式，此驅動程式會將日誌傳送至遠端服務。您必須輸入 Splunk Web 服務的 URL。Splunk 權杖被指定為秘密選項，因其可被視為敏感資料。
- 將日誌匯出至 Amazon Data Firehose – 將任務設定為將容器日誌傳送至 Firehose。系統會提供預設日誌驅動程式選項，用於將日誌傳送至 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。

- 將日誌匯出至 Amazon Kinesis Data Streams – 將任務設定為將容器日誌傳送至 Amazon Kinesis Data Streams。系統會提供預設日誌驅動程式選項，用於將日誌傳送至 Kinesis Data Streams 串流。若要指定不同的串流名稱，請變更驅動程式選項值。
 - 將日誌匯出至 Amazon OpenSearch Service – 將任務設定為將容器日誌傳送至 OpenSearch Service 網域。務必提供日誌驅動程式選項。
 - 將日誌匯出至 Amazon S3 – 將任務設定為將容器日誌傳送至 Simple Storage Service (Amazon S3) 儲存貯體。系統會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	
重新啟動政策 這些選項會定義重新啟動政策，可在容器結束時重新啟動容器。	展開重新啟動政策區段，然後設定下列項目： • 若要為容器啟用重新啟動政策，請開啟啟用重新啟動政策。 • 在忽略的結束代碼欄位中，指定以逗號區隔的整數型容器結束代碼清單。若容器以任一指定結束代碼結束，Amazon ECS 不會嘗試重新啟動該容器。如果未指定任何值，Amazon ECS 不會忽略任何結束代碼。 • 在重設嘗試週期欄位中，指定一個整數時間週期 (以秒為單位)，表示容器結束後，必須持續執行該時間週期才能嘗試重新啟動。Amazon ECS 僅能在每重設嘗試週期秒嘗試重新啟動容器一次。如果未指定任何值，容器必須持續執行 300 秒後，才能嘗試重新啟動。	

若要設定此選項	執行此作業	
HealthCheck 這些是確定容器運作狀態是否良好的命令。如需詳細資訊，請參閱使用容器運作狀態檢查來判斷 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。您可以使用 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。 • 在 Retries (重試次數) 欄位中，輸入失敗時重試	

若要設定此選項	執行此作業	
	運作狀態檢查命令的次數。有效值介於 1 到 10 之間。	
啟動相依性排序 此選項定義容器啟動和關閉的相依性。容器可以包含多個相依性。	展開啟動相依性順序，然後設定下列項目： - 選擇新增容器相依性。 - 在容器中，選擇容器。 - 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。	
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的相依性之前所等待的時間，請在啟動逾時欄位中輸入秒數。 - 若要設定容器在未正常自行結束的情況下停止之前所等待的時間，請在停止逾時欄位中輸入秒數。	

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非完整主機名稱，請在 DNS 搜尋網域欄位中逐行輸入每個網域。 模式是 <code>^[a-zA-Z0-9-.]{0,253}[a-zA-Z0-9-9]\$</code>。 - 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 <code>/etc/hosts</code> 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 組態 這些會覆寫 Dockerfile 中的值。	展開 Docker 組態區段，然後設定下列項目： • 在命令中，輸入容器的可執行命令。 此參數會映射至 Docker Remote API 中 Create a container 區段的 Cmd，以及 docker run 的 COMMAND 選項。此參數會覆寫 Dockerfile 中的 CMD 指令。 • 在進入點欄位中，輸入傳遞至容器的 Docker ENTRYPOINT。 此參數會映射至 Docker Remote API 中 Create a container 區段的 Entrypoint，以及 docker run 的 --entrypoint 選項。此參數會覆寫 Dockerfile 中的 ENTRYPOINT 指令。 • 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。 此參數會映射至 Docker Remote API 中 Create	

若要設定此選項	執行此作業
	a container 區段的 <code>WorkingDir</code>，以及 <code>docker run</code> 的 <code>--workdir</code> 選項。此參數會覆寫 Dockerfile 中的 <code>WORKDIR</code> 指令。
資源限制 (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 <code>Ulimits</code> 以及 docker run 的 <code>--ulimit</code> 選項。	展開資源限制 (ulimits) 區段，然後選擇新增 ulimit。在限制名稱欄位中，選擇限制類型。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 <code>Labels</code> 以及 docker run 的 <code>--label</code> 選項。	展開 Docker 標籤區段，選擇新增鍵值對，然後輸入鍵與值。 若要新增其他 Docker 標籤，請選擇新增鍵值對。

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。

13. (選用) 儲存區段用於為託管於 Fargate 上的任務擴展暫時性儲存容量。您也可以使用此區段來新增任務的資料磁碟區組態。

- 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。

14. (選用) 若要為任務定義新增資料磁碟區組態，請選擇新增磁碟區，然後遵循以下步驟。

- 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. 在磁碟區組態欄位中，選擇是要在建立任務定義時設定磁碟區，還是在部署期間設定磁碟區。

 Note

在建立任務定義時，可設定的磁碟區包括綁定掛載、Docker、Amazon EFS 與 Amazon FSx for Windows File Server。在執行任務或建立/更新服務時，可於部署階段設定的磁碟區包括 Amazon EBS。

- c. 在磁碟區類型欄位中，選擇與您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	
繫結掛載	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. 在驅動程式欄位中，輸入 Docker 磁碟區組態。Windows 容器只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	a. 在檔案系統 ID 中，選擇 FSx for Windows File Server 檔案系統 ID。 b. 在根目錄欄位中，輸入 FSx for Windows File Server 檔案系統中的目錄，此目錄將會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 Secrets Manager 秘密的 Amazon Resource Name (ARN)。 • 若要使用 AWS Systems Manager，請輸入 Systems Manager 參數的 Amazon Resource Name (ARN)。 d. 在網域欄位中，輸入由 AWS Directory Service for Microsoft Active Directory (AWS Managed Microsoft AD) 目錄或自我託管 EC2 Active Directory 託管的完整網域名稱。	

磁碟區類型	步驟	
	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。 - 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)。	

15. 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。

16. (選用) 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您的任務指標，並傳送至 Amazon CloudWatch 或 Amazon Managed Service for Prometheus。選取此選項時，Amazon ECS 會建立 AWS

Distro for OpenTelemetry預先設定為傳送應用程式指標的容器附屬。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. 若選取 Amazon CloudWatch，您的自定應用程式指標將作為自定指標路由至 CloudWatch。如需詳細資訊，請參閱[將應用程式指標匯出至 Amazon CloudWatch](#)。

 Important

若將應用程式指標匯出至 Amazon 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[AWS Distro for OpenTelemetry 與 Amazon CloudWatch 整合所需的 IAM 許可](#)。

- 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。在工作空間遠端寫入端點欄位中，輸入 Prometheus 工作空間的遠端寫入端點 URL。在抓取目標欄位中，輸入 AWS Distro for OpenTelemetry 收集器可用於抓取指標資料的主機與連接埠。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 Distro for OpenTelemetry 與 Amazon Managed Service for Prometheus 整合所需的 IAM 許可](#)。

- c. 若選取 Amazon Managed Service and Prometheus (OpenTelemetry 檢測)，則任務層級 CPU、記憶體、網路和儲存指標以及自訂應用程式指標，都將路由至 Amazon Managed Service for Prometheus。在工作空間遠端寫入端點欄位中，輸入 Prometheus 工作空間的遠端寫入端點 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 Distro for](#)

[OpenTelemetry 與 Amazon Managed Service for Prometheus 整合所需的 IAM 許可。](#)

17. (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。

18. 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. 在建立新任務定義功能表上，選擇使用 JSON 建立新任務定義。
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 Amazon Q Developer 在 Amazon ECS 主控台中提供任務定義建議

在 Amazon ECS 主控台中使用 JSON 編輯器建立任務定義時，您可以透過 Amazon Q Developer 為任務定義提供 AI 產生的程式碼建議。

您可以使用內嵌聊天功能，讓 Amazon Q Developer 透過對話介面產生、解釋或重構任務定義 JSON。您可以在任務定義的任何位置插入產生的建議，並自行決定是否接受提議的變更。Amazon ECS 還增強了現有的內嵌建議功能，可利用 Amazon Q Developer 的智慧能力。

在使用 JSON 編輯器建立任務定義時，您可以讓 Amazon Q Developer 提供建議，協助更快速地建立任務定義。您可以啟用基於屬性的內嵌建議功能，也可以藉助 Amazon Q Developer 建議自動完成範例程式碼的完整區塊。

您可在支援 Amazon Q Developer 的區域中使用此功能。如需詳細資訊，請參閱 [AWS Services by Regions](#)。

先決條件

先決條件如下：

- 除主控台許可外，在主控台中建立任務定義的使用者，若要獲取建議，必須具備 `codewhisperer:GenerateRecommendations` 許可；若要使用內嵌聊天功能，則需具備 `q:SendMessage` 許可。如需詳細資訊，請參閱 [透過 Amazon Q Developer 在主控台中提供建議所需的許可](#)。

程序

- 開啟主控台，網址為 <https://console.aws.amazon.com/ecs/v2>。
- 在導覽窗格中，選擇 Task Definitions (任務定義)。
- 在建立新任務定義功能表上，選擇使用 JSON 建立新任務定義。

建立任務定義頁面隨即開啟。

主控台提供下列預設範本。

```
{
  "requiresCompatibilities": [
    "FARGATE"
  ],
  "family": "",
  "containerDefinitions": [
    {
      "name": "",
      "image": "",
      "essential": true
    }
  ],
  "volumes": [],
  "networkMode": "awsvpc",
  "memory": "3 GB",
  "cpu": "1 vCPU",
  "executionRoleArn": "arn:aws:iam::123456789012:role/ecsTaskExecutionRole"
}
```

4. 在 Amazon Q 內嵌建議快顯視窗中，選擇允許。

如果關閉快顯視窗，您可以在齒輪圖示下啟用 Amazon Q。

5. 在 JSON 編輯器方塊中，編輯 JSON 文件。

若要讓 Amazon Q 建立並填入參數，請輸入註解，註明要新增的內容。在以下範例中，註解內容將觸發 Amazon Q 產生粗體標示的程式碼行。

```
{
  "requiresCompatibilities": [
    "FARGATE"
  ],
  "family": "",
  "containerDefinitions": [
    {
      "name": "",
      "image": "",
      "essential": true
    },
    // add an nginx container using an image from Public ECR, with port 80
    open, and send logs to CloudWatch log group "myproxy"
    {
      "name": "nginx",
      "image": "public.ecr.aws/nginx/nginx:latest",
      "essential": true,
      "portMappings": [
        {
          "containerPort": 80,
          "hostPort": 80,
          "protocol": "tcp"
        }
      ],
      "logConfiguration": {
        "logDriver": "awslogs",
        "options": {
          "awslogs-group": "myproxy",
          "awslogs-region": "us-east-1",
          "awslogs-stream-prefix": "nginx"
        }
      }
    }
  ]
}
```

```
"volumes": [],  
"networkMode": "awsvpc",  
"memory": "3 GB",  
"cpu": "1 vCPU",  
"executionRoleArn": "arn:aws:iam::123456789012:role/ecsTaskExecutionRole"  
}
```

6. 若要使用內嵌聊天功能，可反白顯示這些行，然後選擇星號圖示。

Amazon Q Developer 聊天方塊隨即顯示。

此時輸入您的請求。

Amazon Q Developer 將產生然後更新 JSON 檔案。

若要接受變更，請選擇全部接受

7. 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
- 基礎設施需求
- 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

使用 JSON 編輯器時，您可以讓 Amazon Q 提供建議。如需詳細資訊，請參閱[使用 Amazon Q Developer 在 Amazon ECS 主控台中提供任務定義建議](#)

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。您可以將任務定義相容性更新為 AWS Fargate、受管執行個體 或 Amazon EC2 執行個體。
7. 驗證資訊，然後選擇更新。
8. 如果您的任務定義用於服務，請使用更新的任務定義更新您的服務。如需詳細資訊，請參閱[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 任務定義修訂版

若希望在執行任務或更新服務時，某個任務定義修訂版不再顯示在 ListTaskDefinition API 呼叫或主控台中，可取消註冊該修訂版。

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。

CloudFormation 堆疊

下列行為適用於 2023 年 1 月 12 日之前在新版 Amazon ECS 主控台中建立的任務定義。

建立任務定義後，Amazon ECS 主控台會自動建立名稱開頭為 ECS-Console-V2-TaskDefinition- 的 CloudFormation 堆疊。如果您使用 AWS CLI 或 AWS 開發套件取消註冊任務定義，則必須手動刪除任務定義堆疊。如需詳細資訊，請參閱 CloudFormation User Guide 中的 [Deleting a stack](#)。

在 2023 年 1 月 12 日之後建立的任務定義不會自動建立 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 任務定義修訂版

若不再需要 Amazon ECS 中的特定任務定義修訂版，則可刪除該任務定義修訂版。

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處於 DELETE_IN_PROGRESS 狀態，則任務定義名稱將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

如果存在依賴任務定義修訂版的任何 Amazon ECS 資源，任務定義刪除請求將無法完成。以下資源可能會阻止任務定義遭刪除：

- Amazon ECS 獨立任務 – 需要任務定義才能讓任務保持良好的運作狀態。
- Amazon ECS 服務任務 – 需要任務定義才能讓任務保持良好的運作狀態。
- Amazon ECS 服務部署與任務集 – 當為 Amazon ECS 部署或任務集啟動擴展事件時，需要任務定義。

如果您的任務定義仍處於 DELETE_IN_PROGRESS 狀態，您可以使用 主控台或 AWS CLI 來識別，然後停止封鎖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 服務部署與任務集 – 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. 在任務定義名稱頁面上，選取要刪除的修訂版，然後選擇動作、刪除。

如果無法使用刪除，必須取消註冊任務定義。

6. 驗證刪除確認方塊中的資訊，然後選擇刪除以完成操作。

Amazon ECS 任務定義使用案例

進一步了解如何撰寫各種 AWS 服務和功能的任務定義。

您需要根據工作負載設定某些任務定義參數。若採用 EC2，則必須選擇為該工作負載專門設計的執行個體類型。

主題

- [GPU 工作負載的 Amazon ECS 任務定義](#)
- [影片轉碼工作負載的 Amazon ECS 任務定義](#)

- [AWS Neuron 機器學習工作負載的 Amazon ECS 任務定義](#)
- [深度學習執行個體的 Amazon ECS 任務定義](#)
- [64 位元 ARM 工作負載的 Amazon ECS 任務定義](#)
- [將 Amazon ECS 日誌傳送至 CloudWatch](#)
- [將 Amazon ECS 日誌傳送至 AWS 服務或 AWS Partner](#)
- [在 Amazon ECS 中使用非AWS 容器映像](#)
- [使用容器重新啟動政策在 Amazon ECS 任務中重新啟動個別容器](#)
- [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 Instance Types guide 中的 [Linux Accelerated Computing Instances](#)。

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 最佳化 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](#)、[Amazon EC2 P3 執行個體](#)、[Amazon EC2 P4d 執行個體](#)、[Amazon EC2 P5 執行個體](#)、[Amazon EC2 G3 執行個體](#)、[Amazon EC2 G4 執行個體](#)、[Amazon EC2 G5 執行個體](#)、[Amazon EC2 G6 執行個體](#)與 [Amazon EC2 G6e 執行個體](#)。

執行個體類型	GPU	GPU 記憶體 (GiB)	vCPUs	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

執行個體類型	GPU	GPU 記憶體 (GiB)	vCPUs	記憶體 (GiB)
p4d.24xlarge	8	320	96	1152
p5.48xlarge	8	640	192	2048
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

執行個體類型	GPU	GPU 記憶體 (GiB)	vCPUs	記憶體 (GiB)
g6.xlarge	1	24	4	16
g6.2xlarge	1	24	8	32
g6.4xlarge	1	24	16	64
g6.8xlarge	1	24	32	128
g6.16.xlarge	1	24	64	256
g6.12xlarge	4	96	48	192
g6.24xlarge	4	96	96	384
g6.48xlarge	8	192	192	768
g6.metal	8	192	192	768
gr6.4xlarge	1	24	16	128
g6e.xlarge	1	48	4	32
g6e.2xlarge	1	48	8	64
g6e.4xlarge	1	48	16	128
g6e.8xlarge	1	48	32	256
g6e16.xlarge	1	48	64	512
g6e12.xlarge	4	192	48	384
g6e24.xlarge	4	192	96	768
g6e48.xlarge	8	384	192	1536
gr6.8xlarge	1	24	32	256

您可以查詢 AWS Systems Manager 參數存放區 API，擷取 Amazon ECS 最佳化 AMIs) ID。若使用此參數，您無需手動查詢 Amazon ECS 最佳化 AMI ID。如需 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
```

搭配 Amazon ECS 受管執行個體使用 GPU

Amazon ECS 受管執行個體透過下列 Amazon EC2 執行個體類型，支援適用於機器學習、高效能運算與影片處理等工作負載的 GPU 加速運算。如需有關 Amazon ECS 受管執行個體支援之執行個體類型的詳細資訊，請參閱 [Amazon ECS 受管執行個體的執行個體類型](#)。

以下是 Amazon ECS 受管執行個體上支援的 GPU 型執行個體類型的子集：

- g4dn：採用 NVIDIA T4 GPUs 技術，適用於機器學習推論、電腦視覺與圖形密集型應用程式。
- g5：採用 NVIDIA A10G GPUs 技術，為圖形密集型應用程式與機器學習工作負載提供更高的效能。
- p3：採用 NVIDIA V100 GPUs 技術，專為高效能運算與深度學習訓練而設計。
- p4d：採用 NVIDIA A100 GPUs 技術，為機器學習訓練與高效能運算提供最高效能。

在搭配 Amazon ECS 受管執行個體使用已啟用 GPU 的執行個體類型時，執行個體上會預先安裝 NVIDIA 驅動程式與 CUDA 工具組，讓執行 GPU 加速工作負載更加輕鬆。

啟用 GPU 的執行個體選擇

若要為 Amazon ECS 受管執行個體工作負載選取已啟用 GPU 的執行個體類型，請在容量提供者的啟動範本中使用 `instanceRequirements` 物件。下列程式碼片段展示了可用於選取已啟用 GPU 的執行個體的屬性。

```
{
  "instanceRequirements": {
    "acceleratorTypes": "gpu",
    "acceleratorCount": 1,
    "acceleratorManufacturers": ["nvidia"]
  }
}
```

下列程式碼片段展示了可用於在啟動範本中指定已啟用 GPU 的執行個體類型的屬性。

```
{
  "instanceRequirements": {
    "allowedInstanceTypes": ["g4dn.xlarge", "p4de.24xlarge"]
  }
}
```

啟用 GPU 的容器映像

若要在容器中使用 GPU，則需使用包含必要 GPU 程式庫與工具的容器映像。NVIDIA 提供數個預先建置的容器映像 (可作為 GPU 工作負載的基礎映像)，包括下列項目：

- `nvidia:cuda`：具有 CUDA 工具組的基礎映像，用於 GPU 運算。
- `tensorflow/tensorflow:latest-gpu`：支援 GPU 的 TensorFlow。
- `pytorch/pytorch:latest-cuda`：支援 GPU 的 PyTorch。

如需有關 Amazon ECS 受管執行個體上涉及使用 GPU 的 Amazon ECS 任務定義範例，請參閱[在 Amazon ECS 任務定義中指定 GPU](#)。

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置放任務](#)。

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 [Specialized Configurations with Docker](#)。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 最佳化 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 AWS CLI 或 API，並將執行個體手動註冊到您的叢集。
- 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。

為 Amazon ECS 啟動 GPU 容器執行個體

若要在 Amazon EC2 上的 Amazon ECS 中使用 GPU 執行個體，您需要建立啟動範本、使用者資料檔案並啟動執行個體。

然後，您便可執行使用為 GPU 設定之任務定義的任務。

使用啟動範本

您可以建立啟動範本。

- 建立使用 AMI 的 Amazon ECS 最佳化 GPU AMI ID 的啟動範本。如需有關如何建立啟動範本的資訊，請參閱 Amazon EC2 User Guide 中的 [Create a new launch template using parameters you define](#)。

將從前一步驟取得的 AMI ID 用於 Amazon Machine Image。如需有關如何使用 Systems Manager 參數指定 AMI ID 的資訊，請參閱 Amazon EC2 User Guide 中的 [Specify a Systems Manager parameter in a launch template](#)。

將下列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，取得 GPU AMI ID。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，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。
- 將 `bottlerocket_ami` 取代為從前一步驟取得的 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-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"
}
```

下列任務定義範例展示了可列印可用 GPU 數量的 TensorFlow 容器。在 Amazon ECS 受管執行個體上執行的任務，需要一個 GPU，並使用 g4dn.xlarge 執行個體。

```
{
  "family": "tensorflow-gpu",
  "networkMode": "awsvpc",
  "executionRoleArn": "arn:aws:iam::account-id:role/ecsTaskExecutionRole",
  "containerDefinitions": [
    {
      "name": "tensorflow",
      "image": "tensorflow/tensorflow:latest-gpu",
      "essential": true,
      "command": [
        "python",
        "-c",
        "import tensorflow as tf; print('Num GPUs Available: ',
len(tf.config.list_physical_devices('GPU')))"
      ],
      "resourceRequirements": [
        {
          "type": "GPU",
          "value": "1"
        }
      ],
      "logConfiguration": {
        "logDriver": "awslogs",
        "options": {
          "awslogs-group": "/ecs/tensorflow-gpu",
          "awslogs-region": "region",
          "awslogs-stream-prefix": "ecs"
        }
      }
    }
  ],
  "requiresCompatibilities": [
    "MANAGED_INSTANCES"
  ],
  "cpu": "4096",
  "memory": "8192",
}
```

共用 GPU

若想共用 GPU，您需要設定下列項目。

1. 從任務定義中移除 GPU 資源需求，如此一來，Amazon ECS 無需保留任何應共用的 GPU。
2. 若想共用 GPU，請將下列使用者資料新增至執行個體。此舉會使 `nvidia` 成為容器執行個體上的預設 Docker 容器執行時期，如此一來，所有 Amazon ECS 容器都可以使用 GPU。如需詳細資訊，請參閱 Amazon EC2 User Guide 中的 [Run commands when you launch an EC2 instance with user data input](#)。

```
const userData = ec2.UserData.forLinux();
userData.addCommands(
  'sudo rm /etc/sysconfig/docker',
  'echo DAEMON_MAXFILES=1048576 | sudo tee -a /etc/sysconfig/docker',
  'echo OPTIONS="--default-ulimit nofile=32768:65536 --default-runtime nvidia" |'
  'sudo tee -a /etc/sysconfig/docker',
  'echo DAEMON_PIDFILE_TIMEOUT=10 | sudo tee -a /etc/sysconfig/docker',
  'sudo systemctl restart docker',
);
```

3. 在容器上設定 `NVIDIA_VISIBLE_DEVICES` 環境變數。您可在任務定義中指定該環境變數來完成此操作。如需有效值的相關資訊，請參閱 NVIDIA 文件網站上的 [GPU Enumeration](#)。

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 User Guide 中的 [Run commands when you launch an EC2 instance with user data input](#)。

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
```

- 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
```

影片轉碼工作負載的 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置放任務](#)。

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-ffmpeg-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。

- [Xilinx 在上提供 AMIs 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 Packer 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。您可以遵循 [Xilinx GitHub](#) 的步驟，建置容器映像。

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。下表詳細說明特定於每種執行個體類型的卡和裝置 SoC。

執行個體類型	vCPUs	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/ renderD12 8 ,/dev/ dri/ renderD12 9
vt1.6xlarge	24	48	2	4	/dev/dri/ renderD12 8 ,/dev/ dri/ renderD12 9 ,/dev/ dri/ renderD13 0 ,/dev/ dri/ renderD13 1
vt1.24xlarge	96	182	8	16	/dev/dri/ renderD12 8 ,/dev/ dri/ renderD12 9 ,/dev/ dri/ renderD13 0 ,/dev/ dri/ renderD13 1 ,/dev/ dri/

執行個體類型	vCPUs	RAM (GiB)	U30 加速器卡	可定址 XCU30 SoC 裝置	裝置路徑
					renderD13 2 ,/dev/ dri/ renderD13 3 ,/dev/ dri/ renderD13 4 ,/dev/ dri/ renderD13 5 ,/dev/ dri/ renderD13 6 ,/dev/ dri/ renderD13 7 ,/dev/ dri/ renderD13 8 ,/dev/ dri/ renderD13 9 ,/dev/ dri/ renderD14 0 ,/dev/ dri/ renderD14 1 ,/dev/ dri/ renderD14 2 ,/dev/

執行個體類型	vCPUs	RAM (GiB)	U30 加速器卡	可定址 XCU30 SoC 裝置	裝置路徑
					dri/ renderD14 3

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 任務定義中指定影片轉碼

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": "xilinx-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-ffmpeg-processor --cli-
input-json file://vt1-3xlarge-ffmpeg-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-ffmpeg-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": "xilinx-ffmpeg"
  }
],
"volumes": [
  {

```

```

        "name": "video_file",
        "host": {"sourcePath": "/home/ec2-user"}
    }
]
}

```

2. 註冊任務定義。

```
aws ecs register-task-definition --family vt1-6xlarge-ffmpeg-processor --cli-
input-json file://vt1-6xlarge-ffmpeg-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-ffmpeg-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": "xilinx-ffmpeg"
    }
],
"volumes": [
    {
        "name": "video_file",
        "host": {"sourcePath": "/home/ec2-user"}
    }
]
}

```

2. 註冊任務定義。

```
aws ecs register-task-definition --family vt1-24xlarge-ffmpeg-processor --cli-
input-json file://vt1-24xlarge-ffmpeg-linux.json --region us-east-1
```

AWS Neuron 機器學習工作負載的 Amazon ECS 任務定義

您可以將 [Amazon EC2 Trn1](#)、[Amazon EC2 Trn2](#)、[Amazon EC2 Inf1](#) 和 [Amazon EC2 Inf2](#) 執行個體註冊到您的叢集，以用於機器學習工作負載。

Amazon EC2 Trn1 和 Trn2 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。您可以使用機器學習架構搭配 Trn1 或 Trn2 執行個體上的 AWS Neuron 來訓練機器學習推論模型。然後，您可以在 Inf1 執行個體或 Inf2 執行個體上執行模型，以使用 AWS Inferentia 晶片的加速。

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 包含編譯器、執行時間和分析工具，可最佳化機器學習晶片的 AWS 機器學習效能。AWS Neuron 支援熱門的機器學習架構，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：

- 您的叢集可以包含 Trn1, Trn2, Inf1, Inf2 和其他執行個體的混合。
- 您需要容器中使用支援 AWS Neuron 的機器學習架構的 Linux 應用程式。

Important

使用其他架構的應用程式可能無法改善 Trn1, Trn2, Inf1 和 Inf2 執行個體的效能。

- 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。若為 Inf1，每個晶片具有 4 個 NeuronCores。對於 Trn1, Trn2 和 Inf2，每個晶片都有 2 NeuronCores。您可以對每個 Trn1, Trn2, Inf1 和 Inf2 執行個體執行任意數量的任務。
- 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。這樣做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位於 Trn1, Trn2, Inf1 和 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置放任務](#)。

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。反之，您可以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 Inferentia 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 最佳化 Amazon Linux 2023 (Neuron) AMI

Amazon ECS 提供以 Amazon Linux 2023 為基礎的 Amazon ECS 最佳化 AMI，適用於 AWS Trainium 和 AWS Inferentia 工作負載。它隨附 Docker 的 AWS Neuron 驅動程式和執行期。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 2023 (Neuron) AMI。

您可以使用 AWS CLI 搭配下列命令，擷取目前的 Amazon ECS 最佳化 Amazon Linux 2023 (Neuron) AMI。

```
aws ssm get-parameters --names /aws/service/ecs/optimized-ami/amazon-linux-2023/neuron/
recommended
```

任務定義需求

要在 Amazon ECS 上部署 Neuron，您的任務定義必須包含預先構建的容器的容器定義，該容器為 TensorFlow 提供推論模型。它由 AWS 深度學習容器提供。此容器包含 AWS Neuron 執行期和 TensorFlow Serving 應用程式。啟動時，此容器會從 Amazon S3 中擷取您的模型、使用儲存的模型啟動 Neuron TensorFlow Serving，並等待預測請求。在以下範例中，容器映像擁有 TensorFlow 1.15 和 Ubuntu 18.04。在 GitHub 上維護為 Neuron 最佳化的預先建置的 Deep Learning Containers 完整清單。如需詳細資訊，請參閱[使用 AWS Neuron TensorFlow Serving](#)。

```
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 深度學習 AMIs 開發人員指南》中的[教學課程：Neuron TensorFlow Serving](#)。

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 Inferentia 裝置。您可以使用 `linuxParameters` 參數進行該動作。如需範例任務定義，請參閱[在 Amazon ECS 任務定義中指定 AWS Neuron 機器學習](#)。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	vCPUs	RAM (GiB)	AWS ML 加速器 晶片	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	vCPUs	RAM (GiB)	AWS ML 加速器 晶片	裝置路徑
				on6 , /dev/ neuron7 , / dev/neuron8 , /dev/neur on9 , /dev/ neuron10 , /dev/neur on11 , /dev/ neuron12 , /dev/neur on13 , /dev/ neuron14 , /dev/neur on15

執行個體類型	vCPUs	RAM (GiB)	AWS ML 加速器 晶片	裝置路徑
trn2.48xlarge	192	1536	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
inf1.xlarge	4	8	1	/dev/neuron0
inf1.2xlarge	8	16	1	/dev/neuron0

執行個體類型	vCPUs	RAM (GiB)	AWS ML 加速器晶片	裝置路徑
inf1.6xlarge	24	48	4	/dev/neuron0 , /dev/neuron1 , /dev/neuron2 , /dev/neuron3
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

執行個體類型	vCPUs	RAM (GiB)	AWS ML 加速器晶片	裝置路徑
inf2.xlarge	32	64	1	/dev/neuron0
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 Neuron 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://amzn-s3-demo-bucket/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"
}
]
}

```

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 支援流行的機器學習架構、TensorFlow 和 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置放任務](#)。

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：

- Habana AWS Marketplace AMIs <https://aws.amazon.com/marketplace/pp/prodview-h24gzbggu75zq>。
- 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 Packer 建置由 [GitHub 儲存庫](#) 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 加速深度學習容器，您的任務定義必須包含預先建置容器的容器定義，該容器使用 AWS 深度學習容器提供的 Habana SynapseAI 為 TensorFlow 或 PyTorch 提供深度學習模型。

以下容器映像擁有 TensorFlow 2.7.0 和 Ubuntu 20.04。在 GitHub 上維護為 Habana Gaudi 加速器最佳化的預先建置的 Deep Learning Containers 完整清單。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"
  }
]
}

```

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。您可以在由 [AWS Graviton 處理器](#) 提供支援的平台上執行應用程式。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。
- 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。
- 應用程式可使用 Fluent Bit 或 CloudWatch 進行監控。
- 對於 Fargate，以下 AWS 區域 不支援 64 位元 ARM 工作負載：
 - 美國東部 (維吉尼亞北部)、us-east-1 可用區域
- 對於 EC2，請參閱以下內容，驗證所在區域是否支援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，請參閱《Amazon EC2 命令列參考》中的 [describe-instance-type-offerings](#)。

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": "public.ecr.aws/docker/library/busybox:latest",
      "cpu": 100,
      "memory": 100,
      "essential": true,
      "command": [ "echo hello world" ],
      "entryPoint": [ "sh", "-c" ]
    }
  ],
  "requiresCompatibilities": [ "EC2" ],
  "cpu": "256",
  "memory": "512",
  "runtimePlatform": {
    "operatingSystemFamily": "LINUX",
    "cpuArchitecture": "ARM64"
  },
  "executionRoleArn": "arn:aws:iam::123456789012:role/ecsTaskExecutionRole"
}
```

將 Amazon ECS 日誌傳送至 CloudWatch

您可以在任務中設定容器，將日誌資訊傳送給 CloudWatch Logs。如果正為任務使用 Fargate，您可檢視容器的日誌。如果正使用 EC2，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。awslogs 日誌驅動程式只會將這些日誌從 Docker 傳遞至 CloudWatch Logs。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 容器執行個體傳送至 CloudWatch Logs，請參閱《Amazon CloudWatch Logs 使用者指南》中的[監控日誌檔案](#)和 [CloudWatch Logs 配額](#)。

Fargate

如果正為任務使用 Fargate，您需要將所需的 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：將日誌路由至 CloudWatch](#)。

對於 Fargate 上的 Windows 容器，當任務定義參數包含特殊字元 (例如 & \ < > ^ |) 時，請執行下列任一選項：

- 在整個參數字串前後加上雙引號，並新增逸出字元 (\)

範例

```
"awslogs-multiline-pattern": "\"^[|DEBUG|INFO|WARNING|ERROR\""
```

- 在每個特殊字元前後加上逸出字元 (^)

範例

```
"awslogs-multiline-pattern": "\"^[^|DEBUG^|INFO^|WARNING^|ERROR"
```

EC2

如果對任務使用 EC2，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，或至少具有 1.9.0-1 版 `ecs-init` 套件的自訂 AMI。使用自訂 AMI 時，您必須透過在 `docker run` 陳述式或環境變數檔案中使用下列環境變數，確保啟動代理程式時 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。如需有關任務執行角色的資訊，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。如需有關容器執行個體受管 IAM 政策的資訊，請參閱 [Amazon ECS 容器執行個體 IAM 角色](#)。

Amazon ECS 任務定義範例：將日誌路由至 CloudWatch

您必須先在任務定義中指定容器的 `awslogs` 日誌驅動程式，容器才能將日誌傳送至 CloudWatch。如需有關日誌參數的詳細資訊，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。一個是用於 WordPress 容器，該容器將日誌發送到名為 `awslogs-wordpress` 的日誌群組。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": "public.ecr.aws/docker/library/wordpress:latest",
      "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": "public.ecr.aws/docker/library/mysql:latest",
    "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"
}

```

後續步驟

- 您可以使用 CloudWatch AWS CLI 或 API，選擇性地設定日誌群組的保留政策。如需詳細資訊，請參閱 AWS Command Line Interface Reference 中的 [put-retention-policy](#)。
- 當您在容器定義日誌組態中將任務定義註冊到 awslogs 日誌驅動程式後，就可以執行任務，或使用該任務定義建立服務，以開始將日誌傳送至 CloudWatch Logs。如需詳細資訊，請參閱[將應用程式作為 Amazon ECS 任務執行](#)及[建立 Amazon ECS 滾動更新部署](#)。

將 Amazon ECS 日誌傳送至 AWS 服務或 AWS Partner

您可以使用 FireLens for Amazon ECS 來使用任務定義參數，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。AWS Partner Network 是一個全球合作夥伴社群，利用計劃、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。

依預設，Amazon ECS 會設定容器相依性，確保 Firelens 容器先於所有使用它的容器啟動，並於所有使用它的容器停止後才停止。

若要使用此功能，您必須為任務建立 IAM 角色，以提供使用任務所需的任何 AWS 服務所需的許可。例如，若容器正在將日誌路由至 Firehose，則任務需要呼叫 `firehose:PutRecordBatch` API 的許可。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，任務可能也需要 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，或在日誌組態 AWS Secrets Manager 中參考來自的敏感資料，則必須包含任務執行 IAM 角色。
- 在使用 Amazon S3 中託管的自訂組態檔案時，任務執行 IAM 角色必須包含 `s3:GetObject` 許可。

使用 FireLens for Amazon ECS 時，請考量下列事項：

- 建議將 `my_service_` 新增至日誌容器名稱，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與 FireLens 容器之間新增啟動容器順序相依性。在應用程式容器與 FireLens 容器之間指定容器順序時，會覆寫預設的啟動容器順序。
- 託管於 Linux 上的 AWS Fargate 和 Linux 上的 Amazon EC2 的任務支援 Amazon ECS 的 FireLens。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 CloudFormation 範本來設定 Amazon ECS FireLens 的。如需詳細資訊，請參閱《AWS CloudFormation 使用者指南》中的 [AWS::ECS::TaskDefinition FirelensConfiguration](#)。
- FireLens 在連接埠上監聽 24224，因此為了確保 FireLens 日誌路由器無法在任務之外連線，您不能在允許任務使用的安全群組中連接埠 24224 上的傳入流量。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 配置的容器必須在任何依賴該模式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

如果您在使用 FireLens 配置的容器定義中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"  
  }  
}
```

您可以設定FireLens容器以非根使用者身分執行。考慮下列各項：

- 若要將FireLens容器設定為以非根使用者身分執行，您必須以下列其中一種格式指定使用者：
 - uid
 - uid:gid
 - uid:group

如需在容器定義中指定使用者的詳細資訊，請參閱《Amazon Elastic Container Service API 參考》中的 [ContainerDefinition](#)。

FireLens 容器會透過UNIX通訊端接收應用程式日誌。Amazon ECS 代理程式使用 uid將通訊端目錄的擁有權指派給FireLens容器。

- Amazon ECS Agent 版本 1.96.0和更新版本以及 Amazon ECS 最佳化 AMI 版本 v20250716 和更新版本支援將FireLens容器設定為以非根使用者身分執行。
- 當您為FireLens容器指定使用者時，uid 必須是唯一的，且不會用於任務或容器執行個體中屬於其他容器的其他程序。

如需有關如何搭配 Amazon ECS 使用多個組態檔案 (包括您託管的或 Amazon S3 中的檔案) 的資訊，請參閱 [Init process for Fluent Bit on ECS, multi-config support](#)。

如需範例組態的詳細資訊，請參閱 [Amazon ECS 任務定義範例：將日誌路由至 FireLens](#)。

如需設定高輸送量日誌的詳細資訊，請參閱 [為高輸送量設定 Amazon ECS 日誌](#)。

為高輸送量設定 Amazon ECS 日誌

對於高日誌輸送量案例，我們建議搭配 FireLens 和使用awsfirelens日誌驅動程式Fluent Bit。Fluent Bit 是一種輕量型日誌處理器，可有效處理 資源，並可處理數百萬筆日誌記錄。不過，大規模實現最佳效能需要調校其組態。

本節涵蓋處理高日誌輸送量的進階Fluent Bit最佳化技術，同時維持系統穩定性並確保不會遺失資料。

如需如何搭配 FireLens 使用自訂組態檔案的詳細資訊，請參閱 [使用自訂組態檔案](#)。如需其他範例，請參閱 GitHub 上的 [Amazon ECS FireLens 範例](#)。

Note

本節中的某些組態選項，例如 workers和 threaded，AWS 需要 第 3 Fluent Bit版或更新版本。如需可用版本的資訊，請參閱 [AWS 以取得 Fluent Bit 版本](#)。

了解區塊

Fluent Bit 會以稱為區塊的單位處理資料。當 INPUT 外掛程式接收資料時，引擎會建立區塊，在傳送至 OUTPUT 目的地之前儲存在記憶體或檔案系統上。

緩衝行為取決於 INPUT 區段中的 `storage.type` 設定。根據預設，Fluent Bit 會使用記憶體緩衝。對於高輸送量或生產案例，檔案系統緩衝可提供更好的彈性。

如需詳細資訊，請參閱 Fluent Bit 文件中的 [區塊](#) 和《》中的 [什麼是區塊？AWS](#) 以取得 Fluent Bit 範例儲存庫。

記憶體緩衝（預設）

根據預設，Fluent Bit 會使用記憶體緩衝 (`storage.type memory`)。您可以使用 `Mem_Buf_Limit` 參數限制每個 INPUT 外掛程式的記憶體用量。

下列範例顯示記憶體緩衝的輸入組態：

```
[INPUT]
  Name          tcp
  Tag            ApplicationLogs
  Port          5170
  storage.type  memory
  Mem_Buf_Limit 5MB
```

Important

`Mem_Buf_Limit` 超過外掛程式的時，會 Fluent Bit 暫停輸入，並遺失新記錄。這可能會導致背壓並拖慢您的應用程式。Fluent Bit 日誌中會顯示下列警告：

```
[input] tcp.1 paused (mem buf overlimit)
```

記憶體緩衝適用於日誌輸送量低至中等的簡單使用案例。對於需要資料遺失的高輸送量或生產案例，請改用檔案系統緩衝。

如需詳細資訊，請參閱 Fluent Bit 文件中的 [緩衝和記憶體](#)，以及 [僅限記憶體緩衝 AWS](#)，以取得 Fluent Bit 範例儲存庫。

檔案系統緩衝

對於高輸送量案例，建議使用檔案系統緩衝。如需有關 如何Fluent Bit管理緩衝和儲存的詳細資訊，請參閱 [Fluent Bit 文件中的緩衝和儲存](#)。

檔案系統緩衝提供下列優點：

- 較大的緩衝容量 – 磁碟空間通常比記憶體更豐富。
- 持久性 – 緩衝的資料在Fluent Bit重新啟動後仍然存在。
- 緩慢降級 – 在輸出失敗期間，資料累積在磁碟上，而不是導致記憶體耗盡。

若要啟用檔案系統緩衝，請提供自訂Fluent Bit組態檔案。下列範例顯示建議的組態：

```
[SERVICE]
  # Flush logs every 1 second
  Flush 1
  # Wait 120 seconds during shutdown to flush remaining logs
  Grace 120
  # Directory for filesystem buffering
  storage.path /var/log/flb-storage/
  # Limit chunks stored 'up' in memory (reduce for memory-constrained environments)
  storage.max_chunks_up 32
  # Flush backlog chunks to destinations during shutdown (prevents log loss)
  storage.backlog.flush_on_shutdown On

[INPUT]
  Name forward
  unix_path /var/run/fluent.sock
  # Run input in separate thread to prevent blocking
  threaded true
  # Enable filesystem buffering for persistence
  storage.type filesystem

[OUTPUT]
  Name cloudwatch_logs
  Match *
  region us-west-2
  log_group_name /aws/ecs/my-app
  log_stream_name $(ecs_task_id)
  # Use multiple workers for parallel processing
  workers 2
  # Retry failed flushes up to 15 times
```

```
retry_limit 15
# Maximum disk space for buffered data for this output
storage.total_limit_size 10G
```

金鑰組態參數：

`storage.path`

在磁碟上Fluent Bit存放緩衝區塊的目錄。

`storage.backlog.flush_on_shutdown`

啟用時，會Fluent Bit嘗試在關閉期間將所有待處理日誌檔案系統區塊排清至其目的地。這有助於確保資料在Fluent Bit停止之前交付，但可能會增加關閉時間。

`storage.max_chunks_up`

保留在記憶體中的區塊數量。預設值為 128 個區塊，這可能會耗用 500 MB+ 的記憶體，因為每個區塊最多可使用 4–5 MB。在記憶體受限的環境中，降低此值。例如，如果您有 50 MB 可用於緩衝，請將此設為 8–10 個區塊。

`storage.type filesystem`

啟用輸入外掛程式的檔案系統儲存。儘管名稱為 `filesystem`，還是Fluent Bit使用 `mmap`將區塊映射到記憶體和磁碟，在不犧牲效能的情況下提供持久性。

`storage.total_limit_size`

特定 OUTPUT 外掛程式緩衝資料的最大磁碟空間。達到此限制時，會捨棄該輸出的最舊記錄。如需調整大小的詳細資訊，請參閱 [了解 `storage.total\_limit\_size`](#)。

`threaded true`

在自己的執行緒中執行輸入，與 Fluent Bit的主要事件迴圈分開。這可防止慢速輸入封鎖整個管道。

如需詳細資訊，請參閱 Fluent Bit 文件中的[檔案系統緩衝](#)，以及 AWS 中Fluent Bit的範例儲存庫中的[檔案系統和記憶體緩衝](#)。

了解 **`storage.total_limit_size`**

每個 OUTPUT 外掛程式上的 `storage.total_limit_size` 參數會控制該輸出緩衝資料的最大磁碟空間。達到此限制時，會捨棄該輸出的最舊記錄，為新資料騰出空間。當磁碟空間完全用盡時，Fluent Bit 無法將記錄排入佇列，而且會遺失。

使用以下公式 `storage.total_limit_size`，根據您的日誌速率和所需的復原時段計算適當的：

If log rate is in KB/s, convert to MB/s first:

$\text{log\_rate (MB/s)} = \text{log\_rate (KB/s)} / 1000$

$\text{storage.total\_limit\_size (GB)} = \text{log\_rate (MB/s)} \times \text{duration (hours)} \times 3600 \text{ (seconds/hour)} / 1000 \text{ (MB to GB)}$

下表顯示常見日誌速率和復原時段的範例計算：

日誌速率	1 小時	6 小時	12 小時	24 小時
0.25 MB/s	0.9 GB	5.4 GB	10.8 GB	21.6 GB
0.5 MB/s	1.8 GB	10.8 GB	21.6 GB	43.2 GB
1 MB/s	3.6 GB	21.6 GB	43.2 GB	86.4 GB
5 MB/s	18 GB	108 GB	216 GB	432 GB
10 MB/s	36 GB	216 GB	432 GB	864 GB

若要觀察尖峰輸送量並選擇適當的緩衝區大小，請使用[量值輸送量 FireLens 範例](#)。

使用公式、範例計算和基準測試，選擇適合 `storage.total_limit_size` 的執行通道，以便在中斷期間進行最佳復原。

Amazon ECS 任務儲存需求

在 OUTPUT 區段中加總所有 `storage.total_limit_size` 值，並新增額外負荷緩衝區。此總計會決定 Amazon ECS 任務定義中所需的儲存空間。例如，3 個輸出 × 10 GB，每個 = 30 GB + 緩衝 (5–10 GB) = 總共需要 35–40 GB。如果總計超過可用儲存體，Fluent Bit 可能無法將記錄排入佇列，而且它們將會遺失。

下列儲存選項可供使用：

綁定掛載（暫時性儲存）

- 對於 AWS Fargate，預設值為 20 GB 的暫時性儲存（上限為 200 GB）。在任務定義 `ephemeralStorage` 中使用設定。如需詳細資訊，請參閱 AWS CloudFormation 《使用者指南》中的 [EphemeralStorage](#)。

- 對於 EC2，使用 Amazon ECS 最佳化 AMI（在作業系統和 Docker 之間共用）時，預設值為 30 GB。透過變更根磁碟區大小來增加。

Amazon EBS 磁碟區

- 提供高可用性、耐用、高效能的區塊儲存。
- 需要磁碟區組態，並在任務定義mountPoint中指向 storage.path（預設：/var/log/flb-storage/）。
- 如需詳細資訊，請參閱[在 Amazon ECS 任務定義中將磁碟區組態延至啟動時進行](#)。

Amazon EFS 磁碟區

- 提供簡單、可擴展的檔案儲存。
- 需要磁碟區組態，並在任務定義mountPoint中指向 storage.path（預設：/var/log/flb-storage/）。
- 如需詳細資訊，請參閱[在 Amazon ECS 任務定義中指定 Amazon EFS 檔案系統](#)。

如需資料磁碟區的詳細資訊，請參閱 [Amazon ECS 任務的儲存選項](#)。

最佳化輸出組態

網路問題、服務中斷和目的地限流可防止日誌交付。適當的輸出組態可確保彈性而不會遺失資料。

當輸出排清失敗時，Fluent Bit可以重試 操作。下列參數控制重試行為：

retry_limit

捨棄記錄之前，初次嘗試後的重試次數上限。預設為 1。例如，retry_limit 3表示總共 4 次嘗試（1 次初始 + 3 次重試）。對於生產環境，我們建議使用 15 或更高版本，這涵蓋了幾分鐘的中斷和指數退避。

將False無限次重試設為 no_limits或：

- 透過記憶體緩衝，無限次重試會導致輸入外掛程式在達到記憶體限制時暫停。
- 使用檔案系統緩衝時，最舊的記錄會在達到 storage.total_limit_size 時捨棄。

Important

耗盡所有重試嘗試（1 次初始 + retry_limit 重試）後，會捨棄記錄。AWS 具有 auto_retry_requests true（預設）的外掛程式會在 Fluent Bit的重試機制之前提供額外的重試層。如需詳細資訊，請參閱 Fluent Bit 文件中的[設定重試](#)。

例如，`retry_limit 3`使用預設設定 (`scheduler.base 5`、`scheduler.cap 2000`、`net.connect_timeout 10s`) 提供大約 70 秒的排程器等待時間 (10 秒 + 20 秒 + 40 秒)、40 秒的網路連線逾時 (4 次嘗試 × 10 秒)，加上 AWS 外掛程式重試，根據網路條件和作業系統 TCP 逾時，總計大約 2-10 分鐘。

`scheduler.base`

重試之間的基本秒數（預設值：5）。我們建議 10 秒。

`scheduler.cap`

重試之間的秒數上限（預設值：2000）。我們建議 60 秒。

重試之間的等待時間使用指數退避與抖動：

```
wait_time = random(base, min(base × 2^retry_number, cap))
```

例如，使用 `scheduler.base 10`和 `scheduler.cap 60`：

- 第一次重試：隨機等待 10-20 秒
- 第二次重試：隨機等待 10-40 秒
- 第三次重試和更新時間：隨機等待 10-60 秒（上限）

如需詳細資訊，請參閱 Fluent Bit 文件中的[設定重試和聯網的等待時間](https://docs.fluentbit.io/manual/administration/networking)。<https://docs.fluentbit.io/manual/administration/networking>

`workers`

平行輸出處理的執行緒數目。多個工作者允許並行排清，在處理許多區塊時提高輸送量。

`auto_retry_requests`

AWS 外掛程式特定的設定，可在的Fluent Bit內建重試機制之前提供額外的重試層。預設值為 `true`。啟用時，AWS 輸出外掛程式會在內部重試失敗的請求，再將請求視為失敗的排清並受`retry_limit`組態約束。

[SERVICE] 區段中的 Grace 參數會設定關機期間Fluent Bit等待的時間，以排清緩衝的資料。Grace 期間必須與容器的 協調stopTimeout。在接收 之前，請確定 stopTimeout超過允許 Fluent Bit完成排清的Grace期間SIGKILL。例如，如果 Grace是 120 秒，請將 stopTimeout設定為 150 秒。

下列範例顯示完整Fluent Bit組態，其中包含高輸送量案例的所有建議設定：

[SERVICE]

```
# Flush logs every 1 second
Flush 1
# Wait 120 seconds during shutdown to flush remaining logs
Grace 120
# Directory for filesystem buffering
storage.path /var/log/flb-storage/
# Limit chunks stored 'up' in memory (reduce for memory-constrained environments)
storage.max_chunks_up 32
# Flush backlog chunks to destinations during shutdown (prevents log loss)
storage.backlog.flush_on_shutdown On
# Minimum seconds between retries
scheduler.base 10
# Maximum seconds between retries (exponential backoff cap)
scheduler.cap 60
```

[INPUT]

```
Name forward
unix_path /var/run/fluent.sock
# Run input in separate thread to prevent blocking
threaded true
# Enable filesystem buffering for persistence
storage.type filesystem
```

[OUTPUT]

```
Name cloudwatch_logs
Match *
region us-west-2
log_group_name /aws/ecs/my-app
log_stream_name $(ecs_task_id)
# Use multiple workers for parallel processing
workers 2
# Retry failed flushes up to 15 times
retry_limit 15
# Maximum disk space for buffered data for this output
storage.total_limit_size 10G
```

了解資料遺失案例

長時間中斷或輸出目的地發生問題時，記錄可能會遺失。本指南中的組態建議是將資料遺失降至最低的最佳方法，但無法保證長時間故障期間的零遺失。了解這些案例可協助您設定 Fluent Bit 以最大化彈性。

記錄可能會以兩種方式遺失：儲存填滿時捨棄最舊的記錄，或在系統無法接受更多資料時捨棄最新的記錄。

捨棄的最早記錄

當重試嘗試用盡或 `storage.total_limit_size` 填滿且需要為新資料騰出空間時，會捨棄最舊的緩衝記錄。

超過重試限制

在 AWS 外掛程式重試 (如果 `auto_retry_requests true`) 加上 1 次初始 Fluent Bit 嘗試加上 `retry_limit` 重試後發生。若要緩解，`retry_limit no_limits` 為每個 OUTPUT 外掛程式設定無限次重試：

```
[OUTPUT]
  Name          cloudwatch_logs
  Match         ApplicationLogs
  retry_limit    no_limits
  auto_retry_requests true
```

Important

無限次重試可防止因重試耗盡而捨棄記錄，但可能導致 `storage.total_limit_size` 填滿。

已達到儲存限制 (檔案系統緩衝)

當輸出目的地無法使用的時間超過您設定的 `storage.total_limit_size` 緩衝時間時，便會發生。例如，以 1 MB/s 日誌速率的 10 GB 緩衝區可提供大約 2.7 小時的緩衝時間。若要緩解、增加 `storage.total_limit_size` 每個 OUTPUT 外掛程式並佈建足夠的 Amazon ECS 任務儲存：

```
[OUTPUT]
  Name          cloudwatch_logs
```

Match	ApplicationLogs
storage.total_limit_size	10G

拒絕的最新記錄

當磁碟空間耗盡或輸入因 而暫停時，會捨棄最新的記錄Mem_Buf_Limit。

磁碟空間用盡（檔案系統緩衝）

當磁碟空間完全用盡時，便會發生。Fluent Bit 無法將新記錄排入佇列並遺失。若要緩解，請加總所有storage.total_limit_size值並佈建足夠的 Amazon ECS 任務儲存體。如需詳細資訊，請參閱[Amazon ECS 任務儲存需求](#)。

已達到記憶體限制（記憶體緩衝）

當輸出目的地無法使用且記憶體緩衝區填滿時，便會發生。暫停的輸入外掛程式會停止接受新記錄。若要緩解，請使用 storage.type filesystem以獲得更好的彈性，或增加Mem_Buf_Limit。

將資料遺失降至最低的最佳實務

請考慮下列最佳實務，將資料遺失降至最低：

- 使用檔案系統緩衝 – 設定 storage.type filesystem以在中斷期間獲得更好的彈性。
- 適當大小儲存 – 根據storage.total_limit_size日誌速率和所需的復原時段計算。
- 佈建足夠的磁碟 – 確保 Amazon ECS 任務有足夠的暫時性儲存、Amazon EBS 或 Amazon EFS。
- 設定重試行為 – 平衡 retry_limit（耗盡重試後捨棄記錄）和 no_limits（無限期重試，但可能會填滿儲存體）。

使用多目的地記錄來確保可靠性

將日誌傳送至多個目的地可消除單一失敗點。例如，如果 CloudWatch Logs 遇到中斷，則日誌仍會到達 Amazon S3。

多目的地記錄可提供下列優點。Amazon S3 輸出外掛程式也支援壓縮選項，例如 gzip 和 Parquet 格式，這可以降低儲存成本。如需詳細資訊，請參閱 Fluent Bit 文件中的 [S3 壓縮](#)。

多目的地記錄可提供下列優點：

- 備援 – 如果一個目的地失敗，日誌仍會到達另一個目的地。

- 復原 – 從另一個系統重建一個系統中的差距。
- 耐用性 – 在 Amazon S3 中封存日誌以進行長期保留。
- 成本最佳化 – 在 CloudWatch Logs 等快速查詢服務中保留最近的日誌，保留時間較短，同時將所有日誌存檔到成本較低的 Amazon S3 儲存，以便長期保留。

下列Fluent Bit組態會將日誌傳送至 CloudWatch Logs 和 Amazon S3：

[OUTPUT]

```
Name cloudwatch_logs
Match *
region us-west-2
log_group_name /aws/ecs/my-app
log_stream_name $(ecs_task_id)
workers 2
retry_limit 15
```

[OUTPUT]

```
Name s3
Match *
bucket my-logs-bucket
region us-west-2
total_file_size 100M
s3_key_format /fluent-bit-logs/$(ecs_task_id)/%Y%m%d/%H/%M/$UUID
upload_timeout 10m
# Maximum disk space for buffered data for this output
storage.total_limit_size 5G
```

兩個輸出都使用相同的Match *模式，因此所有記錄都會獨立傳送至兩個目的地。在某個目的地中斷期間，日誌會繼續流向另一個目的地，而失敗的排清會在檔案系統緩衝區中累積以供稍後重試。

搭配尾端輸入外掛程式使用檔案型記錄

對於日誌遺失是重大考量的高輸送量案例，您可以使用替代方法：讓應用程式將日誌寫入磁碟上的檔案，並設定 Fluent Bit 使用tail輸入外掛程式讀取它們。此方法完全略過 Docker 記錄驅動程式層。

使用尾端外掛程式以檔案為基礎的記錄可提供下列優點：

- 位移追蹤 – 尾端外掛程式可以將檔案位移存放在資料庫檔案中（使用 DB選項），在Fluent Bit重新啟動時提供耐用性。這有助於防止在容器重新啟動期間日誌遺失。
- 輸入層級緩衝 – 您可以使用 直接在輸入外掛程式上設定記憶體緩衝區限制Mem_Buf_Limit，以更精細地控制記憶體用量。

- 避免 Docker 額外負荷 – 日誌會直接從檔案移至 `Fluent Bit` 而不會傳遞 Docker 的日誌緩衝區。

若要使用此方法，您的應用程式必須將日誌寫入檔案，而不是 `stdout`。應用程式容器和 `Fluent Bit` 容器都會掛載存放日誌檔案的共用磁碟區。

下列範例顯示具有最佳實務的尾端輸入組態：

```
[INPUT]
  Name tail
  # File path or glob pattern to tail
  Path /var/log/app.log
  # Database file for storing file offsets (enables resuming after restart)
  DB /var/log/flb_tail.db
  # when true, controls that only fluent-bit will access the database (improves
  performance)
  DB.locking true
  # Skip long lines instead of skipping the entire file
  Skip_Long_Lines On
  # How often (in seconds) to check for new files matching the glob pattern
  Refresh_Interval 10
  # Extra seconds to monitor a file after rotation to account for pending flush
  Rotate_Wait 30
  # Maximum size of the buffer for a single line
  Buffer_Max_Size 10MB
  # Initial allocation size for reading file data
  Buffer_Chunk_Size 1MB
  # Maximum memory buffer size (tail pauses when full)
  Mem_Buf_Limit 75MB
```

使用尾端輸入外掛程式時，請考慮下列事項：

- 為您的應用程式日誌實作日誌輪換，以防止磁碟耗盡。監控基礎磁碟區指標以衡量效能。
- 根據您的日誌格式考慮設定 `Ignore_Older`，例如 `Read_from_Head`、和多行剖析器。

如需詳細資訊，請參閱 `Fluent Bit` 文件中的 [Tail](#)。如需最佳實務，請參閱《》中的 [Tail config with best practices](#) AWS for `Fluent Bit` troubleshooting guide。

直接記錄到 FireLens

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。

Note

如果搭配 bridge 網路模式使用 EC2，應用程式容器中的 FLUENT_HOST 環境變數可能會在重新啟動 FireLens 日誌路由器容器 (容器定義中具有 firelensConfiguration 物件的容器) 之後變得不準確。這是因為 FLUENT_HOST 是動態 IP 位址，重新啟動之後可能發生變更。從應用程式容器直接向 FLUENT_HOST IP 位址寫入日誌的操作，可能會在該位址變更後開始失敗。如需有關重新啟動個別容器的詳細資訊，請參閱[使用容器重新啟動政策在 Amazon ECS 任務中重新啟動個別容器](#)。

FLUENT_PORT

流利轉送通訊協定正在接聽的連接埠。

您可以使用這些環境變數，使用 Fluent Forward 通訊協定直接從應用程式程式碼登入 Fluent Bit 日誌路由器，而不是寫入 stdout。此方法略過 Docker 記錄驅動程式層，可提供下列優點：

- 低延遲 – 日誌會直接傳送至 `Fluent Bit` 而不會傳遞 Docker 的記錄基礎設施。
- 結構化記錄 – 原生傳送結構化日誌資料，無需 JSON 編碼開銷。
- 更好的控制 – 您的應用程式可以實作自己的緩衝和錯誤處理邏輯。

下列 Fluent 記錄程式庫支援 Fluent Forward 通訊協定，可用於將日誌直接傳送到 Fluent Bit：

- Go – [fluent-logger-golang](#)
- Python – [fluent-logger-python](#)
- Java – [fluent-logger-java](#)
- Node.js – [fluent-logger-node](#)
- Ruby – [fluent-logger-ruby](#)

設定 Docker 緩衝區限制

建立任務定義時，您可以透過在 `logDriverBufferLimit` 中指定值，來指定在記憶體中緩衝的日誌行數。這會控制 Docker 和 `Fluent Bit` 之間的緩衝區。如需詳細資訊，請參閱 Docker 文件中的 [Fluentd 登入驅動程式](#)。

請在有高輸送量時使用此選項，原因是 Docker 可能會耗盡緩衝區記憶體，並捨棄緩衝區訊息，以便新增新訊息。

使用此選項時，請考慮下列事項：

- EC2 與具有平台版本 1.4.0 或更新版本的 Fargate 類型支援此選項。
- 只有在 `logDriver` 設定為 `awsfirelens` 時，此選項才有效。
- 預設的緩衝區上限為 1048576 日誌行。
- 緩衝區限制必須大於或等於 0 且小於 536870912 日誌行。
- 此緩衝區使用的記憶體容量上限，為每個日誌行大小與緩衝區大小的乘積。例如，如果應用程式的日誌行是平均 2 KiB，緩衝區限制為 4096 最多會使用 8 MiB。在任務層級配置的記憶體總量，應大於為所有容器配置的記憶體容量與日誌驅動程式記憶體緩衝區用量的總和。

下列任務定義顯示如何設定 `log-driver-buffer-limit`：

```
{
  "containerDefinitions": [
    {
      "name": "my_service_log_router",
      "image": "public.ecr.aws/aws-observability/aws-for-fluent-bit:3",
      "cpu": 0,
      "memoryReservation": 51,
      "essential": true,
      "firelensConfiguration": {
        "type": "fluentbit"
      }
    },
    {
      "essential": true,
      "image": "public.ecr.aws/docker/library/httpd:latest",
      "name": "app",
      "logConfiguration": {
        "logDriver": "awsfirelens",
        "options": {
```

```
        "Name": "firehose",
        "region": "us-west-2",
        "delivery_stream": "my-stream",
        "log-driver-buffer-limit": "52428800"
      }
    },
    "dependsOn": [
      {
        "containerName": "my_service_log_router",
        "condition": "START"
      }
    ],
    "memoryReservation": 100
  }
]
```

AWS 適用於 Amazon ECS Fluent Bit 的影像儲存庫

AWS 為 CloudWatch Logs 和 Firehose 提供具有外掛程式 Fluent Bit 的影像。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 [CloudWatch Logs for Fluent Bit](#) 和 [Amazon Kinesis Firehose for Fluent Bit](#)。

AWS for Fluent Bit 映像可在 Amazon ECR Public Gallery 和 Amazon ECR 儲存庫的 Amazon ECR 上取得，以獲得高可用性。

Amazon ECR Public Gallery

AWS 適用於 Fluent Bit 映像的 可在 Amazon ECR Public Gallery 上取得。這是下載 AWS 適用於 Fluent Bit 映像的 的建議位置，因為它是公有儲存庫，可從所有 使用 AWS 區域。如需詳細資訊，請參閱 Amazon ECR Public Gallery 上的 [aws-for-fluent-bit](#)。

Linux

Amazon ECR Public Gallery 中的 AWS for Fluent Bit 映像支援具有 ARM64 或 x86-64 架構的 Amazon Linux 作業系統。

您可以透過指定具有所需 Fluent Bit 映像標籤 AWS 的儲存庫 URL，從 Amazon ECR Public Gallery 提取映像的。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 版本提取「3.x」系列中的最新映像。

```
docker pull public.ecr.aws/aws-observability/aws-for-fluent-bit:3
```

Note

允許未經驗證的提取，但速率限制低於已驗證的提取。若要在提取之前使用 AWS 您的帳戶進行身分驗證，請使用下列命令。

```
aws ecr-public get-login-password --region us-east-1 | docker login --username  
AWS --password-stdin public.ecr.aws
```

AWS 適用於 Fluent Bit 3.0.0

除了 AWS Fluent Bit 版本的現有 2.x，AWS Fluent Bit 還支援新的主要版本 3.x。新的主要版本包括將映像從 Amazon Linux 2 升級至 Amazon Linux 2023，並將 Fluent Bit 版本 1.9.10 升級至 4.1.1。如需詳細資訊，請參閱 [上 AWS 儲存 Fluent Bit 庫的 GitHub](#)。

下列範例示範 Fluent Bit 3.x 映像 AWS 的更新標籤：

您可以針對 AWS Fluent Bit 映像的使用多架構標籤。

```
docker pull public.ecr.aws/aws-observability/aws-for-fluent-bit:3
```

Windows

Amazon ECR Public Gallery 中的 AWS for Fluent Bit 映像支援具有下列作業系統的 AMD64 架構：

- Windows Server 2022 Full
- Windows Server 2022 Core
- Windows Server 2019 Full
- Windows Server 2019 Core

AWS Fargate 上的 Windows 容器不支援 FireLens。

您可以透過指定具有所需 Fluent Bit 映像標籤 AWS 的儲存庫 URL，從 Amazon ECR Public Gallery 提取映像的。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 AWS CLI 命令提取影像的最新穩定 Fluent Bit。

```
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。若要在提取之前使用 AWS 您的帳戶進行身分驗證，請使用下列命令。

```
aws ecr-public get-login-password --region us-east-1 | docker login --username AWS --password-stdin public.ecr.aws
```

Amazon ECR

AWS for Fluent Bit 映像可在 Amazon ECR 上使用，以獲得高可用性。下列命令可用來擷取映像 URIs 並在指定的 中建立映像可用性 AWS 區域。

Linux

您可以使用下列命令擷取最新穩定的 AWS Fluent Bit 映像 URI。

```
aws ssm get-parameters \
  --names /aws/service/aws-for-fluent-bit/stable \
  --region us-east-1
```

您可以使用下列命令來查詢 Systems Manager 參數存放區參數，列出所有版本的 AWS for Fluent Bit 映像。

```
aws ssm get-parameters-by-path \
  --path /aws/service/aws-for-fluent-bit \
  --region us-east-1
```

您可以在範本中 CloudFormation 參考 Systems Manager 參數存放區名稱，以參考最新穩定的 AWS Fluent Bit 映像。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

 Note

如果命令失敗或沒有輸出，則呼叫命令 AWS 區域 的中無法使用映像。

Windows

您可以使用下列命令擷取最新穩定的 AWS Fluent Bit 映像 URI。

```
aws ssm get-parameters \  
  --names /aws/service/aws-for-fluent-bit/windowsservercore-stable \  
  --region us-east-1
```

您可以使用下列命令來查詢 Systems Manager 參數存放區參數，列出所有版本的 AWS for Fluent Bit 映像。

```
aws ssm get-parameters-by-path \  
  --path /aws/service/aws-for-fluent-bit/windowsservercore \  
  --region us-east-1
```

您可以在範本中 CloudFormation 參考 Systems Manager 參數存放區名稱，以參考最新穩定的 AWS Fluent Bit 映像。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：將日誌路由至 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 建立新的任務定義時 AWS 管理主控台，有一個 FireLens 整合區段，可讓您輕鬆地新增日誌路由容器。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。因此為了確保 FireLens 日誌路由器無法在任務之外連線，您不能在任務使用的安全群組中允許連接埠 24224 上的傳入流量。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。

如果不要中繼資料，您可以將 `enable-ecs-log-metadata` 設定為 `false`。

以下任務定義範例定義了一個日誌路由器容器，其使用 Fluent Bit，將其日誌路由至 CloudWatch Logs。其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至 Amazon Data Firehose，並將用於緩衝事件的記憶體設定為 2 MiB。

Note

如需更多任務定義範例，請參閱 GitHub 上的 [Amazon ECS FireLens 範例](#)。

```

{
  "family": "firelens-example-firehose",
  "taskRoleArn": "arn:aws:iam::123456789012:role/ecs_task_iam_role",
  "containerDefinitions": [
    {
      "name": "log_router",
      "image": "public.ecr.aws/aws-observability/aws-for-fluent-bit:3",
      "cpu": 0,
      "memoryReservation": 51,
      "portMappings": [],
      "essential": true,
      "environment": [],
      "mountPoints": [],
      "volumesFrom": [],
      "logConfiguration": {
        "logDriver": "awslogs",
        "options": {
          "awslogs-group": "/ecs/ecs-aws-firelens-sidecar-container",
          "mode": "non-blocking",
          "awslogs-create-group": "true",
          "max-buffer-size": "25m",
          "awslogs-region": "us-east-1",
          "awslogs-stream-prefix": "firelens"
        },
        "secretOptions": []
      },
      "systemControls": [],
      "firelensConfiguration": {
        "type": "fluentbit"
      }
    },
    {
      "essential": true,
      "image": "public.ecr.aws/docker/library/httpd:latest",
      "name": "app",
      "logConfiguration": {
        "logDriver": "awsfirelens",
        "options": {
          "Name": "firehose",
          "region": "us-west-2",
          "delivery_stream": "my-stream",
          "log-driver-buffer-limit": "1048576"
        }
      }
    }
  ]
}

```



```
    },  
    "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 組態。您無需在任務定義中指定 match 組態。

使用自訂組態檔案

您可以指定自訂組態檔案。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 [Fluentd Config File Syntax](#) 與 [YAML Configuration](#)。

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

上託管的任務 AWS Fargate 僅支援 `file` 組態檔案類型。不過，您可以使用 for Fluent Bit init 容器，使用 AWS Fargate 上 Amazon S3 中託管 AWS 的組態檔案。如需詳細資訊，請參閱 [ECS 上的 Fluent Bit 的 Init 程序](#)、[GitHub 上的多組態支援](#)。GitHub

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:3",
      "name": "log_router",
      "firelensConfiguration": {
        "type": "fluentbit",
        "options": {
          "config-file-type": "s3 | file",
          "config-file-value": "arn:aws:s3:::amzn-s3-demo-bucket/fluent.conf"
        }
      }
    }
  ]
}
```

 Note

上託管的任務 AWS Fargate 僅支援file組態檔案類型。不過，您可以使用 for Fluent Bit init 容器，使用 AWS Fargate 上 Amazon S3 中託管 AWS 的組態檔案。如需詳細資訊，請參閱 [ECS 上的 Fluent Bit 的 Init 程序](#)、[GitHub 上的多組態支援](#)。GitHub

在 Amazon ECS 中使用非AWS 容器映像

使用私有登錄檔將您的登入資料存放在 中 AWS Secrets Manager，然後在任務定義中參考它們。這可讓您參考 外部私有登錄檔中存在的容器映像 AWS，這些登錄檔需要在您的任務定義中進行身分驗證。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 User Guide 中的 [Using Amazon ECR Images with 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。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 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。參考的秘密可以來自與使用它的任務不同的 AWS 區域 或不同的 帳戶。

 Note

使用 Amazon ECS API AWS CLI或 AWS SDK 時，如果秘密與您啟動 AWS 區域 的任務位於相同的 中，您可以使用秘密的完整 ARN 或名稱。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 管理主控台，一律必須指定秘密的完整 ARN。

以下是任務定義的程式碼片段，其會顯示所需的參數：

替代下列參數：

- 將 `private-repo` 取代為私有儲存庫的主機名稱
- 將 `private-image` 取代為映像名稱
- 將 `arn:aws:secretsmanager:region:aws_account_id:secret:secret_name` 取代為秘密的 Amazon Resource Name (ARN)

```
"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 Docker 映像檔設定 Amazon ECS 容器執行個體](#)。

使用私有登錄檔

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](#)。

私有登錄檔身分驗證可讓您的 Amazon ECS 任務從需要身分驗證憑證的 AWS（例如 Docker Hub、Quay.io 或您自己的私有登錄檔）外部的私有登錄檔提取容器映像。此功能透過 Secrets Manager 安全地儲存登錄檔憑證，這些憑證隨後會在任務定義中透過 `repositoryCredentials` 參數進行引用。

如需設定私有登錄檔身分驗證的詳細資訊，請參閱[在 Amazon ECS 中使用非AWS 容器映像](#)。

若要存取內含私有登錄檔憑證的秘密，請將以下許可以內嵌政策形式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](#)。

- `secretsmanager:GetSecretValue` – 從 Secrets Manager 擷取私有登錄檔憑證時需要。
- `kms:Decrypt` - 只有在您的秘密使用自訂 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。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "kms:Decrypt",
        "secretsmanager:GetSecretValue"
      ],
      "Resource": [
        "arn:aws:secretsmanager:us-east-1:111122223333:secret:secret_name",
        "arn:aws:kms:us-east-1:111122223333:key/key_id"
      ]
    }
  ]
}
```

2. 使用 AWS Secrets Manager 為您的私有登錄檔登入資料建立秘密。如需有關如何建立秘密的資訊，請參閱 AWS Secrets Manager User Guide 中的 [Create an AWS Secrets Manager secret](#)。

按照下列格式輸入私有登錄檔憑證：

```
{
  "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”](#)。

使用容器重新啟動政策在 Amazon ECS 任務中重新啟動個別容器

您可以為任務定義中定義的每個必要與非必要容器啟用重新啟動政策，以更快地克服暫時性故障並維持任務可用性。在為容器啟用重新啟動政策後，如果容器結束，Amazon ECS 可以重新啟動容器，無需取代任務。

容器預設不會啟用重新啟動政策。在為容器啟用重新啟動政策後，您可以指定不會重新啟動容器的結束代碼。這些可以是指示成功的結束代碼，例如不需要重新啟動的 0 結束代碼。您亦可指定容器必須持續成功執行多久之後才可嘗試重新啟動。如需這些參數的相關資訊，請參閱 [重新啟動政策](#)。如需指定這些值的任務定義範例，請參閱在 [Amazon ECS 任務定義中指定容器重新啟動政策](#)。

您可以使用 Amazon ECS 任務中繼資料端點或 CloudWatch Container Insights 來監控容器重新啟動的次數。如需有關任務中繼資料端點的詳細資訊，請參閱 [Amazon ECS 任務中繼資料端點第 4 版](#)與[Fargate 上任務的 Amazon ECS 任務中繼資料端點第 4 版](#)。如需有關 Amazon ECS Container Insights 指標的詳細資訊，請參閱 Amazon CloudWatch User Guide 中的 [Amazon ECS Container Insights metrics](#)。

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，皆支援容器重新啟動政策。

考量事項

為容器啟用重新啟動政策之前，請考量下列事項：

- Fargate 上的 Windows 容器不支援重新啟動政策。
- 對於託管於 Amazon EC2 執行個體上的任務，此功能要求具備版本 1.86.0 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。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](#)」。
- 如果搭配 bridge 網路模式使用 EC2，應用程式容器中的 FLUENT_HOST 環境變數可能會在重新啟動 FireLens 日誌路由器容器 (容器定義中具有 firelensConfiguration 物件的容器) 之後變得不準確。這是因為 FLUENT_HOST 是動態 IP 位址，重新啟動之後可能發生變更。從應用程式容器直接向 FLUENT_HOST IP 位址寫入日誌的操作，可能會在該位址變更後開始失敗。如需 FLUENT_HOST 的相關資訊，請參閱 [為高輸送量設定 Amazon ECS 日誌](#)。
- Amazon ECS 代理程式會處理容器重新啟動政策。如果某些非預期原因導致 Amazon ECS 代理程式失敗或不再執行，容器將不會重新啟動。
- 政策中定義的重啟嘗試週期，決定容器在 Amazon ECS 重新啟動容器之前必須執行的時長 (以秒為單位)。

在 Amazon ECS 任務定義中指定容器重新啟動政策

若要在任務定義中指定容器重新啟動政策，需在容器定義中指定 `restartPolicy` 物件。如需有關 `restartPolicy` 物件的詳細資訊，請參閱[重新啟動政策](#)。

以下是使用 Fargate 上 Linux 容器並設定 Web 伺服器的任務定義。容器定義包含 `restartPolicy` 物件，並將 `enabled` 設定為 `true` 以啟用容器重新啟動政策。容器必須先執行 180 秒，才能重新啟動；若容器以表示成功的結束代碼 0 結束，則不會被重新啟動。

```
{
  "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": "public.ecr.aws/docker/library/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"
        }
      ],
      "restartPolicy": {
        "enabled": true,
        "ignoredExitCodes": [0],
        "restartAttemptPeriod": 180
      }
    }
  ]
}
```

```
    }  
  }  
],  
"cpu": "256",  
"executionRoleArn": "arn:aws:iam::012345678910:role/ecsTaskExecutionRole",  
"family": "fargate-task-definition",  
"memory": "512",  
"networkMode": "awsvpc",  
"runtimePlatform": {  
  "operatingSystemFamily": "LINUX"  
},  
"requiresCompatibilities": ["FARGATE"]  
}
```

在容器定義中註冊包含 `restartPolicy` 物件的任務定義後，您可以使用該任務定義來執行任務或建立服務。如需詳細資訊，請參閱[將應用程式作為 Amazon ECS 任務執行](#)及[建立 Amazon ECS 滾動更新部署](#)。

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。

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 參數存放區無縫注入容器。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](#)。

建議您在設定秘密管理時執行下列動作。

使用 AWS Secrets Manager 或 AWS Systems Manager 參數存放區來存放秘密資料

您應該將 API 金鑰、資料庫憑證與其他秘密資料安全儲存在 Secrets Manager 中，或將其作為加密參數安全儲存在 Systems Manager Parameter Store 中。這些服務類似，因為它們都是 AWS KMS

用於加密敏感資料的受管金鑰值存放區。不過，Secrets Manager 還具備自動輪換秘密、產生隨機秘密及跨帳戶共用秘密的能力。若要使用這些功能，可選擇 Secrets Manager。反之，則可在 Systems Manager Parameter Store 中使用加密參數。

Important

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：

- 任務 – 先停止任務，再重新啟動。如需詳細資訊，請參閱[停止 Amazon ECS 任務](#)及[將應用程式作為 Amazon ECS 任務執行](#)。
- 服務 – 更新服務並使用「強制執行新部署」選項。如需詳細資訊，請參閱[更新 Amazon ECS 服務](#)。

從加密的 Amazon S3 儲存貯體擷取資料

您應該將秘密儲存於加密的 Amazon S3 儲存貯體中，並使用任務角色來限制對這些秘密的存取。此舉可防止環境變數的值在日誌中意外洩漏，或在執行 `docker inspect` 時遭到暴露。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

由於環境變數的資料外洩風險較高，因此您應該執行附屬容器，從讀取秘密 AWS Secrets Manager 並將其寫入共用磁碟區。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。如需範例，請參閱 [task-def.json](#) 專案。

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。在上 AWS Fargate，磁碟區儲存會使用服務受管金鑰自動加密。

將個別環境變數傳遞至 Amazon ECS 容器

Important

建議您將敏感資料儲存在 AWS Secrets Manager 秘密或 AWS Systems Manager 參數存放區參數中。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 container 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"
        }
      ],
      ...
    }
  ],
  ...
}
```

將環境變數傳遞至 Amazon ECS 容器

Important

建議您將敏感資料儲存在 AWS Secrets Manager 秘密或 AWS Systems Manager 參數存放區參數中。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。

Windows 容器與 Fargate 上的 Windows 容器不支援使用 `environmentFiles` 參數。

您可以建立環境變數檔案，並將其儲存於 Amazon S3 中，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 會在任務清理過程中移除該檔案。

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 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 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。
- 此功能需搭配任務執行角色使用，且該角色必須具備額外的 Amazon S3 許可。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](#)。
- 每個任務定義限制 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。如需有關環境變數檔案語法的詳細資訊，請參閱 Docker 文件中的 [Set environment variables \(-e, --env, --env-file\)](#)。

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：
 - 檔案的處理方式與本機 Docker 環境檔案類似。
 - 若容器定義所引用的環境變數值為空白且儲存於 Amazon S3 中，則容器定義將不會出現在容器中。
 - 不支援 Shell 逸出處理。
 - 容器進入點會解譯 `VARIABLE` 值。

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:::amzn-s3-demo-  
bucket/envfile_object_name.env",
          "type": "s3"
        }
      ],
      ...
    }
  ],
  ...
}
```

在 Amazon ECS 中以程式設計方式傳遞 Secrets Manager 秘密

您可以使用 Secrets Manager 來儲存敏感資料，不必在應用程式中以純文字硬編碼敏感資訊。

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。

- 僅支援存放文字資料的秘密，這些秘密是使用 [CreateSecret](#) API 的 `SecretString` 參數建立的。不支援儲存二進位資料的秘密，即透過 [CreateSecret](#) API 的 `SecretBinary` 參數建立的秘密。
- 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。
- 任務定義必須使用具備 Secrets Manager 額外許可的任務角色。如需詳細資訊，請參閱[Amazon ECS 任務 IAM 角色](#)。

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。如需詳細資訊，請參閱 AWS Secrets Manager User Guide 中的[Retrieve secrets from AWS Secrets Manager](#)。

若要擷取存放在 中的敏感資料 AWS Secrets Manager，請參閱 SDK [程式碼範例程式碼庫中的使用 AWS SDKs 的 AWS Secrets Manager](#) 程式碼範例。AWS

在 Amazon ECS 中以程式設計方式傳遞 Systems Manager Parameter Store 秘密

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、AMI ID、授權碼等資料作為參數值儲存，不必在應用程式中硬編碼此類資訊。您存放的值可以是純文字或加密資料。

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 Systems Manager Parameter Store 參數有更新，應用程式會自動擷取最新版本。

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。
- 對於使用 EC2 的任務，您必須使用 Amazon 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](#)。
- 任務定義必須使用具備 Systems Manager Parameter Store 額外許可的任務角色。如需詳細資訊，請參閱[Amazon ECS 任務 IAM 角色](#)。

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敏感資料，請參閱 SDK [程式碼範例程式碼庫中的使用 AWS SDKs 的 Systems Manager](#) 程式碼範例。AWS

透過 Amazon ECS 環境變數傳遞 Secrets Manager 秘密

在將秘密作為環境變數插入時，可指定秘密的完整內容、秘密內的特定 JSON 金鑰。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 User Guide 中的 [What's in a Secrets Manager secret?](#) 章節。

使用環境變數將 Secrets Manager 秘密插入容器時應考量下列事項。

- 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。
- 容器上執行的應用程式、容器日誌及偵錯工具皆能存取環境變數。
- 對於上的 Amazon ECS 任務 AWS Fargate，請考慮下列事項：
 - 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 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 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。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能力，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 與 Amazon VPC 的詳細資訊，請參閱[How to connect to Secrets Manager service within a Amazon VPC](#)。
- 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。使用下列語法：

```
<powershell>
[Environment]::SetEnvironmentVariable("ECS_ENABLE_AWSLOGS_EXECUTIONROLE_OVERRIDE",
$TRUE, "Machine")
Initialize-ECSAgent -Cluster <cluster name> -EnableTaskIAMRole -LoggingDrivers
'["json-file","awslogs"]'
</powershell>
```

- 任務定義必須使用具備 Secrets Manager 額外許可的任務執行角色。如需詳細資訊，請參閱[Amazon ECS 任務執行 IAM 角色](#)。

建立 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。如需詳細資訊，請參閱 AWS Secrets Manager 《使用者指南》中的[建立 AWS Secrets Manager 秘密](#)。

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。

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 引用完整秘密

以下是任務定義的程式碼片段，顯示引用多個 Secrets Manager 秘密全文時的格式。

```

{
  "containerDefinitions": [{
    "secrets": [
      {
        "name": "environment_variable_name1",
        "valueFrom": "arn:aws:secretsmanager:region:aws_account_id:secret:secret_name-
AbCdEf"
      },
      {
        "name": "environment_variable_name2",
        "valueFrom": "arn:aws:secretsmanager:region:aws_account_id:secret:secret_name-
abcdef"
      },
      {
        "name": "environment_variable_name3",
        "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

`$environment_variable_name1`、`$environment_variable_name2`、`$environment_variable_name3`。

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"
  ],
  "CreateDate": 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 環境變數傳遞 Systems Manager 參數

Amazon ECS 可讓您將敏感資料儲存在 AWS Systems Manager 參數存放區參數中，然後在容器定義中參考，藉此將敏感資料注入容器。

使用環境變數將 Systems Manager 秘密插入容器時，應考量下列事項。

- 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 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 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。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 User Guide 中的 [Improve the security of EC2 instances by using VPC endpoints for Systems Manager](#)。
- 任務定義必須使用具備 Systems Manager Parameter Store 額外許可的任務執行角色。如需詳細資訊，請參閱[Amazon ECS 任務執行 IAM 角色](#)。
- 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。使用下列語法：

```
<powershell>
[Environment]::SetEnvironmentVariable("ECS_ENABLE_AWSLOGS_EXECUTIONROLE_OVERRIDE",
    $TRUE, "Machine")
Initialize-ECSAgent -Cluster <cluster name> -EnableTaskIAMRole -LoggingDrivers
    '["json-file","awslogs"]'
</powershell>
```

建立 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

在任務定義的容器定義內，為 `secrets` 指定要在容器中設定的環境變數名稱，以及 Systems Manager Parameter Store 參數 (含有要提供給容器的敏感資料) 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](#)。

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敏感資料，請參閱 SDK [程式碼範例程式碼庫中的使用 AWS SDKs 的 Systems Manager](#) 程式碼範例。AWS

為 Amazon ECS 記錄組態傳遞秘密

您可以使用 `logConfiguration` 中的 `secretOptions` 參數，傳遞用於記錄的敏感資料。

您可以將秘密儲存於 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。如需有關建立秘密的詳細資訊，請參閱 [Create an AWS Secrets Manager](#)。

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"
      }]
    }]
  }]
}
```

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](#)。

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"
      }]
    }]
  }]
}
```

```
    }  
  }  
}
```

在 Amazon ECS 中使用 Secrets Manager 秘密指定敏感資料

Amazon ECS 可讓您將敏感資料存放在 AWS Secrets Manager 秘密中，然後在容器定義中參考，藉此將敏感資料注入容器。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 秘密、引用 Amazon ECS 任務定義中的秘密，然後查詢容器內顯示秘密內容的環境變數，驗證其是否能運作。

先決條件

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。
- 您的使用者具有必要的 IAM 許可，可建立 Secrets Manager 與 Amazon ECS 資源。

步驟 1：建立 Secrets Manager 機密

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 AWS Secrets Manager 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：將秘密許可新增至任務執行角色

若要讓 Amazon ECS 從 Secrets Manager 秘密擷取敏感資料，您必須擁有任務執行角色的秘密許可。如需詳細資訊，請參閱[Secrets Manager 或 Systems Manager 許可](#)。

步驟 3：建立任務定義

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，確保為在步驟 1 建立的 Secrets Manager 秘密以及在步驟 2 中更新的任務執行角色，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": "public.ecr.aws/docker/library/httpd:2.4",
            "essential": true,
            "name": "ecs-secrets-container"
        }
    ],
}
```

```
"family": "ecs-secrets-tutorial"  
}
```

6. 選擇建立。

步驟 4：建立叢集

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 叢集的資訊，請參閱[the section called “為 Amazon EC2 工作負載建立叢集”](#)。

步驟 5：執行任務

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 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 “將應用程式作為任務執行”](#)。

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 User Guide 中的 [Connect to your Linux instance using PuTTY](#)。

 Important

如需有關執行個體連線問題的詳細資訊，請參閱 Amazon EC2 User Guide 中的 [Troubleshooting Connecting to Your Instance](#)。

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. 選擇許可，然後選擇 ECSSecretsTutorial 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 受管執行個體的 Amazon ECS 任務定義參數

任務定義分為不同的部分：任務系列、AWS Identity and Access Management (IAM) 任務角色、網路模式、容器定義、磁碟區和容量。任務定義中需要系列和容器定義。與此相對，任務角色、網路模式、磁碟區與容量則為選用項目。

您可以在 JSON 檔案中使用這些參數來設定任務定義。

以下是 Amazon ECS 受管執行個體每個任務定義參數的詳細描述。

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。

Capacity

在註冊任務定義時，您可以指定 Amazon ECS 驗證任務定義所依據的容量。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:MANAGED_INSTANCES

驗證任務定義所依據的容量。這會啟動檢查，確保任務定義中使用的所有參數都能滿足 Amazon ECS 受管執行個體的需求。

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](#)。

任務執行 角色

`executionRoleArn`

類型：字串

必要：有條件

任務執行角色的 Amazon Resource Name (ARN)，授予 Amazon ECS 容器代理程式代表您進行 AWS API 呼叫的許可。如需詳細資訊，請參閱[Amazon ECS 任務執行 IAM 角色](#)。

Note

視任務需求而定，任務執行 IAM 角色是必要的項目。私有 ECR 映像提取和使用 `awslogs` 日誌驅動程式需要 角色。

網路模式

networkMode

類型：字串

必要：否

預設：awsvpc

用於任務中容器的聯網模式。對於在 Amazon ECS 受管執行個體上託管的 Amazon ECS 任務，有效值為 awsvpc 與 host。如果未指定網路模式，預設網路模式為 awsvpc。

如果網路模式為 host，任務會略過網路隔離，而容器會直接使用主機的网络堆疊。

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。

如果網路模式是 awsvpc，則任務會配置一個彈性網路介面，而當您使用此任務定義建立服務或執行任務時，您必須指定 NetworkConfiguration。如需詳細資訊，請參閱[Amazon ECS 受管執行個體的 Amazon ECS 任務聯網](#)。

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。

執行時間平台

operatingSystemFamily

類型：字串

必要：否

預設：LINUX

在註冊任務定義時，您會指定作業系統系列。

此欄位的有效值為 LINUX。

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

在註冊任務定義時，您會指定 CPU 架構。有效值為 X86_64 和 ARM64。

如果您未指定值，Amazon ECS 會根據容量提供者組態嘗試將任務放置在可用的 CPU 架構上。為了確保任務放置在特定 CPU 架構上，請在任務定義 `cpuArchitecture` 中指定的值。

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。

如需 ARM64 的相關資訊，請參閱 [the section called “64 位元 ARM 工作負載的任務定義”](#)。

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。

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。

cpu

類型：字串

必要：有條件

針對任務呈現的 CPU 單位硬性限制。您可以在 JSON 檔案中將 CPU 值指定為以 CPU 為單位或虛擬 CPU (vCPU) 為單位的字串。例如，可指定 1024 個 CPU 單位作為 CPU 值，也可指定 1 vCPU 個 vCPU 作為 CPU 值。註冊任務定義時，vCPU 值會轉換為整數，指出 CPU 單位。

此欄位為選用欄位。如果您的叢集沒有任何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。支援的值介於 0.125 個 vCPU 與 10 個 vCPU 之間。

memory

類型：字串

必要：有條件

要分配給任務的記憶體硬性限制。您可以在任務定義中將記憶體值指定為以 MiB 或 GB 為單位的字串。例如，可指定 3072 MiB 作為記憶體值，也可指定 3 GB 作為記憶體值。註冊任務定義時，GB 值會轉換為整數，指出 MiB。

此欄位為選填，且可以使用任何值。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](#)。

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](#)
- [代理組態](#)
- [Tags \(標籤\)](#)
- [Elastic Inference 加速器 \(已棄用\)](#)
- [置放限制條件](#)
- [磁碟區](#)

暫時性儲存

ephemeralStorage

Note

在 Amazon ECS 受管執行個體上執行的任務不支援此參數。

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”](#)。

IPC 模式

ipcMode

Note

在 Amazon ECS 受管執行個體上執行的任務不支援此參數。

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。如果未指定任何值，則 IPC 資源命名空間共用取決於容器執行期組態。

PID 模式

pidMode

類型：字串

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`。如果已指定 `host`，則任務內對同一容器執行個體指定 `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，預設值會是私有命名空間。

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。

代理組態

proxyConfiguration

Note

在 Amazon ECS 受管執行個體上執行的任務不支援此參數。

類型：[ProxyConfiguration](#) 物件

必要：否

App Mesh 代理的組態詳細資訊。

Tags (標籤)

您套用至任務定義的中繼資料，可用於協助您對任務定義進行分類與整理。每個標籤皆包含索引鍵與選用值。您可以兩個都定義。

以下基本限制適用於 標籤：

- 每個資源的標籤數上限 - 50
- 針對每一個資源，每個標籤鍵必須是唯一的，且每個標籤鍵只能有一個值。
- 索引鍵長度上限 - 128 個 UTF-8 Unicode 字元
- 值的長度上限 - 256 個 UTF-8 Unicode 字元
- 如果您的標記結構描述是跨多項服務和資源使用，請記得其他服務可能會有字元使用限制。通常允許的字元為：以 UTF-8 表示的字母、數字和空格，以及下列字元：`+ - = . _ : / @`。

- 標籤鍵與值皆區分大小寫。
- 請不要使用 `aws:`、`AWS:` 或任何大寫或小寫的組合，例如保留供 AWS 使用的索引鍵或值的字首。您不可編輯或刪除具此字首的標籤金鑰或值。具此字首的標籤，不算在受資源限制的標籤計數內。

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。

Elastic Inference 加速器 (已棄用)

Note

Amazon Elastic Inference (EI) 服務已全面終止，不再向客戶提供。

inferenceAccelerator

Note

在 Amazon ECS 受管執行個體上執行的任務不支援此參數。

類型：[InferenceAccelerator](#) 物件

必要：否

要用於任務中容器的 Elastic Inference 加速器。

置放限制條件

placementConstraints

類型：[TaskDefinitionPlacementConstraint](#) 物件陣列

必要：否

要針對任務使用的置放限制條件物件陣列。每項任務您最多可以指定 10 項限制條件 (此限制包含任務定義中的限制條件以及執行時間指定的限制條件)。

Amazon ECS 支援在 Amazon ECS 受管執行個體上執行之任務的 `distinctInstance` 與 `memberOf` 置放限制條件。使用 `memberOf` 置放限制條件的任務支援下列屬性：

- `ecs.subnet-id`
- `ecs.availability-zone`
- `ecs.cpu-architecture`
- `ecs.instance-type`

如需有關置放限制條件的詳細資訊，請參閱[定義 Amazon ECS 用於任務的容器執行個體](#)。

磁碟區

註冊任務定義時，您可以選擇為您的任務指定磁碟區清單。此設定讓您可以在任務中使用資料磁碟區。

如需有關磁碟區類型與其他參數的詳細資訊，請參閱 [the section called “任務的儲存選項”](#)。

name

類型：字串

必要：是

磁碟區名稱。可以包含最多 255 個字母 (大小寫)、數字和連字號。此名稱是參考容器定義 `sourceVolume` 中的 `mountPoints` 參數。

host

類型：[HostVolumeProperties](#) 物件

必要：否

此參數會在您使用綁定掛載主機磁碟區時指定。host 參數的內容決定繫結掛載主機磁碟區是否保留在主機容器執行個體上以及其存放位置。如果 host 參數是空的，則系統會為您的資料磁碟區指派主機路徑。不過，在與其相關聯的容器停止執行後，不保證保留資料。

sourcePath

類型：字串

必要：否

使用 host 參數時，請指定 sourcePath 來宣告呈現給容器之主機執行個體上的路徑。如果此參數為空，則系統會為您指派主機路徑。如果 host 參數包含 sourcePath 檔案位置，則資料磁碟區會保留在主機執行個體上的指定位置，直到您手動刪除為止。如果該 sourcePath 值不存在於主機執行個體上，系統會建立該值。如果位置存在，將匯出來源路徑資料夾的內容。

在 Amazon ECS 受管執行個體上，部分主機檔案系統為唯讀。必須 sourcePath 指向可寫入的目錄，例如 /var 或 /tmp。如需詳細資訊，請參閱 [搭配 Amazon ECS 使用綁定掛載](#)。

dockerVolumeConfiguration

 Note

在 Amazon ECS 受管執行個體上執行的任務不支援此參數。

類型：[DockerVolumeConfiguration](#) 物件

必要：否

此參數會在您使用 Docker 磁碟區時指定。

efsVolumeConfiguration

類型：[EFSVolumeConfiguration](#) 物件

必要：否

此參數會在您使用適用於任務儲存體的 Amazon EFS 檔案系統時指定。

fsxWindowsFileServerVolumeConfiguration

 Note

在 Amazon ECS 受管執行個體上執行的任務不支援此參數。

類型：[FSxWindowsFileServerVolumeConfiguration](#) 物件

必要：否

此參數會在您為任務儲存使用 Amazon FSx for Windows File Server 檔案系統時指定。

`configuredAtLaunch`

類型：布林值

必要：否

指示是否應在啟動時設定磁碟區。此參數用於為獨立任務或作為服務一部分建立的任務建立 Amazon EBS 磁碟區。每個任務定義修訂版只能在磁碟區組態中於啟動時設定一個磁碟區。

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

- [名稱](#)
- [影像](#)
- [記憶體](#)
- [CPU](#)
- [連接埠映射](#)
- [私有儲存庫憑證](#)
- [主要](#)
- [進入點](#)
- [命令](#)
- [工作目錄](#)
- [進階容器定義參數](#)
- [Linux 參數](#)

名稱

name

類型：字串

必要：是

容器的名稱。最多允許 255 個字母（大寫和小寫）、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。

影像

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 create-container 命令中的 Image，以及 docker run 命令的 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。
- 當您未在任務定義中的映像路徑中指定標籤或摘要時，Amazon ECS 容器代理程式會使用 latest 標籤來提取指定的映像。
- 儲存庫映像的後續更新不會傳播至已執行的任務。
- 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`)。

versionConsistency

類型：字串

有效值：enabled|disabled

必要：否

指定 Amazon ECS 是否會將容器定義中提供的容器映像標籤解析為映像摘要。此行為預設為 enabled。如果將容器的值設定為 disabled，Amazon ECS 不會將容器映像標籤解析為摘要，但會使用容器定義中指定的原始映像 URI 進行部署。如需有關容器映像解析的詳細資訊，請參閱[容器映像解析](#)。

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)。此參數會映射至 `docker create-container` 命令中的 `Memory`，以及 `docker run` 的 `--memory` 選項。

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。此參數會映射至 `docker create-container` 命令中的 `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。

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](#)」。

CPU

`cpu`

類型：整數

必要：否

為容器預留的 `cpu` 單位數。此參數會映射至 `docker create-container` 命令中的 `CpuShares`，以及 `docker run` 的 `--cpu-shares` 選項。

對於使用 EC2 容量提供者的任務，此欄位為可選項，唯一要求是任務內為所有容器預留的 CPU 總量必須低於任務層級的 `cpu` 值。

Note

您可以透過將 [Amazon EC2 執行個體](#) 詳細資訊頁面上為執行個體類型列出的 vCPU 乘上 1,024，來判斷每個 EC2 執行個體類型可用的 CPU 單位數。

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，並且該任務是容器執行個體上唯一執行中的任務，該容器便可在任何指定時間內使用完整的 1,024 個 CPU 單位。但是，若您在該容器執行個體上啟動相同任務的另一個複本，可保證每個任務都可以在需要時取得至少 512 個 CPU 單位。此外，每個容器都可在其他容器沒有使用時浮動到更高的 CPU 用量。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 容器代理版本而異：

- 代理程式版本小於或等於 1.1.0：Null 和零 CPU 值會以 0 傳遞給 Docker，然後 Docker 將之轉換為 1,024 個 CPU 共用。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 共用。
- 代理程式版本大於或等於 1.2.0：Null、零和 1 的 CPU 值，會以 2 傳遞至 Docker。

在 Windows 容器執行個體上，CPU 限制會強制為絕對限制或配額。Windows 容器只能存取任務定義的描述中所指定之 CPU 數量。將空或零的 CPU 值以 0 的形式傳遞給 Docker，Windows 會將此值解釋為一個 CPU 的 1%。

連接埠映射

portMappings

類型：物件陣列

必要：否

連接埠映射會將容器的網路連接埠對外公開，讓用戶端可以存取您的應用程式。這也適用於同一任務中的容器間通訊。

針對使用 `awsvpc` 網路模式的任務定義，只指定 `containerPort`。一律略過 `hostPort`，且容器連接埠會自動映射至主機上的隨機高連接埠號碼。

此參數的大部分欄位 (包括 `containerPort`、`hostPort`、`protocol`) 會映射至 `docker create-container` 命令中的 `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 `CloudWatch` 中新增通訊協定專屬遙測。

如果您沒有為此參數設定值，系統會使用 `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。

對於使用 awsvpc 網路模式的任務，您可以透過 containerPort 指定公開的連接埠。

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 awsvpc 網路模式。
- 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。
- 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 GitHub 上的 [Issue #11185](#)。

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。

連接埠範圍未包含在傳送到 EventBridge 的 Amazon ECS 任務事件中。如需詳細資訊，請參閱 [the section called “使用 EventBridge 自動化因應 Amazon ECS 錯誤”](#)。

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。

hostPort 可維持空白，或與 containerPort 的值相同。

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 與 VPC Lattice

用於連接埠映射的名稱。此參數僅適用於 Service Connect 與 VPC Lattice。此參數是您在服務的 Service Connect 與 VPC Lattice 組態中使用的名稱。

如需詳細資訊，請參閱[使用 Service Connect 以利用簡稱連線 Amazon ECS 服務](#)。

下列範例中同時使用了 Service Connect 與 VPC Lattice 的必要欄位。

```
"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。

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憑證。

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、AWS CLI 或 AWS SDKs 時，如果秘密與您啟動的任務位於相同的區域中，您可以使用完整的 ARN 或秘密的名稱。使用時 AWS 管理主控台，您必須指定秘密的完整 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"  
    }  
  }  
]
```

主要

essential

類型：布林值

必要：否

若容器的 `essential` 參數標記為 `true`，並且該容器因為任何理由失敗或停止，則所有為任務之一部分的其他容器都會停止。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。若您的應用程式由多個容器組成，將一般用途用的容器分組成元件，並將不同的元件分成多個任務定義。如需詳細資訊，請參閱[the section called “架構應用程式”](#)。

進入點

entryPoint

類型：字串陣列

必要：否

傳遞至容器的進入點。此參數會映射至 `docker create-container` 命令中的 `Entrypoint`，以及 `docker run` 的 `--entrypoint` 選項。

```
"entryPoint": ["string", ...]
```


命令

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 create-container` 命令中的 `Cmd`，以及 `docker run` 的 `COMMAND` 參數。如果有多個引數，每個引數在陣列中都是分開的字串。

```
"command": ["string", ...]
```

工作目錄

workingDirectory

類型：字串

必要：否

容器內要執行命令的工作目錄。此參數會映射至 `docker create-container` 命令中的 `WorkingDir`，以及 `docker run` 的 `--workdir` 選項。

進階容器定義參數

下列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

- [重新啟動政策](#)
- [運作狀態檢查](#)
- [Environment](#)
- [安全](#)
- [網路設定](#)
- [儲存與記錄](#)
- [資源需求](#)

- [容器逾時](#)
- [容器相依性](#)
- [系統控制](#)
- [互動性](#)
- [虛擬終端機](#)

重新啟動政策

restartPolicy

容器重新啟動政策及相關組態參數。在為容器設定重新啟動政策後，Amazon ECS 可以重新啟動容器，無需取代任務。如需詳細資訊，請參閱[使用容器重新啟動政策在 Amazon ECS 任務中重新啟動個別容器](#)。

enabled

類型：布林值

必要：是

指定是否已為容器啟用重新啟動政策。

ignoredExitCodes

類型：Integer array

必要：否

Amazon ECS 會略過且不嘗試重新啟動的結束代碼清單。最多可以指定 50 個容器結束代碼。Amazon ECS 預設不會略過任何結束代碼。

restartAttemptPeriod

類型：整數

必要：否

容器必須執行一段時間 (以秒為單位) 後，才能嘗試重新啟動。容器每 restartAttemptPeriod 秒只能重新啟動一次。如果容器未能執行達此時長即提前結束，則不會重新啟動。restartAttemptPeriod 最少可設定 60 秒，restartAttemptPeriod 最多可設定 1800 秒。容器預設必須執行 300 秒後才能重新啟動。

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來判斷 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 中註冊任務定義時 AWS 管理主控台，請使用逗號分隔的命令清單。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 JSON AWS 管理主控台 面板 AWS CLI、或 APIs 註冊任務定義時，請將命令清單括在括號中。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。

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。您可以指定介於 0 到 300 秒之間的值。預設情況下，startPeriod 是停用的。

如果運作狀態檢查在 `startPeriod` 內成功，則代表容器運作狀態良好，之後的任何故障都會計入，累積至重試次數上限。

Environment

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，此參數會映射至 [Create a container](#) 區段中的 `CpuShares`。

對於在 Amazon ECS 受管執行個體上執行的任務，此欄位為選用項目。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。

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。Linux 核心允許的有效 CPU 共用值，最小為 2，最大為 262144。但是，CPU 參數並非必要項目，且您可以在容器定義中使用小於 2 且大於 262144 的 CPU 值。針對小於 2 且大於 262144 的 CPU 值 (包含 null)，行為會隨您的 Amazon ECS 容器代理程式版本而異：

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](#)。

Elastic Inference accelerator

Note

在 Amazon ECS 受管執行個體上託管的容器不支援此參數。

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（已棄用）”](#)。

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。

```
"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 create-container 命令中的 Cmd，以及 docker run 的 COMMAND 參數。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。此參數會映射至 docker run 命令的 --env-file 選項。

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。此參數會映射至 docker create-container 命令中的 Env 參數，以及 docker run 命令的 --env 選項。

Important

不建議使用純文字環境變數儲存敏感資訊 (例如憑證)。

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。支援的值為 AWS Secrets Manager 秘密的完整 Amazon Resource Name (ARN) 或 AWS Systems Manager 參數存放區中參數的完整 ARN。

Note

如果 Systems Manager 參數存放區參數或 Secrets Manager 參數與您啟動 AWS 區域的任務位於相同的 中，您可以使用秘密的完整 ARN 或名稱。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"  
  }  
]
```



```
] ]
```

安全

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許可 (類似 `root` 使用者)。此參數會映射至 `docker create-container` 命令中的 `Privileged`，以及 `docker run` 的 `--privileged` 選項。

user

類型：字串

必要：否

要在容器內使用的使用者。此參數會映射至 `docker create-container` 命令中的 `User`，以及 `docker run` 的 `--user` 選項。

Important

使用 `host` 網路模式執行任務時，請勿使用根使用者 (UID 0) 執行容器。建議使用非根使用者以提高安全性。

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`

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一讀存取權。此參數會映射至 `docker create-container` 命令中的 `ReadonlyRootfs`，以及 `docker run` 的 `--read-only` 選項。

`dockerSecurityOptions`

 Note

在 Amazon ECS 受管執行個體上執行的任務不支援此參數。

類型：字串陣列

必要：否

提供 SELinux 和 AppArmor 多層級安全系統自訂標籤的字串清單。此欄位對使用 Fargate 的任務中的容器無效。

`ulimits`

類型：[Ulimit](#) 物件陣列

必要：否

容器中要設定的 `ulimits` 清單。如果在任務定義中指定 `ulimit` 值，該值會覆寫 Docker 設定的預設值。此參數會映射至 `docker create-container` 命令中的 `Ulimits`，以及 `docker run` 的 `--ulimit` 選項。有效命名值會顯示在 [Ulimit](#) 資料類型中。

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 Fargate 會覆寫的 `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。預設的 `nofile` 軟性限制為 1024，預設的硬性限制為 65535。

此參數需要容器執行個體上的 Docker Remote API 1.18 版或更新版本。若要檢查容器執行個體的 Docker Remote API 版本，請登入容器執行個體，並執行下列命令：`sudo docker version --format '{{.Server.APIVersion}}'`

`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。此參數會映射至 `docker create-container` 命令中的 `Labels`，以及 `docker run` 的 `--label` 選項。

此參數需要容器執行個體上的 Docker Remote API 1.18 版或更新版本。

```
"dockerLabels": {"string": "string"
  ...}
```

網路設定

`disableNetworking`

Note

在 Amazon ECS 受管執行個體上執行的任務不支援此參數。

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。

預設值為 `false`。

```
"disableNetworking": true|false
```

`links`

Note

在 Amazon ECS 受管執行個體上執行的任務不支援此參數。

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。最多可輸入 255 個字母 (大小寫)、數字、連字號與底線。

 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

 Note

在 Amazon ECS 受管執行個體上執行的任務不支援此參數。

類型：字串

必要：否

您容器要使用的主機名稱。此參數會映射至 docker create-container 中的 Hostname，以及 docker run 的 --hostname 選項。

```
"hostname": "string"
```

dnsServers

 Note

在 Amazon ECS 受管執行個體上執行的任務不支援此參數。

類型：字串陣列

必要：否

提供給容器的 DNS 伺服器清單。

```
"dnsServers": ["string", ...]
```

extraHosts

Note

使用 awsvpc 網路模式的任務不支援此參數。

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。

此參數會映射至 docker create-container 命令中的 ExtraHosts，以及 docker run 的 --add-host 選項。

```
"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一讀存取權。此參數會映射至 `docker create-container` 命令中的 `ReadonlyRootfs`，以及 `docker run` 的 `--read-only` 選項。

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載點。此參數會映射至 Docker API 中 `create-container` 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，且掛載點不能跨磁碟機使用。您必須指定掛載點，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)

掛載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一讀存取權。如果此值為 `false`，則容器可寫入磁碟區。預設值為 `false`。

對於在執行 Windows 作業系統之 EC2 執行個體上執行的任務，將該值保留為預設值 `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。此參數會映射至 `docker create-container` 命令中的 `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一讀存取權。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](#)。

此參數會映射至 docker create-container 命令中的 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)。

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。
- 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。

```
"logConfiguration": {
  "logDriver": "awslogs","splunk", "awsfirelens",
  "options": {"string": "string"
    ...},
  "secretOptions": [{
    "name": "string",
    "valueFrom": "string"
  }]
}
```

logDriver

類型：字串

有效值："awslogs","splunk","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。

支援的日誌驅動程式為 `awslogs`、`splunk` 與 `awsfirelens`。

如需有關如何在任務定義中使用 `awslogs` 日誌驅動程式將您的容器日誌傳送到 CloudWatch Logs 的詳細資訊，請參閱[將 Amazon ECS 日誌傳送至 CloudWatch](#)。

如需有關使用 `awsfirelens` 日誌驅動程式的詳細資訊，請參閱[將 Amazon ECS 日誌傳送至 AWS 服務或 AWS Partner](#)。

 Note

如果您有未列出的自訂驅動程式，您可以分支 [GitHub 上可取得的](#) Amazon ECS 容器代理程式專案，並自訂以搭配此驅動程式一起使用。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。

此參數需要容器執行個體上的 Docker Remote API 1.18 版或更新版本。

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。

可供指定的選項視日誌驅動程式而定。使用 `awslogs` 路由器將日誌路由至 Amazon CloudWatch 時，可指定的部分選項包括：

`awslogs-create-group`

必要：否

指定您是否希望自動建立日誌群組。若未指定此選項，則預設為 `false`。

 Note

在您嘗試使用 `awslogs-create-group` 之前，您的 IAM 政策必須包含 `logs:CreateLogGroup` 許可。

awslogs-region

必要：是

指定 AWS 區域 awslogs 日誌驅動程式要傳送 Docker 日誌的。您可以選擇將所有的日誌從不同區域中的叢集傳送至 CloudWatch Logs 中的單一區域。如此一來，日誌全都顯示在單一位置中。或者，您可以依照區域分隔日誌，以獲得更高的精細程度。請確定指定的日誌群組存在於您使用此選項指定的區域。

awslogs-group

必要：是

請務必指定 awslogs 日誌驅動程式傳送其日誌串流的日誌群組。

awslogs-stream-prefix

必要：是

使用 awslogs-stream-prefix 選項將日誌串流與指定字首、容器名稱以及容器所屬 Amazon ECS 任務的 ID 建立關聯。如果您使用此選項指定前綴，則日誌串流會使用下列格式。

```
prefix-name/container-name/ecs-task-id
```

如果您未使用此選項指定字首，則該日誌串流會以容器執行個體上 Docker 常駐程式指派的容器 ID 命名。因為只使用 Docker 容器 ID (僅容器執行個體提供) 很難對日誌回溯追蹤到傳送該日誌的容器，所以建議您使用此選項指定前綴。

對於 Amazon ECS 服務，您可以使用服務名稱作為前綴。藉此這可讓您對日誌串流回溯追蹤到容器所屬的服務、傳送該日誌之容器的名稱，以及容器所屬任務的 ID。

您必須指定日誌的串流字首，才可在使用 Amazon ECS 主控台時，讓日誌顯示在 Log (日誌) 窗格中。

awslogs-datetime-format

必要：否

此選項會以 Python strftime 格式定義多行開始模式。日誌訊息由符合模式的一行以及不符合模式的任何後續行所組成。符合的行是日誌訊息之間的分隔符號。

使用此格式的一個使用案例範例是用於剖析輸出，例如堆疊傾印，在其他情形下這可能會記錄在多個項目中。正確的模式可允許將它擷取在單一項目中。

如需詳細資訊，請參閱 [awslogs-datetime-format](#)。

無法同時設定 `awslogs-datetime-format` 和 `awslogs-multiline-pattern` 選項。

 Note

多行記錄會執行常規運算式剖析並比對所有日誌訊息。這可能會對記錄效能造成負面影響。

`awslogs-multiline-pattern`

必要：否

此選項定義使用常規運算式的多行開始模式。日誌訊息由符合模式的一行以及不符合模式的任何後續行所組成。符合的行是日誌訊息之間的分隔符號。

如需詳細資訊，請參閱 [awslogs-multiline-pattern](#)。

如果同時設定 `awslogs-datetime-format`，會忽略此選項。

無法同時設定 `awslogs-datetime-format` 和 `awslogs-multiline-pattern` 選項。

 Note

多行記錄會執行常規運算式剖析並比對所有日誌訊息。這可能會對記錄效能造成負面影響。

`mode`

必要：否

有效值：`non-blocking` | `blocking`

此選項定義從容器到 `awslogs` 日誌驅動程式的日誌訊息傳送模式。當從容器的日誌流程中斷時，您選擇的傳送模式會影響應用程式可用性。

如果使用 blocking 模式，且 CloudWatch 的日誌流程中斷，則從容器程式碼寫入 stdout 與 stderr 串流的呼叫將會封鎖。應用程式的日誌記錄執行緒將因此封鎖。這可能會導致應用程式沒有回應，並導致容器運作狀態檢查失敗。

如果您使用 non-blocking 模式，則容器的日誌將儲存在使用 max-buffer-size 選項設定的記憶體內中間緩衝區中。這可以防止當日誌無法傳送至 CloudWatch 時應用程式發生無回應的狀況。如果您想要確保服務的可用性且可以接受一些日誌遺失，我們建議您使用此模式。如需詳細資訊，請參閱 [Preventing log loss with non-blocking mode in the awslogs container log driver](#)。

max-buffer-size

必要：否

預設值：1m

使用 non-blocking 模式時，max-buffer-size 日誌選項會控制用於中繼訊息儲存的緩衝區大小。請務必根據您的應用程式指定適當的緩衝區大小。當緩衝區填滿時，無法儲存進一步的日誌。無法儲存的日誌將會遺失。

要使用 splunk 日誌路由器路由日誌，需指定 splunk-token 與 splunk-url。

當您使用 awsfirelens 日誌路由器將日誌路由到 AWS 服務 或 AWS Partner Network 目的地以進行日誌儲存和分析時，您可以設定 log-driver-buffer-limit 選項來限制記憶體中緩衝的事件數量，然後再傳送至日誌路由器容器。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 “為高輸送量設定日誌”](#)。

使用 awsfirelens 路由日誌時，可指定的其他選項取決於目的地。當您將日誌匯出至 Amazon Data Firehose 時，您可以使用 指定 AWS 區域，region 並使用 指定日誌串流的名稱 delivery_stream。

在將日誌匯出至 Amazon Kinesis Data Streams 時，可透過 region 指定 AWS 區域，透過 stream 指定日誌串流名稱。

在將日誌匯出至 Amazon OpenSearch Service 時，可指定多個選項，例如 Name、Host (不含通訊協定的 OpenSearch Service 端點) Port、Index、Type、Aws_auth、Aws_region、Suppress_Type_Name 與 tls。

在將日誌匯出至 Amazon S3 時，可透過 bucket 選項指定儲存貯體。您也可以指定 region、total_file_size、upload_timeout 與 use_put_object 作為選項。

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

容器的 FireLens 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 日誌傳送至 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:::*amzn-s3-demo-bucket*/fluent.conf|filepath"}。如需詳細資訊，請參閱[Amazon ECS 任務定義範例：將日誌路由至 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。

資源需求

resourceRequirements

類型：[ResourceRequirement](#) 物件陣列

必要：否

指派給容器的資源類型和數量。目前僅支援資源 GPU。

type

類型：字串

必要：是

要指派給容器的資源類型。支援的值為 GPU。

value

類型：字串

必要：是

所指定資源類型的值。

如果使用 GPU 類型，值是 Amazon ECS 容器代理程式將為容器預留的實體 GPUs 數目。為任務中所有容器預留的 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。

GPU 不適用於在 Fargate 上執行的任務。

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。

最大值為 600 秒 (10 分鐘)。

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)。

如果未指定參數，則會使用預設值 30 秒。最大值為 86400 秒 (24 小時)。

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。容器可以包含多個相依性。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。

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。此條件僅在任務啟動時確認。

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。此參數會映射至 docker create-container 命令中的 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：

- 如果設定任何容器的 systemControls，該設定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。

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。

```
"systemControls": [  
  {  
    "namespace": "string",  
    "value": "string"  
  }  
]
```

namespace

類型：字串

必要：否

要設定 value 的命名空間核心參數。

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：以 "net.*" 開頭的 Sysctls。在 Fargate 上，僅接受容器內既有的具命名空間的 Sysctls。

Fargate 支援所有這些值。

value

類型：字串

必要：否

namespace 中所指定命名空間核心參數的值。

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。此參數會映射至 docker create-container 命令中的 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。此參數會映射至 docker create-container 命令中的 Tty，以及 docker run 的 --tty 選項。

預設值為 false。

Linux 參數

linuxParameters

類型：[LinuxParameters](#) 物件

必要：否

套用到容器的 Linux 特定修改項目，例如 Linux 核心功能。

capabilities

類型：[KernelCapabilities](#) 物件

必要：否

新增至 Docker 或從其提供的預設組態中卸除的容器 Linux 功能。

devices

類型：[Device](#) 物件的陣列

必要：否

任何要公開給容器的主機裝置。此參數會映射至 `docker create-container` 命令中的 `Devices`，以及 `docker run` 的 `--device` 選項。

`initProcessEnabled`

類型：布林值

必要：否

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。此參數會對應 `docker run` 的 `--init` 選項。

`maxSwap`

 Note

在 Amazon ECS 受管執行個體上執行的任務不支援此參數。

類型：整數

必要：否

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。

`swappiness`

 Note

在 Amazon ECS 受管執行個體上執行的任務不支援此參數。

類型：整數

必要：否

這可讓您調整容器的記憶體交換行為。`swappiness` 值若為 0 將導致交換不會發生，除非絕對需要。`swappiness` 值若為 100 將導致非常積極地交換頁面。有效的值為介於 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。此參數會對應 `docker run` 的 `--memory-swappiness` 選項。

`sharedMemorySize`

 Note

在 Amazon ECS 受管執行個體上執行的任務不支援此參數。

類型：整數

必要：否

`/dev/shm` 磁碟區大小 (以 MiB 為單位)。此參數會對應 `docker run` 的 `--shm-size` 選項。

`tmpfs`

類型：[Tmpfs](#) 物件的陣列

必要：否

`tmpfs` 掛載的容器路徑、掛載選項和大小 (以 MiB 為單位)。此參數會對應 `docker run` 的 `--tmpfs` 選項。

Fargate 的 Amazon ECS 任務定義參數

任務定義分為不同的部分：任務系列、AWS Identity and Access Management (IAM) 任務角色、網路模式、容器定義、磁碟區和啟動類型。任務定義中需要系列和容器定義。與此相對，任務角色、網路模式、磁碟區與啟動類型則為選用項目。

您可以在 JSON 檔案中使用這些參數來設定任務定義。

以下是 Fargate 每個任務定義參數的詳細描述。

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。

Capacity

在註冊任務定義時，您可以指定 Amazon ECS 驗證任務定義所依據的容量。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。

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:FARGATE

驗證任務定義所依據的容量。這會啟動檢查，確保任務定義中使用的所有參數都能滿足 Fargate 的需求

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](#)。

任務執行 角色

`executionRoleArn`

類型：字串

必要：有條件

任務執行角色的 Amazon Resource Name (ARN)，授予 Amazon ECS 容器代理程式代表您進行 AWS API 呼叫的許可。

 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：是

任務中的容器所使用的 Docker 聯網模式。對於在 Fargate 上託管的 Amazon ECS 任務，需要 awsvpc 網路模式。

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。

有效值為

LINUX、WINDOWS_SERVER_2025_FULL、WINDOWS_SERVER_2025_CORE、WINDOWS_SERVER_2022_與 WINDOWS_SERVER_2019_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

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。

若有 Linux 任務，可將值設定為 ARM64。如需詳細資訊，請參閱[the section called “64 位元 ARM 工作負載的任務定義”](#)。

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 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。

任務定義允許使用以下參數：

cpu

類型：字串

必要：是

Note

任務層級 CPU 與記憶體參數為必填項，用於判斷任務執行所在的執行個體類型與大小。對於 Windows 任務，這些值不會在執行時期強制執行，因為 Windows 沒有能輕易對一組容器強制執行集合資源限制的原生機制。若要強制執行資源限制，建議使用 Windows 容器的容器層級資源。

針對任務呈現的 CPU 單位硬性限制。您可以在 JSON 檔案中將 CPU 值指定為以 CPU 為單位或虛擬 CPU (vCPU) 為單位的字串。例如，可指定 1024 個 CPU 單位作為 CPU 值，也可指定 1 vCPU 個 vCPU 作為 CPU 值。註冊任務定義時，vCPU 值會轉換為整數，指出 CPU 單位。

此欄位為必要項目，而且您必須使用下列其中一個值，從而決定 memory 參數受支援的值範圍。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
 Note 此選項需要 Linux 平台 1.4.0 或更新版本。		
16384 (16vCPU)	介於 32 GB 與 120 GB 之間，以 8 GB 為單位遞增	Linux
 Note 此選項需要 Linux 平台 1.4.0 或更新版本。		

memory

類型：字串

必要：是

Note

任務層級 CPU 與記憶體參數為必填項，用於判斷任務執行所在的執行個體類型與大小。對於 Windows 任務，這些值不會在執行時期強制執行，因為 Windows 沒有能輕易對一組容器強制執行集合資源限制的原生機制。若要強制執行資源限制，建議使用 Windows 容器的容器層級資源。

要分配給任務的記憶體硬性限制。您可以在任務定義中將記憶體值指定為以 MiB 或 GB 為單位的字串。例如，可指定 3072 MiB 作為記憶體值，也可指定 3 GB 作為記憶體值。註冊任務定義時，GB 值會轉換為整數，指出 MiB。

此欄位為必要項目，而且您必須使用下列其中一個值，從而決定 cpu 參數受支援的值範圍：

記憶體值 (以 MiB 為單位，且附註以 GB 為單位的約略換算值)	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 (4 GB)、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

記憶體值 (以 MiB 為單位，且附註以 GB 為單位的約略換算值)	CPU 數值	Fargate 支援的作業系統
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  Note 此選項需要 Linux 平台 1.4.0 或更新版本。	16384 (16vCPU)	Linux

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](#)
- [影像](#)
- [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。最多允許 255 個字母（大寫和小寫）、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。

影像

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 create-container 命令中的 Image，以及 docker run 命令的 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。
- 當您未在任務定義中的映像路徑中指定標籤或摘要時，Amazon ECS 容器代理程式會使用 latest 標籤來提取指定的映像。
- 儲存庫映像的後續更新不會傳播至已執行的任務。

- 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)。

versionConsistency

類型：字串

有效值：enabled|disabled

必要：否

指定 Amazon ECS 是否會將容器定義中提供的容器映像標籤解析為映像摘要。此行為預設為 enabled。如果將容器的值設定為 disabled，Amazon ECS 不會將容器映像標籤解析為摘要，但會使用容器定義中指定的原始映像 URI 進行部署。如需有關容器映像解析的詳細資訊，請參閱[容器映像解析](#)。

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)。此參數會映射至 docker create-container 命令中的 Memory，以及 docker run 的 --memory 選項。

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。此參數會映射至 `docker create-container` 命令中的 `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

連接埠映射會將容器的網路連接埠對外公開，讓用戶端可以存取您的應用程式。這也適用於同一任務中的容器間通訊。

針對使用 awsvpc 網路模式的任務定義，只指定 containerPort。一律略過 hostPort，且容器連接埠會自動映射至主機上的隨機高連接埠號碼。

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 (包括 containerPort、hostPort、protocol) 會映射至 docker create-container 命令中的 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 CloudWatch 中新增通訊協定專屬遙測。

如果您沒有為此參數設定值，系統會使用 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。

對於使用 awsvpc 網路模式的任務，您可以透過 containerPort 指定公開的連接埠。

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。

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 `awsvpc` 網路模式。
- 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。
- `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 GitHub 上的 [Issue #11185](#)。

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。

連接埠範圍未包含在傳送到 EventBridge 的 Amazon ECS 任務事件中。如需詳細資訊，請參閱 [the section called “使用 EventBridge 自動化因應 Amazon ECS 錯誤”](#)。

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。

hostPort 可維持空白，或與 containerPort 的值相同。

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 與 VPC Lattice

用於連接埠映射的名稱。此參數僅適用於 Service Connect 與 VPC Lattice。此參數是您在服務的 Service Connect 與 VPC Lattice 組態中使用的名稱。

如需詳細資訊，請參閱[使用 Service Connect 以利用簡稱連線 Amazon ECS 服務](#)。

下列範例中同時使用了 Service Connect 與 VPC Lattice 的必要欄位。

```
"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。

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憑證。

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 AWS CLI 或 AWS SDKs 時，如果秘密與您啟動的任務位於相同的區域中，您可以使用完整的 ARN 或秘密的名稱。使用時 AWS 管理主控台，您必須指定秘密的完整 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

下列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

- [重新啟動政策](#)
- [運作狀態檢查](#)
- [Environment](#)
- [網路設定](#)
- [儲存與記錄](#)
- [安全](#)
- [資源限制](#)

- [Docker 標籤](#)

重新啟動政策

restartPolicy

容器重新啟動政策及相關組態參數。在為容器設定重新啟動政策後，Amazon ECS 可以重新啟動容器，無需取代任務。如需詳細資訊，請參閱[使用容器重新啟動政策在 Amazon ECS 任務中重新啟動個別容器](#)。

enabled

類型：布林值

必要：是

指定是否已為容器啟用重新啟動政策。

ignoredExitCodes

類型：Integer array

必要：否

Amazon ECS 會略過且不嘗試重新啟動的結束代碼清單。最多可以指定 50 個容器結束代碼。Amazon ECS 預設不會略過任何結束代碼。

restartAttemptPeriod

類型：整數

必要：否

容器必須執行一段時間 (以秒為單位) 後，才能嘗試重新啟動。容器每 restartAttemptPeriod 秒只能重新啟動一次。如果容器未能執行達此時長即提前結束，則不會重新啟動。restartAttemptPeriod 最少可設定 60 秒，restartAttemptPeriod 最多可設定 1800 秒。容器預設必須執行 300 秒後才能重新啟動。

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來判斷 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 中註冊任務定義時 AWS 管理主控台，請使用逗號分隔的命令清單。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 JSON AWS 管理主控台 面板 AWS CLI、或 APIs 註冊任務定義時，請將命令清單括在括號中。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。

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。您可以指定介於 0 到 300 秒之間的值。預設情況下，startPeriod 是停用的。

如果運作狀態檢查在 startPeriod 內成功，則代表容器運作狀態良好，之後的任何故障都會計入，累積至重試次數上限。

Environment

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，此參數會映射至 [Create a container](#) 區段中的 CpuShares。

對於使用 Fargate 的任務，此欄位為選用項目。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。

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。Linux 核心允許的有效 CPU 共用值，最小為 2，最大為 262144。但是，CPU 參數並非必要項目，且您可以在容器定義中使用小於 2 且大於 262144 的 CPU 值。針對小於 2 且大於 262144 的 CPU 值 (包含 null)，行為會隨您的 Amazon ECS 容器代理程式版本而異：

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

在 Fargate 上託管的容器不支援此參數。

類型：[ResourceRequirement](#) 物件

必要：否

Amazon ECS 容器代理程式保留給容器的實體 GPU 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](#)。

Elastic Inference accelerator

在 Fargate 上託管的容器不支援此參數。

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 \(已棄用\)”](#)。

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。

```
"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 create-container 命令中的 Cmd，以及 docker run 的 COMMAND 參數。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

此參數不適用於 Fargate 上的 Windows 容器。

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。此參數會映射至 docker run 命令的 --env-file 選項。

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。此參數會映射至 docker create-container 命令中的 Env 參數，以及 docker run 命令的 --env 選項。

 Important

不建議使用純文字環境變數儲存敏感資訊 (例如憑證)。

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。支援的值為 AWS Secrets Manager 秘密的完整 Amazon Resource Name (ARN) 或 AWS Systems Manager 參數存放區中參數的完整 ARN。

Note

如果 Systems Manager 參數存放區參數或 Secrets Manager 參數與您啟動 AWS 區域的任務位於相同的 中，您可以使用秘密的完整 ARN 或名稱。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"  
  }  
]
```

網路設定

disableNetworking

在 Fargate 上執行的任務不支援此參數。

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。

預設值為 false。

```
"disableNetworking": true|false
```

links

使用 awsvpc 網路模式的任務不支援此參數。

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。最多可輸入 255 個字母 (大小寫)、數字、連字號與底線。

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。此參數會映射至 `docker create-container` 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

在 Fargate 上執行的任務不支援此參數。

類型：字串陣列

必要：否

提供給容器的 DNS 伺服器清單。

```
"dnsServers": ["string", ...]
```

extraHosts

使用 `awsvpc` 網路模式的任務不支援此參數。

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。

此參數會映射至 `docker create-container` 命令中的 `ExtraHosts`，以及 `docker run` 的 `--add-host` 選項。

```
"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一讀存取權。此參數會映射至 `docker create-container` 命令中的 `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載點。此參數會映射至 Docker API 中 create-container 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，且掛載點不能跨磁碟機使用。您必須指定掛載點，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)

掛載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。

對於在執行 Windows 作業系統之 EC2 執行個體上執行的任務，將該值保留為預設值 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。此參數會映射至 `docker create-container` 命令中的 `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一讀存取權。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](#)。

此參數會映射至 `docker create-container` 命令中的 `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)。

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。
- 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。
- 您必須在任務外部安裝任何額外軟體。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","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","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。

支援的日誌驅動程式為 awslogs、splunk 與 awsfirelens。

如需有關如何在任務定義中使用 awslogs 日誌驅動程式將您的容器日誌傳送到 CloudWatch Logs 的詳細資訊，請參閱[將 Amazon ECS 日誌傳送至 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，您可以分支 [GitHub 上可取得的](#) Amazon ECS 容器代理程式專案，並自訂以搭配此驅動程式一起使用。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。

此參數需要容器執行個體上的 Docker Remote API 1.18 版或更新版本。

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。

可供指定的選項視日誌驅動程式而定。使用 `awslogs` 路由器將日誌路由至 Amazon CloudWatch 時，可指定的部分選項包括：

`awslogs-create-group`

必要：否

指定您是否希望自動建立日誌群組。若未指定此選項，則預設為 `false`。

Note

在您嘗試使用 `awslogs-create-group` 之前，您的 IAM 政策必須包含 `logs:CreateLogGroup` 許可。

`awslogs-region`

必要：是

指定 AWS 區域 `awslogs` 日誌驅動程式要傳送 Docker 日誌的。您可以選擇將所有的日誌從不同區域中的叢集傳送至 CloudWatch Logs 中的單一區域。如此一來，日誌全都顯示在單一位置中。或者，您可以依照區域分隔日誌，以獲得更高的精細程度。請確定指定的日誌群組存在於您使用此選項指定的區域。

`awslogs-group`

必要：是

請務必指定 `awslogs` 日誌驅動程式傳送其日誌串流的日誌群組。

`awslogs-stream-prefix`

必要：是

使用 `awslogs-stream-prefix` 選項將日誌串流與指定字首、容器名稱以及容器所屬 Amazon ECS 任務的 ID 建立關聯。如果您使用此選項指定前綴，則日誌串流會使用下列格式。

```
prefix-name/container-name/ecs-task-id
```

如果您未使用此選項指定字首，則該日誌串流會以容器執行個體上 Docker 常駐程式指派的容器 ID 命名。因為只使用 Docker 容器 ID (僅容器執行個體提供) 很難對日誌回溯追蹤到傳送該日誌的容器，所以建議您使用此選項指定前綴。

對於 Amazon ECS 服務，您可以使用服務名稱作為前綴。藉此這可讓您對日誌串流回溯追蹤到容器所屬的服務、傳送該日誌之容器的名稱，以及容器所屬任務的 ID。

您必須指定日誌的串流字首，才可在使用 Amazon ECS 主控台時，讓日誌顯示在 Log (日誌) 窗格中。

`awslogs-datetime-format`

必要：否

此選項會以 Python `strftime` 格式定義多行開始模式。日誌訊息由符合模式的一行以及不符合模式的任何後續行所組成。符合的行是日誌訊息之間的分隔符號。

使用此格式的一個使用案例範例是用於剖析輸出，例如堆疊傾印，在其他情形下這可能會記錄在多個項目中。正確的模式可允許將它擷取在單一項目中。

如需詳細資訊，請參閱 [awslogs-datetime-format](#)。

無法同時設定 `awslogs-datetime-format` 和 `awslogs-multiline-pattern` 選項。

Note

多行記錄會執行常規運算式剖析並比對所有日誌訊息。這可能會對記錄效能造成負面影響。

`awslogs-multiline-pattern`

必要：否

此選項定義使用常規運算式的多行開始模式。日誌訊息由符合模式的一行以及不符合模式的任何後續行所組成。符合的行是日誌訊息之間的分隔符號。

如需詳細資訊，請參閱 [awslogs-multiline-pattern](#)。

如果同時設定 `awslogs-datetime-format`，會忽略此選項。

無法同時設定 `awslogs-datetime-format` 和 `awslogs-multiline-pattern` 選項。

 Note

多行記錄會執行常規運算式剖析並比對所有日誌訊息。這可能會對記錄效能造成負面影響。

下列選項適用於所有受支援的日誌驅動程式。

`mode`

必要：否

有效值：`non-blocking` | `blocking`

此選項定義從容器到日誌驅動程式 (透過 `logDriver` 指定) 的日誌訊息傳送模式。當從容器的日誌流程中斷時，您選擇的傳送模式會影響應用程式可用性。

如果使用 `blocking` 模式，且日誌流程中斷，則從容器程式碼寫入 `stdout` 與 `stderr` 串流的呼叫將會封鎖。應用程式的日誌記錄執行緒將因此封鎖。這可能會導致應用程式沒有回應，並導致容器運作狀態檢查失敗。

如果您使用 `non-blocking` 模式，則容器的日誌將儲存在使用 `max-buffer-size` 選項設定的記憶體內中間緩衝區中。這可以防止當無法傳送日誌時應用程式發生無回應的狀況。如果您想要確保服務的可用性且可以接受一些日誌遺失，我們建議您使用此模式。如需詳細資訊，請參閱 [Preventing log loss with non-blocking mode in the awslogs container log driver](#)。

您可透過 `defaultLogDriverMode` 帳戶設定，為特定 AWS 區域中的所有容器設定預設 `mode`。若未在 `logConfiguration` 中指定 `mode` 選項，或未對帳戶進行設定，則 Amazon ECS 將預設採用 `non-blocking` 模式。如需有關帳戶設定的詳細資訊，請參閱 [預設日誌驅動程式模式](#)。

使用 `non-blocking` 模式時，`max-buffer-size` 日誌選項會控制用於中繼訊息儲存的緩衝區大小。請務必根據您的應用程式指定適當的緩衝區大小。在任務層級配置的記憶體總量，應大於為所有容器配置的記憶體容量與日誌驅動程式記憶體緩衝區用量的總和。

Note

自 2025 年 6 月 25 日起，Amazon ECS 將預設日誌驅動程式模式從 blocking 變更為 non-blocking，優先保障任務可用性而非日誌完整性。若要在此變更後繼續使用 blocking 模式，請執行下列任一操作：

- 將容器定義 logConfiguration 中的 mode 選項設定為 blocking。
- 將 defaultLogDriverMode 帳戶設定設定為 blocking。

max-buffer-size

必要：否

預設值：10m

使用 non-blocking 模式時，max-buffer-size 日誌選項會控制用於中繼訊息儲存的緩衝區大小。請務必根據您的應用程式指定適當的緩衝區大小。當緩衝區填滿時，無法儲存進一步的日誌。無法儲存的日誌將會遺失。

要使用 splunk 日誌路由器路由日誌，需指定 splunk-token 與 splunk-url。

當您使用 awsfirelens 日誌路由器將日誌路由到 AWS 服務 或 AWS Partner Network 目的地以進行日誌儲存和分析時，您可以設定 log-driver-buffer-limit 選項，以限制在記憶體中緩衝的日誌行數，然後再傳送至日誌路由器容器。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 “為高輸送量設定日誌”](#)。

使用 awsfirelens 路由日誌時，可指定的其他選項取決於目的地。當您將日誌匯出至 Amazon Data Firehose 時，您可以使用 指定 AWS 區域，region 並使用 指定日誌串流的名稱 delivery_stream。

在將日誌匯出至 Amazon Kinesis Data Streams 時，可透過 region 指定 AWS 區域，透過 stream 指定日誌串流名稱。

在將日誌匯出至 Amazon OpenSearch Service 時，可指定多個選項，例如 Name、Host (不含通訊協定的 OpenSearch Service 端點) Port、Index、Type、Aws_auth、Aws_region、Suppress_Type_Name 與 tls。

在將日誌匯出至 Amazon S3 時，可透過 bucket 選項指定儲存貯體。您也可以指定 region、total_file_size、upload_timeout 與 use_put_object 作為選項。

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

容器的 FireLens 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 日誌傳送至 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:::*amzn-s3-demo-bucket*/fluent.conf|filepath"}。如需詳細資訊，請參閱[Amazon ECS 任務定義範例：將日誌路由至 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 任務與容器安全最佳實務](#)。

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。

`credentialsspecdomainless` : MyARN

您使用 `credentialsspecdomainless: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。

`credentialsspec` : MyARN

您使用 `credentialsspec: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](#)。

`user`

類型：字串

必要：否

要在容器內使用的使用者。此參數會映射至 `docker create-container` 命令中的 `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"
```

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。此參數會映射至 `docker create-container` 命令中的 `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 65535，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](#)。

此參數需要容器執行個體上的 Docker Remote API 1.18 版或更新版本。

 Note

Windows 容器不支援此參數。

```
"ulimits": [  
  {  
    "name":  
"core"|"cpu"|"data"|"fsize"|"locks"|"memlock"|"msgqueue"|"nice"|"nofile"|"nproc"|"rss"|"rtpr  
    "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。可依據 ulimit 的 type，以位元組、秒數或計數單位指定該數值。

softLimit

類型：整數

必要：是 (當使用 ulimits 時)

ulimit 類型的軟性限制。可依據 ulimit 的 type，以位元組、秒數或計數單位指定該數值。

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。此參數會映射至 `docker create-container` 命令中的 `Labels`，以及 `docker run` 的 `--label` 選項。

此參數需要容器執行個體上的 Docker Remote API 1.18 版或更新版本。

```
"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

套用到容器的 Linux 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 提供的預設組態中移除容器的 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："SYS_PTRACE"

必要：否

要新增至 Docker 提供之預設組態容器的 Linux 功能。此參數會映射至 docker create-container 命令中的 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。此參數會映射至 `docker create-container` 命令中的 `CapDrop`，以及 `docker run` 的 `--cap-drop` 選項。

devices

任何要公開給容器的主機裝置。此參數會映射至 `docker create-container` 命令中的 `Devices`，以及 `docker run` 的 `--device` 選項。

Note

使用 Fargate 啟動類型時，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。此參數會對應 `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

在 Fargate 上執行的任務不支援此參數。

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。

sharedMemorySize

/dev/shm 磁碟區的大小值 (以 MiB 為單位)。此參數會對應 docker run 的 `--shm-size` 選項。

Note

如果使用在 Fargate 上執行的任務，則不支援 sharedMemorySize 參數。

類型：整數

tmpfs

tmpfs 掛載的容器路徑、掛載選項和大小上限 (以 MiB 為單位)。此參數會對應 docker run 的 `--tmpfs` 選項。

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。容器可以包含多個相依性。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。

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。此條件僅在任務啟動時確認。

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。

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)。

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。

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。此參數會映射至 `docker create-container` 命令中的 `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：

- 如果設定任何容器的 `systemControls`，該設定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。

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 value 的命名空間核心參數。

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：以 "net.*" 開頭的 Sysctls。在 Fargate 上，僅接受容器內既有的具命名空間的 Sysctls。

Fargate 支援所有這些值。

value

類型：字串

必要：否

namespace 中所指定命名空間核心參數的值。

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。此參數會映射至 docker create-container 命令中的 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。此參數會映射至 docker create-container 命令中的 Tty，以及 docker run 的 --tty 選項。

預設值為 false。

Elastic Inference 加速器名稱（已棄用）

任務定義的 Elastic Inference 加速器資源要求。

Note

Amazon Elastic Inference (EI) 服務已全面終止，不再向客戶提供。

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。

代理組態

proxyConfiguration

類型：[ProxyConfiguration](#) 物件

必要：否

App Mesh 代理的組態詳細資訊。

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：[KeyValuePair](#) 物件陣列

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

註冊任務定義時，您可以選擇指定磁碟區清單，此類清單會傳送至容器執行個體上的 Docker 常駐程式，且可供同一容器執行個體中的其他容器存取。

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具成本效益、持久且高效能的區塊儲存服務。執行獨立任務或建立/更新服務時，每個 Amazon ECS 任務最多可掛載 1 個 Amazon EBS 磁碟區。Fargate 上託管的 Linux 任務支援 Amazon EBS 磁碟區。如需詳細資訊，請參閱[將 Amazon EBS 磁碟區與 Amazon ECS 搭配使用](#)。
- 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。您的應用程式可以擁有所需的儲存體，以及何時需要它。Fargate 上託管的任務支援 Amazon EFS 磁碟區。如需詳細資訊，請參閱[將 Amazon EFS 磁碟區與 Amazon ECS 搭配使用](#)。
- 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[搭配 Amazon ECS 使用 FSx for Windows File Server 磁碟區](#)。

Fargate 上的 Windows 容器不支援此選項。

- 綁定掛載 – 主機上掛載至容器的檔案或目錄。執行任務時，支援綁定掛載主機磁碟區。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[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。最多可輸入 255 個字母 (大小寫)、數字、連字號 (-) 與底線 (_)。在容器定義 `mountPoints` 物件的 `sourceVolume` 參數中參考此名稱。

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

只有在使用託管在 Amazon EC2 執行個體或 Amazon ECS 受管執行個體上的任務時，才支援 `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

指定磁碟區可否在啟動時設定。若設定為 `true`，您可以在執行獨立任務或建立/更新服務時設定磁碟區。若設定為 `true`，您無法在任務定義中提供另一個磁碟區組態。此參數必須設定為 `true`，以設定要連接至任務的 Amazon EBS 磁碟區。將 `configuredAtLaunch` 設定為 `true` 並且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任務定義。此做法使任務定義可在不同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：[DockerVolumeConfiguration](#) 物件

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 容器只支援使用 `local` 驅動程式。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)。此欄位只有在 `scope` 的值為 `shared` 才會使用。如果 `scope` 的值為 `task`，則此參數必須省略。

`driver`

類型：字串

必要：否

要使用的 Docker 磁碟區驅動程式。驅動程式值必須符合 Docker 提供的驅動程式名稱，因為此名稱會用於任務置放。如果使用 Docker 外掛程式 CLI 安裝驅動程式，則使用 `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，則使用 Docker 外掛程式探索來擷取驅動程式名稱。

`driverOpts`

類型：字串

必要：否

一組需傳遞的、專用於 Docker 驅動程式的選項映射。此參數會映射至 Docker 中「Create a volume」區段的 `DriverOpts`。

`labels`

類型：字串

必要：否

自訂中繼資料，新增到您的 Docker 磁碟區。

`efsVolumeConfiguration`

類型：[EFSVolumeConfiguration](#) 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 `authorizationConfig` 中指定了 EFS 存取點，則必須省略根目錄參數或將其設定為 `/`，後者將強制執行 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。如果未指定傳輸加密連接埠，任務會使用 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：[EFSAuthorizationConfig](#) 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，或將其設定為 `/`，後者將強制執行在 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](#)。

`FSxWindowsFileServerVolumeConfiguration`

類型：[FSxWindowsFileServerVolumeConfiguration](#) 物件

必要：是

此參數在您為任務儲存使用 [Amazon FSx for Windows File Server](#) 檔案系統時指定。

`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：

- [AWS Secrets Manager](#) 秘密的 Amazon Resource Name (ARN)。
- [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 EC2 Active Directory 託管的完整網域名稱。

Tags (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 服務都可以存取標籤，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](#)
- [故障注入](#)

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

僅平台版本 1.4.0 或更新版本 (Linux) 或 1.0.0 或更新版本 (Windows) 支援此參數。

IPC 模式

ipcMode

在 Fargate 上執行的任務不支援此參數。

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。

- 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 容器上，唯一的有效值為 `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 `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。

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。

故障注入

enableFaultInjection

類型：布林值

有效值: true | false

必要：否

如果此參數設定為 true，在任務的承載中，Amazon ECS 與 Fargate 皆接受來自任務容器的故障注入請求。根據預設，此參數會設定為 false。

Amazon EC2 的 Amazon ECS 任務定義參數

任務定義分為不同的部分：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、任務置放限制和容量。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 Amazon EC2 每個任務定義參數的詳細描述

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。

Capacity

在註冊任務定義時，您可以指定 Amazon ECS 驗證任務定義所依據的容量。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。

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

驗證任務定義所依據的容量。這會啟動檢查，確保任務定義中使用的所有參數都能滿足 Amazon EC2 的需求。

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 Windows 執行個體的額外組態](#)。

任務執行 角色

executionRoleArn

類型：字串

必要：有條件

任務執行角色的 Amazon Resource Name (ARN)，授予 Amazon ECS 容器代理程式代表您進行 AWS API 呼叫的許可。

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 與 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 與 awsvpc。如果未指定網路模式，則使用 default 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。

如果網路模式是 awsvpc，則任務會配置一個彈性網路介面，而當您使用此任務定義建立服務或執行任務時，您必須指定 NetworkConfiguration。如需詳細資訊，請參閱[EC2 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。

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。

允許的網路模式取決於底層 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

在註冊任務定義時，您會指定作業系統系列。

有效值為

LINUX、WINDOWS_SERVER_2025_FULL、WINDOWS_SERVER_2025_CORE、WINDOWS_SERVER_2022_與 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

註冊任務定義時，您可以指定 CPU 架構。有效值為 X86_64 和 ARM64。如果您未指定值，Amazon ECS 會根據容量提供者組態嘗試將任務放置在可用的 CPU 架構上。為了確保任務放置在特定 CPU 架構上，請在任務定義cpuArchitecture中指定 的值。

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。

若有 Linux 任務，可將值設定為 ARM64。如需詳細資訊，請參閱[the section called “64 位元 ARM 工作負載的任務定義”](#)。

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。

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。

`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。您可以在 JSON 檔案中將 CPU 值指定為以 CPU 為單位或虛擬 CPU (vCPU) 為單位的字串。例如，可指定 1024 個 CPU 單位作為 CPU 值，也可指定 1 vCPU 個 vCPU 作為 CPU 值。註冊任務定義時，vCPU 值會轉換為整數，指出 CPU 單位。

此欄位為選用欄位。如果您的叢集沒有任何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。支援的值介於 0.125 個 vCPU 與 192 個 vCPU 之間。

`memory`

類型：字串

必要：有條件

Note

Windows 容器不支援此參數。

要分配給任務的記憶體硬性限制。您可以在任務定義中將記憶體值指定為以 MiB 或 GB 為單位的字串。例如，可指定 3072 MiB 作為記憶體值，也可指定 3 GB 作為記憶體值。註冊任務定義時，GB 值會轉換為整數，指出 MiB。

此欄位為選填，且可以使用任何值。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](#)。

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](#)
- [影像](#)
- [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。最多允許 255 個字母（大寫和小寫）、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。

影像

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 create-container 命令中的 Image，以及 docker run 命令的 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。
- 當您未在任務定義中的映像路徑中指定標籤或摘要時，Amazon ECS 容器代理程式會使用 latest 標籤來提取指定的映像。
- 儲存庫映像的後續更新不會傳播至已執行的任務。
- 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)。

versionConsistency

類型：字串

有效值：enabled|disabled

必要：否

指定 Amazon ECS 是否會將容器定義中提供的容器映像標籤解析為映像摘要。此行為預設為 enabled。如果將容器的值設定為 disabled，Amazon ECS 不會將容器映像標籤解析為摘要，但會使用容器定義中指定的原始映像 URI 進行部署。如需有關容器映像解析的詳細資訊，請參閱[容器映像解析](#)。

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)。此參數會映射至 docker create-container 命令中的 Memory，以及 docker run 的 --memory 選項。

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。此參數會映射至 `docker create-container` 命令中的 `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

連接埠映射會將容器的網路連接埠對外公開，讓用戶端可以存取您的應用程式。這也適用於同一任務中的容器間通訊。

針對使用 `awsvpc` 網路模式的任務定義，只指定 `containerPort`。一律略過 `hostPort`，且容器連接埠會自動映射至主機上的隨機高連接埠號碼。

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 (包括 `containerPort`、`hostPort`、`protocol`) 會映射至 `docker create-container` 命令中的 `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 `CloudWatch` 中新增通訊協定專屬遙測。

如果您沒有為此參數設定值，系統會使用 `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。

對於使用 awsvpc 網路模式的任務，您可以透過 containerPort 指定公開的連接埠。

假設您在 EC2 容量提供者的任務中使用容器，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 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 GitHub 上的 [Issue #11185](#)。

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。

連接埠範圍未包含在傳送到 EventBridge 的 Amazon ECS 任務事件中。如需詳細資訊，請參閱 [the section called “使用 EventBridge 自動化因應 Amazon ECS 錯誤”](#)。

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。

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。您可以在 ECS 容器代理程式組態中使用 `ECS_DYNAMIC_HOST_PORT_RANGE` 設定，為動態指派的主機連接埠指定自訂範圍。如果您的任務因為連接埠與容器執行個體上的其他程序衝突而無法啟動，例如也使用暫時性連接埠範圍連接埠的傳出連線，這可能會有所幫助。如需詳細資訊，請參閱 [Amazon ECS 容器代理程式組態](#)。

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 與 VPC Lattice

用於連接埠映射的名稱。此參數僅適用於 Service Connect 與 VPC Lattice。此參數是您在服務的 Service Connect 與 VPC Lattice 組態中使用的名稱。

如需詳細資訊，請參閱 [使用 Service Connect 以利用簡稱連線 Amazon ECS 服務](#)。

下列範例中同時使用了 Service Connect 與 VPC Lattice 的必要欄位。

```
"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憑證。

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、AWS CLI 或 AWS SDKs 時，如果秘密與您啟動的任務位於相同的區域中，您可以使用完整的 ARN 或秘密的名稱。使用時 AWS 管理主控台，您必須指定秘密的完整 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

下列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

- [重新啟動政策](#)
- [運作狀態檢查](#)
- [Environment](#)
- [網路設定](#)
- [儲存與記錄](#)
- [安全](#)
- [資源限制](#)
- [Docker 標籤](#)

重新啟動政策

restartPolicy

容器重新啟動政策及相關組態參數。在為容器設定重新啟動政策後，Amazon ECS 可以重新啟動容器，無需取代任務。如需詳細資訊，請參閱[使用容器重新啟動政策在 Amazon ECS 任務中重新啟動個別容器](#)。

enabled

類型：布林值

必要：是

指定是否已為容器啟用重新啟動政策。

ignoredExitCodes

類型：Integer array

必要：否

Amazon ECS 會略過且不嘗試重新啟動的結束代碼清單。最多可以指定 50 個容器結束代碼。Amazon ECS 預設不會略過任何結束代碼。

restartAttemptPeriod

類型：整數

必要：否

容器必須執行一段時間 (以秒為單位) 後，才能嘗試重新啟動。容器每 restartAttemptPeriod 秒只能重新啟動一次。如果容器未能執行達此時長即提前結束，則

不會重新啟動。`restartAttemptPeriod` 最少可設定 60 秒，`restartAttemptPeriod` 最多可設定 1800 秒。容器預設必須執行 300 秒後才能重新啟動。

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來判斷 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 中註冊任務定義時 AWS 管理主控台，請使用逗號分隔的命令清單。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 JSON AWS 管理主控台 面板 AWS CLI、或 APIs 註冊任務定義時，請將命令清單括在括號中。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。

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。

如果運作狀態檢查在 startPeriod 內成功，則代表容器運作狀態良好，之後的任何故障都會計入，累積至重試次數上限。

Environment

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，此參數會映射至 [Create a container](#) 區段中的 CpuShares。

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。Linux 核心允許的有效 CPU 共用值，最小為 2，最大為 262144。但是，CPU 參數並非必要項目，且您可以在容器定義中使用小於 2 且大於 262144 的 CPU 值。針對小於 2 且大於 262144 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 $\leq 1.1.0$ ：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 $\geq 1.2.0$: Null、零和值為一的 CPU 值會以兩個 CPU 共享傳遞給 Docker。
- 代理程式版本 $\geq 1.84.0$: 大於 256 個 vCPU 的 CPU 值，會以 256 傳遞至 Docker，相當於 262144 個 CPU 共用數。

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 \(已棄用\)”](#)。

Note

Windows 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。

```
"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 `create-container` 命令中的 `Cmd`，以及 `docker run` 的 `COMMAND` 參數。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。此參數會映射至 docker run 命令的 --env-file 選項。

啟用 FIPS 後，不支援具有句點 (.) 的儲存貯體名稱 (例如 amzn-s3-demo-bucket1.name.example)。儲存貯體名稱包含句點 (.) 會導致任務無法啟動，因為代理程式無法從 Amazon S3 提取環境變數檔案。

這不適用於 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。此參數會映射至 docker create-container 命令中的 Env 參數，以及 docker run 命令的 --env 選項。

Important

不建議使用使用純文字環境變數儲存敏感資訊 (例如憑證)。

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。支援的值為 AWS Secrets Manager 秘密的完整 Amazon Resource Name (ARN) 或 AWS Systems Manager 參數存放區中參數的完整 ARN。

 Note

如果 Systems Manager 參數存放區參數或 Secrets Manager 參數與您啟動 AWS 區域的任務位於相同的 中，您可以使用秘密的完整 ARN 或名稱。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"  
  }  
]
```

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。

 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。最多可輸入 255 個字母 (大小寫)、數字、連字號與底線。

 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。此參數會映射至 `docker create-container` 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。

 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。此參數會映射至 `docker create-container` 命令中的 `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。

此參數會映射至 `docker create-container` 命令中的 `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一讀存取權。此參數會映射至 docker create-container 命令中的 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載點。此參數會映射至 Docker API 中 create-container 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，且掛載點不能跨磁碟機使用。您必須指定掛載點，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)

掛載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一讀存取權。如果此值為 `false`，則容器可寫入磁碟區。預設值為 `false`。

對於執行 Windows 作業系統的任務，請將該值保留為預設值 `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。此參數會映射至 `docker create-container` 命令中的 `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一讀存取權。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](#)。

此參數會映射至 docker create-container 命令中的 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)。

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。
- 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。
- 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_AVAILABLE_LOGGING_DRIVERS 環境變數註冊該執行個體上可用的記錄驅動程式，置放在該執行個體上的容器才能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](#)。

```
"logConfiguration": {
  "logDriver": "awslogs","fluentd","gelf","json-
file","journald","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","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。

支援的記錄驅動程式為 `awslogs`、`fluentd`、`gelf`、`json-file`、`journald`、`syslog`、`splunk` 和 `awsfirelens`。

如需有關如何在任務定義中使用 `awslogs` 日誌驅動程式將您的容器日誌傳送到 CloudWatch Logs 的詳細資訊，請參閱[將 Amazon ECS 日誌傳送至 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，您可以分支 [GitHub 上可取得的](#) Amazon ECS 容器代理程式專案，並自訂以搭配此驅動程式一起使用。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。

此參數需要容器執行個體上的 Docker Remote API 1.18 版或更新版本。

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。

可供指定的選項視日誌驅動程式而定。使用 `awslogs` 路由器將日誌路由至 Amazon CloudWatch 時，可指定的部分選項包括：

`awslogs-create-group`

必要：否

指定您是否希望自動建立日誌群組。若未指定此選項，則預設為 `false`。

 Note

在您嘗試使用 `awslogs-create-group` 之前，您的 IAM 政策必須包含 `logs:CreateLogGroup` 許可。

awslogs-region

必要：是

指定 AWS 區域 awslogs 日誌驅動程式要傳送 Docker 日誌的。您可以選擇將所有的日誌從不同區域中的叢集傳送至 CloudWatch Logs 中的單一區域。如此一來，日誌全都顯示在單一位置中。或者，您可以依照區域分隔日誌，以獲得更高的精細程度。請確定指定的日誌群組存在於您使用此選項指定的區域。

awslogs-group

必要：是

請務必指定 awslogs 日誌驅動程式傳送其日誌串流的日誌群組。

awslogs-stream-prefix

必要：選用

使用 awslogs-stream-prefix 選項將日誌串流與指定字首、容器名稱以及容器所屬 Amazon ECS 任務的 ID 建立關聯。如果您使用此選項指定前綴，則日誌串流會使用下列格式。

```
prefix-name/container-name/ecs-task-id
```

如果您未使用此選項指定字首，則該日誌串流會以容器執行個體上 Docker 常駐程式指派的容器 ID 命名。因為只使用 Docker 容器 ID (僅容器執行個體提供) 很難對日誌回溯追蹤到傳送該日誌的容器，所以建議您使用此選項指定前綴。

對於 Amazon ECS 服務，您可以使用服務名稱作為前綴。藉此這可讓您對日誌串流回溯追蹤到容器所屬的服務、傳送該日誌之容器的名稱，以及容器所屬任務的 ID。

您必須指定日誌的串流字首，才可在使用 Amazon ECS 主控台時，讓日誌顯示在 Log (日誌) 窗格中。

awslogs-datetime-format

必要：否

此選項會以 Python strftime 格式定義多行開始模式。日誌訊息由符合模式的一行以及不符合模式的任何後續行所組成。符合的行是日誌訊息之間的分隔符號。

使用此格式的一個使用案例範例是用於剖析輸出，例如堆疊傾印，在其他情形下這可能會記錄在多個項目中。正確的模式可允許將它擷取在單一項目中。

如需詳細資訊，請參閱 [awslogs-datetime-format](#)。

無法同時設定 `awslogs-datetime-format` 和 `awslogs-multiline-pattern` 選項。

 Note

多行記錄會執行常規運算式剖析並比對所有日誌訊息。這可能會對記錄效能造成負面影響。

`awslogs-multiline-pattern`

必要：否

此選項定義使用常規運算式的多行開始模式。日誌訊息由符合模式的一行以及不符合模式的任何後續行所組成。符合的行是日誌訊息之間的分隔符號。

如需詳細資訊，請參閱 [awslogs-multiline-pattern](#)。

如果同時設定 `awslogs-datetime-format`，會忽略此選項。

無法同時設定 `awslogs-datetime-format` 和 `awslogs-multiline-pattern` 選項。

 Note

多行記錄會執行常規運算式剖析並比對所有日誌訊息。這可能會對記錄效能造成負面影響。

下列選項適用於所有受支援的日誌驅動程式。

`mode`

必要：否

有效值：`non-blocking` | `blocking`

此選項定義從容器到日誌驅動程式 (透過 `logDriver` 指定) 的日誌訊息傳送模式。當從容器的日誌流程中斷時，您選擇的傳送模式會影響應用程式可用性。

如果使用 `blocking` 模式，且日誌流程中斷，則從容器程式碼寫入 `stdout` 與 `stderr` 串流的呼叫將會封鎖。應用程式的日誌記錄執行緒將因此封鎖。這可能會導致應用程式沒有回應，並導致容器運作狀態檢查失敗。

如果您使用 non-blocking 模式，則容器的日誌將儲存在使用 max-buffer-size 選項設定的記憶體內中間緩衝區中。這可以防止當無法傳送日誌時應用程式發生無回應的狀況。如果您想要確保服務的可用性且可以接受一些日誌遺失，我們建議您使用此模式。如需詳細資訊，請參閱 [Preventing log loss with non-blocking mode in the awslogs container log driver](#)。

您可透過 defaultLogDriverMode 帳戶設定，為特定 AWS 區域中的所有容器設定預設 mode。若未在 logConfiguration 中指定 mode 選項，或未對帳戶進行設定，則 Amazon ECS 將預設採用 non-blocking 模式。如需有關帳戶設定的詳細資訊，請參閱[預設日誌驅動程式模式](#)。

使用 non-blocking 模式時，max-buffer-size 日誌選項會控制用於中繼訊息儲存的緩衝區大小。請務必根據您的應用程式指定適當的緩衝區大小。在任務層級配置的記憶體總量，應大於為所有容器配置的記憶體容量與日誌驅動程式記憶體緩衝區用量的總和。

 Note

自 2025 年 6 月 25 日起，Amazon ECS 將預設日誌驅動程式模式從 blocking 變更為 non-blocking，優先保障任務可用性而非日誌完整性。若要在此變更後繼續使用 blocking 模式，請執行下列任一操作：

- 將容器定義 logConfiguration 中的 mode 選項設定為 blocking。
- 將 defaultLogDriverMode 帳戶設定設為 blocking。

max-buffer-size

必要：否

預設值：10 m

使用 non-blocking 模式時，max-buffer-size 日誌選項會控制用於中繼訊息儲存的緩衝區大小。請務必根據您的應用程式指定適當的緩衝區大小。當緩衝區填滿時，無法儲存進一步的日誌。無法儲存的日誌將會遺失。

要使用 splunk 日誌路由器路由日誌，需指定 splunk-token 與 splunk-url。

當您使用awsfirelens日誌路由器將日誌路由到 AWS 服務 或 AWS Partner Network 目的地以進行日誌儲存和分析時，您可以設定 log-driver-buffer-limit 選項，以限制在記憶體中緩衝的日誌行數，然後再傳送至日誌路由器容器。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 “為高輸送量設定日誌”](#)。

使用 `awsfirelens` 路由日誌時，可指定的其他選項取決於目的地。當您將日誌匯出至 Amazon Data Firehose 時，您可以使用指定 AWS 區域，`region` 並使用指定日誌串流的名稱 `delivery_stream`。

在將日誌匯出至 Amazon Kinesis Data Streams 時，可透過 `region` 指定 AWS 區域，透過 `stream` 指定日誌串流名稱。

在將日誌匯出至 Amazon OpenSearch Service 時，可指定多個選項，例如 `Name`、`Host` (不含通訊協定的 OpenSearch Service 端點) `Port`、`Index`、`Type`、`Aws_auth`、`Aws_region`、`Suppress_Type_Name` 與 `tls`。

在將日誌匯出至 Amazon S3 時，可透過 `bucket` 選項指定儲存貯體。您也可以指定 `region`、`total_file_size`、`upload_timeout` 與 `use_put_object` 作為選項。

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

容器的 FireLens 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 日誌傳送至 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:::amzn-s3-demo-bucket/fluent.conf|filepath"}。`如需詳細資訊，請參閱[Amazon ECS 任務定義範例：將日誌路由至 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 任務與容器安全最佳實務](#)。

`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。

此參數會映射至 docker create-container 命令中的 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。此參數會映射至 docker create-container 命令中的 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」|「AppArmor：描述檔」|「標籤：*#*」|「憑證規格：*#####*」

必要：否

提供多個安全系統自訂組態的字串清單。

對於 Linux 任務，此參數可用於引用 SELinux 與 AppArmor 多層級安全系統的自訂標籤。

此參數可在設定 Active Directory 身分驗證的容器時用於引用憑證規格檔案。如需詳細資訊，請參閱[了解如何在 Amazon ECS 上為 EC2 Windows 容器使用 gMSA](#)及[在 Amazon ECS 上為 EC2 Linux 容器使用 gMSA](#)。

此參數會映射至 `docker create-container` 命令中的 `SecurityOpt`，以及 `docker run` 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。此參數會映射至 docker create-container 命令中的 Ulimits，以及 docker run 的 --ulimit 選項。

此參數需要容器執行個體上的 Docker Remote API 1.18 版或更新版本。

Note

Windows 容器不支援此參數。

```
"ulimits": [  
  {  
    "name":  
    "core"|"cpu"|"data"|"fsize"|"locks"|"memlock"|"msgqueue"|"nice"|"nofile"|"nproc"|"rss"|"rtpr  
    "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。可依據 `ulimit` 的 `type`，以位元組、秒數或計數單位指定該數值。

`softLimit`

類型：整數

必要：是 (當使用 `ulimits` 時)

`ulimit` 類型的軟性限制。可依據 `ulimit` 的 `type`，以位元組、秒數或計數單位指定該數值。

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。此參數會映射至 `docker create-container` 命令中的 `Labels`，以及 `docker run` 的 `--label` 選項。

此參數需要容器執行個體上的 Docker Remote API 1.18 版或更新版本。

```
"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

套用到容器的 Linux 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。此參數會映射至 docker create-container 命令中的 CapAdd，以及 docker run 的 --cap-add 選項。

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。此參數會映射至 docker create-container 命令中的 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。此參數會映射至 `docker create-container` 命令中的 `CapDrop`，以及 `docker run` 的 `--cap-drop` 選項。

devices

任何要公開給容器的主機裝置。此參數會映射至 `docker create-container` 命令中的 `Devices`，以及 `docker run` 的 `--device` 選項。

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。此參數會對應 `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。

`sharedMemorySize`

`/dev/shm` 磁碟區的大小值 (以 MiB 為單位)。此參數會對應 `docker run` 的 `--shm-size` 選項。

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。此參數會對應 `docker run` 的 `--memory-swappiness` 選項。

Note

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)。此參數會對應 `docker run` 的 `--tmpfs` 選項。

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。容器可以包含多個相依性。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。

執行個體至少需要 1.26.0 版的容器代理程式，才能啟用容器相依性。不過，我們建議您使用最新版的容器代理。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 最佳化 Linux AMI](#)。

```
"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。此條件僅在任務啟動時確認。

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。

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)。

如果未指定 stopTimeout 參數，則會使用為 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。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 最佳化 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。此參數會映射至 `docker create-container` 命令中的 `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 `awsvpc` 網路模式的任務，如果您設定任何容器的 `systemControls`，則它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。

```
"systemControls": [  
  {  
    "namespace": "string",  
    "value": "string"  
  }  
]
```

namespace

類型：字串

必要：否

要設定 `value` 的命名空間核心參數。

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

value

類型：字串

必要：否

namespace 中所指定命名空間核心參數的值。

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。此參數會映射至 docker create-container 命令中的 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。此參數會映射至 docker create-container 命令中的 Tty，以及 docker run 的 --tty 選項。

預設值為 false。

Elastic Inference 加速器名稱（已棄用）

任務定義的 Elastic Inference 加速器資源要求。

 Note

Amazon Elastic Inference (EI) 服務已全面終止，不再向客戶提供。

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。

您可以使用限制條件，根據可用區域、執行個體類型或自訂屬性來置放任務。如需詳細資訊，請參閱 [定義 Amazon ECS 用於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 [建立表達式以定義 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 的任務，容器執行個體至少需要 1.26.0 版的容器代理程式與 1.26.0-1 版的 `ecs-init` 套件，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 最佳化 Linux AMI](#)。

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：[KeyValuePair](#) 物件陣列

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

註冊任務定義時，您可以選擇指定磁碟區清單，此類清單會傳送至容器執行個體上的 Docker 常駐程式，且可供同一容器執行個體中的其他容器存取。

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具成本效益、持久且高效能的區塊儲存服務。執行獨立任務或建立/更新服務時，每個 Amazon ECS 任務最多可掛載 1 個 Amazon EBS 磁碟區。Linux 任務支援 Amazon EBS 磁碟區。如需詳細資訊，請參閱[將 Amazon EBS 磁碟區與 Amazon ECS 搭配使用](#)。
- 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。您的應用程式可以擁有所需的儲存體，以及何時需要它。支援 Amazon EFS 磁碟區。如需詳細資訊，請參閱[將 Amazon EFS 磁碟區與 Amazon ECS 搭配使用](#)。
- 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[搭配 Amazon ECS 使用 FSx for Windows File Server 磁碟區](#)。

Fargate 上的 Windows 容器不支援此選項。

- Docker 磁碟區 – 在主機 Amazon EC2 執行個體上的 `/var/lib/docker/volumes` 下建立的 Docker 受管磁碟區。Docker 磁碟區驅動程式（也稱為外掛程式）用於整合磁碟區與外部儲存系統，例如 Amazon EBS。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 容器只支援使用 local 驅動程式。若要使用 Docker 磁碟區，請在您的任務定義中指定 `dockerVolumeConfiguration`。
- 綁定掛載 – 主機上掛載至容器的檔案或目錄。支援綁定掛載主機磁碟區。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[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。最多可輸入 255 個字母 (大小寫)、數字、連字號 (-) 與底線 (_)。在容器定義 `mountPoints` 物件的 `sourceVolume` 參數中參考此名稱。

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

只有在使用託管在 Amazon EC2 執行個體或 Amazon ECS 受管執行個體上的任務時，才支援 `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

指定磁碟區可否在啟動時設定。若設定為 `true`，您可以在執行獨立任務或建立/更新服務時設定磁碟區。若設定為 `true`，您無法在任務定義中提供另一個磁碟區組態。此參數必須設定為 `true`，以設定要連接至任務的 Amazon EBS 磁碟區。將 `configuredAtLaunch` 設定為 `true` 並且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任務定義。此做法使任務定義可在不同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：[DockerVolumeConfiguration](#) 物件

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 容器只支援使用 `local` 驅動程式。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)。此欄位只有在 `scope` 的值為 `shared` 才會使用。如果 `scope` 的值為 `task`，則此參數必須省略。

`driver`

類型：字串

必要：否

要使用的 Docker 磁碟區驅動程式。驅動程式值必須符合 Docker 提供的驅動程式名稱，因為此名稱會用於任務置放。如果使用 Docker 外掛程式 CLI 安裝驅動程式，則使用 `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，則使用 Docker 外掛程式探索來擷取驅動程式名稱。

`driverOpts`

類型：字串

必要：否

一組需傳遞的、專用於 Docker 驅動程式的選項映射。此參數會映射至 Docker 中「Create a volume」區段的 `DriverOpts`。

`labels`

類型：字串

必要：否

自訂中繼資料，新增到您的 Docker 磁碟區。

`efsVolumeConfiguration`

類型：[EFSVolumeConfiguration](#) 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 `authorizationConfig` 中指定了 EFS 存取點，則必須省略根目錄參數或將其設定為 `/`，後者將強制執行 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。如果未指定傳輸加密連接埠，任務會使用 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：[EFSAuthorizationConfig](#) 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，或將其設定為 `/`，後者將強制執行在 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](#)。

`FSxWindowsFileServerVolumeConfiguration`

類型：[FSxWindowsFileServerVolumeConfiguration](#) 物件

必要：是

此參數在您為任務儲存使用 [Amazon FSx for Windows File Server](#) 檔案系統時指定。

`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：

- [AWS Secrets Manager](#) 秘密的 Amazon Resource Name (ARN)。
- [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 EC2 Active Directory 託管的完整網域名稱。

Tags (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 服務都可以存取標籤，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

- [IPC 模式](#)
- [PID 模式](#)
- [故障注入](#)

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。

- 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。

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`。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。

Note

Windows 容器不支援此參數。

故障注入

`enableFaultInjection`

類型：布林值

有效值: `true` | `false`

必要：否

如果此參數設定為 `true`，在任務的承載中，Amazon ECS 接受來自任務容器的故障注入請求。根據預設，此參數會設定為 `false`。

Amazon ECS 任務定義範本

空白的任務定義範本如下所示。您可以使用此範本來建立任務定義，然後將其貼入主控台 JSON 輸入區域，或儲存到檔案，並搭配 AWS CLI `--cli-input-json` 選項使用。如需詳細資訊，請參閱 [Fargate 的 Amazon ECS 任務定義參數](#)。

EC2 範本

```
{
  "family": "",
  "taskRoleArn": "",
  "executionRoleArn": "",
  "networkMode": "none",
  "containerDefinitions": [
    {
      "name": "",
      "image": "",
      "repositoryCredentials": {
        "credentialsParameter": ""
      },
      "cpu": 0,
      "memory": 0,
      "memoryReservation": 0,
      "links": [""],
      "portMappings": [
        {
          "containerPort": 0,
          "hostPort": 0,
          "protocol": "tcp"
        }
      ],
      "restartPolicy": {
        "enabled": true,
        "ignoredExitCodes": [0],
        "restartAttemptPeriod": 180
      },
      "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 範本

Important

對於 Fargate，您必須在 `operatingSystemFamily` 參數中指定以下任一值：

- 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

您可以從複製範例與程式碼片段開始，著手建立自己的任務定義。

您可以複製範例，然後在使用主控台時的透過 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[Fargate 的 Amazon ECS 任務定義參數](#)。

如需更多任務定義範例，請參閱 GitHub 上的 [AWS 範例任務定義](#)。

主題

- [Web 伺服器](#)
- [splunk 日誌驅動程式](#)
- [fluentd 日誌驅動程式](#)

- [gelf 日誌驅動程式](#)
- [外部執行個體上的工作負載](#)
- [Amazon ECR 映像與任務定義 IAM 角色](#)
- [進入點與命令](#)
- [容器相依性](#)
- [任務定義中的磁碟區](#)
- [Windows 任務定義範例](#)

Web 伺服器

以下是使用 Fargate 上 Linux 容器並設定 Web 伺服器的任務定義範例：

```
{
  "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": "public.ecr.aws/docker/library/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 上 Windows 容器並設定 Web 伺服器的任務定義範例：

```

{
  "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 日誌驅動程式

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 日誌驅動程式

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 日誌驅動程式

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作負載

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 映像與任務定義 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"
  }
}
```

進入點與命令

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 example.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"
}

```

任務定義中的磁碟區

請參閱下列主題，了解如何在任務中指定磁碟區。

- 如需有關如何設定 Amazon EBS 磁碟區的資訊，請參閱[在部署 Amazon ECS 時指定 Amazon EBS 磁碟區組態](#)。
- 如需有關如何設定 Amazon EFS 磁碟區的資訊，請參閱[使用主控台為 Amazon ECS 設定 Amazon EFS 檔案系統](#)。
- 如需有關如何設定 FSx for Windows File Server 磁碟區的資訊，請參閱[了解如何為 Amazon ECS 設定 FSx for Windows File Server 檔案系統](#)。

- 如需有關如何設定 Docker 磁碟區的資訊，請參閱 [Amazon ECS 的 Docker 磁碟區範例](#)。
- 如需有關如何設定綁定掛載的資訊，請參閱 [Amazon ECS 的綁定掛載範例](#)。

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 叢集是任務或服務邏輯分組，可為容器化應用程式提供基礎結構容量。建立叢集時，可從三種主要基礎結構類型中選擇，每種類型都針對不同的使用案例與操作需求進行最佳化。

選擇合適的叢集類型

Amazon ECS 為叢集提供三種基礎結構類型。選擇最符合工作負載需求、操作偏好設定與成本最佳化目標的類型：

Amazon ECS 受管執行個體 (建議)

最適合大多數工作負載 - AWS 完整管理基礎 Amazon EC2 執行個體，包括佈建、修補和擴展。此選項能在效能、成本效益與操作簡易性之間實現最佳平衡。

使用時機：

- AWS 您想要處理基礎設施管理
- 需要具成本效益、帶自動最佳化功能的運算
- 想要專注於應用程式而不是基礎結構
- 需要具有彈性擴展的可預測效能

Fargate

無伺服器運算 – 只需為任務實際使用的資源付費，無需管理任何基礎結構。適用於變動性工作負載與需要快速上手的案例。

使用時機：

- 想要完全無伺服器營運
- 有無法預測或可變的工作負載
- 想要將營運開銷降至最低
- 需要快速部署與擴展

Amazon EC2 執行個體

完全控制 – 您可以直接管理底層 Amazon EC2 執行個體，包括執行個體選擇、設定與維護。

使用時機：

- 需要特定的執行個體類型或組態

- 現有 Amazon EC2 基礎結構可以利用
- 需要自訂 AMI 或專用軟體
- 需要對底層基礎結構擁有最大控制

Note

Amazon ECS 受管執行個體是大多數新工作負載的建議選擇，因為它提供最佳的效能、成本最佳化和操作簡單性，同時允許 AWS 處理基礎設施管理任務。

叢集元件

除基礎結構容量外，叢集還包含下列資源：

- 執行任務和服務的網路 (VPC 和子網路)

當您將 Amazon ECS 受管執行個體或 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 Service Connect 進行服務間通訊。

- 監控選項

CloudWatch Container Insights 需要額外付費，且為全受管服務。此服務會自動收集、彙總及總結 Amazon ECS 指標和日誌。

叢集概念

以下是有關 Amazon ECS 叢集的一般概念。

- 可透過建立叢集來分隔資源。
- 叢集是 AWS 區域 特定的。
- 叢集可以處於如下任一狀態。

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。此行為未來可能會變更，因此請確保您不會依賴 INACTIVE 叢集的留存。

- 叢集可以包含託管於 Amazon ECS Managed Instances、AWS Fargate、Amazon EC2 執行個體或外部執行個體上的任務組合。任務可以當作啟動類型或容量提供者策略在 Amazon ECS 受管執行個體、Fargate 或 EC2 基礎結構上執行。如果您使用 EC2 容量提供者，Amazon ECS 不會追蹤和擴展 Amazon EC2 Auto Scaling 群組的容量。
- 叢集可以同時包含 Amazon ECS 受管執行個體容量提供者、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 Amazon ECS 受管執行個體容量提供者、Auto Scaling 群組容量提供者或 Fargate 容量提供者。
- 您可以針對 Amazon ECS 受管執行個體與 EC2 或 Auto Scaling 群組容量提供者，分別採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 政策來限制對叢集的存取。如需詳細資訊，請參閱 [Amazon ECS 叢集範例](#) 中的 [Amazon Elastic Container Service 身分型政策範例](#) 一節。
- 您可以使用服務自動擴展功能實現 Fargate 任務擴展。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 Service 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 Service Connect 以利用簡稱連線 Amazon ECS 服務](#)。

容量提供者

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和選用的容量提供者策略。您可以將預設容量提供者策略指派給叢集。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。僅當您未為任務或服務指定啟動類型或容量提供者策略時，叢集的預設容量提供者策略才適用。若提供任一參數，則不會使用預設策略。

Amazon ECS 為叢集提供三種類型的容量提供者：

Amazon ECS 受管執行個體容量提供者

AWS 完整管理基礎 Amazon EC2 執行個體，包括佈建、修補、擴展和生命週期管理。這能在效能、成本效益與操作簡易性之間實現最佳平衡。Amazon ECS 受管執行個體容量提供者會依據工作負載需求，自動最佳化執行個體選擇與擴展。

透過 Amazon ECS 受管執行個體可享如下優勢：

- 執行個體自動佈建與擴展
- 受管修補與安全性更新
- 透過智慧型執行個體選擇實現成本最佳化
- 降低營運開銷

Fargate 容量提供者

無伺服器運算：只需為任務實際使用的資源付費，無需管理任何基礎結構。您只需要將預先定義的容量提供者 (Fargate 及 Fargate Spot) 與叢集建立關聯。

Auto Scaling 群組容量提供者

當您為容量使用 Amazon EC2 執行個體時，可以使用 Auto Scaling 群組管理 Amazon EC2 執行個體。Auto Scaling 能確保您有正確的 Amazon EC2 執行個體數量來處理應用程式負載。您可以完全控制底層基礎結構。

叢集可以包含託管於 Amazon ECS Managed Instances、AWS Fargate、Amazon EC2 執行個體或外部執行個體上的任務組合。任務可以當作啟動類型或容量提供者策略在 Amazon ECS 受管執行個體、Fargate 或 EC2 基礎結構上執行。如果您使用 EC2 作為啟動類型，Amazon ECS 就不會追蹤和擴展 Amazon EC2 Auto Scaling 群組的容量。

叢集可以同時包含 Amazon ECS 受管執行個體容量提供者、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 Amazon ECS 受管執行個體容量提供者、Auto Scaling 群組容量提供者或 Fargate 容量提供者。

Amazon ECS 受管執行個體叢集

Amazon ECS 容量提供者會管理叢集中任務的基礎設施擴展。建立叢集之後，您需為該叢集建立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。

Amazon ECS 受管執行個體具有如下容量提供者。

Type	執行個體選擇準則
預設	符合下列任務定義與服務參數需求且最具成本效益的執行個體： • 任務定義 • <code>operatingSystemFamily</code> • <code>cpuArchitecture</code> • <code>cpu</code> • <code>memory</code> • 服務定義 • <code>placementConstraints</code> • <code>placementStrategy</code>
Custom	符合您建立叢集時指定的屬性與類型需求的執行個體。如需有關這些屬性的資訊，請參閱 Amazon ECS 容器執行個體屬性。如需有關執行個體類型的資訊，請參閱 Amazon EC2 Instance Types 中的 Amazon EC2 instance type specifications。

Amazon ECS 會啟動執行個體，並將這些執行個體與 Amazon ECS 受管執行個體容量提供者建立關聯。對於自訂容量提供者，Amazon ECS 也會建立容量提供者。

為 Amazon ECS 受管執行個體建立叢集

您可以透過建立叢集，定義任務與服務執行所在的基礎結構。

當您為 Amazon ECS 受管執行個體建立叢集時，Amazon ECS 會自動使用預設組態建立受管執行個體容量提供者。此容量提供者會為您的工作負載選取成本最佳化最高的執行個體類型。如果您需要特定的執行個體屬性或類型，也可以建立自訂容量提供者。

為了盡量簡化叢集建立程序，主控台有許多選擇的預設選項。

- 在 中建立與叢集 AWS Cloud Map 同名的預設命名空間。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](#)。

- 為您的受管儲存指派 AWS KMS 金鑰。如需有關如何建立金鑰的資訊，請參閱 AWS Key Management Service User Guide 中的 [Create a KMS key](#)。
- 新增標籤以協助您識別叢集。

先決條件

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 叢集範例”](#)。

建立叢集的使用者必須具有額外許可：iam:CreateServiceLinkedRole。

當您未指定 `instanceRequirements`，Amazon ECS 會根據您的任務定義需求，自動選取成本最佳化最高的執行個體類型。若要使用特定執行個體屬性或類型，請使用 建立容量提供者 `instanceRequirements`。

了解如何選擇執行個體。如需詳細資訊，請參閱[Amazon ECS 受管執行個體的執行個體選擇最佳實務](#)。

您有 Amazon ECS 受管執行個體所需的 IAM 角色。其中包含：

- 基礎設施角色 - 允許 Amazon ECS 代表您呼叫 AWS 服務，以管理 Amazon ECS 受管執行個體基礎設施。

如需詳細資訊，請參閱[Amazon ECS 基礎結構 IAM 角色](#)。

- 執行個體設定檔 – 為在受管執行個體上執行的 Amazon ECS 容器代理程式與 Docker 常駐程式提供許可。

如需詳細資訊，請參閱[Amazon ECS 受管執行個體執行個體設定檔](#)。

主控台程序

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. 若是自訂容量提供者，請執行以下步驟：
 - 對於選取 EC2 容量取得方式，選擇 Amazon ECS 受管執行個體。
 - 對於「執行個體設定檔」，選擇對應的執行個體設定檔角色。
 - 在「基礎結構角色」欄位中，選擇對應的基礎結構角色。
 - 若要使用自訂容量提供者，請在執行個體選擇欄位中選擇使用自訂。然後，為各屬性輸入對應的屬性值。
7. (選用) 使用 Container Insights，展開監控區段，然後選擇下列任一選項：
 - 若要使用建議的具有增強可觀測性的 Container Insights，請選擇具有增強可觀測性的 Container Insights。
 - 若要使用 Container Insights，請選擇 Container Insights。
8. (選用) 加密受管儲存上的資料。在加密下，針對受管儲存，輸入您要用來加密受管儲存資料的 AWS KMS 金鑰 ARN。

9. (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：

- 在金鑰欄位中，輸入金鑰名稱。
- 在值欄位中，輸入金鑰值。

10. 選擇建立。

AWS CLI 程序

您可以使用 AWS CLI 為 Amazon ECS 受管執行個體建立叢集。使用 AWS CLI 的最新版本。如需有關如何升級至最新版本的詳細資訊，請參閱 [Installing or updating to the latest version of the AWS CLI](#)。

Note

您可以使用雙堆疊服務端點，透過 IPv4 和 IPv6 AWS CLI 從、SDKs 和 Amazon ECS API 與 Amazon ECS 互動。如需詳細資訊，請參閱 [the section called “使用雙堆疊端點”](#)。

建立新的叢集 (AWS CLI)

1. 使用下列命令以唯一名稱建立叢集：

```
aws ecs create-cluster --cluster-name managed-instances-cluster
```

輸出：

```
{
  "cluster": {
    "status": "ACTIVE",
    "defaultCapacityProviderStrategy": [],
    "statistics": [],
    "capacityProviders": [],
    "tags": [],
    "clusterName": "managed-instances-cluster",
    "settings": [
      {
        "name": "containerInsights",
        "value": "disabled"
      }
    ]
  }
}
```



```
    ],
    "registeredContainerInstancesCount": 0,
    "pendingTasksCount": 0,
    "runningTasksCount": 0,
    "activeServicesCount": 0,
    "clusterArn": "arn:aws:ecs:region:aws_account_id:cluster/managed-instances-
cluster"
  }
}
```

2. (選用) 若要為叢集啟用具有增強可觀測性的 Container Insights，請使用下列命令：

```
aws ecs put-account-setting --name containerInsights --value enhanced
```

3. (選用) 若要將標籤新增至叢集，請使用下列命令：

```
aws ecs tag-resource --resource-arn
arn:aws:ecs:region:aws_account_id:cluster/managed-instances-cluster --tags
key=Environment,value=Production
```

後續步驟

為 Amazon ECS 受管執行個體建立任務定義。如需詳細資訊，請參閱[使用主控台建立 Amazon ECS 任務定義](#)。

將應用程式作為獨立任務執行，或作為服務的組成分執行。如需詳細資訊，請參閱下列內容：

- [將應用程式作為 Amazon ECS 任務執行](#)
- [建立 Amazon ECS 滾動更新部署](#)

更新叢集以使用 Amazon ECS 受管執行個體

您可以將現有叢集更新為使用 Amazon ECS 受管理執行個體。

當您將 Amazon ECS 受管執行個體新增至叢集時，Amazon ECS 會自動使用預設組態建立受管執行個體容量提供者。此容量提供者會為您的工作負載選取成本最佳化程度最高的一般用途執行個體類型。如果您需要特定的執行個體屬性或類型，也可以建立自訂容量提供者。

先決條件

當您未指定 `instanceRequirements`，Amazon ECS 會根據您的任務定義需求，自動選取成本最佳化最高的執行個體類型。若要使用特定執行個體屬性或類型，請使用 建立容量提供者 `instanceRequirements`。

您有 Amazon ECS 受管執行個體所需的 IAM 角色。其中包含：

- 基礎設施角色 - 允許 Amazon ECS 代表您呼叫 AWS 服務，以管理 Amazon ECS 受管執行個體基礎設施。

如需詳細資訊，請參閱[Amazon ECS 基礎結構 IAM 角色](#)。

- 執行個體設定檔 – 為在受管執行個體上執行的 Amazon ECS 容器代理程式與 Docker 常駐程式提供許可。

如需詳細資訊，請參閱[Amazon ECS 受管執行個體執行個體設定檔](#)。

更新考量

為 Amazon ECS 受管執行個體更新叢集時，建議考量下列事項：

- 執行中的任務 – 更新叢集設定不會影響目前執行中的任務。變更將套用至更新後啟動的新任務。
- 容量提供者變更 – 如果修改容量提供者設定，現有的受管執行個體將持續執行，但新的執行個體將使用更新的組態。
- 監控變更 – 啟用或停用 Container Insights 將影響整個叢集的指標集合。

主控台程序

更新叢集 (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. 在叢集頁面上選取要更新的叢集。
5. 選擇更新叢集。
6. (選用) 若要修改容量提供者設定，請在自訂容量提供者區段中視需要更新下列項目：

- 在執行個體設定檔欄位中，視需要選擇不同的執行個體設定檔角色。
- 在基礎結構角色欄位中，視需要選擇不同的基礎結構角色。
- 若要使用自訂容量提供者，請在執行個體選擇欄位中更新屬性值設定。

7. 選擇更新。

AWS CLI 程序

您可以使用 AWS CLI 為 Amazon ECS 受管執行個體更新叢集。使用 AWS CLI 的最新版本。如需有關如何升級至最新版本的詳細資訊，請參閱 [Installing or updating to the latest version of the AWS CLI](#)。

Note

您可以使用雙堆疊服務端點，透過 IPv4 和 IPv6 AWS CLI 從、SDKs 和 Amazon ECS API 與 Amazon ECS 互動。如需詳細資訊，請參閱 [the section called “使用雙堆疊端點”](#)。

更新叢集 (AWS CLI)

1. 為叢集建立容量提供者。執行以下命令：

將 *user-input* 取代為實際值。

```
aws ecs create-capacity-provider \  
  --name my-managed-instances-provider \  
  --managed-instances-provider \  
  --instance-profile arn:aws:iam::123456789012:instance-profile/  
ecsInstanceProfile \  
  --infrastructure-role-arn arn:aws:iam::123456789012:role/ecsInfrastructureRole  
 \  
  --instance-requirements '{  
    "vCpuCount": {"min": 2, "max": 8},  
    "memoryMiB": {"min": 4096, "max": 16384}  
  }'
```

2. 使用下列命令將容量提供者新增至叢集：

將 *user-input* 取代為實際值。

```
aws ecs put-cluster-capacity-providers --cluster managed-instances-cluster --  
capacity-providers my-managed-instances-provider --default-capacity-provider-  
strategy capacityProvider=my-managed-instances-provider,weight=1
```

Amazon ECS 受管執行個體容量提供者

Amazon ECS 受管執行個體容量提供者提供容器運算模型，可讓您存取完整範圍 AWS 的功能和 Amazon EC2 產品，同時 AWS 管理操作和安全性責任。會 AWS 處理軟體和作業系統修補、執行個體擴展和維護，讓您享有 Fargate 的操作優勢，同時維持對所有 AWS 功能和整合的存取。

Amazon ECS 會為 Amazon ECS 受管執行個體建立啟動範本。這定義了 Amazon ECS 啟動 Amazon ECS 受管執行個體的方式，包括任務的執行個體設定檔、網路與儲存組態、容量選項，以及彈性執行個體類型選擇的執行個體需求。

自訂容量提供者適用時機

若工作負載需要下列項目時，建議採用自訂容量提供者：

- 特定運算需求：需要加速運算、特定 CPU 指令集、高網路效能或大型記憶體組態的應用程式，這些組態不適用於標準 Fargate 選項。
- GPU 工作負載：機器學習推論、即時影像轉譯、影片編碼，或其他需要存取 NVIDIA 或 AMD GPUs 的 GPU 加速應用程式。
- 容量保留：需要可預測容量可用性的任務關鍵工作負載。
- 進階可觀測性：需要對基礎作業系統進行特殊權限存取的安全性和監控工具，例如 eBPF 型監控解決方案或網路分析工具。
- 成本最佳化：可以受益於多任務置放、共用基礎設施元件，或需要最大化較大執行個體類型的使用率的工作負載。

監控選項

Amazon ECS 受管執行個體提供全方位的監控功能，可協助追蹤容器化工作負載的效能、運作狀態與資源使用率。您可以根據營運需求，從不同的監控層級中進行選擇。

- 基本監控 – 針對大多數指標以 5 分鐘為間隔提供基本指標，針對狀態檢查以 1 分鐘為間隔提供基本指標。此功能為預設啟用，不會產生額外的費用。

- 詳細監控 – 以 1 分鐘為間隔提供所有指標，藉此提供增強的可見性，從而能夠更快地偵測並回應營運問題。如需詳細資訊，請參閱[Amazon ECS 受管執行個體的詳細監控](#)。

這兩個監控選項都與 CloudWatch 無縫整合，可提供儀表板、警示與自動化回應，有助於維持最佳的應用程式效能與可用性。

容量提供者考量事項

叢集可以同時包含 Amazon ECS 受管執行個體容量提供者、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 Amazon ECS 受管執行個體容量提供者、Auto Scaling 群組容量提供者或 Fargate 容量提供者。

標籤傳播

Amazon ECS 受管執行個體的容量提供者支援標籤傳播功能。透過標籤傳播功能，容量提供者管理的所有資源 (受管執行個體、Amazon ECS 容器執行個體、啟動範本、磁碟區、彈性網路介面)，皆會以在容量提供者層級指定的相同標籤進行標記。您可以在容量提供者建立期間指定標籤，同時將 `propagateTags` 參數指定為 `CAPACITY_PROVIDER` 來啟用標籤傳播功能。

如需有關標記 Amazon ECS 受管執行個體的詳細資訊，請參閱 [Amazon ECS 受管執行個體的標籤](#)。

更新 Amazon ECS 受管執行個體容量提供者的最佳實務

對於最高層級的安全與復原支援，建議將容量提供者視為不可變資源。若需更新容量提供者組態，請遵循以下建議的工作流程：

1. 透過更新後的組態建立新容量提供者，而不是修改現有的容量提供者。
2. 更新每個服務以採用新的容量提供者，並允許完成部署作業。
3. 在確認新組態如預期般運作後，刪除舊的容量提供者。

此方法具備如下優勢：

- 受控推展 – 每次更新一個服務，並監控相關影響。
- 輕易復原 – 發生問題時可快速將服務復原為使用先前的容量提供者。
- 縮小影響範圍 – 新組態的問題不會立即影響所有工作負載。

Note

如果您使用的是 CloudFormation，請考慮保留舊容量提供者，直到稍後部署為止，以保留復原堆疊變更的能力。

雖然您可以就地更新容量提供者，但此方法會導致更大的未受控影響範圍。就地更新會將新設定套用至往後佈建的所有新容量，但不會觸發服務部署。這意味著，您可能要等到服務需要擴展時，才會發現組態問題。

為 Amazon ECS 受管執行個體建立容量提供者

Amazon ECS 受管執行個體會使用容量提供者來管理工作負載的運算容量。當您在不指定的情況下建立容量提供者時 `instanceRequirements`，Amazon ECS 會自動選取最具成本最佳化的一般用途執行個體類型。您可以使用 `instanceRequirements` 建立容量提供者，以指定執行個體屬性，例如執行個體類型、CPU 製造商、加速器類型和其他需求。

自訂容量提供者採用屬性型執行個體類型選擇機制，可讓您將執行個體需求表達為一組屬性。這些需求會自動轉換為所有相符的 Amazon EC2 執行個體類型，由此簡化執行個體類型組態的建立與維護。若要進一步了解執行個體需求與屬性型選擇，請參閱 Amazon EC2 User Guide 中的 [Amazon EC2 Fleet attribute-based instance type selection](#) 文件。

先決條件

開始之前，請確保您已完成下列事項：

- 決定要使用的監控類型。如需詳細資訊，請參閱 [the section called “Amazon ECS 受管執行個體的詳細監控”](#)。
- 已有現有的叢集，或計劃建立新叢集。如需詳細資訊，請參閱 [為 Amazon ECS 受管執行個體建立叢集](#)。
- 您有 Amazon ECS 受管執行個體所需的 IAM 角色。其中包含：
 - 基礎設施角色 - 允許 Amazon ECS 代表您呼叫 AWS 服務，以管理 Amazon ECS 受管執行個體基礎設施。

如需詳細資訊，請參閱 [Amazon ECS 基礎結構 IAM 角色](#)。

- 執行個體設定檔 – 為在受管執行個體上執行的 Amazon ECS 容器代理程式與 Docker 常駐程式提供許可。

如需詳細資訊，請參閱 [Amazon ECS 受管執行個體執行個體設定檔](#)。

了解如何選擇執行個體。如需詳細資訊，請參閱[Amazon ECS 受管執行個體的執行個體選擇最佳實務](#)。

主控台程序

為 Amazon ECS 受管執行個體建立容量提供者 (Amazon ECS 主控台)

1. 開啟主控台，網址為 <https://console.aws.amazon.com/ecs/v2>。
2. 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. 在叢集頁面上，選擇叢集的名稱。
5. 在叢集頁面上，選擇基礎結構索引標籤。
6. 在容量提供者區段，選擇建立容量提供者。
7. 在容量提供者組態區段，設定下列項目：
 - 在容量提供者名稱欄位中，輸入容量提供者的唯一名稱。
 - 在容量提供者類型欄位中，選擇 Amazon ECS 受管執行個體。
8. 在執行個體組態區段，設定下列項目：
 - 在執行個體設定檔欄位中，選擇為 Amazon ECS 受管執行個體建立的執行個體設定檔角色。
 - 在基礎結構角色欄位中，選擇為 Amazon ECS 受管執行個體建立的基礎結構角色。
9. 在執行個體需求區段下，指定執行個體的屬性。您可以設定下列項目的任意組合：
 - vCPU 計數 – 指定 vCPU 的數量 (例如 4 或 8-16 的範圍)。
 - 記憶體 (MiB) – 指定 MiB 中的記憶體數量 (例如 8192 或 16384-32768 的範圍)。
 - 執行個體類型 – 指定特定執行個體類型 (例如 m5.large, m5.xlarge, c5.large)。
 - CPU 製造商 – 從 intel、amd 或 amazon-web-services 中選擇。
 - 加速器類型 – 指定加速器類型 (例如 gpu、fpga 或 inference)。
 - 加速器計數 – 指定加速器數量 (例如 1 或 2-4 的範圍)。
10. 在進階組態區段，選擇下列任一監控選項：
 - 若要讓 CloudWatch 傳送狀態檢查指標，請選擇基本。
 - 若要讓 CloudWatch 傳送所有指標指標，請選擇詳細。
11. (選用) 為協助識別容量提供者，請展開標籤區段，然後設定標籤。

若要啟用從容量提供者到受管資源的標籤傳播，例如從容量提供者啟動的執行個體，請在傳播標籤來源欄位中選擇容量提供者。

[新增標籤] 選擇新增標籤，並執行下列動作：

- 在金鑰欄位中，輸入金鑰名稱。
- 在值欄位中，輸入金鑰值。

12. 選擇建立。

AWS CLI 程序

您可以使用 AWS CLI 為 Amazon ECS 受管執行個體建立容量提供者。使用 AWS CLI 的最新版本。如需有關如何升級至最新版本的詳細資訊，請參閱 [Installing or updating to the latest version of the AWS CLI](#)。

為 Amazon ECS 受管執行個體建立容量提供者 (AWS CLI)

1. 執行以下命令：

```
aws ecs create-capacity-provider --cli-input-json file://capacity-provider-definition.json
```

下列 capacity-provider-definition.json 可用來指定基本執行個體需求與執行個體儲存體大小，以及啟用標籤傳播：

```
{
  "name": "my-managed-instances-provider",
  "cluster": "my-cluster",
  "tags": [
    {
      "key": "version",
      "value": "test"
    }
  ],
  "managedInstancesProvider": {
    "infrastructureRoleArn": "arn:aws:iam::123456789012:role/ecsInfrastructureRole",
    "instanceLaunchTemplate": {
      "ec2InstanceProfileArn": "arn:aws:iam::123456789012:instance-profile/ecsInstanceRole",
```



```
    "instanceRequirements": {
      "vCpuCount": {
        "min": 4,
        "max": 8
      },
      "memoryMiB": {
        "min": 8192,
        "max": 16384
      }
    },
    "networkConfiguration": {
      "subnets": [
        "subnet-abcdef01234567",
        "subnet-bcdefa98765432"
      ],
      "securityGroups": [
        "sg-0123456789abcdef"
      ]
    },
    "storageConfiguration": {
      "storageSizeGiB": 100
    },
    "monitoring": "basic"
  },
  "propagateTags": "CAPACITY_PROVIDER"
}
```

2. 驗證容量提供者是否建立成功：

```
aws ecs describe-capacity-providers \
  --capacity-providers my-managed-instances-provider
```

後續步驟

建立容量提供者之後，即可在建立服務或執行任務時使用：

- 若要將容量提供者用於服務，請參閱[建立 Amazon ECS 滾動更新部署](#)。
- 若要將容量提供者用於獨立任務，請參閱[將應用程式作為 Amazon ECS 任務執行](#)。

更新 Amazon ECS 受管執行個體監控

您可以修改 Amazon ECS 受管執行個體容量提供者的監控選項，在基本監控與詳細監控之間進行變更。您可以透過監控選項調整監控資料的收集層級，無需重新建立容量提供者。

如需監控選項的詳細資訊，請參閱[監控 Amazon ECS 受管執行個體](#)。

主控台程序

更新 Amazon ECS 受管執行個體的監控 (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. 在叢集頁面上，選擇叢集的名稱。
5. 在叢集頁面上，選擇基礎結構索引標籤。
6. 在進階組態區段，選擇下列任一監控選項：
 - 若要讓 CloudWatch 傳送狀態檢查指標，請選擇基本。
 - 若要讓 CloudWatch 傳送所有指標指標，請選擇詳細。
7. 選擇更新。

若要更新與現有 Amazon ECS 受管執行個體容量提供者相關聯的標籤，請遵循如下步驟：

1. 在導覽窗格中，選擇叢集。
2. 在叢集頁面上，選擇基礎結構。
3. 在基礎結構頁面上，選擇已建立的容量提供者。
4. 在容量提供者頁面上，選擇標籤。
5. 在 Tags (標籤) 下，選擇 Manage tags (管理標籤)。
6. 若要新增標籤，請指定要新增之標籤的鍵值，然後選擇儲存。若要一次新增多個標籤，請為每個要新增的標籤選擇新增標籤。您最多可新增 50 個標籤。

若要移除標籤，請選擇 Remove (移除)。

 Note

如果啟用標籤傳播，在建立容量提供者之後新增或移除的標籤，皆不適用於容量提供者先前建立的資源。

AWS CLI 程序

您可以使用 AWS CLI 為 Amazon ECS 受管執行個體更新容量提供者。使用 AWS CLI 的最新版本。如需有關如何升級至最新版本的詳細資訊，請參閱 [Installing or updating to the latest version of the AWS CLI](#)。

更新 Amazon ECS 受管執行個體的監控 (AWS CLI)

1. 若要啟用詳細監控，請使用如下命令：

```
aws ecs update-capacity-provider \
  --name my-managed-instances-provider \
  --managed-instances-provider '{
    "instanceLaunchTemplateUpdate": {
      "monitoring": "DETAILED"
    }
  }'
```

2. 若要啟用基本監控，請使用如下命令：

```
aws ecs update-capacity-provider \
  --name my-managed-instances-provider \
  --managed-instances-provider '{
    "instanceLaunchTemplateUpdate": {
      "monitoring": "BASIC"
    }
  }'
```

刪除 Amazon ECS 受管執行個體容量提供者

如果 Amazon ECS 受管執行個體容量提供者已使用完畢，即可將其刪除。刪除群組後，Amazon ECS 受管執行個體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 Amazon ECS 受管執行個體容量提供者之前，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。使用強制執行新部署選項，可確保使用容量提供者提供之 Amazon ECS 受管執行個體容量的任何任務都會轉換，以使用剩餘容量提供者的容量。

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. 在叢集：**##**頁面上，選擇基礎結構、Amazon ECS 受管執行個體容量提供者，然後選擇刪除。
5. 在確認方塊中，輸入 delete **Amazon ECS Managed Instances capacity provider name**
6. 選擇 刪除。

安全地停止在 Amazon ECS 受管執行個體上執行的 Amazon ECS 工作負載

您可以使用 Amazon ECS 任務縮減保護來保護任務，避免任務因服務擴展或部署的縮減事件終止。

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。
- 遊戲應用程式會以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。在將 ProtectionEnabled 設定為 true 時，任務預設會受到 2 小時保護。您隨後可以使用 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)。如果您使用的是 AWS CLI，您可以指定 --protection-enabled 選項。

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。如果您使用的是 AWS CLI，您可以指定 --no-protection-enabled 選項。

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。在任務完成工作之後，您可以使用相同的端點清除對 `ProtectionEnabled` 屬性的設定，使任務符合在後續縮減事件期間終止的資格。

如需有關 Amazon ECS 容器代理程式端點的詳細資訊，請參閱 [Amazon ECS 任務縮減保護端點](#)。

- Amazon ECS API

如果應用程式具有追蹤作用中任務狀態的元件，則可使用 Amazon ECS API 來設定與擷取任務縮減保護。使用 `UpdateTaskProtection` 將一或多個任務標記為受保護。使用 `GetTaskProtection` 擷取保護狀態。

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 Elastic Container Service API Reference 中的 [UpdateTaskProtection](#) 與 [GetTaskProtection](#)。

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：

- 任務縮減保護僅支援從服務部署的任務。
- 任務縮減保護支援從在 Amazon ECS 受管執行個體上執行的服務部署的任務。

- 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。
- 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。
 - 如果您使用 CloudFormation，則更新堆疊會有 3 小時的逾時。因此，如果您將任務保護設定為 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 堆疊會顯示 `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

任務必須具備具有如下許可的 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 任務縮減保護端點

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。

從容器內對此 URI 發送 PUT 請求，設定任務縮減保護。對此 URI 的 GET 請求將傳回任務的目前保護狀態。

任務縮減保護請求參數

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` 來標記任務以實現保護。指定 `false` 以移除保護，並讓任務符合終止資格。

類型：布林值

必要：是

ExpiresInMinutes

任務受到保護的分鐘數。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任務的範例

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}'
```

60 分鐘保護任務的範例

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}'
```

24 小時保護任務的範例

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 請求將傳回以下回應。

```
{
  "protection": {
    "ExpirationDate": "2023-12-20T21:57:44.837Z",
    "ProtectionEnabled": true,
    "TaskArn": "arn:aws:ecs:us-west-2:111122223333:task/1234567890abcdef0"
  }
}
```

任務縮減保護回應參數

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)。如果任務未受到保護，則此值將為空。

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

造成例外狀況的 Amazon ECS API 呼叫 AWS 請求 ID。

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"
  }
}
```

Fargate 的 Amazon ECS 叢集

使用 AWS Fargate 容量提供者的 Amazon ECS，您可以將 Fargate 和 Fargate Spot 容量與 Amazon ECS 任務搭配使用。

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，您的任務會因兩分鐘警告而中斷。

當使用 Fargate 和 Fargate Spot 容量提供者的任務停止時，系統會將任務狀態變更事件傳送至 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：

- 將容量提供者與容量提供者策略相關聯之前，必須先將容量提供者與叢集建立關聯。
- 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，0 則會使用的預設值。
- 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。任何權重為零的容量提供者都不會用來置放任務。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。
- 一個叢集可以包含同時使用這兩種容量提供者的服務與獨立任務。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 Spot 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。發生此狀況時，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，並作為 SIGTERM 訊號傳送給正在執行的任務。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 任務狀態變更事件之 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。如需有關建立 EventBridge 規則的詳細資訊，請參閱 Amazon EventBridge User Guide 中的 [Getting started with Amazon EventBridge](#)。

```

{
  "source": [
    "aws.ecs"
  ],
  "detail-type": [
    "ECS Task State Change"
  ],
  "detail": {
    "clusterArn": [
      "arn:aws:ecs:us-west-2:111122223333:cluster/default"
    ]
  }
}

```

為 Fargate 工作負載建立 Amazon ECS 叢集

您可以透過建立叢集，定義任務與服務執行所在的基礎結構。

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 叢集範例”](#)。Amazon ECS 主控台會透過建立 CloudFormation 堆疊來建立 Amazon ECS 叢集所需的資源。

主控台會自動將 Fargate 及 Fargate Spot 容量提供者與叢集建立關聯。

您可以修改下列選項：

- 將命名空間新增至叢集。

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](#)。

- 啟用任務事件以接收任務狀態變更的 EventBridge 通知。
- 新增標籤以協助您識別叢集。
- 為您的受管儲存指派 AWS KMS 金鑰。如需有關如何建立金鑰的資訊，請參閱 AWS Key Management Service User Guide 中的 [Create a KMS key](#)。
- 為您的 Fargate 暫時性儲存指派 AWS KMS 金鑰。如需有關如何建立金鑰的資訊，請參閱 AWS Key Management Service User Guide 中的 [Create a KMS key](#)。
- 設定 ECS Exec 的 AWS KMS 金鑰和記錄。

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 Service Connect 預設值區段的預設命名空間欄位中，選擇或輸入命名空間名稱。若要使用共用命名空間，請選擇或輸入命名空間 ARN。如需有關使用共用命名空間的詳細資訊，請參閱 [具有共用 AWS Cloud Map 命名空間的 Amazon ECS Service Connect](#)。
6. (選用) 使用 Container Insights，展開監控區段，然後選擇下列任一選項：
 - 若要使用建議的具有增強可觀測性的 Container Insights，請選擇具有增強可觀測性的 Container Insights。
 - 若要使用 Container Insights，請選擇 Container Insights。

7. (選用) 若要啟用任務事件，請展開任務事件，然後開啟啟用任務事件。

當您啟用任務事件時，Amazon ECS 會將任務狀態變更事件傳送至 EventBridge。這可讓您自動監控和回應任務生命週期變更。

8. (選用) 若要使用 ECS Exec 在叢集中偵錯任務，請展開疑難排解組態區段，然後設定下列項目：

- (選用) 對於 AWS KMS ECS Exec 的金鑰，輸入您要用來加密 ECS Exec 工作階段資料的 AWS KMS 金鑰 ARN。
- (選用) 在 ECS Exec 記錄欄位中，選擇日誌目的地：
 - 若要將日誌傳送至 CloudWatch Logs，請選擇 Amazon CloudWatch。
 - 若要將日誌傳送至 Amazon S3，請選擇 Amazon S3。
 - 若要停用記錄，請選擇無。

9. (選用) 在加密區段，您可以設定下列項目：

- 在 Fargate 暫時性儲存上加密資料。在加密下，針對 Fargate 暫時性儲存，輸入您要用來加密 Fargate 暫時性儲存資料的 AWS KMS 金鑰 ARN。
- 加密受管儲存上的資料。在加密下，針對受管儲存，輸入您要用來加密受管儲存資料的 AWS KMS 金鑰 ARN。

10. (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：

- 在金鑰欄位中，輸入金鑰名稱。
- 在值欄位中，輸入金鑰值。

[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](#)
- [將應用程式作為 Amazon ECS 任務執行](#)
- [建立 Amazon ECS 滾動更新部署](#)

EC2 工作負載的 Amazon ECS 容量提供者

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 能確保您有正確的 Amazon EC2 執行個體數量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 群組的縮減和橫向擴充動作，也可以自行管理擴展動作。如需詳細資訊，請參閱[使用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 CloudFormation 堆疊的一部分。名稱字首皆為 EC2ContainerService-*<ClusterName>*。您可以使用 Auto Scaling 群組作為該叢集的容量提供者。

建議使用受管執行個體排空功能，允許優雅終止 Amazon EC2 執行個體，不會中斷工作負載。預設會啟用此功能。如需詳細資訊，請參閱[安全地停止在 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。
- 將容量提供者與容量提供者策略相關聯之前，必須先將容量提供者與叢集建立關聯。

- 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，0 則會使用的預設值。
- 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。任何權重為零的容量提供者都不會用來置放任務。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。每當應用程式需要橫向擴充時，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)。這可讓任何最終初始化程序執行，然後將執行個體投入使用。如需詳細資訊，請參閱 [為 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 User Guide 中的 [Auto Scaling launch templates](#)。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 Amazon 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 執行期監控來偵測 AWS 環境中叢集和容器的威脅。執行時期監控使用 GuardDuty 安全代理程式，將執行時期可見性新增至個別 Amazon ECS 工作負載，例如檔案存取、程序執行與網路連線。如需詳細資訊，請參閱 GuardDuty User Guide 中的 [GuardDuty Runtime Monitoring](#)。

為 Amazon EC2 工作負載建立 Amazon ECS 叢集

您可以透過建立叢集，定義任務與服務執行所在的基礎結構。

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 叢集範例”](#)。Amazon ECS 主控台提供簡單的方法，透過建立 CloudFormation 堆疊來建立 Amazon ECS 叢集所需的資源。

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：

- 變更執行個體啟動所在的子網路。
- 變更為控制容器執行個體流量的安全群組。
- 將命名空間新增至叢集。

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](#)。

- 啟用任務事件以接收任務狀態變更的 EventBridge 通知。
- 為您的受管儲存指派 AWS KMS 金鑰。如需有關如何建立金鑰的資訊，請參閱 AWS Key Management Service User Guide 中的 [Create a KMS key](#)。
- 為您的 Fargate 暫時性儲存指派 AWS KMS 金鑰。如需有關如何建立金鑰的資訊，請參閱 AWS Key Management Service User Guide 中的 [Create a KMS key](#)。
- 設定 ECS Exec 的 AWS KMS 金鑰和記錄。
- 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。如需有關如何建立 SSH 金鑰的資訊，請參閱 Amazon EC2 User Guide 中的[Amazon EC2 key pairs and Linux instances](#)。
- 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 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。

聯網選項

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群組，此群組可控制容器執行個體的流量。您可以變更執行個體的字網路和安全群組。

您可以選擇現有的子網路。您可以使用現有的安全群組，或建立新的安全群組。若要以純 IPv6 組態建立任務，請使用僅包含 IPv6 CIDR 區塊的子網路。

建立新的安全群組時，必須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
- 傳入流量 (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 叢集範例”](#)。

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 Service Connect 預設值區段的預設命名空間欄位中，選擇或輸入命名空間名稱。若要使用共用命名空間，請選擇或輸入命名空間 ARN。如需有關使用共用命名空間的詳細資訊，請參閱 [具有共用 AWS Cloud Map 命名空間的 Amazon ECS Service Connect](#)。
6. 將 Amazon EC2 執行個體新增至叢集，展開基礎結構區段，然後選取 Fargate 與自行管理執行個體。

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：
 - 在佈建模型欄位中，選擇要使用隨需執行個體或是 Spot 執行個體。
 - 如果選擇使用 Spot 執行個體，請在配置策略欄位中，選擇執行個體要使用的 Spot 容量集區 (執行個體類型與可用區域)。

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](#)。

- 在容器執行個體 Amazon Machine Image (AMI) 欄位中，選擇適用於 Auto Scaling 群組執行個體的 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。

- 在 EC2 執行個體角色欄位中，您可以選擇現有的容器執行個體角色，也可以建立新的容器執行個體角色。

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。
- 若要允許較大的映像與儲存空間，在根 EBS 磁碟區大小欄位中，請輸入以 GiB 為單位的值。

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。對於純 IPv6 組態，選擇具有 IPv6 CIDR 區塊的 VPC，以及僅具有 IPv6 CIDR 區塊的子網路。
- 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 [聯網選項](#)。

- 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 地址指派給執行個體。
 - 開啟：將公有 IP 地址指派給執行個體。
- 8. (選用) 使用 Container Insights，展開監控區段，然後選擇下列任一選項：
 - 若要使用建議的具有增強可觀測性的 Container Insights，請選擇具有增強可觀測性的 Container Insights。
 - 若要使用 Container Insights，請選擇 Container Insights。
- 9. (選用) 若要啟用任務事件，請展開任務事件，然後開啟啟用任務事件。

當您啟用任務事件時，Amazon ECS 會將任務狀態變更事件傳送至 EventBridge。這可讓您自動監控和回應任務生命週期變更。
- 10. (選用) 若要使用 ECS Exec 在叢集中偵錯任務，請展開疑難排解組態區段，然後設定下列項目：
 - (選用) 對於 AWS KMS ECS Exec 的金鑰，輸入您要用來加密 ECS Exec 工作階段資料的 AWS KMS 金鑰 ARN。
 - (選用) 在 ECS Exec 記錄欄位中，選擇日誌目的地：
 - 若要將日誌傳送至 CloudWatch Logs，請選擇 Amazon CloudWatch。
 - 若要將日誌傳送至 Amazon S3，請選擇 Amazon S3。
 - 若要停用記錄，請選擇無。
- 11. (選用)

如果搭配手動選項使用執行時期監控，而且想讓 GuardDuty 監控此叢集，請選擇新增標籤並執行下列動作：

- 在鍵欄位中，輸入 **guardDutyRuntimeMonitoringManaged**
 - 針對數值，輸入 **true**。
12. (選用) 加密受管儲存上的資料。在加密下，針對受管儲存，輸入您要用來加密受管儲存資料的 AWS KMS 金鑰 ARN。
13. (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：

- 在金鑰欄位中，輸入金鑰名稱。
- 在值欄位中，輸入金鑰值。

[移除標籤] 選擇標籤「金鑰」和「值」右側的移除。

14. 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](#)
- [將應用程式作為 Amazon ECS 任務執行](#)
- [建立 Amazon ECS 滾動更新部署](#)

使用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。在建立 Amazon ECS Auto Scaling 群組容量提供者時，您可以啟用受管擴展。然後，您可以為此 Auto Scaling 群組中的執行個體使用率設定目標百分比 (targetCapacity)。Amazon ECS 會為 Auto Scaling 群組建立兩個自訂 CloudWatch 指標與一個目標追蹤擴展政策。Amazon ECS 接著會根據任務使用的資源使用率來管理縮減與橫向擴充動作。

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：

- 低指標值 CloudWatch 警示
- 高指標值 CloudWatch 警示
- 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。

如果關閉受管擴展或解除容量提供者與叢集的關聯，Amazon ECS 會移除 CloudWatch 指標和目標追蹤擴展政策資源。

Amazon ECS 使用以下指標來判定要採取的動作：

CapacityProviderReservation

特定容量提供者正在使用的容器執行個體的百分比。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。之後，Amazon ECS 會將指標發佈到 CloudWatch。如需有關指標計算方式的詳細資訊，請參閱 [Deep Dive on Amazon ECS Cluster Auto Scaling](#)。

```
CapacityProviderReservation = (number of instances needed) / (number of running instances) x 100
```

DesiredCapacity

Auto Scaling 群組的容量。此指標不會發布至 CloudWatch。

之後，Amazon ECS 會在 AWS/ECS/ManagedScaling 命名空間中將 CapacityProviderReservation 指標發佈到 CloudWatch。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 指標增加的值會導致相關聯的 CloudWatch 警示採取動作。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 指標減少值會導致相關聯的 CloudWatch 警示採取動作。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 終止的執行個體](#)。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。
- 當 Amazon ECS 從 0 個執行個體橫向擴充時，會自動啟動 2 個執行個體。
- Amazon ECS 會將 `AWSServiceRoleForECS` 服務連結 IAM 角色用於 AWS Auto Scaling 代表您呼叫所需的許可。如需詳細資訊，請參閱 [使用 Amazon ECS 的服務連結角色](#)。
- 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。如果已移除此標籤，則 Amazon ECS 無法管理擴展。

- 叢集自動擴展不會修改群組的 `MinimumCapacity` (容量下限) 或 `MaximumCapacity` (容量上限)。為了讓群組進行橫向擴展，`MaximumCapacity` (容量上限) 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。
- 就容量而言，`binpack` 策略是最有效率的策略。
- 當目標容量小於 100% 時，在置放策略中，`binpack` 策略的有限順序必須高於 `spread` 策略。這可防止容量提供者橫向擴充，直至每個任務都有專用執行個體或達到限制為止。

開啟叢集自動擴展

您可以使用主控台或 AWS CLI 啟用叢集自動擴展。

當您透過主控台建立使用 EC2 容量提供者的叢集時，Amazon ECS 會代為建立 Auto Scaling 群組並設定目標容量。如需詳細資訊，請參閱[為 Amazon EC2 工作負載建立 Amazon ECS 叢集](#)。

您也可以建立 Auto Scaling 群組，然後將群組指派給叢集。如需詳細資訊，請參閱[更新 Amazon ECS 容量提供者](#)。

當您使用 時 AWS CLI，在您建立叢集之後

1. 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](#)。
2. 使用 `put-cluster-capacity-providers` 修改叢集容量提供者。如需詳細資訊，請參閱[啟用 Amazon ECS 叢集自動擴展](#)。

最佳化 Amazon ECS 叢集自動擴展

在 Amazon EC2 上執行 Amazon ECS 的客戶，可以利用叢集自動擴展來管理 Amazon EC2 Auto Scaling 群組的擴展。透過叢集自動擴展，您可以設定 Amazon ECS 自動擴展 Auto Scaling 群組，從而專注於執行任務。Amazon ECS 將確保 Auto Scaling 群組視需求進行擴展與縮減，無需額外人工介入。Amazon ECS 容量提供者可用來管理叢集中的基礎結構，確保有足夠的容器執行個體可滿足應用程式需求。若要了解叢集自動擴展的底層運作機制，請參閱 [Deep Dive on Amazon ECS Cluster Auto Scaling](#)。

叢集自動擴展依賴於 CloudWatch 與 Auto Scaling 群組的整合機制來調整叢集容量。因此存在以下固有延遲：

- 發布 CloudWatch 指標。
- 指標 `CapacityProviderReservation` 觸發 CloudWatch 高低閾值警報所需的處理時間
- 新啟動的 Amazon EC2 執行個體完成暖機所需的時間。您可以採取下列動作提高叢集自動擴展的回應速度，加速實現部署：

容量提供者步進擴展大小

Amazon ECS 容量提供者將擴增/縮小容器執行個體，以滿足應用程式的需求。將 Amazon ECS 會啟動的執行個體數量下限預設為 1。如果需要多個執行個體來放置待處理任務，可能會增加部署所需的時間。您可以透過 Amazon ECS API 增加 [minimumScalingStepSize](#)，提高 Amazon ECS 每次可縮減或擴展的執行個體數量下限。太低的 [maximumScalingStepSize](#) 會限制每次可縮減或擴展的容器執行個體數量，進而拖慢部署速度。

Note

此組態目前只能透過 [CreateCapacityProvider](#) 或 [UpdateCapacityProvider](#) API 使用。

執行個體暖機期間

執行個體暖機期間是指新啟動的 Amazon EC2 執行個體納入 Auto Scaling 群組 CloudWatch 指標統計前的一段時間。在指定的暖機期到期後，執行個體才會納入 Auto Scaling 群組的彙總指標計算；而叢集自動擴展會接著進行其下一輪計算，以估算所需的執行個體數量。

[instanceWarmupPeriod](#) 的預設值為 300 秒，您可以透過 [CreateCapacityProvider](#) 或 [UpdateCapacityProvider](#) API 將其設定為較低的值，從而實現更即時的擴展。建議將該值設定為大於 60 秒，避免過度佈建。

備用容量

如果容量提供者沒有可用於置放任務的容器執行個體，就必須透過即時啟動 Amazon EC2 執行個體來增加 (橫向擴充) 叢集容量，並等待這些執行個體啟動完成後才能在其上部署容器。這會大幅降低任務啟動速率。對此，您有兩種選擇。

在這種情況下，若事先啟動並準備好可用於執行任務的 Amazon EC2 備用容量，即能提高有效的任務啟動速率。您可以透過 Target Capacity 組態來指定希望叢集維持的備用容量。例如，將 Target Capacity 設定為 80%，即指定叢集需隨時保留 20% 的備用容量。這些備用容量可讓任何獨立任務立即啟動，確保任務啟動不會受到限流。但此方法的權衡在於，維持叢集備用容量可能會增加成本。

另一種替代方法是為服務增加緩衝空間，而非為容量提供者設定。這意味著，您不必透過降低 Target Capacity 組態來啟動備用容量，而是可以藉由修改服務自動擴展的目標追蹤擴展指標或步進擴展閾值，直接增加服務中的副本數量。請注意，此方法僅對突發性工作負載有效；在部署新服務並首次從 0 擴展至 N 個任務時，則無法發揮作用。如需有關相關擴展政策的詳細資訊，請參閱《Amazon Elastic Container Service 開發人員指南》中的 [目標追蹤擴展政策](#) 或 [步進擴展政策](#)。

Amazon ECS 受管擴展行為

如果您的 Auto Scaling 群組容量提供者使用受管擴展，Amazon ECS 會估計要新增至叢集的最佳執行個體數量的下限，並使用此值來決定要請求或發放的執行個體數量。

受管向外擴展行為

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 會針對單一容量提供者執行下列後續步驟。

沒有容量提供者策略的任務會被容量提供者忽略。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。如需有關如何選擇執行個體類別的詳細資訊，請參閱 [Amazon ECS 的 Amazon EC2 容器執行個體](#)。

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。
- 如果是下列情況之一，Amazon ECS 會依照 minimumScalingStepSize 將 CapacityProviderReservation 指標發佈到 CloudWatch：
 - 計算出的執行個體數量上限，小於擴展步驟大小下限。
 - 為 maximumScalingStepSize 或計算出的執行個體數量上限中的較小值。

- CloudWatch 警示會使用容量提供者的 CapacityProviderReservation 指標。當 CapacityProviderReservation 指標大於 targetCapacity 值時，警示也會增加 Auto Scaling 群組的 DesiredCapacity。targetCapacity 值是在叢集自動擴展活動期間，傳送到 CloudWatch 警示的容量提供者設定。

預設 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 確保最近啟動的 EC2 執行個體容量已穩定，然後才會啟動新執行個體。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)。處於 instanceWarmupPeriod 內的執行個體會封鎖橫向擴充。

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。如需縮短此時間，請確保該 instanceWarmupPeriod 足夠大，足以 EC2 執行個體啟動並執行 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` 啟動一個任務，而容量提供者容器執行個體沒有足夠的資源來啟動任務，則 Amazon ECS 會限制每個叢集具有此狀態的任務數目，防止任何任務執行時超越此限制。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 縮減警示需要 15 個資料點 (15 分鐘)，然後才能開始 Auto Scaling 群組的縮減程序。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。之後，Amazon ECS 會將指標發佈到 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](#)

$$\text{CapacityProviderReservation} = (\text{number of instances needed}) / (\text{number of running instances}) \times 100$$

3. `CapacityProviderReservation` 指標會產生 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。

控制 Amazon ECS 終止的執行個體

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自動擴展控制哪些執行個體會終止。使用受管終止保護時，Amazon ECS 只會終止沒有執行 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 會先針對 Auto Scaling 群組中的 EC2 執行個體啟用執行個體縮減保護選項。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，保有備用容量。此設定能加速後續任務啟動，因為 Auto Scaling 群組無需額外啟動執行個體。Amazon ECS 使用目標容量值來管理服務建立的 CloudWatch 指標。Amazon ECS 會管理 CloudWatch 指標。Auto Scaling 群組被視為穩定狀態，無需觸發擴展動作。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](#)。

更新 Amazon ECS 容量提供者的受管終止保護

使用受管終止保護時，您需要更新現有容量提供者的設定。

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. 在叢集頁面上，選擇基礎結構索引標籤。
4. 選擇對應的容量提供者。
5. 選擇更新，修改容量提供者設定。
6. 在 Auto Scaling 群組設定區段，切換受管終止保護開關，啟用或停用此功能。
7. 選擇更新。

AWS CLI

您可以使用 `update-capacity-provider` 命令更新容量提供者的受管終止保護設定：

啟用受管終止保護：

```
aws ecs update-capacity-provider \
  --name CapacityProviderName \
  --auto-scaling-group-provider
"managedScaling={status=ENABLED,targetCapacity=70,minimumScalingStepSize=1,maximumScalingStepS
```

停用受管終止保護：

```
aws ecs update-capacity-provider \
  --name CapacityProviderName \
  --auto-scaling-group-provider
"managedScaling={status=ENABLED,targetCapacity=70,minimumScalingStepSize=1,maximumScalingStepS
```


Note

可能需要幾分鐘的時間，變更才能在叢集中生效。啟用受管終止保護時，正在執行任務的執行個體將受到保護，不受縮減事件影響。停用受管終止保護時，保護旗標將於下個 ECS 容量提供者管理週期時從執行個體中移除。

執行任務的主控制台

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. 在叢集頁面上，選擇任務索引標籤。
4. 選擇對應任務。
5. 在組態區段，切換受管終止保護開關，啟用或停用此功能。
6. 選擇設定任務縮減保護。

設定任務縮減保護對話方塊隨即顯示

- a. 在任務縮減保護區段，切換啟用開關。
- b. 在有效時長 (分鐘) 欄位中，輸入任務縮減保護到期前的分鐘數。
- c. 選擇 Update (更新)

啟用 Amazon ECS 叢集自動擴展

您需要啟用叢集自動擴展，以便 Amazon ECS 管理已註冊至叢集的 Amazon EC2 執行個體的擴展作業。

若想透過主控台啟用叢集自動擴展，請參閱[建立 Amazon ECS 容量提供者](#)。

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 工作負載的 Amazon ECS 容量提供者”](#)。

若要啟用叢集自動擴展，請將容量提供者與叢集建立關聯，然後啟用叢集自動擴展。

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，請在請求中包含 `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 新增 Auto Scaling 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 \  
  --name CapacityProviderName \  
  --auto-scaling-group-provider "managedScaling={status=ENABLED}"
```

關閉 Amazon ECS 叢集自動擴展

如需更精細地控制已註冊至叢集的 EC2 執行個體，您可以停用叢集自動擴展。

若要關閉叢集的叢集自動擴展，您可以解除容量提供者與透過叢集啟用之受管擴展的關聯，或者更新容量提供者來關閉受管擴展。

解除與容量提供者之間的關聯

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 \  
  --name CapacityProviderName \  
  --auto-scaling-group-provider "managedScaling={status=DISABLED}"
```

建立 Amazon ECS 容量提供者

叢集建立完成後，即可為 EC2 建立新的容量提供者 (Auto Scaling 群組)。容量提供者可協助管理並擴展應用程式的基礎結構。

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 Set target capacity (設定目標容量) 欄位中，輸入在 Amazon ECS 受管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 容量提供者

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 新主控台)

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 Set target capacity (設定目標容量) 欄位中，輸入在 Amazon ECS 受管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 群組容量提供者將轉換為 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 主控台內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。使用強制執行新部署選項，可確保使用容量提供者提供之 Amazon EC2 執行個體容量的任何任務都會轉換，以使用剩餘容量提供者的容量。

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。

安全地停止在 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優雅終止 Amazon EC2 執行個體。該機制可讓工作負載安全地停止，再重新排程至未終止的執行個體。基礎結構維護與更新作業可順利執行，且無需擔心工作負載受到中斷。透過受管執行個體排空，您可以簡化需要替代 Amazon EC2 執行個體的基礎結構管理工作流程，同時確保應用程式的彈性與可用性。

Amazon ECS 受管執行個體排空適用於 Auto Scaling 群組執行個體替代。根據執行個體重新整理與執行個體生命週期上限，客戶可以確保容量符合最新的作業系統與安全性要求。

受管執行個體排空只能與 Amazon ECS 容量提供者搭配使用。當您使用 Amazon ECS 主控台或 SDK 建立或更新 Auto Scaling 群組容量提供者時 AWS CLI，可以開啟受管執行個體耗盡。

Amazon ECS 受管執行個體排空會涵蓋下列事件。

- [Auto Scaling 群組執行個體重新整理](#) – 使用執行個體重新整理來執行 Auto Scaling 群組中 Amazon EC2 執行個體的滾動替代，而不是手動分批執行。當您需要替換大量執行個體時，此方法非常實用。執行個體重新整理透過 Amazon EC2 主控台或 `StartInstanceRefresh` API 啟動。如果使用受管終止保護，請務必在呼叫 `StartInstanceRefresh` 時為縮減保護選取 `Replace`。
- [執行個體生命週期上限](#) – 您可以在替換 Auto Scaling 群組執行個體時定義生命週期上限。此設定有助於根據內部安全政策或合規要求來排程替代執行個體。
- [Auto Scaling 群組縮減](#) – 根據擴展政策與排程擴展動作，Auto Scaling 群組支援執行個體的自動擴展。使用 Auto Scaling 群組作為 Amazon ECS 容量提供者，您可以在其中沒有任務執行時縮減 Auto Scaling 群組執行個體。
- [Auto Scaling 群組運作狀態檢查](#) – Auto Scaling 群組支援許多運作狀態檢查，可管理運作狀態不良的執行個體的終止。
- [CloudFormation 堆疊更新](#) - 您可以在 CloudFormation 堆疊中新增 `UpdatePolicy` 屬性，以在群組變更時執行滾動更新。
- [Spot 容量重新平衡](#) – Auto Scaling 群組會根據 Amazon EC2 容量重新平衡通知，主動嘗試替換中斷風險較高的 Spot 執行個體。Auto Scaling 群組會在替代執行個體完成啟動且運作狀態良好時，終止舊執行個體。Amazon ECS 受管執行個體排空功能排空 Spot 執行個體的方式，與排空非 Spot 執行個體的方式相同。
- [Spot 中斷](#) – Spot 執行個體會兩分鐘通知後終止。Amazon ECS 受管執行個體排空會讓執行個體進入排空狀態，以此作為回應。

具有受管執行個體排空的 Amazon EC2 Auto Scaling lifecycle hook

Auto Scaling 群組 lifecycle hook 可讓客戶建立由執行個體生命週期中特定事件觸發的解決方案，並在發生此特定事件時執行自訂動作。Auto Scaling 群組最多允許 50 個勾點。可同時存在多組終止勾點平行執行，Auto Scaling 群組會等待所有勾點完成後才終止執行個體。

除 Amazon ECS 受管勾點終止外，您也可以自行設定生命週期終止勾點。Lifecycle hook 具有 default action，因此建議將 continue 設定為預設值，確保 Amazon ECS 受管勾點等其他勾點不受自訂勾點任何錯誤的影響。

如果您已設定 Auto Scaling 群組終止 lifecycle hook 並啟用 Amazon ECS 受管執行個體排空，則會執行兩個 lifecycle hook。不過，相對時間點無法保證。Lifecycle hook 具有 default action 設定，可指定逾時之時要採取的動作。如果失敗，建議在自訂勾點中使用 continue 作為預設結果。這可確保其他勾點，特別是 Amazon ECS 受管勾點不受自訂 lifecycle hook 中任何錯誤的影響。abandon 的替代結果會略過所有其他勾點，應避免此類狀況發生。如需有關 Auto Scaling 群組 lifecycle hook 的詳細資訊，請參閱 Amazon EC2 Auto Scaling User Guide 中的 [Amazon EC2 Auto Scaling lifecycle hook](#)。

任務與受管執行個體排空

Amazon ECS 受管執行個體排空功能會使用在容器執行個體中找到的現有排空功能。[容器執行個體排空](#)功能會針對屬於 Amazon ECS 服務的副本任務執行替代與停止作業。處於 PENDING 或 RUNNING 狀態的獨立任務與 RunTask 調用的任務一樣，不會受到影響。您必須等待這些任務完成，或手動將其停止。容器執行個體會維持 DRAINING 狀態，直至所有任務停止或超過 48 小時為止。常駐程式任務是所有副本任務停止後最後停止的任務。

受管執行個體排空與受管終止保護

即使停用受管終止，受管執行個體排空仍可運作。如需有關受管終止保護的資訊，請參閱[控制 Amazon ECS 終止的執行個體](#)。

下表摘要說明受管終止與受管排空的不同組合的行為。

受管終止	受管排空	結果
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

受管終止	受管排空	結果
		因縮減事件而終止。任何正在終止的執行個體，例如未設定終止保護、收到 Spot 中斷通知或受執行個體重新整理強制處理的執行個體，皆會優雅排空。
Disabled	已啟用	Amazon ECS 不會保護執行任務的 Amazon EC2 執行個體免受縮減影響。不過，任何正在終止的執行個體皆會優雅排空。

受管終止	受管排空	結果
已啟用	Disabled	Amazon ECS 可保護正在執行任務的 Amazon EC2 執行個體，避免因縮減事件而終止。不過，如果發生 Spot 中斷、強制執行個體重新整理或執行個體未執行任何任務，執行個體仍會終止。Amazon ECS 不會對這些執行個體執行優雅排空，並會在其停止後啟動替代服務任務。

受管終止	受管排空	結果
Disabled	Disabled	Amazon EC2 執行個體可以隨時縮減或終止，即使正在執行 Amazon ECS 任務。Amazon ECS 會在這些執行個體停止後啟動替代服務任務。

受管執行個體排空與 Spot 執行個體排空

透過 Spot 執行個體排空功能，您可以在 Amazon ECS 代理程式上設定環境變數 `ECS_ENABLE_SPOT_INSTANCE_DRAINING`，這能讓 Amazon ECS 因應為期兩分鐘的 Spot 中斷通知時，將執行個體設為排空狀態。Amazon ECS 受管執行個體排空功能可促進 Amazon EC2 執行個體的優雅關閉，這些執行個體的終止可能源於多種原因，而非僅限於 Spot 中斷。例如，您可以使用 Amazon EC2 Auto Scaling 容量重新平衡功能，在高中斷風險的情況下主動替換 Spot 執行個體，且受管執行個體排空會對要替換的 Spot 執行個體執行優雅關閉。透過受管執行個體排空功能，您不需要個別啟用 Spot 執行個體排空，因此 Auto Scaling 群組使用者資料中的 `ECS_ENABLE_SPOT_INSTANCE_DRAINING` 是多餘的。如需有關 Spot 執行個體排空的詳細資訊，請參閱 [Spot 執行個體](#)。

受管執行個體排空如何與 EventBridge 搭配使用

Amazon ECS 受管執行個體排空事件會發布至 Amazon EventBridge，Amazon ECS 會在您帳戶的預設匯流排中建立 EventBridge 受管規則，用以支援受管執行個體排空。您可以將這些事件篩選到 Lambda、Amazon SNS 和 Amazon SQS 等 AWS 其他服務，以監控和疑難排解。

- Amazon EC2 Auto Scaling 會在調用 lifecycle hook 時，將事件傳送至 EventBridge。
- Spot 中斷通知會發布至 EventBridge。

- Amazon ECS 會產生錯誤訊息，您可以透過 Amazon ECS 主控台與 API 擷取這些訊息。
- EventBridge 內建重試機制，可作為暫時故障的緩解措施。

設定 Amazon ECS 容量提供者以安全地關閉執行個體

當您透過 Amazon ECS 主控台與 AWS CLI 建立或更新 Auto Scaling 群組容量提供者時，即可啟用受管執行個體排空。

Note

當您建立容量提供者時，受管執行個體排空預設為啟用。

以下是使用 AWS CLI 建立已啟用受管執行個體耗盡的容量提供者，以及為叢集現有容量提供者啟用受管執行個體耗盡的範例。

建立已啟用受管執行個體排空的容量提供者

若要建立已啟用受管執行個體排空的容量提供者，請使用 `create-capacity-provider` 命令。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"
  }
}

```

為叢集現有容量提供者啟用受管執行個體排空

使用 `update-capacity-provider` 命令為叢集的現有容量提供者啟用受管執行個體排空。您會看到 `managedDraining` 目前顯示 `DISABLED`，`updateStatus` 顯示 `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 `describe-clusters` 命令，並包含 `ATTACHMENTS` 選項。受管執行個體排空連接的 `status` 為 `PRECREATED`，而整體的 `attachmentsStatus` 為 `UPDATING`。

```
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": []
}

```

更新完成後，請使用 `describe-capacity-providers`，您會看到 `managedDraining` 現在為 `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 受管執行個體排空疑難排解

您可能需要針對受管執行個體排空的問題進行疑難排解。以下是您在使用時可能遇到的問題與解決方案範例。

使用自動擴展時，執行個體不會在超過執行個體生命週期上限後終止。

如果在使用 Auto Scaling 群組時，您的執行個體即使在達到並超過執行個體生命週期上限之後仍未終止，可能是因為執行個體受到縮減保護。您可以停用受管終止功能，同時允許受管排空功能處理執行個體回收。

Amazon ECS 受管執行個體的排空行為

Amazon ECS 受管執行個體終止可確保正常的工作負載轉換，同時最佳化成本和維護系統運作狀態。終止系統為執行個體終止提供三種不同的決策路徑，每種都具有不同的時間特性與客戶影響設定檔。

終止決策路徑

客戶啟動的終止

在您需要從服務中立即移除容器執行個體時，提供執行個體移除的直接控制權。您可以在強制旗標設定為 true 的情況下調用 `DeregisterContainerInstance` API，表示即使有任何執行中的工作負載，仍需要立即終止。

系統啟動的閒置終止

Amazon ECS 受管執行個體透過終止未執行任何任務的閒置 Amazon ECS 容器執行個體，持續監控並主動最佳化成本。ECS 使用啟發式延遲，讓容器執行個體有機會在終止之前取得新啟動的任務。這可以使用 `scaleInAfter` Amazon ECS 受管執行個體容量提供者組態參數來自訂。

基礎結構重新整理終止

Amazon ECS 受管執行個體會自動管理和更新受管容器執行個體上的軟體，以確保安全性和合規性，同時維持工作負載可用性。如需詳細資訊，請參閱 [Amazon ECS 受管執行個體中的修補](#)。

優雅排空與工作負載遷移

優雅排空系統會實作與 Amazon ECS 服務管理的複雜協調，確保服務受管任務正確遷移至已排程終止的執行個體之外。

服務任務排空協調

當執行個體轉換為排空狀態時，Amazon ECS 排程器會自動停止在該執行個體上置放新任務，同時為現有服務任務實作優雅關閉程序。服務任務排空包括與服務部署策略、運作狀態檢查需求和排空偏好設定的協調，確保最佳的遷移時間與成功率。

獨立任務處理

獨立任務需要不同的處理方式，因為此類任務無法受益於自動服務管理機制。系統會評估獨立任務特性，包括任務持續時間預估、完成機率分析與客戶影響評估。優雅完成策略允許獨立任務在延長的寬限期內自然完成，而強制終止可確保任務尚未自然完成時，基礎結構重新整理會在可接受的時間範圍內發生。

兩階段完成策略

終止系統會實作兩階段方法，可平衡工作負載持續性與基礎結構管理需求。

階段 1：優雅完成期間

在此階段，系統會實作優先考量工作負載持續性的優雅排空策略。服務任務會透過正常的 Amazon ECS 排程程序優雅排空，獨立任務會持續執行且可能自然完成，系統將監控所有任務透過自然完成流程進入已停止狀態。

階段 2：硬性截止日期強制執行

當優雅完成在可接受的時間範圍內未達成終止目標時，系統會實作硬性截止日期強制執行。硬性截止日期通常設定為排空啟動時間加 7 天，為優雅完成提供大量時間，同時維持營運需求。強制執行包括自動調用強制取消註冊程序並立即終止所有剩餘的任務，無論完成狀態如何。

使用 為 Amazon ECS 叢集自動擴展建立資源 AWS 管理主控台

了解如何使用 AWS 管理主控台建立叢集自動擴展資源。資源需要名稱時，我們將使用字首 ConsoleTutorial 來確保這些資源都具有唯一的名稱，讓使用者能輕鬆找到資源。

主題

- [先決條件](#)
- [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](#)

先決條件

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 IAM 使用者擁有 [AmazonECS_FullAccess](#) IAM 政策範例中指定的所需許可。
- 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 Auto Scaling 啟動範本和 Auto Scaling 群組，做為 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：

- 在金鑰欄位中，輸入金鑰名稱。
- 在值欄位中，輸入金鑰值。

[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": "public.ecr.aws/amazonlinux/amazonlinux:latest",
      "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-cluster`。
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. 從系列下拉式清單中選擇 `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。

我們可以透過檢視 CloudWatch 指標、Auto Scaling 群組設定，最後是 Amazon ECS 叢集任務計數，以確認所有項目都正常運作。

檢視叢集的 CloudWatch 指標

1. 透過 <https://console.aws.amazon.com/cloudwatch/> 開啟 CloudWatch 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 `CapacityProviderName, ClusterName`。
6. 選取與 `ConsoleTutorial-cluster` `ClusterName` 對應的核取方塊。
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-cluster。

4. 在 ConsoleTutorial-cluster 頁面上，選擇任務索引標籤，然後依序選擇停止、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-cluster。
7. 在頁面右上角，選擇刪除叢集。
8. 在確認方塊中，輸入 delete ConsoleTutorial-cluster，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 Amazon EC2 容器執行個體

Amazon ECS 容器執行個體是執行 Amazon ECS 容器代理程式並已在叢集中註冊的 Amazon EC2 執行個體。使用容量提供者、外部容量提供者或 Auto Scaling 群組容量提供者透過 Amazon ECS 執行任務時，任務會置放在作用中的容器執行個體上。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 AMIs 是由 AWS 工程師在 Amazon ECS 上預先設定和測試。這是讓您快速上手並在 AWS 上執行容器最簡單的方法。

透過主控台建立叢集時，Amazon ECS 會使用與所選作業系統相關聯的最新 AMI 為執行個體建立啟動範本。

當您使用 CloudFormation 建立叢集時，SSM 參數是 Auto Scaling 群組執行個體的 Amazon EC2 啟動範本的一部分。您可以設定範本，透過動態 Systems Manager 參數來決定要部署的 Amazon ECS 最佳化 AMI。此參數可確保每次部署堆疊時，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 AWS CloudFormation User Guide 中的 [Create an Amazon ECS cluster with the Amazon ECS-optimized Amazon Linux 2023 AMI](#)。

- [擷取 Amazon ECS 最佳化 Linux AMI 中繼資料](#)
- [擷取 Amazon ECS 最佳化 Bottlerocket AMI 中繼資料](#)
- [擷取 Amazon ECS 最佳化 Windows 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。使用較小的執行個體，您能夠以更精細的橫向擴充方式節省成本。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。相反地，您可以建立多個 Auto Scaling 群組，每個群組使用不同的執行個體類型。然後，您可以為這些群組中的每一個建立 Amazon EC2 容量提供者。

依據以下準則，決定要使用的執行個體系列類型與執行個體類型：

- 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。
- 考量包括網路輸送量與儲存空間在內的需求。
- 考量 CPU 與記憶體。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 執行個體與任務可能只會在收到警告兩分鐘後隨即終止。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。

如需有關如何在容器執行個體上設定 Spot 終止通知的資訊，請參閱：

- [設定 Amazon ECS Linux 容器執行個體接收 Spot 執行個體通知](#)
- [設定 Amazon ECS Windows 容器執行個體接收 Spot 執行個體通知](#)

Amazon ECS 最佳化 Linux AMI

Important

Amazon ECS 最佳化 Amazon Linux 2 AMI 將於 2026 年 6 月 30 日終止服務，與上游 Amazon Linux 2 作業系統的相同終止服務日期同步 (如需詳細資訊，請參閱 [Amazon Linux 2 FAQs](#))。建議客戶將應用程式升級至 Amazon Linux 2023，該版本提供直至 2028 年的長期技術支援。如需有關從 Amazon Linux 2 遷移至 Amazon Linux 2023 的資訊，請參閱[從 Amazon Linux 2 Amazon ECS 最佳化 AMI 遷移至 Amazon Linux 2023 Amazon ECS 最佳化 AMI](#)。

根據預設，所有 Amazon ECS 最佳化 AMIs 的棄用日期都會設為 AMI 建立日期的兩年後。您可以使用 Amazon EC2 DescribeImages API 來檢查 AMI 的棄用狀態和日期。如需詳細資訊，請參閱《Amazon Elastic Compute Cloud API 參考》中的 [DescribeImages](#)。

Amazon ECS 會提供根據執行容器工作負載的需求與建議而預先設定的 Amazon ECS 最佳化 AMI。建議為 Amazon EC2 執行個體使用 Amazon ECS 最佳化 Amazon Linux 2023 AMI。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](#)。

透過主控台建立叢集時，Amazon ECS 會使用與所選作業系統相關聯的最新 AMI 為執行個體建立啟動範本。

當您使用 CloudFormation 建立叢集時，SSM 參數是 Auto Scaling 群組執行個體的 Amazon EC2 啟動範本的一部分。您可以設定範本，透過動態 Systems Manager 參數來決定要部署的 Amazon ECS 最佳化 AMI。此參數可確保每次部署堆疊時，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 AWS CloudFormation User Guide 中的 [Create an Amazon ECS cluster with the Amazon ECS-optimized Amazon Linux 2023 AMI](#)。

如需自訂 Amazon ECS 最佳化 AMI，請參閱 GitHub 上的 [Amazon ECS Optimized AMI Build Recipes](#)。

下列 Amazon ECS 最佳化 AMI 的變體適用於搭配 Amazon Linux 2023 作業系統的 Amazon EC2 執行個體。

作業系統	AMI	Description	儲存組態
Amazon Linux 2023	Amazon ECS 最佳化 Amazon Linux 2023 AMI	Amazon Linux 2023 是新一代的 Amazon Linux 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，增加或減少您的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 2023 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 Linux 2023 (arm64)	Amazon ECS 最佳化 Amazon Linux 2023 (arm64) AMI	此 AMI 以 Amazon Linux 2023 為基礎，建議在啟動 Amazon	根據預設，Amazon ECS 最佳化 Amazon Linux 2023 AMI 附有

作業系統	AMI	Description	儲存組態
		EC2 執行個體 (採用 ARM 型 AWS Graviton/Graviton 2/Graviton 3/Graviton 4 處理器的執行個體) 時為 Amazon ECS 工作負載採用。如需詳細資訊，請參閱 Amazon EC2 Instance Types guide 中的 Specifications for the Amazon EC2 general purpose instances 。	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 ECS 最佳化 Amazon Linux 2023 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	Description	儲存組態
Amazon Linux 2023 (Neuron)	Amazon ECS 最佳化 Amazon Linux 2023 AMI	此 AMI 以 Amazon Linux 2023 為基礎，適用於 Amazon EC2 Inf1、Trn1 或 Inf2 執行個體。它預先設定了 AWS Inferentia 和 AWS Trainium 驅動程式，以及 Docker 的 AWS Neuron 執行時間，這使得在 Amazon ECS 上執行機器學習推論工作負載更加輕鬆。如需詳細資訊，請參閱AWS Neuron 機器學習工作負載的 Amazon ECS 任務定義。 Amazon ECS 最佳化 Amazon Linux 2023 (Neuron) AMI 未預先安裝 AWS CLI。	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 ECS 最佳化 Amazon Linux 2023 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	Description	儲存組態
Amazon Linux 2023 GPU	Amazon ECS 最佳化 Amazon Linux 2023 GPU AMI	根據 Amazon Linux 2023，建議在為 Amazon ECS 工作負載啟動 Amazon 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 。	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 ECS 最佳化 Amazon Linux 2023 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。

下列 Amazon ECS 最佳化 AMI 的變體適用於搭配 Amazon Linux 2 作業系統的 Amazon EC2 執行個體。

作業系統	AMI	Description	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	依預設，基於 Amazon Linux 2 的 Amazon ECS 最佳化 AMI (Amazon ECS 最佳化 Amazon Linux 2

作業系統	AMI	Description	儲存組態
		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。	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。

作業系統	AMI	Description	儲存組態
Amazon Linux 2	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。	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。

作業系統	AMI	Description	儲存組態
Amazon Linux 2 (arm64)	Amazon ECS 最佳化 Amazon Linux 2 核心 5.10 (arm64) AMI	以 Amazon Linux 2 為基礎，此 AMI 適用於您的 Amazon EC2 執行個體，其採用 Arm 型 AWS Graviton/Graviton 3/Graviton 4 處理器，而且您想要針對 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 Instance Types guide 中的 Specifications for Amazon EC2 general purpose instances。 Amazon ECS 最佳化 Amazon Linux 2 (arm64) AMI 不會隨附 AWS CLI 預先安裝的。	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。

作業系統	AMI	Description	儲存組態
Amazon Linux 2 (arm64)	Amazon ECS 最佳化 Amazon Linux 2 (arm64) AMI	根據 Amazon Linux 2，此 AMI 適用於啟動 Amazon EC2 執行個體時，這些執行個體採用 Arm 型 AWS Graviton/Graviton 2/Graviton 3/Graviton 4 處理器，適用於 Amazon ECS 工作負載。 Amazon ECS 最佳化 Amazon Linux 2 (arm64) AMI 不會隨附 AWS CLI 預先安裝的。	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。

作業系統	AMI	Description	儲存組態
Amazon Linux 2 (GPU)	Amazon ECS GPU 最佳化核心 5.10 AMI	此 AMI 以 Amazon Linux 2 為基礎，建議在啟動基於 Amazon EC2 GPU 的執行個體時，搭配 Linux 核心 5.10 為 Amazon ECS 工作負載採用。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 。	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。

作業系統	AMI	Description	儲存組態
Amazon Linux 2 (GPU)	Amazon ECS GPU 最佳化 AMI	此 AMI 以 Amazon Linux 2 為基礎，建議在啟動基於 Amazon EC2 GPU 的執行個體時，搭配 Linux 核心 4.14 為 Amazon ECS 工作負載採用。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 。	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。

作業系統	AMI	Description	儲存組態
Amazon Linux 2 (Neuron)	Amazon ECS 最佳化 Amazon Linux 2 (Neuron) 核心 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。它預先設定了 AWS Inferentia 搭配 Linux 核心 5.10 和 AWS Trainium 驅動程式，以及 Docker 的 AWS Neuron 執行期，這使得在 Amazon ECS 上執行機器學習推論工作負載更加輕鬆。如需詳細資訊，請參閱 AWS Neuron 機器學習工作負載的 Amazon ECS 任務定義 。Amazon ECS 最佳化 Amazon Linux 2 (Neuron) AMI 不會隨附 AWS CLI 預先安裝的。	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。

作業系統	AMI	Description	儲存組態
Amazon Linux 2 (Neuron)	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。它預先設定了 AWS Inferentia 和 AWS Trainium 驅動程式，以及 Docker 的 AWS Neuron 執行期，這使得在 Amazon ECS 上執行機器學習推論工作負載更加輕鬆。如需詳細資訊，請參閱 AWS Neuron 機器學習工作負載的 Amazon ECS 任務定義 。Amazon ECS 最佳化 Amazon Linux 2 (Neuron) AMI 不會隨附 AWS CLI 預先安裝的。	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 在 GitHub 上提供了 Amazon ECS 最佳化 AMI 的 Linux 變體變更記錄。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 Parameter Store API，即可擷取每個變體的 AMI 名稱。如需詳細

資訊，請參閱[擷取 Amazon ECS 最佳化 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：

- 在 GitHub 上取得[來源 AMI 發行說明](#)
- Docker 文件中的[Docker 引擎版本備註](#)
- NVIDIA 文件中的[NVIDIA 驅動程式文件](#)
- GitHub 上的[Amazon ECS agent changelog](#)

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 ecs-init 舊版本和封裝，請參閱 GitHub 上的[Amazon ecs-init 變更紀錄](#)

將安全性更新套用至 Amazon ECS 最佳化 AMI

Amazon ECS 最佳化 AMI 以 Amazon Linux 為基礎，包含自訂版本的 cloud-init。Cloud-init 是一種套件，用於在雲端運算環境中引導 Linux 映像，並在啟動執行個體時執行所需的動作。依預設，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，皆會在執行個體啟動時套用所有「關鍵」與「重要」安全性更新。

自 2024 年 6 月 12 日起，以 Amazon Linux 2 為基礎的 Amazon ECS 最佳化 AMI 版本將不再預設在啟動時更新套件。因此，建議在新版 Amazon ECS 最佳化 AMI 發布時，及時更新至最新版本。Amazon ECS 最佳化 AMI 會在有可用安全性更新或基礎 AMI 變更時發行。這將確保您收到最新的套件版本與安全性更新，而且套件版本在執行個體啟動時不可變。如需有關擷取最新 Amazon ECS 最佳化 AMI 的詳細資訊，請參閱[擷取 Amazon ECS 最佳化 Linux AMI 中繼資料](#)。

建議將環境自動化，以便在可用時自動更新至新 AMI。如需有關可用選項的資訊，請參閱[Amazon ECS enables easier EC2 capacity management, with managed instance draining](#)。

若要在 AMI 版本上繼續手動套用「關鍵」與「重要」安全性更新，您可以在 Amazon EC2 執行個體上執行下列命令。

```
yum update --security
```

 Warning

更新 docker 或容器化套件會停止主機上所有執行中的容器，這表示所有執行中的 Amazon ECS 任務都會停止。據以規劃，將服務中斷降至最低。

若要在啟動時重新啟用安全性更新，您可以在啟動 Amazon EC2 執行個體時，將以下一行新增至 cloud-init 使用者資料的 `#cloud-config` 區段。如需詳細資訊，請參閱 Amazon Linux User Guide 中的 [Using cloud-init on Amazon Linux 2](#)。

```
#cloud-config
repo_upgrade: security
```

Amazon ECS 最佳化 AL2023 GPU AMIs 中的版本鎖定套件

某些套件對於 Amazon ECS 最佳化 AL2023 GPU AMIs 中 GPU 功能的正確、高效能行為至關重要。其中包含：

- NVIDIA 驅動程式 (nvidia*)
- 核心模組 (kmod*)
- NVIDIA 程式庫 (libnvidia*)
- 核心套件 (kernel*)

 Note

這並非一份詳盡無遺的清單。鎖定套件的完整清單可透過取得 `dnf versionlock list`

這些套件為版本鎖定，以確保穩定性，並防止可能中斷 GPU 工作負載的意外變更。因此，這些套件通常應該在受管程序的範圍內進行修改，以正常處理潛在問題並維護 GPU 功能。

為了防止意外修改，`dnf versionlock` 外掛程式會用於這些套件。

如果您想要修改鎖定的套件，您可以：

```
# unlock a single package
sudo dnf versionlock delete $PACKAGE_NAME
```

```
# unlock all packages
sudo dnf versionlock clear
```

Important

需要更新這些套件時，客戶應考慮使用包含必要更新的最新 AMI 版本。如果需要更新現有的執行個體，則應採用涉及解鎖、更新和重新鎖定套件的謹慎方法，始終確保在整個過程中維護 GPU 功能。

擷取 Amazon ECS 最佳化 Linux AMI 中繼資料

您能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 執行時期版本。

透過主控台建立叢集時，Amazon ECS 會使用與所選作業系統相關聯的最新 AMI 為執行個體建立啟動範本。

當您使用 CloudFormation 建立叢集時，SSM 參數是 Auto Scaling 群組執行個體的 Amazon EC2 啟動範本的一部分。您可以設定範本，透過動態 Systems Manager 參數來決定要部署的 Amazon ECS 最佳化 AMI。此參數可確保每次部署堆疊時，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 AWS CloudFormation User Guide 中的 [Create an Amazon ECS cluster with the Amazon ECS-optimized Amazon Linux 2023 AMI](#)。

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 2023 (GPU) AMI 中繼資料：

```
/aws/service/ecs/optimized-ami/amazon-linux-2023/gpu/<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/amazon-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 最佳化核心 5.10 AMI 中繼資料：

```
/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 最佳化 Amazon Linux 2 (Neuron) 核心 5.10 AMI 中繼資料：

```
/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` 來擷取最新的建議 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-2017.09.1-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。

擷取最新的建議 Amazon ECS 最佳化 AMI 的中繼資料

您可以使用 AWS CLI 搭配下列 AWS CLI 命令，擷取最新建議的 Amazon ECS 最佳化 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
```

- Amazon ECS 最佳化 Amazon Linux 2023 (Neuron) AMI :

```
aws ssm get-parameters --names /aws/service/ecs/optimized-ami/amazon-linux-2023/  
neuron/recommended --region us-east-1
```

- Amazon ECS 最佳化 Amazon Linux 2023 GPU AMI :

```
aws ssm get-parameters --names /aws/service/ecs/optimized-ami/amazon-linux-2023/gpu/  
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/amazon-linux-2/arm64/  
recommended --region us-east-1
```

- 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
```

- Amazon ECS 最佳化 Amazon Linux 2 (Neuron) 核心 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 AWS CLI 搭配下列 AWS CLI 命令，擷取特定 Amazon ECS 最佳化 Amazon Linux AMI 版本的中繼資料。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
```

利用 Systems Manager GetParametersByPath API 來擷取 Amazon ECS 最佳化 Amazon Linux 2 核心 5.10 AMI 中繼資料

使用 Systems Manager GetParametersByPath API 搭配 AWS CLI 下列命令擷取 Amazon ECS 最佳化 Amazon Linux 2 AMI 中繼資料。

```
aws ssm get-parameters-by-path --path /aws/service/ecs/optimized-ami/amazon-linux-2/
kernel-5.10/ --region us-east-1
```

擷取最新的建議 Amazon ECS 最佳化 Amazon Linux 2 核心 5.10 AMI 的映像 ID

您可以使用子參數 `image_id` 來擷取最新的建議 Amazon ECS 最佳化 Amazon Linux 2 核心 5.10 AMI ID 的映像 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 CloudFormation 範本中使用最新建議的 Amazon ECS 最佳化 AMI

透過參考 Systems Manager 參數存放區名稱，可以在 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
```

從 Amazon Linux 2 遷移至 Amazon Linux 2023 Amazon ECS 最佳化 AMI

遵循 [Amazon Linux](#) 支援政策，Amazon ECS 將於 2026 年 6 月 30 日起終止對 Amazon Linux 2 Amazon ECS 最佳化 AMI 的標準支援。此日期之後，Amazon ECS 代理程式版本將被固定，且僅當來

源 Amazon Linux 2 AMI 進行更新時，才會發布新的 Amazon Linux 2 Amazon ECS 最佳化 AMI。完整終止服務 (EOL) 將與 2026 年 6 月 30 日生效，此後將不再發布 Amazon ECS 最佳化的 Amazon Linux 2 AMI，即使來源 AMI 有更新亦是如此。

Amazon Linux 2023 採用預設安全的方式，具備預先設定的安全政策、處於寬容模式的 SELinux、預設啟用的僅 IMDSv2 模式、最佳化的開機時間以及改善的套件管理功能，可大幅提升安全性與效能。

Amazon Linux 2 與 Amazon Linux 2023 Amazon ECS 最佳化 AMI 之間具有高度相容性，大多數客戶在這兩個作業系統之間遇到的工作負載變更很小，甚至為零。

如需詳細資訊，請參閱 Amazon Linux 2023 User Guide 中的 [Comparing Amazon Linux 2 and Amazon Linux 2023](#)，以及 [AL2023 FAQs](#)。

相容性考量

套件管理與作業系統更新

與舊版本的 Amazon Linux 不同，Amazon ECS 最佳化 Amazon Linux 2023 AMI 會鎖定到特定版本的 Amazon Linux 儲存庫。這可以防止使用者意外更新套件，導致非必要變更或重大變更。如需詳細資訊，請參閱 Amazon Linux 2023 User Guide 中的 [Managing repositories and OS updates in Amazon Linux 2023](#)。

Linux 核心版本

Amazon Linux 2 AMI 以 Linux 核心 4.14 與 5.10 為基礎，而 Amazon Linux 2023 使用 Linux 核心 6.1 與 6.12。如需詳細資訊，請參閱 Amazon Linux 2023 User Guide 中的 [Comparing Amazon Linux 2 and Amazon Linux 2023 kernels](#)。

套件可用性變更

以下是 Amazon Linux 2023 中的重要套件變更：

- Amazon Linux 2 中的部分來源二進位套件在 Amazon Linux 2023 中不再提供。如需詳細資訊，請參閱 Amazon Linux 2023 Release Notes 中的 [Packages removed from Amazon Linux 2023](#)。
- Amazon Linux 支援不同版本套件方式的變更。Amazon Linux 2 中使用的 `amazon-linux-extras` 系統不存在於 Amazon Linux 2023 中。所有套件皆可在「核心」儲存庫中使用。
- Amazon Linux 2023 不支援適用於 Enterprise Linux (EPEL) 的額外套件。如需詳細資訊，請參閱 Amazon Linux 2023 User Guide 中的 [EPEL compatibility in Amazon Linux 2023](#)。
- Amazon Linux 2023 不支援 32 位元應用程式。如需詳細資訊，請參閱 Amazon Linux 2023 User Guide 中的 [Deprecated features from Amazon Linux 2](#)。

控制群組 (cgroups) 變更

控制群組 (cgroup) 是 Linux 核心功能，用於階層式整理程序並為其分配系統資源。控制組廣泛用於執行容器執行期，並由 systemd 執行。

Amazon ECS 代理程式、Docker 與 containerd 都支援 cgroupv1 及 cgroupv2。Amazon ECS 代理程式與容器執行時期會代為管理 cgroup，因此 Amazon ECS 客戶不需要為此底層 cgroup 升級進行任何變更。

如需有關 cgroupv2 的更多詳細資訊，請參閱 Amazon Linux 2023 User Guide 中的 [Control groups v2 in Amazon Linux 2023](#)。

執行個體中繼資料服務 (IMDS) 變更

Amazon Linux 2023 預設需要執行個體中繼資料服務第 2 版 (IMDSv2)。IMDSv2 有多項優勢，有助於改善安全狀態。它使用面向工作階段的身分驗證方法，需要透過一個簡單的 HTTP PUT 請求來建立一個秘密權杖以啟動工作階段。工作階段的權杖有效期可設定為 1 秒至 6 小時之間的任何時間。

如需有關從 IMDSv1 轉換至 IMDSv2 的詳細資訊，請參閱 Amazon EC2 User Guide 中的 [Transition to using Instance Metadata Service Version 2](#)。

若您仍想使用 IMDSv1，可以透過使用執行個體中繼資料選項啟動屬性，手動覆寫設定來實現。

記憶體交換積極性變更

Amazon Linux 2023 與 cgroups v2 不支援容器層級的記憶體交換積極性。如需詳細資訊，請參閱在 [Amazon ECS 上管理容器交換記憶體空間](#)。

FIPS 驗證變更

Amazon Linux 2 已通過 FIPS 140-2 認證，Amazon Linux 2023 已通過 FIPS 140-3 認證。

若要在 Amazon Linux 2023 上啟用 FIPS 模式，需在 Amazon EC2 執行個體上安裝必要套件，並遵循 Amazon Linux 2023 User Guide 中 [Enable FIPS Mode on Amazon Linux 2023](#) 的說明執行設定步驟。

加速執行個體支援

Amazon ECS 最佳化 Amazon Linux 2023 AMI 同時支援 Neuron 與 GPU 兩種加速執行個體類型。如需詳細資訊，請參閱 [Amazon ECS 最佳化 Linux AMI](#)。

建置自訂 AMI

雖然建議遷移至官方支援與發布的 Amazon Linux 2023 Amazon ECS 最佳化 AMI，但也可透過用於建置 Amazon ECS 最佳化 AMI Linux 變體的開放原始碼建置指令碼，繼續建置自訂的 Amazon Linux 2 Amazon ECS 最佳化 AMI。如需詳細資訊，請參閱[Amazon ECS 最佳化 Linux AMI 建置指令碼](#)。

遷移策略

建議建立並實作包含完整應用程式測試的遷移計畫。以下各節根據您管理 Amazon ECS 基礎結構的方式概述不同的遷移策略。

透過 Amazon ECS 容量提供者進行遷移

1. 建立使用新啟動範本的新容量提供者。這個容量提供者應參考一個 Auto Scaling 群組，其啟動範本與您現有的類似，但需指定其中一種 Amazon Linux 2023 變體，而非 Amazon Linux 2 Amazon ECS 最佳化 AMI。將此新容量提供者新增至現有的 Amazon ECS 叢集。
2. 更新叢集的預設容量提供者策略，使其同時包含現有的 Amazon Linux 2 容量提供者與新的 Amazon Linux 2023 容量提供者。初始權重設定應以 Amazon Linux 2 提供者為主 (例如權重 80)，Amazon Linux 2023 提供者為輔 (例如權重 20)。這會使 Amazon ECS 在排程新任務時，開始佈建 Amazon Linux 2023 執行個體。確認這些執行個體能正確註冊，且任務能在新的執行個體上順利執行。
3. 隨著時間推移逐漸調整叢集預設策略中的容量提供者權重，增加 Amazon Linux 2023 提供者的權重，同時減少 Amazon Linux 2 提供者權重 (例如從 60/40 調整為 40/60，然後是 20/80)。您也可以更新個別服務容量提供者策略，排定 Amazon Linux 2023 執行個體的優先順序。監控任務置放，確保任務在 Amazon Linux 2023 執行個體上順利執行。
4. 選擇性地排空 Amazon Linux 2 容器執行個體，加速實現任務遷移。如果您有足夠的 Amazon Linux 2023 替換容量，您可以透過 Amazon ECS 主控台手動耗盡 Amazon Linux 2 容器執行個體 AWS CLI，或加速將任務從 Amazon Linux 2 轉移至 Amazon Linux 2023。遷移完成後，從叢集中移除 Amazon Linux 2 容量提供者，同時刪除相關聯的 Auto Scaling 群組。

透過 Amazon EC2 Auto Scaling 群組進行遷移

1. 建立使用新啟動範本的新 Amazon EC2 Auto Scaling 群組。這應該與您現有的啟動範本類似，但需指定其中一種 Amazon Linux 2023 變體，而非 Amazon Linux 2 Amazon ECS 最佳化 AMI。這個新建的 Auto Scaling 群組可將執行個體啟動至現有的叢集。
2. 向上擴展 Auto Scaling 群組，使 Amazon Linux 2023 執行個體開始向叢集進行註冊。確認這些執行個體能正確註冊，且任務能在新的執行個體上順利執行。

3. 確認任務在 Amazon Linux 2023 上運作後，請向上擴展 Amazon Linux 2023 Auto Scaling 群組，同時逐步縮減 Amazon Linux 2 Auto Scaling 群組規模，直至完全替換所有 Amazon Linux 2 執行個體為止。
4. 如果有足夠的 Amazon Linux 2023 替代容量，建議您明確排空容器執行個體，加速將任務從 Amazon Linux 2 轉換至 Amazon Linux 2023。如需詳細資訊，請參閱[排空 Amazon ECS 容器執行個體](#)。

透過手動受管執行個體進行遷移

1. 透過 Amazon ECS 最佳化 Amazon Linux 2023 AMI 而非 Amazon Linux 2 手動啟動 (或調整啟動的指令碼) 新的 Amazon EC2 執行個體。確保這些執行個體採用與現有 Amazon Linux 2 執行個體相同的安全群組、子網路、IAM 角色與叢集組態。執行個體應在啟動時自動註冊至現有的 Amazon ECS 叢集。
2. 確認新的 Amazon Linux 2023 執行個體已成功註冊至 Amazon ECS 叢集且處於 ACTIVE 狀態。透過等待自然任務置放或手動停止/啟動部分任務來觸發重新排程，測試任務是否可以排程並在這些新執行個體上正常執行。
3. 透過隨需啟動額外的 Amazon Linux 2023 執行個體，逐步取代 Amazon Linux 2 執行個體，接著手動逐台排空並終止 Amazon Linux 2 執行個體。您可以透過 Amazon ECS 主控台將執行個體設定為 DRAINING 狀態來排空執行個體，這會停止在其中置放新任務，並允許現有任務完成或在別處重新排程。

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。這些建置指令碼現可於 GitHub 取得。如需詳細資訊，請參閱 GitHub 上的 [amazon-ecs-ami](#)。

如需自訂 Amazon ECS 最佳化 AMI，請參閱 GitHub 上的 [Amazon ECS Optimized AMI Build Recipes](#)。

此建置指令碼儲存庫包含 [HashiCorp Packer](#) 範本及用於產生 Amazon ECS 最佳化 AMI 每個 Linux 變數的建置指令碼。這些指令碼是 Amazon ECS 最佳化 AMI 建置的真實來源，因此您可遵循 GitHub 儲存庫來監控我們 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。

如需詳細資訊，請參閱 GitHub 上的 Amazon ECS AMI 儲存庫，網址為 [aws/amazon-ecs-ami](#)。

建置 Amazon ECS 最佳化 Linux AMI

1. 複製 aws/amazon-ecs-ami GitHub 儲存庫。

```
git clone https://github.com/aws/amazon-ecs-ami.git
```

2. 為建立 AMI 時要使用 AWS 的區域新增環境變數。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 a12
```

- Amazon ECS 最佳化 Amazon Linux 2 (arm64) AMI

```
make a12arm
```

- Amazon ECS GPU 最佳化 AMI

```
make a12gpu
```

- Amazon ECS 最佳化 Amazon Linux 2 (Neuron) AMI

```
make a12inf
```

- Amazon ECS 最佳化 Amazon Linux 2023 AMI

```
make a12023
```

- Amazon ECS 最佳化 Amazon Linux 2023 (arm64) AMI

```
make a12023arm
```

- Amazon ECS 最佳化 Amazon Linux 2023 GPU AMI

```
make a12023gpu
```

- Amazon ECS 最佳化 Amazon Linux 2023 (Neuron) AMI


```
make al2023neu
```

Amazon ECS 最佳化 Bottlerocket AMI

Bottlerocket 是以 Linux 為基礎的開放原始碼作業系統，專為在虛擬機器或裸機主機上執行容器 AWS 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](#)。

有多種 Amazon ECS 最佳化 Bottlerocket AMI 核心 6.1 與核心 5.10 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 5.10：

- 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，您可以使用額外管理工具取得 SSH 管理員存取權並執行啟動引導。

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引的詳細資訊，請參閱 GitHub 上的[安全性功能](#)和[安全性指引](#)。

- 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。
- Bottlerocket AMI 版本 1.19.0 或更新版本支援 initProcessEnabled 任務定義參數。
- Bottlerocket AMI 也不支援下列服務和功能：
 - ECS Anywhere
 - Service Connect
 - Amazon EFS 處於加密模式
 - awsvpc 網路模式中的 Amazon EFS
 - 無法掛載 Amazon EBS 磁碟區
 - Elastic Inference 加速器

擷取 Amazon ECS 最佳化 Bottlerocket AMI 中繼資料

您可以查詢 AWS Systems Manager 參數存放區 API，擷取 Amazon ECS 最佳化 AMIs) ID。若使用此參數，您無需手動查詢 Amazon ECS 最佳化 AMI ID。如需 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 Bottlerocket AMI 變體

您可以透過 AWS 區域 和使用 AWS CLI 或 的架構，擷取最新的穩定 aws-ecs-2 Bottlerocket AMI 變體 AWS 管理主控台。

- AWS CLI – 您可以使用子參數，使用以下 AWS CLI 命令擷取最新建議的 Amazon ECS 最佳化 Bottlerocket AMI 影像 ID `image_id`。將 *region* 替換為您需要 AMI ID 的區域代碼。

如需支援 AWS 區域的相關資訊，請參閱 GitHub 上的[尋找 AMI](#)。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 管理主控台：您可以使用 AWS 管理主控台中的 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 支援的資訊 AWS 區域，請參閱[在 GitHub 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 Bottlerocket AMI 變體

您可以使用 AWS CLI 或，依區域和架構擷取最新的穩定 aws-ecs-2-nvidia Bottlerocket AMI 變體 AWS 管理主控台。

- AWS CLI – 您可以使用子參數，使用以下 AWS CLI 命令擷取最新建議的 Amazon ECS 最佳化 Bottlerocket AMI 的影像 ID `image_id`。將 *region* 替換為您需要 AMI ID 的區域代碼。

如需支援 AWS 區域的相關資訊，請參閱 GitHub 上的[尋找 AMI](#)。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 管理主控台：您可以使用 AWS 管理主控台中的 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。

如需支援的資訊 AWS 區域，請參閱[在 GitHub 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 Bottlerocket AMI 變體

您可以使用 AWS CLI 或擷取最新的穩定 `aws-ecs-1` Bottlerocket AMI 變體 AWS 區域 和架構 AWS 管理主控台。

- AWS CLI – 您可以使用子參數，使用以下 AWS CLI 命令擷取最新建議的 Amazon ECS 最佳化 Bottlerocket AMI 影像 ID `image_id`。將 *region* 替換為您需要 AMI ID 的區域代碼。

如需支援 AWS 區域的相關資訊，請參閱 GitHub 上的[尋找 AMI](#)。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 管理主控台：您可以使用 AWS 管理主控台內的 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。

如需支援的資訊 AWS 區域，請參閱[在 GitHub 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 Bottlerocket AMI 變體

您可以使用 AWS CLI 或 [AWS 管理主控台](#)，依區域和架構擷取最新的穩定 aws-ecs-1-nvidia Bottlerocket AMI 變體。

- AWS CLI – 您可以使用子參數 `--image-id`，使用以下 AWS CLI 命令擷取最新建議的 Amazon ECS 最佳化 Bottlerocket AMI 的影像 ID `image_id`。將 *region* 替換為您需要 AMI ID 的區域代碼。

如需支援的資訊 AWS 區域，請參閱[在 GitHub 上尋找 AMI](#)。GitHub

- 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 管理主控台：您可以使用 AWS 管理主控台中的 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。

如需支援的資訊 AWS 區域，請參閱[在 GitHub 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

如需如何在 Amazon ECS 上開始使用 Bottlerocket 作業系統的詳細教學課程，請參閱 GitHub 上的 [Using a Bottlerocket AMI with Amazon ECS](#)，以及 AWS 部落格網站上的 [Getting started with Bottlerocket and Amazon ECS](#)。

如需有關 Bottlerocket 執行個體啟動方式的資訊，請參閱[啟動 Amazon ECS 的 Bottlerocket 執行個體](#)

啟動 Amazon ECS 的 Bottlerocket 執行個體

您可以啟動 Bottlerocket 執行個體，以便執行容器工作負載。

您可以使用 AWS CLI 來啟動 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 最佳化 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 作業系統的詳細演練，請參閱 GitHub 上的 [搭配使用 Bottlerocket AMI 與 Amazon ECS](#)，以及 AWS 部落格網站上的開始使用 [Bottlerocket 和 Amazon ECS](#)。

Amazon ECS Linux 容器執行個體管理

在將 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 容器執行個體傳遞資料](#)
- [設定 Amazon ECS Linux 容器執行個體接收 Spot 執行個體通知](#)
- [啟動 Amazon ECS Linux 容器執行個體時執行指令碼](#)
- [增加 Amazon ECS Linux 容器執行個體網路介面數量](#)
- [保留 Amazon ECS Linux 容器執行個體記憶體](#)
- [使用 從遠端管理 Amazon ECS 容器執行個體 AWS Systems Manager](#)
- [為 Amazon ECS Linux 容器執行個體使用 HTTP Proxy](#)
- [為 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 18.01.0 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 Amazon ECS 容器執行個體。

您可以透過各種方法啟動執行個體 AWS CLI，包括 Amazon EC2 主控台和 SDK。如需有關啟動執行個體其他方法的資訊，請參閱 Amazon EC2 User Guide 中的 [Launch your instance](#)。

如需有關啟動精靈的詳細資訊，請參閱 Amazon EC2 User Guide 中的 [Launch an instance using the new launch instance wizard](#)。

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 \(登入\)](#)
- [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 發佈的適當 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 [Amazon ECS 最佳化 Linux AMI](#) 中概述的要求。

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《Amazon EC2 使用者指南》中的[執行個體類型](#)。如果您想要執行純 IPv6 工作負載，某些執行個體類型不支援 IPv6 位址。如需詳細資訊，請參閱《Amazon EC2 使用者指南》中的[IPv6 地址](#)。

- 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。

網路設定

選擇表單的網路設定區段的編輯按鈕後，視需要設定網路設定。

- 在 VPC 欄位中，選擇執行個體啟動所在的 VPC。若要執行純 IPv6 工作負載，請選擇同時包含 IPv4 與 IPv6 CIDR 區塊的雙堆疊 VPC。
- 在子網路欄位中，請選擇要在其中啟動執行個體的字網路。您可以在與可用區域、Local Zone、Wavelength Zone 或 Outpost 相關聯的子網路中啟動執行個體。

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字網路。若要建立新的子網路，請選擇建立新的子網路，前往 Amazon VPC 主控台。完成後，請返回啟動執行個體精靈並選擇「重新整理」圖示，載入清單中的子網路。

在 Local Zone 中啟動執行個體，選取您在 Local Zone 中建立的字網路。

若要在 Outpost 中啟動執行個體，請在與 Outpost 相關聯的 VPC 中選取子網路。

若要執行純 IPv6 工作負載，請選擇僅包含 IPv6 CIDR 區塊的字網路。

- 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)。

i 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 [為 Amazon ECS Linux 容器執行個體使用 HTTP Proxy](#)。

- 如果選擇雙堆疊 VPC 與純 IPv6 子網路，請在自動指派 IPv6 IP 欄位中選取啟用。

- 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。
- 如果要為純 IPv6 工作負載啟動執行個體，請選擇進階網路組態，然後在指派主要 IPv6 IP 欄位中選擇是。

Note

如果沒有主要 IPv6 位址，在容器執行個體上以主機或橋接網路模式執行的任務，將無法向負載平衡器或 AWS Cloud Map 註冊。

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](#)。

- 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，將下列指令碼貼到使用者資料欄位中，將 *your_bucket_name* 取代為儲存貯體的名稱，以安裝 AWS CLI 並在啟動時寫入您的組態檔案。

Note

如需此組態的詳細資訊，請參閱「[將 Amazon ECS 容器執行個體組態儲存於 Amazon S3 中](#)」。

```
#!/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
```

- 依預設，Amazon ECS 容器代理程式會透過偵測執行個體的預設 IPv4 與 IPv6 路由，判斷容器執行個體是否相容純 IPv6 組態。若要覆寫此行為，您可以在執行個體的 `/etc/ecs/ecs.config` 檔案中將 ECS_INSTANCE_IP_COMPATIBILITY 參數設定為 `ipv4` 或 `ipv6`。

```
#!/bin/bash
cat <<'EOF' >> /etc/ecs/ecs.config
ECS_CLUSTER=your_cluster_name
ECS_INSTANCE_IP_COMPATIBILITY=ipv6
EOF
```

如需詳細資訊，請參閱[引導 Amazon ECS Linux 容器執行個體傳遞資料](#)。

引導 Amazon ECS Linux 容器執行個體傳遞資料

當您啟動 Amazon EC2 執行個體時，可以將使用者資料傳送到 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 啟動精靈時傳遞此使用者資料，請參閱[啟動 Amazon ECS Linux 容器執行個體](#)。

您可以設定容器執行個體，在容器代理程式組態或 Docker 常駐程式組態中傳遞資料。

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](#)。

Note

AWS 不支援自訂 Docker 組態，因為它們有時可能會與未來的 Amazon ECS 變更或功能衝突，而不會出現警告。

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
```

設定 Amazon ECS Linux 容器執行個體接收 Spot 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。如果在執行個體上啟用 Amazon ECS Spot 執行個體排空，則 Amazon 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

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。

Spot 執行個體排空預設為停用。

您可以在啟動執行個體時啟用 Spot 執行個體排空。將下列指令碼新增至使用者資料欄位。將 *MyCluster* 取代為叢集名稱，以完成容器執行個體註冊。

```
#!/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 “容器自我檢查”](#)。

```
curl http://localhost:51678/v1/metadata
```

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，以處理營運或安全性相關事項，例如監控、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任務。

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。如需詳細資訊，請參閱 AWS Identity and Access Management User Guide 中的 [Update permissions for a role](#)。
2. 使用 Amazon ECS 最佳化 Amazon Linux 2 AMI 啟動一或多個容器執行個體。啟動新的容器執行個體，並在 EC2 使用者資料中使用如下範例指令碼。將 `your_cluster_name` 取代為容器執行個體要註冊的目標叢集，並將 `my_task_def` 取代為要在啟動時於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 容器執行個體網路介面數量

Note

此功能不適用於 Fargate。

每個使用 `awsvpc` 網路模式的任務都會收到專用的彈性網路介面 (ENI)，該介面會連接至託管此任務的容器執行個體。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介面計算在內，所

以您可以再連接兩個 ENI 到執行個體。因為每個使用 `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。

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。當您使用這些執行個體類型並啟用 `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。若要使用主控台啟用此功能，請參閱[修改 Amazon ECS 帳戶設定](#)。若要使用 AWS CLI 開啟功能，請參閱[使用 管理 Amazon ECS 帳戶 設定 AWS CLI](#)。

例如，啟用 `awsvpcTrunking` 的 `c5.large` 執行個體，其 ENI 限制會增加至 12 個。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[EC2 的 Amazon ECS 任務聯網選項](#)。

考量事項

使用 ENI 主幹功能時，建議考量下列事項。

- 只有 Amazon ECS 最佳化 AMI 的 Linux 變體，或具有 1.28.1 版本或更新版本容器代理程式和版本 1.28.1-2 或更新版本 `ecs-init` 套件的其他 Amazon Linux 變體，才支援更高的 ENI 限制。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 `awsvpcTrunking` 後啟動的新 Amazon EC2 執行個體，才會收到更高的 ENI 限制和幹線網路介面。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 Amazon EC2 主控台中建立新執行個體時，請清除啟用資源型 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。
- 您的任務必須使用 `awsvpc` 網路模式和 EC2。使用 Fargate 的任務，會一律收到專用的 ENI (不論啟動多少)，因此不需要此功能。
- 您的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。容器執行個體成功取消註冊或終止後，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。
- 如果具有更高 ENI 限制之公有子網路中的容器執行個體停止再重新啟動，執行個體會遺失其公有 IP 位址，且容器代理程式將失去其連線。
- 啟用 `awsvpcTrunking` 時，容器執行個體會收到使用 VPC 預設安全群組的額外 ENI，並由 Amazon ECS 管理。

預設 VPC 在每個可用區域附帶一個公有子網路、一個網際網路閘道，以及啟用 DNS 解析的設定。子網路是公有子網路，因為主路由表會將以網際網路為目標的子網路流量傳送至網際網路閘道。您可將路由從目標 `0.0.0.0/0` 移至網際網路閘道，將預設子網路變成私有子網路。但若如此做，在該子網路中執行的任何容器執行個體都無法存取網際網路。您可以新增或刪除安全群組規則，控制進出子網路的流量。如需詳細資訊，請參閱 Amazon Virtual Private Cloud User Guide 中的 [Security group rules](#)。

先決條件

在啟動具有更高 ENI 限制的容器執行個體之前，必須完成下列先決條件。

- 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 提供代表您呼叫其他 AWS 服務的許可。這個角色會在您建立叢集，或在 AWS 管理主控台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。我們建議您建立 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 主幹功能的任務。如需有關容器執行個體的詳細資訊，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受支援的 Amazon EC2 執行個體類型來啟動新容器執行個體，且執行個體將具有更高的 ENI 限制。如需支援的執行個體類型清單，請參閱 [援更多 Amazon ECS 容器網路介面的執行個體](#)。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
```

若要使用 檢視具有更高 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 ecs.aws-vpc-trunk-id 屬性來列出具有更高 ENI 限制的容器執行個體，該屬性指出其具有幹線網路介面。

- [list-attributes](#) (AWS CLI)

```
aws ecs list-attributes \
    --target-type container-instance \
    --attribute-name ecs.aws-vpc-trunk-id \
    --cluster cluster_name \
    --region us-east-1
```

- [Get-ECSAttributeList](#) (AWS Tools for Windows PowerShell)

```
Get-ECSAttributeList -TargetType container-instance -AttributeName ecs.aws-vpc-trunk-id -Region us-east-1
```

援更多 Amazon ECS 容器網路介面的執行個體

下列顯示受支援的 Amazon EC2 執行個體類型，以及在啟用 aws-vpc-Trunking 帳戶設定的前後，可以在每個執行個體類型上啟動使用 aws-vpc 網路模式的任務數量。

⚠ 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.24xlarge 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、r5t 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	未啟用 ENI 主幹功能時的任務數量上限	啟用 ENI 主幹功能時的任務數量上限
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

執行個體類型	未啟用 ENI 主幹功能時的 任務數量上限	啟用 ENI 主幹功能時的任務數量上限
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
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

執行個體類型	未啟用 ENI 主幹功能時的 任務數量上限	啟用 ENI 主幹功能時的任務數量上限
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
m5d.24xlarge	14	120
m5d.metal	14	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
m6a.16xlarge	14	120
m6a.24xlarge	14	120
m6a.32xlarge	14	120

執行個體類型	未啟用 ENI 主幹功能時的 任務數量上限	啟用 ENI 主幹功能時的任務數量上限
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
m6gd.12xlarge	7	60
m6gd.16xlarge	14	120
m6gd.metal	14	120

執行個體類型	未啟用 ENI 主幹功能時的 任務數量上限	啟用 ENI 主幹功能時的任務數量上限
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
m6id.24xlarge	14	120
m6id.32xlarge	14	120
m6id.metal	14	120

執行個體類型	未啟用 ENI 主幹功能時的 任務數量上限	啟用 ENI 主幹功能時的任務數量上限
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
m6in.24xlarge	14	120
m6in.32xlarge	15	120
m6in.metal	15	120

執行個體類型	未啟用 ENI 主幹功能時的 任務數量上限	啟用 ENI 主幹功能時的任務數量上限
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
m7g.8xlarge	7	60
m7g.12xlarge	7	60
m7g.16xlarge	14	120

執行個體類型	未啟用 ENI 主幹功能時的 任務數量上限	啟用 ENI 主幹功能時的任務數量上限
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
m7gd.metal	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
m7i.24xlarge	14	120
m7i.48xlarge	14	120
m7i.metal-24xl	14	120

執行個體類型	未啟用 ENI 主幹功能時的 任務數量上限	啟用 ENI 主幹功能時的任務數量上限
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
m7i-flex.12xlarge	7	120
m7i-flex.16xlarge	14	120
m8a.medium	1	4
m8a.large	2	10
m8a.xlarge	3	20
m8a.2xlarge	3	40
m8a.4xlarge	7	60
m8a.8xlarge	9	90
m8a.12xlarge	11	120
m8a.16xlarge	15	120
m8a.24xlarge	15	120
m8a.48xlarge	23	120
m8a.metal-24xl	15	120
m8a.metal-48xl	23	120

執行個體類型	未啟用 ENI 主幹功能時的 任務數量上限	啟用 ENI 主幹功能時的任務數量上限
m8azn.medium	2	4
m8azn.large	3	10
m8azn.xlarge	3	20
m8azn.3xlarge	7	40
m8azn.6xlarge	7	60
m8azn.12xlarge	15	120
m8azn.24xlarge	15	120
m8azn.metal-12xl	15	120
m8azn.metal-24xl	15	120
m8g.medium	1	4
m8g.large	2	10
m8g.xlarge	3	20
m8g.2xlarge	3	40
m8g.4xlarge	7	60
m8g.8xlarge	7	60
m8g.12xlarge	7	60
m8g.16xlarge	14	120
m8g.24xlarge	14	120
m8g.48xlarge	14	120
m8g.metal-24xl	14	120

執行個體類型	未啟用 ENI 主幹功能時的 任務數量上限	啟用 ENI 主幹功能時的任務數量上限
m8g.metal-48xl	14	120
m8gb.medium	1	4
m8gb.large	2	10
m8gb.xlarge	3	20
m8gb.2xlarge	3	40
m8gb.4xlarge	7	60
m8gb.8xlarge	9	60
m8gb.12xlarge	11	60
m8gb.16xlarge	15	120
m8gb.24xlarge	23	120
m8gb.48xlarge	23	120
m8gb.metal-24xl	23	120
m8gb.metal-48xl	23	120
m8gd.medium	1	4
m8gd.large	2	10
m8gd.xlarge	3	20
m8gd.2xlarge	3	40
m8gd.4xlarge	7	60
m8gd.8xlarge	7	60
m8gd.12xlarge	7	60

執行個體類型	未啟用 ENI 主幹功能時的 任務數量上限	啟用 ENI 主幹功能時的任務數量上限
m8gd.16xlarge	14	120
m8gd.24xlarge	14	120
m8gd.48xlarge	14	120
m8gd.metal-24xl	14	120
m8gd.metal-48xl	14	120
m8gn.medium	1	4
m8gn.large	2	10
m8gn.xlarge	3	20
m8gn.2xlarge	3	40
m8gn.4xlarge	7	60
m8gn.8xlarge	9	60
m8gn.12xlarge	11	60
m8gn.16xlarge	15	120
m8gn.24xlarge	23	120
m8gn.48xlarge	23	120
m8gn.metal-24xl	23	120
m8gn.metal-48xl	23	120
m8i.large	2	10
m8i.xlarge	3	20
m8i.2xlarge	3	40

執行個體類型	未啟用 ENI 主幹功能時的 任務數量上限	啟用 ENI 主幹功能時的任務數量上限
m8i.4xlarge	7	60
m8i.8xlarge	9	90
m8i.12xlarge	11	120
m8i.16xlarge	15	120
m8i.24xlarge	15	120
m8i.32xlarge	23	120
m8i.48xlarge	23	120
m8i.96xlarge	23	120
m8i.metal-48xl	23	120
m8i.metal-96xl	23	120
m8id.large	2	10
m8id.xlarge	3	20
m8id.2xlarge	3	40
m8id.4xlarge	7	60
m8id.8xlarge	9	90
m8id.12xlarge	11	120
m8id.16xlarge	15	120
m8id.24xlarge	15	120
m8id.32xlarge	23	120
m8id.48xlarge	23	120

執行個體類型	未啟用 ENI 主幹功能時的 任務數量上限	啟用 ENI 主幹功能時的任務數量上限
m8id.96xlarge	23	120
m8id.metal-48xl	23	120
m8id.metal-96xl	23	120
m8i-flex.large	2	4
m8i-flex.xlarge	3	10
m8i-flex.2xlarge	3	20
m8i-flex.4xlarge	7	40
m8i-flex.8xlarge	9	60
m8i-flex.12xlarge	11	120
m8i-flex.16xlarge	15	120
mac2.metal	7	12
mac2-m1ultra.metal	7	12
mac2-m2.metal	7	12
mac2-m2pro.metal	7	12
mac-m4.metal	7	12
mac-m4pro.metal	7	12
mac-m4max.metal	7	12

運算最佳化

執行個體類型	未啟用 ENI 主幹功能時的 任務數量上限	啟用 ENI 主幹功能時的任務數量上限
c5.large	2	10
c5.xlarge	3	20
c5.2xlarge	3	40
c5.4xlarge	7	60
c5.9xlarge	7	60
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

執行個體類型	未啟用 ENI 主幹功能時的 任務數量上限	啟用 ENI 主幹功能時的任務數量上限
c5ad.12xlarge	7	60
c5ad.16xlarge	14	120
c5ad.24xlarge	14	120
c5d.large	2	10
c5d.xlarge	3	20
c5d.2xlarge	3	40
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

執行個體類型	未啟用 ENI 主幹功能時的 任務數量上限	啟用 ENI 主幹功能時的任務數量上限
c6a.48xlarge	14	120
c6a.metal	14	120
c6g.medium	1	4
c6g.large	2	10
c6g.xlarge	3	20
c6g.2xlarge	3	40
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

執行個體類型	未啟用 ENI 主幹功能時的 任務數量上限	啟用 ENI 主幹功能時的任務數量上限
c6gn.medium	1	4
c6gn.large	2	10
c6gn.xlarge	3	20
c6gn.2xlarge	3	40
c6gn.4xlarge	7	60
c6gn.8xlarge	7	60
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

執行個體類型	未啟用 ENI 主幹功能時的 任務數量上限	啟用 ENI 主幹功能時的任務數量上限
c6id.2xlarge	3	40
c6id.4xlarge	7	60
c6id.8xlarge	7	90
c6id.12xlarge	7	120
c6id.16xlarge	14	120
c6id.24xlarge	14	120
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

執行個體類型	未啟用 ENI 主幹功能時的 任務數量上限	啟用 ENI 主幹功能時的任務數量上限
c7a.xlarge	3	20
c7a.2xlarge	3	40
c7a.4xlarge	7	60
c7a.8xlarge	7	90
c7a.12xlarge	7	120
c7a.16xlarge	14	120
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

執行個體類型	未啟用 ENI 主幹功能時的 任務數量上限	啟用 ENI 主幹功能時的任務數量上限
c7gd.large	2	10
c7gd.xlarge	3	20
c7gd.2xlarge	3	40
c7gd.4xlarge	7	60
c7gd.8xlarge	7	60
c7gd.12xlarge	7	60
c7gd.16xlarge	14	120
c7gd.metal	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
c7gn.metal	14	120
c7i.large	2	10
c7i.xlarge	3	20
c7i.2xlarge	3	40

執行個體類型	未啟用 ENI 主幹功能時的 任務數量上限	啟用 ENI 主幹功能時的任務數量上限
c7i.4xlarge	7	60
c7i.8xlarge	7	90
c7i.12xlarge	7	120
c7i.16xlarge	14	120
c7i.24xlarge	14	120
c7i.48xlarge	14	120
c7i.metal-24xl	14	120
c7i.metal-48xl	14	120
c7i-flex.large	2	4
c7i-flex.xlarge	3	10
c7i-flex.2xlarge	3	20
c7i-flex.4xlarge	7	40
c7i-flex.8xlarge	7	60
c7i-flex.12xlarge	7	120
c7i-flex.16xlarge	14	120
c8a.medium	1	4
c8a.large	2	10
c8a.xlarge	3	20
c8a.2xlarge	3	40
c8a.4xlarge	7	60

執行個體類型	未啟用 ENI 主幹功能時的 任務數量上限	啟用 ENI 主幹功能時的任務數量上限
c8a.8xlarge	9	90
c8a.12xlarge	11	120
c8a.16xlarge	15	120
c8a.24xlarge	15	120
c8a.48xlarge	23	120
c8a.metal-24xl	15	120
c8a.metal-48xl	23	120
c8g.medium	1	4
c8g.large	2	10
c8g.xlarge	3	20
c8g.2xlarge	3	40
c8g.4xlarge	7	60
c8g.8xlarge	7	60
c8g.12xlarge	7	60
c8g.16xlarge	14	120
c8g.24xlarge	14	120
c8g.48xlarge	14	120
c8g.metal-24xl	14	120
c8g.metal-48xl	14	120
c8gb.medium	1	4

執行個體類型	未啟用 ENI 主幹功能時的 任務數量上限	啟用 ENI 主幹功能時的任務數量上限
c8gb.large	2	10
c8gb.xlarge	3	20
c8gb.2xlarge	3	40
c8gb.4xlarge	7	60
c8gb.8xlarge	9	60
c8gb.12xlarge	11	60
c8gb.16xlarge	15	120
c8gb.24xlarge	23	120
c8gb.48xlarge	23	120
c8gb.metal-24xl	23	120
c8gb.metal-48xl	23	120
c8gd.medium	1	4
c8gd.large	2	10
c8gd.xlarge	3	20
c8gd.2xlarge	3	40
c8gd.4xlarge	7	60
c8gd.8xlarge	7	60
c8gd.12xlarge	7	60
c8gd.16xlarge	14	120
c8gd.24xlarge	14	120

執行個體類型	未啟用 ENI 主幹功能時的 任務數量上限	啟用 ENI 主幹功能時的任務數量上限
c8gd.48xlarge	14	120
c8gd.metal-24xl	14	120
c8gd.metal-48xl	14	120
c8gn.medium	1	4
c8gn.large	2	10
c8gn.xlarge	3	20
c8gn.2xlarge	3	40
c8gn.4xlarge	7	60
c8gn.8xlarge	9	60
c8gn.12xlarge	11	60
c8gn.16xlarge	15	120
c8gn.24xlarge	23	120
c8gn.48xlarge	23	120
c8gn.metal-24xl	23	120
c8gn.metal-48xl	23	120
c8i.large	2	10
c8i.xlarge	3	20
c8i.2xlarge	3	40
c8i.4xlarge	7	60
c8i.8xlarge	9	90

執行個體類型	未啟用 ENI 主幹功能時的 任務數量上限	啟用 ENI 主幹功能時的任務數量上限
c8i.12xlarge	11	120
c8i.16xlarge	15	120
c8i.24xlarge	15	120
c8i.32xlarge	23	120
c8i.48xlarge	23	120
c8i.96xlarge	23	120
c8i.metal-48xl	23	120
c8i.metal-96xl	23	120
c8id.large	2	10
c8id.xlarge	3	20
c8id.2xlarge	3	40
c8id.4xlarge	7	60
c8id.8xlarge	9	90
c8id.12xlarge	11	120
c8id.16xlarge	15	120
c8id.24xlarge	15	120
c8id.32xlarge	23	120
c8id.48xlarge	23	120
c8id.96xlarge	23	120
c8id.metal-48xl	23	120

執行個體類型	未啟用 ENI 主幹功能時的 任務數量上限	啟用 ENI 主幹功能時的任務數量上限
c8id.metal-96xl	23	120
c8i-flex.large	2	4
c8i-flex.xlarge	3	10
c8i-flex.2xlarge	3	20
c8i-flex.4xlarge	7	40
c8i-flex.8xlarge	9	60
c8i-flex.12xlarge	11	120
c8i-flex.16xlarge	15	120

記憶體最佳化

執行個體類型	未啟用 ENI 主幹功能時的 任務數量上限	啟用 ENI 主幹功能時的任務數量上限
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

執行個體類型	未啟用 ENI 主幹功能時的 任務數量上限	啟用 ENI 主幹功能時的任務數量上限
r5a.xlarge	3	20
r5a.2xlarge	3	40
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

執行個體類型	未啟用 ENI 主幹功能時的 任務數量上限	啟用 ENI 主幹功能時的任務數量上限
r5d.8xlarge	7	60
r5d.12xlarge	7	60
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

執行個體類型	未啟用 ENI 主幹功能時的 任務數量上限	啟用 ENI 主幹功能時的任務數量上限
r6g.4xlarge	7	60
r6g.8xlarge	7	60
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

執行個體類型	未啟用 ENI 主幹功能時的 任務數量上限	啟用 ENI 主幹功能時的任務數量上限
r6i.16xlarge	14	120
r6i.24xlarge	14	120
r6i.32xlarge	14	120
r6i.metal	14	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
r6idn.large	2	10
r6idn.xlarge	3	20
r6idn.2xlarge	3	40
r6idn.4xlarge	7	60
r6idn.8xlarge	7	90
r6idn.12xlarge	7	120

執行個體類型	未啟用 ENI 主幹功能時的 任務數量上限	啟用 ENI 主幹功能時的任務數量上限
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
r6in.32xlarge	15	120
r6in.metal	15	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

執行個體類型	未啟用 ENI 主幹功能時的 任務數量上限	啟用 ENI 主幹功能時的任務數量上限
r7a.12xlarge	7	120
r7a.16xlarge	14	120
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

執行個體類型	未啟用 ENI 主幹功能時的 任務數量上限	啟用 ENI 主幹功能時的任務數量上限
r7gd.8xlarge	7	60
r7gd.12xlarge	7	60
r7gd.16xlarge	14	120
r7gd.metal	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
r7iz.large	2	10
r7iz.xlarge	3	20
r7iz.2xlarge	3	40
r7iz.4xlarge	7	60
r7iz.8xlarge	7	90

執行個體類型	未啟用 ENI 主幹功能時的 任務數量上限	啟用 ENI 主幹功能時的任務數量上限
r7iz.12xlarge	7	120
r7iz.16xlarge	14	120
r7iz.32xlarge	14	120
r7iz.metal-16xl	14	120
r7iz.metal-32xl	14	120
r8a.medium	1	4
r8a.large	2	10
r8a.xlarge	3	20
r8a.2xlarge	3	40
r8a.4xlarge	7	60
r8a.8xlarge	9	90
r8a.12xlarge	11	120
r8a.16xlarge	15	120
r8a.24xlarge	15	120
r8a.48xlarge	23	120
r8a.metal-24xl	15	120
r8a.metal-48xl	23	120
r8g.medium	1	4
r8g.large	2	10
r8g.xlarge	3	20

執行個體類型	未啟用 ENI 主幹功能時的 任務數量上限	啟用 ENI 主幹功能時的任務數量上限
r8g.2xlarge	3	40
r8g.4xlarge	7	60
r8g.8xlarge	7	60
r8g.12xlarge	7	60
r8g.16xlarge	14	120
r8g.24xlarge	14	120
r8g.48xlarge	14	120
r8g.metal-24xl	14	120
r8g.metal-48xl	14	120
r8gb.medium	1	4
r8gb.large	2	10
r8gb.xlarge	3	20
r8gb.2xlarge	3	40
r8gb.4xlarge	7	60
r8gb.8xlarge	9	60
r8gb.12xlarge	11	60
r8gb.16xlarge	15	120
r8gb.24xlarge	23	120
r8gb.48xlarge	23	120
r8gb.metal-24xl	23	120

執行個體類型	未啟用 ENI 主幹功能時的 任務數量上限	啟用 ENI 主幹功能時的任務數量上限
r8gb.metal-48xl	23	120
r8gd.medium	1	4
r8gd.large	2	10
r8gd.xlarge	3	20
r8gd.2xlarge	3	40
r8gd.4xlarge	7	60
r8gd.8xlarge	7	60
r8gd.12xlarge	7	60
r8gd.16xlarge	14	120
r8gd.24xlarge	14	120
r8gd.48xlarge	14	120
r8gd.metal-24xl	14	120
r8gd.metal-48xl	14	120
r8gn.medium	1	4
r8gn.large	2	10
r8gn.xlarge	3	20
r8gn.2xlarge	3	40
r8gn.4xlarge	7	60
r8gn.8xlarge	9	60
r8gn.12xlarge	11	60

執行個體類型	未啟用 ENI 主幹功能時的 任務數量上限	啟用 ENI 主幹功能時的任務數量上限
r8gn.16xlarge	15	120
r8gn.24xlarge	23	120
r8gn.48xlarge	23	120
r8gn.metal-24xl	23	120
r8gn.metal-48xl	23	120
r8i.large	2	10
r8i.xlarge	3	20
r8i.2xlarge	3	40
r8i.4xlarge	7	60
r8i.8xlarge	9	90
r8i.12xlarge	11	120
r8i.16xlarge	15	120
r8i.24xlarge	15	120
r8i.32xlarge	23	120
r8i.48xlarge	23	120
r8i.96xlarge	23	120
r8i.metal-48xl	23	120
r8i.metal-96xl	23	120
r8id.large	2	10
r8id.xlarge	3	20

執行個體類型	未啟用 ENI 主幹功能時的 任務數量上限	啟用 ENI 主幹功能時的任務數量上限
r8id.2xlarge	3	40
r8id.4xlarge	7	60
r8id.8xlarge	9	90
r8id.12xlarge	11	120
r8id.16xlarge	15	120
r8id.24xlarge	15	120
r8id.32xlarge	23	120
r8id.48xlarge	23	120
r8id.96xlarge	23	120
r8id.metal-48xl	23	120
r8id.metal-96xl	23	120
r8i-flex.large	2	4
r8i-flex.xlarge	3	10
r8i-flex.2xlarge	3	20
r8i-flex.4xlarge	7	40
r8i-flex.8xlarge	9	60
r8i-flex.12xlarge	11	120
r8i-flex.16xlarge	15	120
u-3tb1.56xlarge	7	12
u-6tb1.56xlarge	14	12

執行個體類型	未啟用 ENI 主幹功能時的 任務數量上限	啟用 ENI 主幹功能時的任務數量上限
u-18tb1.112xlarge	14	12
u-18tb1.metal	14	12
u-24tb1.112xlarge	14	12
u-24tb1.metal	14	12
u7i-6tb.112xlarge	14	120
u7i-8tb.112xlarge	14	120
u7i-12tb.224xlarge	14	120
u7in-16tb.224xlarge	15	120
u7in-24tb.224xlarge	15	120
u7in-32tb.224xlarge	15	120
u7inh-32tb.480xlarge	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
x2gd.16xlarge	14	120
x2gd.metal	14	120

執行個體類型	未啟用 ENI 主幹功能時的 任務數量上限	啟用 ENI 主幹功能時的任務數量上限
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
x8g.medium	1	4
x8g.large	2	10

執行個體類型	未啟用 ENI 主幹功能時的 任務數量上限	啟用 ENI 主幹功能時的任務數量上限
x8g.xlarge	3	20
x8g.2xlarge	3	40
x8g.4xlarge	7	60
x8g.8xlarge	7	60
x8g.12xlarge	7	60
x8g.16xlarge	14	120
x8g.24xlarge	14	120
x8g.48xlarge	14	120
x8g.metal-24xl	14	120
x8g.metal-48xl	14	120
x8aecz.large	3	10
x8aecz.xlarge	3	20
x8aecz.3xlarge	7	40
x8aecz.6xlarge	7	60
x8aecz.12xlarge	15	120
x8aecz.24xlarge	15	120
x8aecz.metal-12xl	15	120
x8aecz.metal-24xl	15	120
x8i.large	2	10
x8i.xlarge	3	20

執行個體類型	未啟用 ENI 主幹功能時的 任務數量上限	啟用 ENI 主幹功能時的任務數量上限
x8i.2xlarge	3	40
x8i.4xlarge	7	60
x8i.8xlarge	9	90
x8i.12xlarge	11	120
x8i.16xlarge	15	120
x8i.24xlarge	15	120
x8i.32xlarge	23	120
x8i.48xlarge	23	120
x8i.64xlarge	23	120
x8i.96xlarge	23	120
x8i.metal-48xl	23	120
x8i.metal-96xl	23	120

儲存最佳化

執行個體類型	未啟用 ENI 主幹功能時的 任務數量上限	啟用 ENI 主幹功能時的任務數量上限
i4g.large	2	10
i4g.xlarge	3	20
i4g.2xlarge	3	40
i4g.4xlarge	7	60

執行個體類型	未啟用 ENI 主幹功能時的 任務數量上限	啟用 ENI 主幹功能時的任務數量上限
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
i7i.large	2	10
i7i.xlarge	3	20
i7i.2xlarge	3	40
i7i.4xlarge	7	60
i7i.8xlarge	7	90
i7i.12xlarge	7	90
i7i.16xlarge	14	120
i7i.24xlarge	14	120
i7i.48xlarge	14	120

執行個體類型	未啟用 ENI 主幹功能時的 任務數量上限	啟用 ENI 主幹功能時的任務數量上限
i7i.metal-24xl	14	120
i7i.metal-48xl	14	120
i7ie.large	2	20
i7ie.xlarge	3	29
i7ie.2xlarge	3	29
i7ie.3xlarge	3	29
i7ie.6xlarge	7	60
i7ie.12xlarge	7	60
i7ie.18xlarge	14	120
i7ie.24xlarge	14	120
i7ie.48xlarge	14	120
i7ie.metal-24xl	14	120
i7ie.metal-48xl	14	120
i8g.large	2	10
i8g.xlarge	3	20
i8g.2xlarge	3	40
i8g.4xlarge	7	60
i8g.8xlarge	7	60
i8g.12xlarge	7	60
i8g.16xlarge	14	120

執行個體類型	未啟用 ENI 主幹功能時的 任務數量上限	啟用 ENI 主幹功能時的任務數量上限
i8g.24xlarge	14	120
i8g.48xlarge	14	120
i8g.metal-24xl	14	120
i8g.metal-48xl	14	120
i8ge.large	2	20
i8ge.xlarge	3	29
i8ge.2xlarge	3	29
i8ge.3xlarge	5	29
i8ge.6xlarge	9	60
i8ge.12xlarge	11	60
i8ge.18xlarge	15	120
i8ge.24xlarge	15	120
i8ge.48xlarge	23	120
i8ge.metal-24xl	15	120
i8ge.metal-48xl	23	120
im4gn.large	2	10
im4gn.xlarge	3	20
im4gn.2xlarge	3	40
im4gn.4xlarge	7	60
im4gn.8xlarge	7	60

執行個體類型	未啟用 ENI 主幹功能時的 任務數量上限	啟用 ENI 主幹功能時的任務數量上限
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	未啟用 ENI 主幹功能時的 任務數量上限	啟用 ENI 主幹功能時的任務數量上限
dl1.24xlarge	59	120
dl2q.24xlarge	14	120
f2.6xlarge	7	90
f2.12xlarge	7	120
f2.48xlarge	14	120
g4ad.xlarge	1	12
g4ad.2xlarge	1	12
g4ad.4xlarge	2	12
g4ad.8xlarge	3	12

執行個體類型	未啟用 ENI 主幹功能時的 任務數量上限	啟用 ENI 主幹功能時的任務數量上限
g4ad.16xlarge	7	12
g5.xlarge	3	6
g5.2xlarge	3	19
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
g6.xlarge	3	20
g6.2xlarge	3	40
g6.4xlarge	7	60
g6.8xlarge	7	90
g6.12xlarge	7	120

執行個體類型	未啟用 ENI 主幹功能時的 任務數量上限	啟用 ENI 主幹功能時的任務數量上限
g6.16xlarge	14	120
g6.24xlarge	14	120
g6.48xlarge	14	120
g6e.xlarge	3	20
g6e.2xlarge	3	40
g6e.4xlarge	7	60
g6e.8xlarge	7	90
g6e.12xlarge	9	120
g6e.16xlarge	14	120
g6e.24xlarge	19	120
g6e.48xlarge	39	120
g6f.large	1	10
g6f.xlarge	3	20
g6f.2xlarge	3	40
g6f.4xlarge	7	60
gr6.4xlarge	7	60
gr6.8xlarge	7	90
gr6f.4xlarge	7	60
g7e.2xlarge	3	242
g7e.4xlarge	7	242

執行個體類型	未啟用 ENI 主幹功能時的 任務數量上限	啟用 ENI 主幹功能時的任務數量上限
g7e.8xlarge	7	242
g7e.12xlarge	9	242
g7e.24xlarge	19	242
g7e.48xlarge	39	242
inf2.xlarge	3	20
inf2.8xlarge	7	90
inf2.24xlarge	14	120
inf2.48xlarge	14	120
p4d.24xlarge	59	120
p4de.24xlarge	59	120
p5.4xlarge	3	60
p5.48xlarge	63	242
p5e.48xlarge	63	242
p5en.48xlarge	63	242
p6-b200.48xlarge	31	242
p6-b300.48xlarge	67	242
p6e-gb200.36xlarge	38	120
trn1.2xlarge	3	19
trn1.32xlarge	39	120
trn1n.32xlarge	79	242

執行個體類型	未啟用 ENI 主幹功能時的任務數量上限	啟用 ENI 主幹功能時的任務數量上限
trn2.3xlarge	1	14
trn2.48xlarge	31	242
trn2u.48xlarge	31	242
vt1.3xlarge	3	40
vt1.6xlarge	7	60
vt1.24xlarge	14	120

高效能運算

執行個體類型	未啟用 ENI 主幹功能時的任務數量上限	啟用 ENI 主幹功能時的任務數量上限
hpc6a.48xlarge	1	120
hpc6id.32xlarge	1	120
hpc7g.4xlarge	3	120
hpc7g.8xlarge	3	120
hpc7g.16xlarge	3	120
hpc8a.96xlarge	3	-2

保留 Amazon ECS Linux 容器執行個體記憶體

當 Amazon ECS 容器代理程式將容器執行個體註冊至叢集時，代理程式必須判斷容器執行個體有多少記憶體可保留給您的任務。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。但在容器執行個體註冊時，未必恰好有 8192 MiB 的記憶體可供任務使用。

ECS 受管執行個體記憶體資源判斷

Amazon ECS 受管執行個體使用階層方法來判斷任務的記憶體資源需求。與依賴 Docker 記憶體自我檢查的 EC2 上的 ECS 不同，ECS 受管執行個體會在排程決策期間直接從任務承載計算記憶體需求。

ECS 受管執行個體代理程式收到任務時，會使用下列優先順序來計算記憶體需求：

1. 任務層級記憶體（最高優先順序）- 如果任務層級記憶體是在任務定義中指定，代理程式會直接使用此值。這優先於所有容器層級的記憶體設定。
2. 容器層級記憶體總和（備用）- 如果未指定任務層級記憶體（或為 0），代理程式會加總任務中所有容器的記憶體需求。對於每個容器，它使用：
 - a. 記憶體保留（軟性限制）- 如果容器在其組態 `memoryReservation` 中指定，代理程式會使用此值。
 - b. 容器記憶體（硬性限制）- 如果 `memoryReservation` 未指定，代理程式會使用容器 `memory` 的欄位。

Example 指定的任務層級記憶體

指定任務層級記憶體時，其優先順序會高於容器層級設定：

```
{
  "family": "my-task",
  "memory": "2048",
  "containerDefinitions": [
    {
      "name": "container1",
      "memory": 1024,
      "memoryReservation": 512
    }
  ]
}
```

代理程式保留 2048 MiB（任務層級記憶體優先）。

Example 具有保留的容器層級記憶體

未指定任務層級記憶體時，代理程式會加總容器記憶體需求：

```
{
  "family": "my-task",
```

```
"containerDefinitions": [  
  {  
    "name": "container1",  
    "memory": 1024,  
    "memoryReservation": 512  
  },  
  {  
    "name": "container2",  
    "memory": 512  
  }  
]  
}
```

代理程式保留 512 MiB (容器 1 保留) + 512 MiB (容器 2 記憶體) = 總計 1024 MiB。

ECS 受管執行個體代理程式會分三個階段執行記憶體計算：

1. 任務接收 - 當任務承載從 ECS 控制平面抵達時，代理程式會立即計算所需的記憶體。
2. 資源儲存 - 計算的記憶體需求會存放在任務模型中，以供日後在資源會計操作中使用。
3. 排程決策 - 在接受任務之前，客服人員會檢查是否有足夠記憶體可用。如果可用的記憶體不足，任務會遭到拒絕，並保留在 ECS 服務佇列中，直到資源可用為止。

Note

與 EC2 上的 ECS 不同，ECS 受管執行個體不會使用 ECS_RESERVED_MEMORY 組態變數。系統程序的記憶體保留是透過基礎平台的資源管理來處理，而代理程式會根據任務定義執行準確的資源會計。

對於 EC2 上的 ECS，Amazon ECS 容器代理程式提供名為 `ECS_RESERVED_MEMORY` 的組態變數，可用來從配置給您任務 MiB 的集區中移除指定數量的記憶體。這可為重要系統程序有效地預留記憶體。

若您的任務佔用了容器執行個體上的所有記憶體，任務可能會與重要的系統程序爭奪記憶體，也許會引發系統故障。

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 容器執行個體傳遞資料](#)。

若指定 8192 MiB 供任務使用，而您沒有任何容器執行個體擁有 8192 MiB 或更多記憶體可滿足此需求，則任務便無法置放到叢集中。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請求。

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 均有提供命令列公用程式來判斷記憶體總量。

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 Windows Server AMI 的 `m4.large` 執行個體的輸出範例。

```
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。

檢視容器執行個體記憶體

您可以在 Amazon ECS 主控台中檢視容器執行個體註冊的記憶體量 (也可以使用 [DescribeContainerInstances](#) API 操作)。若要為特定執行個體類型的任務提供盡可能多的記憶體，以

最大化資源使用率，您可以觀察該容器執行個體有多少可用的記憶體量，然後再向任務指派盡可能多的記憶體。

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. 在導覽窗格中選擇叢集，再選取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. 資源區段會顯示容器執行個體的已註冊和可使用的記憶體。

已註冊的記憶體值是容器執行個體首次啟動時向 Amazon ECS 註冊的記憶體量，而可用記憶體的值則是尚未分配給任務的記憶體量。

使用 從遠端管理 Amazon ECS 容器執行個體 AWS Systems Manager

您可以在 AWS Systems Manager (Systems Manager) 中使用執行命令功能，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](#)。

以下是搭配 Amazon ECS 使用 Systems Manager 的先決條件。

1. 您必須授予容器執行個體角色 (ecsInstanceRole) 存取 Systems Manager API 的許可。您可以透過將 AmazonSSMManagedInstanceCore 指派給 ecsInstanceRole 角色來執行此操作。如需

有關如何將政策連接至角色的資訊，請參閱 AWS Identity and Access Management User Guide 中的 [Update permissions for a role](#)

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 [Manually installing and uninstalling SSM Agent on EC2 instances for Linux](#)。

將 Systems Manager 受管政策連接到 `ecsInstanceRole` 並確認 AWS Systems Manager 代理程式 (SSM 代理程式) 已安裝在您的容器執行個體之後，您就可以開始使用 Run Command 將命令傳送至您的容器執行個體。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](#)。

常見的使用案例是透過 Run Command 更新容器執行個體軟體。您可以使用下列參數，遵循 AWS Systems Manager 使用者指南中的程序。

參數	Value	
命令文件	AWS-RunShellScript	
命令	<pre>\$ yum update -y</pre>	
目標執行個體	您的容器執行個體	

為 Amazon ECS Linux 容器執行個體使用 HTTP Proxy

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案，再重新啟動代理程式。

`/etc/ecs/ecs.config` (Amazon Linux 2 與 Amazon Linux 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=169.254.169.254,169.254.170.2"
```

將此數值設為 169.254.169.254,169.254.170.2，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 169.254.169.254,169.254.170.2，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。

如需有關如何設定代理的資訊，請參閱[如何在 Amazon Linux 2 或 Amazon Linux 2023 中，為 Docker 和 Amazon ECS 容器代理程式設定 HTTP Proxy](#)。

為 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。將 *MyCluster* 取代為您的叢集名稱。

```
#!/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，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 主控台下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 容器自我檢查](#)。

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 最佳化 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 AWS CLI AWS SDKs，使用 UpdateContainerAgent API 操作更新容器代理程式。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 AWS CLI AWS SDKs 請求代理程式更新時，`UpdateContainerAgentAPI` 程序就會開始。Amazon ECS 會比對您目前的代理程式版本與最新可用的代理程式版本，判斷是否可執行更新。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)

有時，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。

 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[將 Amazon ECS 容器執行個體組態儲存於 Amazon S3 中](#)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。因此 Amazon ECS 最佳化 AMI 會使用 ecs-init RPM，您可以在 GitHub 上檢視[此 RPM 的來源程式碼](#)。

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 最佳化 Windows 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 AMIs 是由 AWS 工程師在 Amazon ECS 上預先設定和測試。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 ECS 最佳化 Windows AMI 中繼資料”](#)。

Important

2022 年 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 2025 Full AMI
- Amazon ECS 最佳化 Windows Server 2025 Core AMI
- 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

Windows Server 2016 不支援最新版本的 Docker，例如 25.x.x。因此，Windows Server 2016 Full AMI 不會收到 Docker 執行時期的安全性修補程式或錯誤修補程式。建議遷移至下列任一 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 2025、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 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[EC2 Windows 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。您的容器必須在它們使用該功能時執行部分提供的 PowerShell 程式碼。如需詳細資訊，請參閱[Amazon EC2 Windows 執行個體的額外組態](#)。
- 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[使用負載平衡分佈 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 ECS 最佳化 Windows AMI 中繼資料”](#)。

擷取 Amazon ECS 最佳化 Windows 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 2025 完整 AMI 中繼資料：

```
/aws/service/ami-windows-latest/Windows_Server-2025-English-Full-ECS_Optimized
```

- Windows Server 2025 Core AMI 中繼資料：

```
/aws/service/ami-windows-latest/Windows_Server-2025-English-Core-ECS_Optimized
```

- 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 AWS CLI 搭配下列 AWS CLI 命令，擷取最新的穩定 Amazon ECS 最佳化 AMI。

- 對於 Amazon ECS 最佳化 Windows Server 2025 Full AMI：

```
aws ssm get-parameters --names /aws/service/ami-windows-latest/Windows_Server-2025-English-Full-ECS_Optimized --region us-east-1
```

- 對於 Amazon ECS 最佳化 Windows Server 2025 Core AMI：

```
aws ssm get-parameters --names /aws/service/ami-windows-latest/Windows_Server-2025-English-Core-ECS_Optimized --region us-east-1
```

- 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 CloudFormation 範本中使用最新建議的 Amazon ECS 最佳化 AMI

透過參考 Systems Manager 參數存放區名稱，可以在 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
```

Amazon ECS 最佳化 AMI 版本

檢視目前和舊版的 Amazon ECS 最佳化 AMI，及其對應的 Amazon ECS 容器代理程式、Docker 和 ecs-init 套件版本。

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 ECS 最佳化 Windows 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 CloudFormation 範本中參考 Systems Manager 參數存放區參數的詳細資訊，請參閱 [在 CloudFormation 範本中使用最新建議的 Amazon ECS 最佳化 AMI](#)。

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。

Windows Server 2016 不支援最新版本的 Docker，例如 25.x.x。因此，Windows Server 2016 Full AMI 不會收到 Docker 執行時期的安全性修補程式或錯誤修補程式。建議遷移至下列任一 Windows 平台：

- Windows Server 2022 Full

- Windows Server 2022 Core
- Windows Server 2019 Full
- Windows Server 2019 Core

Note

繼 2025 年 8 月 AMI 版本後，gMSA 外掛程式記錄已從檔案型記錄 (C:\ProgramData\Amazon\gmsa) 遷移至 Windows Event logging。公開版日誌收集器指令碼會收集所有 gMSA 日誌。如需詳細資訊，請參閱[使用 Amazon ECS 日誌收集器收集容器日誌](#)。

Windows Server 2025 Full AMI versions

下表列出 Amazon ECS 最佳化 Windows Server 2025 Full AMI 的目前和先前版本，以及其 Amazon ECS 容器代理程式和 Docker 的對應版本。

Amazon ECS 最佳化 Windows Server 2025 Full AMI	Amazon ECS 容器代理程式版本	Docker 版本	Visibility
Windows_Server-2025-English-Full-ECS_Optimized-2026.03.13	1.102.0	25.0.6 (Docker CE)	公有
Windows_Server-2025-English-Full-ECS_Optimized-2026.02.13	1.101.3	25.0.6 (Docker CE)	公有
Windows_Server-2025-English-Full-ECS_Optimized-2026.01.16	1.101.2	25.0.6 (Docker CE)	公有
Windows_Server-2025-English-Full-ECS	1.101.0	25.0.6 (Docker CE)	公有

Amazon ECS 最佳化 Windows Server 2025 Full AMI	Amazon ECS 容器代理程式版本	Docker 版本	Visibility
_Optimized-2025.12.13			

使用下列 AWS CLI 命令擷取目前的 Amazon ECS 最佳化 Windows Server 2025 Full AMI。

```
aws ssm get-parameters --names /aws/service/ami-windows-latest/Windows_Server-2025-English-Full-ECS_Optimized
```

Windows Server 2025 Core AMI versions

下表列出 Amazon ECS 最佳化 Windows Server 2025 Core AMI 的目前和先前版本，以及其 Amazon ECS 容器代理程式和 Docker 的對應版本。

Amazon ECS 最佳化 Windows Server 2025 Core AMI	Amazon ECS 容器代理程式版本	Docker 版本	Visibility
Windows_Server-2025-English-Core-ECS_Optimized-2026.03.13	1.102.0	25.0.6 (Docker CE)	公有
Windows_Server-2025-English-Core-ECS_Optimized-2026.02.13	1.101.3	25.0.6 (Docker CE)	公有
Windows_Server-2025-English-Core-ECS_Optimized-2026.01.16	1.101.2	25.0.6 (Docker CE)	公有
Windows_Server-2025-English-Core-ECS	1.101.0	25.0.6 (Docker CE)	公有

Amazon ECS 最佳化 Windows Server 2025 Core AMI	Amazon ECS 容器代理程式版本	Docker 版本	Visibility
_Optimized-2025.12.13			

使用下列 AWS CLI 命令來擷取目前的 Amazon ECS 最佳化 Windows Server 2025 Core AMI。

```
aws ssm get-parameters --names /aws/service/ami-windows-latest/Windows_Server-2025-English-Core-ECS_Optimized
```

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
Windows_Server-2022-English-Full-ECS_Optimized-2026.03.13	1.102.0	25.0.6 (Docker CE)	公有
Windows_Server-2022-English-Full-ECS_Optimized-2026.02.13	1.101.3	25.0.6 (Docker CE)	公有
Windows_Server-2022-English-Full-ECS_Optimized-2026.01.16	1.101.2	25.0.6 (Docker CE)	公有
Windows_Server-2022-English-Full-ECS	1.101.0	25.0.6 (Docker CE)	公有

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-2025.12.13			

使用下列 AWS CLI 命令擷取目前的 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
```

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
Windows_Server-2022-English-Core-ECS_Optimized-2026.03.13	1.102.0	25.0.6 (Docker CE)	公有
Windows_Server-2022-English-Core-ECS_Optimized-2026.02.13	1.101.3	25.0.6 (Docker CE)	公有
Windows_Server-2022-English-Core-ECS_Optimized-2026.01.16	1.101.2	25.0.6 (Docker CE)	公有
Windows_Server-2022-English-Core-ECS	1.101.0	25.0.6 (Docker CE)	公有

Amazon ECS 最佳化 Windows Server 2022 Core AMI	Amazon ECS 容器代理程式版本	Docker 版本	Visibility
_Optimized-2025.12.13			

使用以下 AWS CLI 命令擷取目前的 Amazon ECS 最佳化 Windows Server 2022 Full 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
Windows_Server-2019-English-Full-ECS_Optimized-2026.03.13	1.102.0	25.0.6 (Docker CE)	公有
Windows_Server-2019-English-Full-ECS_Optimized-2026.02.13	1.101.3	25.0.6 (Docker CE)	公有
Windows_Server-2019-English-Full-ECS_Optimized-2026.01.16	1.101.2	25.0.6 (Docker CE)	公有
Windows_Server-2019-English-Full-ECS	1.101.0	25.0.6 (Docker CE)	公有

Amazon ECS 最佳化 Windows Server 2019 Full AMI	Amazon ECS 容器代理程式版本	Docker 版本	Visibility
_Optimized-2025.12.13			

使用下列 AWS CLI 命令來擷取目前的 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
```

Windows Server 2019 Core AMI versions

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
Windows_Server-2019-English-Core-ECS_Optimized-2026.03.13	1.102.0	25.0.6 (Docker CE)	公有
Windows_Server-2019-English-Core-ECS_Optimized-2026.02.13	1.101.3	25.0.6 (Docker CE)	公有
Windows_Server-2019-English-Core-ECS_Optimized-2026.01.16	1.101.2	25.0.6 (Docker CE)	公有
Windows_Server-2019-English-Core-ECS	1.101.0	25.0.6 (Docker CE)	公有

Amazon ECS 最佳化 Windows Server 2019 Core AMI	Amazon ECS 容器代理程式版本	Docker 版本	Visibility
_Optimized-2025.12.13			

使用下列 AWS CLI 命令來擷取目前的 Amazon ECS 最佳化 Windows Server 2019 Full 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

Windows Server 2016 不支援最新版本的 Docker，例如 25.x.x。因此，Windows Server 2016 Full AMI 不會收到 Docker 執行時期的安全性修補程式或錯誤修補程式。建議遷移至下列任一 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
Windows_Server-2016-English-Full-ECS_Optimized-2026.03.13	1.102.0	20.10.23 (Docker CE)	公有

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-2026.02.13	1.101.3	20.10.23 (Docker CE)	公有
Windows_Server-2016-English-Full-ECS_Optimized-2026.01.16	1.101.2	20.10.23 (Docker CE)	公有
Windows_Server-2016-English-Full-ECS_Optimized-2025.12.13	1.101.0	20.10.23 (Docker CE)	公有

使用下列 AWS CLI 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
```

建立自己的 Amazon ECS 最佳化 Windows AMI

使用 EC2 Image Builder 自行建立自訂 Amazon ECS 最佳化 Windows 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：

- 來源映像應以 Windows Server 2019 Core、Windows Server 2019 Full、Windows Server 2022 Core 或 Windows Server 2022 Full 為基礎。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 管理主控台。

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 容器代理程式和 Docker 版本。

Amazon ECS Windows 容器執行個體管理

在將 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 Windows 容器執行個體傳遞資料](#)
- [為 Amazon ECS Windows 容器執行個體使用 HTTP Proxy](#)
- [設定 Amazon ECS Windows 容器執行個體接收 Spot 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 User Guide 中的 [Launch an instance using the new launch instance wizard](#)。

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](#)》。

使用搜尋列來尋找由發佈的適當 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。

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。

- 子網路：您可以在與可用區域、Local Zone、Wavelength 區域或 Outpost 相關聯的子網路中啟動執行個體。

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子網路。若要建立新的子網路，請選擇建立新的子網路，前往 Amazon VPC 主控台。完成後，請返回啟動執行個體精靈並選擇「重新整理」圖示，載入清單中的子網路。

在 Local Zone 中啟動執行個體，選取您在 Local Zone 中建立的子網路。

若要在 Outpost 中啟動執行個體，請在與 Outpost 相關聯的 VPC 中選取子網路。

- 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 [為 Amazon ECS Linux 容器執行個體使用 HTTP Proxy](#)。

- 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 AMI，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 Windows 容器執行個體傳遞資料

當您啟動 Amazon EC2 執行個體時，可以將使用者資料傳送到 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 ccloud-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。它已從 [Amazon ECS Container Agent GitHub 儲存庫](#) 讀我檔案頁面的代理程式安裝說明調整。

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 工作負載建立 Amazon ECS 叢集](#)。

如需有關如何在 Windows Server 2022 Full 上安裝 Amazon ECS 代理程式的資訊，請參閱 GitHub 上的 [Issue 3753](#)。

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>
```

為 Amazon ECS Windows 容器執行個體使用 HTTP Proxy

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

以下範例使用者資料 PowerShell 指令碼會設定 Amazon ECS 容器代理程式和 Docker 常駐程式，以使用您指定的 HTTP 代理。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。只要複製以下 PowerShell 指令碼，並在 User data (使用者資料) 欄位中貼上即可 (請確認您使用自己的代理和叢集資訊取代紅色的範例數值)。

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 Windows 執行個體的額外組態](#)。


```
<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 Windows 容器執行個體接收 Spot 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啟用 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”](#)。

外部執行個體的 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。

支援的作業系統和系統架構

以下是支援的作業系統清單。支援 x86_64 和 ARM64 CPU 架構。

- Amazon Linux 2023
- Ubuntu 20、Ubuntu 22、Ubuntu 24
- RHEL 9 — 在執行 [ECS Anywhere 安裝指令碼](#) 之前，您必須確保已安裝 Docker。如需詳細資訊，請參閱 [Docker 文件中的在 RHEL 上安裝 Docker 引擎](#)。

從 2026 年 8 月 7 日開始，Amazon ECS Anywhere 不再支援下列作業系統：

- Amazon Linux 2
- CentOS Stream 9
- RHEL 7、RHEL 8
- Fedora 32、Fedora 33、Fedora 40
- openSUSE Tumbleweed
- Ubuntu 18
- Debian 9、Debian 10、Debian 11、Debian 12
- SUSE Enterprise Server 15
- 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允許其與 AWS APIs 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。
- 若要將容器日誌傳送至 CloudWatch Logs，請確定您已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[將標籤新增至 Amazon ECS 的外部容器執行個體](#)。
- 外部執行個體支援 ECS Exec。如需詳細資訊，請參閱[使用 ECS Exec 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失去與的連線 AWS，SSM Agent 會在重新建立連線後自動重新整理登入資料。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 and 虛擬機器](#)。
 - 只要執行個體位於純 IPv6 子網路中，即可在採用純 IPv6 組態的外部執行個體上執行 Linux 任務。如需詳細資訊，請參閱[在純 IPv6 模式下使用 VPC](#)。
- 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。

- 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[外部執行個體的額外組態](#)。

```
$ 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。

如需有關每個網路模式的詳細資訊，請參閱[EC2 執行個體的 Amazon ECS 任務聯網選項](#)。

- 只要執行個體位於純 IPv6 子網路中，即可在採用純 IPv6 組態的外部執行個體上執行 Linux 任務。如需詳細資訊，請參閱[在純 IPv6 模式下使用 VPC](#)。
- 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失去與的連線 AWS，SSM Agent 會在重新建立連線後自動重新整理登入資料。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 and 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` — 這是服務端點，AWS Systems Manager 用於在 Systems Manager 代理程式和雲端中的 Systems Manager 服務之間進行通訊。
- `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。
- 如果您的任務需要與任何其他 AWS 服務通訊，請確定允許這些服務端點。應用程式範例包括使用 Amazon ECR 來提取容器映像或將 CloudWatch 用於 CloudWatch Logs。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 搭配 ECS Anywhere

Important

Amazon ECS Anywhere 的 Windows 支援已棄用。本節不再適用。

若要 Amazon FSx for Windows File Server 搭配 Amazon ECS 外部執行個體使用，您必須在內部部署資料中心與之間建立連線 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

Important

Amazon ECS Anywhere 的 Windows 支援已棄用。本節不再適用。

當 Windows 是支援的作業系統時，ECS Anywhere 支援下列使用案例。

- Active Directory 位於 AWS 雲端 - 對於此組態，您可以使用連線在內部部署網路與 AWS 雲端之間建立 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。如需有關如何開始使用的資訊 AWS Directory Service，請參閱《AWS Directory Service 管理指南》中的[設定 AWS Directory Service](#)。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 “了解如何為 EC2 Windows 容器使用 gMSA”](#)。

- 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。

為外部執行個體工作負載建立 Amazon ECS 叢集

您可以透過建立叢集，定義任務與服務執行所在的基礎結構。

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 叢集範例”](#)。Amazon ECS 主控台提供簡單的方法，透過建立 CloudFormation 堆疊來建立 Amazon ECS 叢集所需的資源。

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。

您可以修改下列選項：

- 將命名空間新增至叢集。

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](#)。

- 設定外部執行個體的叢集
- 為您的受管儲存指派 AWS KMS 金鑰。如需有關如何建立金鑰的資訊，請參閱 AWS Key Management Service User Guide 中的 [Create a KMS key](#)。
- 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. (選用) 使用 Container Insights，展開監控區段，然後選擇下列任一選項：
- 若要使用建議的具有增強可觀測性的 Container Insights，請選擇具有增強可觀測性的 Container Insights。
 - 若要使用 Container Insights，請選擇 Container Insights。
7. (選用) 若要使用 ECS Exec 在叢集中偵錯任務，請展開疑難排解組態區段，然後設定下列項目：
- 選取啟用 ECS Exec。
 - (選用) 對於 AWS KMS ECS Exec 的金鑰，輸入您要用來加密 ECS Exec 工作階段資料的 AWS KMS 金鑰 ARN。
 - (選用) 在 ECS Exec 記錄欄位中，選擇日誌目的地：
 - 若要將日誌傳送至 CloudWatch Logs，請選擇 Amazon CloudWatch。
 - 若要將日誌傳送至 Amazon S3，請選擇 Amazon S3。
 - 若要停用記錄，請選擇無。
8. (選用) 加密受管儲存上的資料。在加密下，針對受管儲存，輸入您要用來加密受管儲存資料的 AWS KMS 金鑰 ARN。
9. (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：

- 在金鑰欄位中，輸入金鑰名稱。
- 在值欄位中，輸入金鑰值。

10. 選擇建立。

後續步驟

您必須向叢集註冊執行個體。如需詳細資訊，請參閱[將外部執行個體註冊到 Amazon ECS 叢集](#)。

為外部啟動類型建立任務定義。如需詳細資訊，請參閱[使用主控台建立 Amazon ECS 任務定義](#)

將應用程式作為獨立任務執行，或作為服務的組成分執行。如需詳細資訊，請參閱下列內容：

- [將應用程式作為 Amazon ECS 任務執行](#)
- [建立 Amazon ECS 滾動更新部署](#)

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 管理主控台

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
gpg:                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

在執行安裝指令碼以完成外部執行個體註冊程序之前，AWS CLI 可用來建立 Systems Manager 啟用。

取消註冊 Amazon ECS 外部執行個體

我們建議您在完成執行個體 AWS Systems Manager 之後，從 Amazon ECS 和 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。

取消註冊執行個體之後，請清除執行個體上剩餘的 AWS 資源。之後，您可以將該執行個體註冊至新的叢集。

程序

AWS 管理主控台

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 User Guide 中的 [Deregistering managed nodes in a hybrid and multicloud environment](#)。

8. 取消註冊執行個體後，請清除現場部署伺服器或 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。 <pre>sudo systemctl stop ecs amazon-ssm-agent</pre> b. 移除 Amazon ECS 和 Systems Manager 套件。 對於 CentOS 7、CentOS 8 和 RHEL 7 <pre>sudo yum remove -y amazon-ecs-init amazon-ssm-agent</pre> 對於 SUSE Enterprise Server 15 <pre>sudo zypper remove -y amazon-ecs-init amazon-ssm-agent</pre> 對於 Debian 和 Ubuntu

作業系統	步驟	
	<pre>sudo apt remove -y amazon-ecs-init amazon-ssm-agent</pre> c. 移除剩餘的目錄。 <pre>sudo rm -rf /var/ lib/ecs /etc/ecs / var/lib/amazon/ss m /var/log/ecs / var/log/amazon/ssm</pre>	
Windows	a. 停用執行個體上的 Amazon ECS 容器代理程式和 SSM Agent 服務。 <pre>Stop-Service AmazonECS</pre> <pre>Stop-Service AmazonSSMAgent</pre> b. 移除 Amazon ECS 套件。 <pre>.\ecs-anywhere-ins tall.ps1 -Uninstal l</pre>	

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，請清除現場部署伺服器或 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。 <pre>sudo systemctl stop ecs amazon-ssm-agent</pre> b. 移除 Amazon ECS 和 Systems Manager 套件。 <pre>sudo (yum/apt/ zypper) remove amazon-ecs-init amazon-ssm-agent</pre> c. 移除剩餘的目錄。 <pre>sudo rm -rf /var/ lib/ecs /etc/ecs / var/lib/amazon/ss m /var/log/ecs / var/log/amazon/ssm</pre>	

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，您的內部部署伺服器或 VM 必須同時執行 AWS Systems Manager 代理程式 (SSM 代理程式) 和 Amazon ECS 容器代理程式。新增或更新任何功能時，會 AWS 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 建議您自動執行更新執行個體上 SSM Agent 的程序。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)	amazon-ecs-init-latest.x86_64.rpm
CentOS 8 (x86_64)	
CentOS Stream 9 (x86_64)	
SUSE Enterprise Server 15 (x86_64)	
RHEL 7 (x86_64)	
RHEL 8 (x86_64)	
CentOS 7 (aarch64)	amazon-ecs-init-latest.aarch64.rpm
CentOS 8 (aarch64)	
CentOS Stream 9 (aarch64)	
RHEL 7 (aarch64)	
Debian 9 (x86_64)	amazon-ecs-init-latest.amd64.deb
Debian 10 (x86_64)	
Debian 11 (x86_64)	
Debian 12 (x86_64)	
Ubuntu 18 (x86_64)	

作業系統 (架構)	ecs-init 套件
Ubuntu 20 (x86_64)	ecs-init 套件
Ubuntu 22 (x86_64)	
Ubuntu 24 (x86_64)	
Debian 9 (aarch64)	amazon-ecs-init-latest.arm64.deb
Debian 10 (aarch64)	
Debian 11 (aarch64)	
Debian 12 (aarch64)	
Ubuntu 18 (aarch64)	
Ubuntu 20 (aarch64)	
Ubuntu 22 (aarch64)	
Ubuntu 24 (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和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型服務收集、彙總及總結指標和日誌。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 “使用具有增強可觀測性的 Container Insights 監控 Amazon ECS”](#)。

- 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：

- 在金鑰欄位中，輸入金鑰名稱。
- 在值欄位中，輸入金鑰值。

[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 “停止任務”](#)。
- 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，選擇刪除叢集。

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*。

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 User Guide 中的 [Terminate your instance](#)，以及 [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 CloudFormation 堆疊管理，請透過更新 Auto Scaling 群組或 CloudFormation 堆疊來終止執行個體。否則，Auto Scaling 群組或 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[終止您的執行個體](#)。

排空 Amazon ECS 容器執行個體

有時候您可能需要從叢集中移除容器執行個體，以便執行系統更新或叢集容量規模縮減等作業。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。

Amazon ECS 受管執行個體的排空行為

Amazon ECS 受管執行個體終止可確保正常的工作負載轉換，同時最佳化成本和維護系統運作狀態。終止系統為執行個體終止提供三種不同的決策路徑，每種都具有不同的時間特性與客戶影響設定檔。

客戶啟動的終止

在您需要從服務中立即移除容器執行個體時，提供執行個體移除的直接控制權。執行 `deregister-container-instance` 時，`force` 請求參數會設為 `true`。這表示即使有任何執行中的工作負載，仍需要立即終止。

系統啟動的閒置終止

Amazon ECS 受管執行個體透過終止未執行任何任務的閒置 Amazon ECS 容器執行個體，持續監控並主動最佳化成本。ECS 使用啟發式延遲，讓容器執行個體有機會在終止之前取得新啟動的任務。這可以使用 `scaleInAfter` Amazon ECS 受管執行個體容量提供者組態參數來自訂。

基礎結構重新整理終止

Amazon ECS 受管執行個體會自動管理和更新受管容器執行個體上的軟體，以確保安全性和合規性，同時維持工作負載可用性。如需詳細資訊，請參閱 [Amazon ECS 受管執行個體中的修補](#)。

終止系統會實作兩階段方法，可平衡工作負載持續性與基礎結構管理需求。

階段 1：優雅完成期間

在此階段，系統會實作優先考量工作負載持續性的優雅排空策略。服務任務會透過正常的 Amazon ECS 排程程序優雅排空。獨立任務可能會自然完成，因此會持續執行。系統會監控所有任務，透過自然完成程序達到停止狀態。

階段 2：硬性截止日期強制執行

當優雅完成在可接受的時間範圍內未達成終止目標時，系統會實作硬性截止日期強制執行。硬性截止日期通常設定為排空啟動時間加 7 天，為優雅完成提供大量時間，同時維持營運需求。強制執行包括自動強制取消註冊程序，並立即終止所有剩餘的任務，無論完成狀態如何。

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。您可以確認容器執行個體上的任務狀態，方法是搭配使用 [ListTasks](#) 操作與 `containerInstance` 參數以取得執行個體上的任務清單，然後搭配使用 [DescribeTasks](#) 操作與每個任務的 Amazon Resource Name (ARN) 或 ID 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状态從 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置放。

程序

透過運用新的 AWS 管理主控台，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 [UpdateContainerInstancesState](#) API 動作或 [update-container-instances-state](#) 命令將容器執行個體的状态變更成 DRAINING。

AWS 管理主控台

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 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。

在 Auto Scaling 群組之外變更執行個體類型或大小

AWS 建議您保持基礎設施不變。如果您需要變更執行個體大小，您可以：

- 水平擴展並新增其他執行個體。然後，在這些執行個體上放置其他任務，或者
- 啟動新的較大/較小執行個體，然後耗盡舊執行個體，以垂直擴展。

這兩種方法都有助於將應用程式可用性的影響降至最低。

如果您使用其他方法來變更執行個體，您可能會收到下列錯誤：

```
Container instance type changes are not supported.
```

當您收到此錯誤時，請執行下列步驟：

1. 啟動具有所需執行個體類型的新執行個體。
2. 耗盡舊執行個體類型。如需詳細資訊，請參閱 [the section called “容器執行個體耗盡”](#)。

Amazon ECS EC2 容器執行個體

Amazon ECS 代理程式是在每個已註冊至叢集的容器執行個體上執行的程序。該程序有助於容器執行個體與 Amazon ECS 之間的通訊。

 Note

在 Linux 容器執行個體上，代理程式容器會掛載最上層目錄，例如 `/lib`、`/lib64` 與 `/proc`。這對於 Amazon EBS 磁碟區、awsipc 網路模式、Amazon ECS Service Connect 與 FireLens for Amazon ECS 等 ECS 功能是必要的。

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](#)」。

Amazon ECS 容器代理程式包含 `amazon-ecs-pause` 映像。Amazon ECS 會將此映像用於採用 awsipc 網路模式的任務。

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 18.01.0 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。

如果您將容器執行個體的状态變更為 `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 容器執行個體](#)。

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。

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](#)。

Docker 支援

Amazon ECS 支援在 Amazon Linux 上發布的 Docker 最近兩個主要版本。目前包括 Docker 20.10.x 與 Docker 25.x。

您可以在 GitHub 的 [Amazon ECS 代理程式規格檔案](#) 中找到 Amazon ECS 所需的最低 Docker 版本。

使用 Amazon ECS 最佳化 AMI 時，會預先安裝 Docker 並設定為搭配 Amazon ECS 容器代理程式使用。AMI 包含經 Amazon ECS 測試且支援的 Docker 版本。

Note

雖然 Amazon ECS 支援多個 Docker 版本，但建議使用 Amazon ECS 最佳化 AMI 內建的 Docker 版本，以取得最佳相容性與技術支援。

Amazon ECS 最佳化 AMI

如需 Amazon ECS 最佳化 AMI 的詳細資訊，請參閱 [Amazon ECS 最佳化 Linux AMIs](#)。

其他資訊

下列頁面提供變更的其他資訊：

- GitHub 上的 [Amazon ECS 代理程式變更記錄](#)
- [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

適用對象：EC2 執行個體

Amazon ECS 容器代理程式支援數個組態選項，其中大部分透過環境變數進行設定。

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。您也可以在此啟動時使用 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 容器執行個體傳遞資料](#)。

如果是透過 Windows 版的 Amazon ECS 最佳化 AMI 來啟動容器執行個體，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 User Guide 中的[Run commands when you launch an EC2 instance with user data input](#)，以及[the section called “引導容器執行個體”](#)。

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
```

透過主機 PID 命名空間執行 Amazon ECS 代理程式

依預設，Amazon ECS 代理程式會透過專用的 PID 命名空間執行。在下列組態中，您可以設定 Amazon ECS 代理程式採用主機 PID 命名空間執行：

- SELinux 強制執行模式已啟用。
- Docker 的 SELinux 安全政策設定為 `true`。

您可以在 `/etc/ecs/ecs.config` 檔案中將 `ECS_AGENT_PID_NAMESPACE_HOST` 環境變數設定為 `true`，實現對此行為進行設定。啟用此變數時，`ecs-init` 會使用主機 PID 命名空間 (`--pid=host`) 啟動 Amazon ECS 代理程式容器，允許代理程式在 SELinux 強制執行環境中正確引導自身。此功能在 Amazon ECS 代理程式版本 1.94.0 及更新版本中可用。

若要啟用此功能，請將以下一行新增至 `/etc/ecs/ecs.config` 檔案：

```
ECS_AGENT_PID_NAMESPACE_HOST=true
```

進行此變更後，請重新啟動 Amazon ECS 代理程式，讓變更生效：

```
sudo systemctl restart ecs
```

若啟用了 SELinux 強制模式，且 Docker 安全政策設定為 `true`，那麼即使設定了 `ECS_AGENT_PID_NAMESPACE_HOST=true`，下列功能也無法運作。

- Amazon ECS Exec
- Amazon EBS 任務連接
- Service Connect
- FireLens for Amazon ECS

可用參數

如需有關可用 Amazon ECS 容器代理程式組態參數的資訊，請參閱 GitHub 上的 [Amazon ECS 容器代理程式](#)。

將 Amazon ECS 容器執行個體組態儲存於 Amazon S3 中

使用上述環境變數控制 Amazon ECS 容器代理程式組態。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。某些不敏感的環境變數 (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，且不會造成任何問題。不過，其他敏感資訊，例如您的 AWS 登入資料或 `ECS_ENGINE_AUTH_DATA` 變數，絕不應該傳遞至使用者資料中的執行個體，或以允許它們出現在 `.bash_history` 檔案中 `/etc/ecs/ecs.config` 的方式寫入。

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，AWS CLI 並在執行個體啟動 `/etc/ecs/ecs.config` 時將您的組態資訊複製到。

若要在 Amazon S3 中存放 **`ecs.config`** 檔案

1. 您必須授予容器執行個體角色 (`ecsInstanceRole`) 許可，才能取得 Amazon S3 的唯讀存取權。您可以將 `AmazonS3ReadOnlyAccess` 指派給 `ecsInstanceRole` 角色來執行此操作。如需有關

如何將政策連接至角色的資訊，請參閱 AWS Identity and Access Management User Guide 中的 [Update permissions for a role](#)

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 GitHub 上的 [Amazon ECS 容器代理程式](#)。

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 User Guide 中的 [Creating a bucket](#)。
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 [上傳物件](#)。

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 使用者資料中使用如下範例指令碼。指令碼會安裝 `awscli`，AWS CLI 並將您的組態檔案複製到 `/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

若想向 Amazon ECS 叢集註冊 Amazon EC2 執行個體，且該執行個體未使用基於 Amazon ECS 最佳化 AMI 的 AMI，您可以透過如下程序手動安裝 Amazon ECS 容器代理程式。為此，您可以從其中一

個區域 S3 儲存貯體或 Amazon Elastic Container Registry Public 下載代理程式。如果從其中一個區域 Amazon S3 儲存貯體下載，您可選擇性使用 PGP 簽章驗證容器代理程式的有效性。

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 儲存貯體中下載代理程式並加以安裝。

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 [GitHub 上的 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 x86_64 (x86_64)
		Amazon ECS init arm64 (arm64)	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 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)
us-west-1	美國西部 (加利佛尼亞北部)	Amazon ECS init amd64 (amd64)	Amazon ECS init x86_64 (x86_64)
		Amazon ECS init arm64 (arm64)	Amazon ECS aarch64 (aarch64)
us-west-2	美國西部 (奧勒岡)	Amazon ECS init amd64 (amd64)	Amazon ECS init x86_64 (x86_64)
		Amazon ECS init arm64 (arm64)	Amazon ECS aarch64 (aarch64)
ap-east-1	亞太地區 (香港)	Amazon ECS init amd64 (amd64)	Amazon ECS init x86_64 (x86_64)
		Amazon ECS init arm64 (arm64)	Amazon ECS aarch64 (aarch64)
ap-northeast-1	亞太地區 (東京)	Amazon ECS init amd64 (amd64)	Amazon ECS init x86_64 (x86_64)
		Amazon ECS init arm64 (arm64)	Amazon ECS aarch64 (aarch64)
ap-northeast-2	亞太地區 (首爾)	Amazon ECS init amd64 (amd64)	Amazon ECS init x86_64 (x86_64)
		Amazon ECS init arm64 (arm64)	Amazon ECS aarch64 (aarch64)

區域	區域名稱	Amazon ECS init deb 檔案	Amazon ECS init rpm 檔案
ap-south-1	亞太地區 (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	亞太地區 (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	亞太地區 (悉尼)	Amazon ECS init amd64 (amd64) Amazon ECS init arm64 (arm64)	Amazon ECS init x86_64 (x86_64) Amazon ECS aarch64 (aarch64)
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)

區域	區域名稱	Amazon ECS init deb 檔案	Amazon ECS init rpm 檔案
eu-west-2	歐洲 (倫敦)	Amazon ECS init amd64 (amd64)	Amazon ECS init x86_64 (x86_64)
		Amazon ECS init arm64 (arm64)	Amazon ECS aarch64 (aarch64)
eu-west-3	Europe (Paris)	Amazon ECS init amd64 (amd64)	Amazon ECS init x86_64 (x86_64)
		Amazon ECS init arm64 (arm64)	Amazon ECS aarch64 (aarch64)
sa-east-1	南美洲 (聖保羅)	Amazon ECS init amd64 (amd64)	Amazon ECS init x86_64
		Amazon ECS init arm64 (arm64)	Amazon ECS aarch64 (aarch64)
us-gov-east-1	AWS GovCloud (美國東部)	Amazon ECS init amd64 (amd64)	Amazon ECS init x86_64 (x86_64)
		Amazon ECS init arm64 (arm64)	Amazon ECS aarch64 (aarch64)
us-gov-west-1	AWS GovCloud (美國西部)	Amazon ECS init amd64 (amd64)	Amazon ECS init x86_64 (x86_64)
		Amazon ECS init arm64 (arm64)	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。如需有關 Docker 支援的詳細資訊，請參閱 [Amazon ECS EC2 容器執行個體](#)。

```
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] 區段結尾處新增以下一行。

```
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[將 Amazon ECS 容器執行個體組態儲存於 Amazon S3 中](#)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。如需有關所有可用組態參數的詳細資訊，請參閱 [Amazon ECS 容器代理程式組態](#) 或 GitHub 上的 [Amazon ECS Agent README](#)。

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"}
```

為私有 Docker 映像檔設定 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 的任務才支援此功能。

另一種啟用私有登錄檔身分驗證的方法，是使用 AWS Secrets Manager 安全地存放私有登錄檔登入資料，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 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 [將 Amazon ECS 容器執行個體組態儲存於 Amazon S3 中](#)。

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.com":
{"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 “容器自我檢查”](#)。

```
curl http://localhost:51678/v1/metadata
```

Amazon ECS 任務與映像自動清理

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`

等待刪除已停止任務容器的預設時間。如果該值設定為低於 1 秒，則會略過該值。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)。

Note

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。待容器移除後，所有未被引用的映像都將根據映像清理組態參數，成為待清理候選對象。

`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 上為容器排程

Amazon Elastic Container Service (Amazon ECS) 是一個共用狀態的開放式並行存取系統，可為容器化工作負載提供彈性排程功能。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。該系統還針對批次任務或單一執行任務提供執行獨立任務或排程任務的功能。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	服務排程器適合用於長時間執行的無狀態服務及應用程式。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	獨立任務適合用於如批次任務這類執行工作然後停止的程序。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	排程任務適用於下列情況： 若叢集中已有依照設定時間間隔來執行的任務，可以使用 EventBridge 排程器來建立排程。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時間執行叢集中的一個或多個任務。您的排程事件可以設定為特定的時間間隔 (每隔 <i>N</i> 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 排程器排程 Amazon ECS 任務

運算選項

透過 Amazon ECS，您可以指定任務或服務執行所在的基礎結構。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 與 Fargate Spot。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任務分佈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。

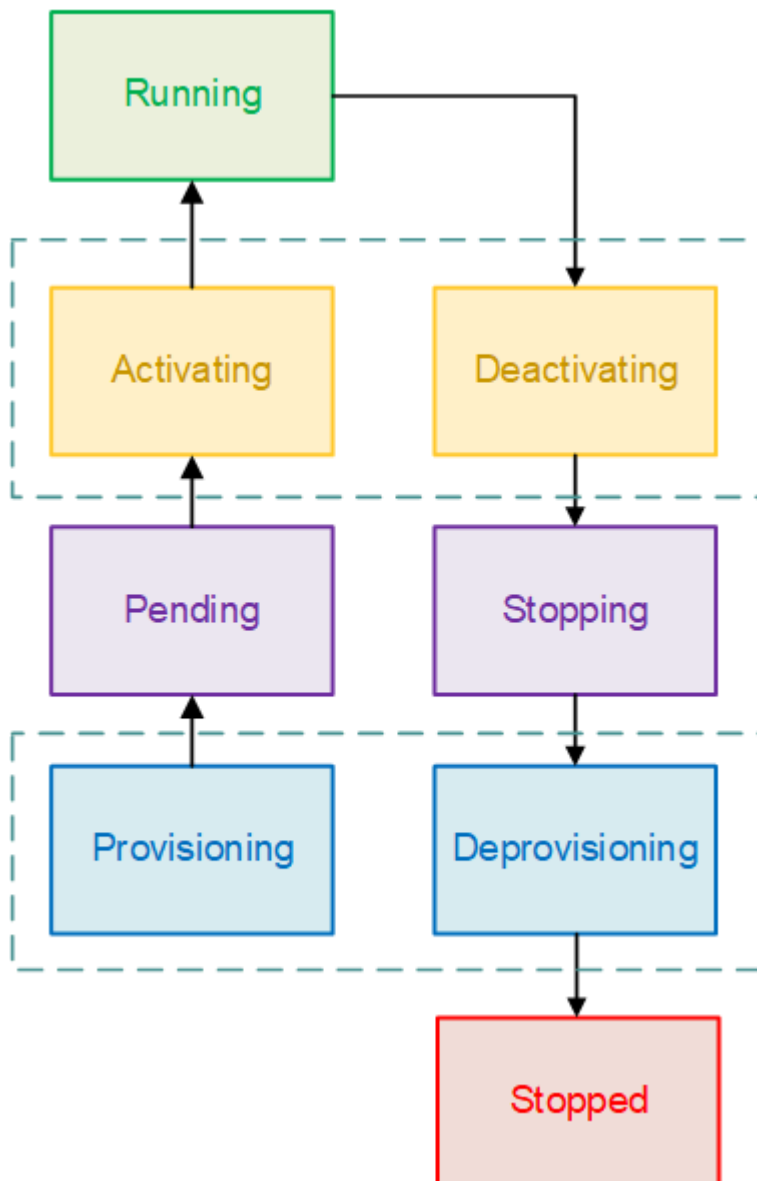
運算選項會直接在 Fargate 上啟動任務，或在向叢集手動註冊的 Amazon EC2 執行個體上啟動任務。

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。在此狀態下，Amazon ECS 會提取容器映像、建立容器、設定任務聯網、註冊負載平衡器目標群組，以及設定服務探索。

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，任務若屬於設定為使用 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，容器代理程式會傳送容器映像中定義的停止訊號，以通知應用程式需要使用 STOPSIGNAL 指示完成和關閉。這是預設 SIGTERM 的。然後，它會在等待任務定義中設定的 StopTimeout 持續時間 SIGKILL 後傳送。

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。

如果任務因錯誤而停止，請參閱[檢視 Amazon ECS 已停止任務錯誤](#)。

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置放任務

您可以使用任務置放來設定 Amazon ECS，將任務置放在符合特定條件 (例如可用區域或執行個體類型) 的容器執行個體上。

下列是任務置放元件：

- 任務置放策略 – 用於選取容器執行個體進行任務置放或選取任務進行終止的演算法。例如，Amazon ECS 可以隨機選取容器執行個體，也可以選取讓任務均勻分佈到一組執行個體的容器執行個體。
- 任務群組 – 一組相關任務，例如資料庫任務。
- 任務置放限制條件 – 必須符合這些規則才能在容器執行個體上置放任務。若不符合限制條件，則任務不會置放且保持 PENDING 狀態。例如，您可以使用限制條件，僅將任務置放在特定執行個體類型上。

Amazon ECS 針對不同容量選項有不同的演算法。

Amazon ECS 受管執行個體

對於在 Amazon ECS 受管執行個體上執行的任務，Amazon ECS 必須決定任務的置放位置，以及縮減任務計數時要終止的任務。Amazon ECS 會根據容量提供者啟動範本中指定的執行個體需求、任務定義中指定的需求 (例如 CPU 與記憶體)，以及任務置放限制條件來作出此決定。

Note

Amazon ECS 受管執行個體不支援任務置放策略。Amazon ECS 將盡力將任務分散至可存取的可用區域。

Amazon ECS 放置任務時，會使用下列程序來選取容器執行個體：

1. 識別滿足任務定義中 CPU、GPU、記憶體與連接埠需求的容器執行個體。
2. 識別滿足任務置放限制條件的容器執行個體。
3. 識別滿足容量提供者啟動範本中指定執行個體需求的容器執行個體。
4. 選取容器執行個體進行任務置放。

EC2

對於採用 EC2 啟動類型的任務，Amazon ECS 必須根據任務定義中所指定的需求 (例如 CPU 與記憶體)，決定任務置放位置。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。

預設任務置放策略取決於是手動執行任務 (獨立任務) 還是在某項服務中執行任務。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。不在服務中的任務沒有預設任務置放限制條件。如需詳細資訊，請參閱[在 Amazon ECS 上為容器排程](#)。

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. 識別滿足任務定義中 CPU、GPU、記憶體與連接埠需求的容器執行個體。
2. 識別滿足任務置放限制條件的容器執行個體。
3. 識別滿足任務置放策略的容器執行個體。
4. 選取容器執行個體進行任務置放。

Fargate

使用 Fargate 的任務不支援任務置放策略與限制條件。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。

Amazon ECS 受管執行個體自動擴展與任務置放

Amazon ECS 受管執行個體使用智慧型演算法自動擴展叢集容量，並在整個基礎結構中有效率地置放任務。了解這些演算法的運作方式可協助您最佳化服務組態，並對置放行為進行疑難排解。

任務置放演算法

Amazon ECS 受管執行個體使用複雜的置放演算法，在排程任務時平衡可用性、資源使用率與網路需求。

可用區域分佈

依預設，Amazon ECS 受管執行個體會將任務分佈到多個可用區域，以便優先考慮可用性：

- 對於具有多項任務的服務，Amazon ECS 受管執行個體會盡可能確保分佈在不同可用區域中的至少 3 個執行個體
- 此行為提供了容錯能力，但可能導致每個執行個體的資源使用率降低
- 優先考慮可用區域分佈，而非裝箱最佳化

裝箱行為

雖然 Amazon ECS 受管執行個體可以執行裝箱以最大化資源使用率，但此行為會受到網路組態影響：

- 若要實現裝箱，請將服務設定為使用單一子網路
- 多子網路組態會優先考慮可用區域分佈，而非資源密度
- 裝箱更可能發生在初始服務啟動期間，而非事件擴展期間

ENI 密度考量

對於使用 `awsvpc` 網路模式的服務，Amazon ECS 受管執行個體在作出置放決策時會考慮彈性網路介面 (ENI) 密度：

- `awsvpc` 模式中的每項任務都需要專用的 ENI
- 執行個體類型具有影響任務密度的不同 ENI 限制
- 選取目標執行個體時，Amazon ECS 受管執行個體會考量 ENI 可用性

Note

為最佳化置放決策，我們正在持續改進 ENI 密度計算。

容量提供者決策邏輯

Amazon ECS 受管執行個體容量提供者會基於多種因素作出擴展與置放決策：

資源需求

待定任務的 CPU、記憶體與網路需求

執行個體可用性

現有執行個體的目前容量與使用率

網路限制條件

子網路組態與 ENI 可用性

可用區域分佈

跨多個可用區域維持容錯能力

組態選項

子網路選取策略

子網路組態會顯著影響任務置放行為：

多個子網路 (預設)

優先考慮可用區域分佈以實現高可用性

可能會導致每個執行個體的資源使用率降低

建議用於需要容錯能力的生產工作負載

單一子網路

啟用裝箱以提高資源使用率

將任務集中在一個可用區域，以降低容錯能力

適用於開發或成本最佳化工作負載

網路模式考量

您選擇的網路模式會影響置放決策：

- awsvpc 模式 – 每項任務都需要專用 ENI，用於限制每個執行個體的任務密度
- host 模式 – 任務會直接使用主機的網路，置放主要由資源可用性驅動

CPU 架構考量事項

`cpuArchitecture` 您在任務定義中指定的 用於在特定架構上放置任務。如果您未指定 `cpuArchitecture`，Amazon ECS 會根據容量提供者組態，嘗試在任何可用的 CPU 架構上放置任務。您可指定為 X86_64 或 ARM64。

對任務置放進行疑難排解

常見置放模式

了解預期的置放模式有助於區分正常行為與潛在問題：

分散式分佈

在部分利用的多個執行個體上分佈任務

使用多個子網路時的正常行為

表示優先考慮可用性而非資源效率

集中式置放

在使用率較高的較少執行個體上置放多項任務

使用單一子網路組態時的預期行為

可能在初始服務啟動期間發生

不均勻分佈

某些執行個體使用率很高，而某些執行個體未得到充分利用

可能表示 ENI 限制或資源限制條件

建議檢閱執行個體類型與網路組態

最佳化置放行為

若要針對特定需求最佳化任務置放：

1. 評估可用性需求與成本最佳化需求
2. 根據優先順序選擇適當的子網路組態
3. 為網路模式選取具有足夠 ENI 容量的執行個體類型
4. 監控置放模式並視需要調整組態

最佳實務

- 對於生產工作負載 – 跨不同可用區域使用多個子網路以確保高可用性，接受資源使用率方面的權衡
- 對於開發或測試 – 考慮使用單一子網路組態，最大化資源使用率並降低成本

- 對於 **awsvpc** 模式 – 選擇具有足夠 ENI 容量的執行個體類型，避免出現置放限制條件
- 對於成本最佳化 – 監控使用率模式並調整服務組態，平衡可用性與效率
- 對於疑難排解 – 調查意外置放模式時，請檢閱子網路組態與網路模式

使用策略來定義 Amazon ECS 任務置放

對於採用 EC2 啟動類型的任務，Amazon ECS 必須根據任務定義中所指定的需求 (例如 CPU 與記憶體)，決定任務置放位置。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。

預設任務置放策略取決於是手動執行任務 (獨立任務) 還是在某項服務中執行任務。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。不在服務中的任務沒有預設任務置放限制條件。如需詳細資訊，請參閱[在 Amazon ECS 上為容器排程](#)。

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. 識別滿足任務定義中 CPU、GPU、記憶體與連接埠需求的容器執行個體。
2. 識別滿足任務置放限制條件的容器執行個體。
3. 識別滿足任務置放策略的容器執行個體。
4. 選取容器執行個體進行任務置放。

您可以使用 `placementStrategy` 參數，在服務定義或任務定義中指定任務置放策略。

```
"placementStrategy": [  
  {  
    "field": "The field to apply the placement strategy against",  
    "type": "The placement strategy to use"  
  }  
]
```

]

您可以在執行任務 ([RunTask](#))、建立新服務 ([CreateService](#)) 或更新現有服務 ([UpdateService](#)) 時指定策略。

下表說明可用的類型與欄位。

type	有效欄位值	
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 將相關的 Amazon ECS 任務分在同一組。	- 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 <code>attribute:ecs.availability-zone</code>	

type	有效欄位值	
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置放任務](#)。

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 任務置放策略範例](#)。

Amazon ECS 任務置放策略範例

您可以使用下列動作指定任務置放策略：[CreateService](#)、[UpdateService](#) 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"
  }
]
```

將相關的 Amazon ECS 任務分在同一組

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[使用策略來定義 Amazon ECS 任務置放](#)。

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 任務置放限制條件範例](#)。

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。

任務群組需符合下列需求。

- 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 用於任務的容器執行個體

任務置放限制條件是有關容器執行個體的規則，Amazon ECS 透過其判斷是否允許任務在該執行個體上執行。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置放限制條件。

```
"placementConstraints": [  
  {  
    "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)	可指定的時機	
distinctInstance	• 執行任務 RunTask • 建立新的服務 CreateService	
將每個作用中任務置放在不同的容器執行個體上。	vice ,	

Constraint type (限制條件類型)	可指定的時機	
Amazon ECS 會查看任務置放所需的任務狀態。例如，如果現有任務的所需狀態為 STOPPED (但上次狀態不是)，則儘管存在 <code>distinctInstance</code> 置放限制條件，新的傳入任務仍可置放於同一執行個體上。因此，您可能會在同一執行個體上看到 2 項上次狀態為 RUNNING 的任務。  Important 建議需要為任務實現嚴格隔離的客戶採用 Fargate。Fargate 會在硬體虛擬化環境中執行每個任務。此舉可確保這些容器化工作負載，不會與其他任務共用網路介面、Fargate 暫時性儲存、CPU 或記憶體。如需詳細資訊，請參閱 的安全概觀 AWS Fargate。		
<code>memberOf</code> 在滿足運算式的容器執行個體上放置任務。	- 執行任務 RunTask - 建立新的服務 CreateService， - 建立新的任務定義 RegisterTaskDefinition - 建立任務定義的新修訂版 RegisterTaskDefinition - 更新服務 UpdateService	

使用 `memberOf` 限制條件類型時，可以使用叢集查詢語言來建立表達式，該語言定義了 Amazon ECS 可以置放任務的容器執行個體。表達式可讓您依屬性將容器執行個體分組。表達式位於 `placementConstraint` 的 `expression` 參數中。

Amazon ECS 容器執行個體屬性

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`、`WINDOWS_SERVER_2016_FULL`。

這對 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必須符合主機的作業系統版本。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 ECS 最佳化 Windows 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`。

Note

Amazon ECS 受管執行個體支援下列屬性子集：

- `ecs.subnet-id`
- `ecs.availability-zone`
- `ecs.instance-type`
- `ecs.cpu-architecture`

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 容器執行個體網路介面數量](#)。

`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 “上的 Amazon Elastic Container Service 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[外部執行個體的 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。

建立表達式以定義 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 容器執行個體屬性](#)。

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 [將應用程式作為 Amazon ECS 任務執行](#) 和 [建立 Amazon ECS 滾動更新部署](#)。您也可以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 容器執行個體屬性](#)。

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[將相關的 Amazon ECS 任務分在同一組](#)。

運算子

比較運算子。下列是支援的運算子。

運算子	Description
==、equals	字串相等
!=、not_equals	字串不相等

運算子	Description
>、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](https://docs.oracle.com/javase/7/docs/api/java/util/regex/Pattern.html)。

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 任務置放限制條件範例

下列是任務置放條件限制的範例。

此範例使用 `memberOf` 限制條件在 T2 執行個體上置放任務。可以使用下列動作來指定：[CreateService](#)、[UpdateService](#)、[RegisterTaskDefinition](#) 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 [將相關的 Amazon ECS 任務分在同一組](#)。可以使用下列動作來指定：[CreateService](#)、[UpdateService](#)、[RegisterTaskDefinition](#) 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。可以使用下列動作來指定：[CreateService](#)、[UpdateService](#) 和 [RunTask](#)。

Amazon ECS 會查看任務置放所需的任務狀態。例如，如果現有任務的所需狀態為 `STOPPED` (但上次狀態不是)，則儘管存在 `distinctInstance` 置放限制條件，新的傳入任務仍可置放於同一執行個體上。因此，您可能會在同一執行個體上看到 2 項上次狀態為 `RUNNING` 的任務。

```
"placementConstraints": [
  {
    "type": "distinctInstance"
  }
]
```


]

Amazon ECS 獨立任務

當您的應用程式執行一些工作，然後停止，例如批次程序時，您可以將應用程式做為任務執行。如果您想要執行任務一次，您可以使用 主控台、AWS CLI、APIs 或 SDKs。

如果需要以速率、Cron 或一次性排程為基礎執行應用程式，可以使用 EventBridge 排程器建立排程。

任務工作流程

當您啟動 Amazon ECS 任務 (獨立任務或 Amazon ECS 服務) 時，系統會建立任務並最初將其移至 PROVISIONING 狀態。當任務處於 PROVISIONING 狀態時，任務與容器都不存在，因為 Amazon ECS 需要尋找運算容量來置放任務。

Amazon ECS 會根據您的啟動類型或容量提供者組態，為您的任務選取適當的運算容量。Fargate 與 EC2 都可以搭配容量提供者及容量提供者策略使用。使用 Fargate 時，無需考慮佈建、設定與擴展叢集容量。Fargate 會為您的任務處理所有基礎設施管理。對於 EC2，您可以藉由將 Amazon EC2 執行個體註冊至叢集來管理叢集容量，也可以使用叢集自動擴展功能來簡化運算容量管理。叢集自動擴展負責動態擴展叢集容量，讓您可以專注於執行任務。Amazon ECS 會根據您在任務定義中指定的需求，例如 CPU 與記憶體，以及置放限制條件與策略，來決定任務置放位置。如需詳細資訊，請參閱 [Amazon ECS 如何在容器執行個體上置放任務](#)。

如果您使用已啟用受管擴展的容量提供者，由於運算容量不足而無法啟動的任務會移至 PROVISIONING 狀態，而不會立即失敗。找到用於置放任務的容量後，Amazon ECS 會佈建必要的附件，例如 awsvpc 模式中任務的彈性網路介面 (ENI)。該附件會使用 Amazon ECS 容器代理程式來提取容器映像，然後啟動容器。佈建完成且相關容器啟動後，Amazon ECS 會將任務移至 RUNNING 狀態。如需有關任務狀態的資訊，請參閱 [Amazon ECS 任務生命週期](#)。

將應用程式作為 Amazon ECS 任務執行

您可以透過 AWS 管理主控台為一次性程序建立任務。

建立獨立任務 (AWS 管理主控台)

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。根據選擇的資源頁面，使用下列步驟建立獨立任務。

啟動服務的資源	步驟	
叢集詳細資訊頁面	- 在叢集頁面，選取您要在哪個叢集中建立服務。 - 在任務索引標籤中，選擇執行任務。	
任務定義修訂版頁面	- 在任務定義頁面上，選擇任務定義系列以顯示該系列的修訂版。 - 選取要使用的修訂版。 - 從部署功能表中，選擇執行任務。	

- 在現有叢集欄位中選擇叢集。

選擇建立叢集以在新叢集上執行任務

- 選擇任務在叢集基礎結構中的分佈方式。在運算組態下，選擇選項。若要使用容量提供者策略，必須在叢集層級設定容量提供者。

如果尚未將叢集設定為使用容量提供者，請改用啟動類型。

如果想要在 Amazon ECS 受管執行個體上執行工作負載，必須使用容量提供者策略選項。

分發方法	步驟	
容量提供者策略	- 在 Compute options (運算選項) 區段中，選取 Capacity provider strategy (容量提供者策略)。 - 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	

分發方法	步驟	
	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。	
啟動類型	- 在運算選項區段中，選取啟動類型。 - 針對 啟動類型，選擇啟動類型。 (選用) 使用 Fargate 時，在平台版本欄位中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。	

5. 在部署組態下，執行下列動作：

- 在任務定義欄位中輸入任務定義。

⚠ 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。

- b. 針對 Desired tasks (所需任務)，輸入要啟動的任務數目。
 - c. 在任務群組欄位中輸入任務群組名稱。
6. 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。
 - 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 在公有子網路中執行時，任務可以指派公有 IP 地址，以便他們有網際網路的路由。無法使用此欄位將公有 IP 指派給 EC2 任務。如需詳細資訊，請參閱 [Fargate 的 Amazon ECS 任務聯網選項](#)，以及 [為 Amazon ECS 任務配置網路介面](#)。

7. 如果任務使用的資料磁碟區與部署時的組態相容，則可透過展開磁碟區區段對磁碟區進行設定。

磁碟區名稱與磁碟區類型會在建立任務定義修訂版時設定，且無法在執行獨立任務時變更。若要更新磁碟區名稱與類型，必須建立新的任務定義修訂版，並使用新修訂版執行任務。

設定此磁碟區類型	執行此作業	
Amazon EBS	a. 在 EBS 磁碟區類型欄位中，選擇要連接至任務的 EBS 磁碟區類型。 b. 在大小 (GiB) 欄位中，輸入磁碟區大小的有效值 (以 GiB 為單位)。您可以指定磁碟區大小 (大小下限 1 GiB，大小上限 16,384 GiB)。除非提供快照 ID，否則需要此值。	

設定此磁碟區類型	執行此作業	
	c. 在 IOPS 欄位中，輸入磁碟區應提供的輸入/輸出操作次數 (IOPS) 上限。此值僅可針對 io1、io2 與 gp3 磁碟區類型設定。 d. 在 輸送量 (MiB/s) 欄位中，輸入磁碟區應提供的輸送量 (以 MiBps 或 MiB/s 為單位)。此值僅可針對 gp3 磁碟區類型設定。 e. 在快照 ID 欄位中，若想從快照建立磁碟區，請選擇現有的 Amazon EBS 磁碟區快照或輸入快照的 ARN。您也可以不選擇或不輸入快照 ID，建立新的空白磁碟區。 f. 如果指定了快照 ID，則可指定磁碟區初始化速率 (MiB/s)。輸入一個介於 100 至 300 MiB/s 之間的值，該值將決定從使用快照 ID 指定的快照在建立磁碟區時載入資料的速度。 g. 在終止政策欄位中，如果希望在任務終止後保留為連接至任務而設定的磁碟區，請取消選取核取方塊。依預設，在任務終止後，會刪除連接至任務的 EBS 磁碟區。 h. 在檔案系統類型欄位中，選擇用於磁碟區上資料儲存與擷取的檔案系統類	

設定此磁碟區類型	執行此作業	
	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。如果有檔案系統類型不相符，則任務將無法啟動。 i. 在基礎結構角色欄位中，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 <code>AmazonECSInfrastructureRolePolicyForVolumes</code> 受管政策連接至該角色，也可參照該政策建立並連接自己的政策，該政策需具有符合特定需求的許可。如需有關必要許可的詳細資訊，請參閱 Amazon ECS 基礎結構 IAM 角色。 j. 若想依預設設定使用 Amazon EBS 加密，請在加密欄位中選擇預設值。如果您的帳戶已依預設設定加密，則會使用設定中指定的 AWS Key Management Service (AWS KMS) 金鑰來加密磁碟區。如果選擇預設，但	

設定此磁碟區類型	執行此作業	
	未開啟 Amazon EBS 預設加密，則磁碟區將取消加密。 如果您選擇自訂，您可以為磁碟區加密指定 AWS KMS key 您選擇的。 如果選擇無，除非您已設定為預設加密，或者從加密快照建立磁碟區，否則磁碟區將不會加密。 k. 如果您已選擇自訂加密，則必須指定要使用 AWS KMS key 的。在 KMS 金鑰欄位中選擇 AWS KMS key 或輸入金鑰 ARN。如果您選擇使用對稱客戶受管金鑰來加密磁碟區，請確定您已在 AWS KMS key 政策中定義正確的許可。如需詳細資訊，請參閱 Amazon EBS 磁碟區的資料加密。 l. (選用) 在標籤下，您可以透過從任務定義傳播標籤或提供自己的標籤，將標籤新增至 Amazon EBS 磁碟區。 如果要從任務定義傳播標籤，請在傳播標籤來源欄位中選擇任務定義。如果選擇不傳播，或者未選	

設定此磁碟區類型	執行此作業	
	擇任何值，則不會傳播標籤。 如果要提供自己的標籤，請選擇新增標籤，然後為每個新增的標籤提供鍵與值。 如需有關標記 Amazon EBS 磁碟區的詳細資訊，請參閱標記 Amazon EBS 磁碟區。	

8. (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置放任務](#)。

- AZ 平衡分散 - 在可用區域及可用區域中的容器執行個體之間分散任務。
- AZ 平衡 BinPack - 使用最低可用記憶體，在可用區域及可用區域中的容器執行個體之間分散任務。
- BinPack - 根據最低可用的 CPU 或記憶體數分散任務。
- 每個主機一個任務 - 在每個容器執行個體上最多置放一個服務的任務。
- 自訂 - 定義自己的任務置放策略。

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。

9. (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](#)。

10. (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 (命令覆寫)，請輸入 Docker 命令。
- 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 (" ") 括住字串，並以下列格式將字串傳遞至容器：

```
MY_ENV_VAR="This variable contains a string."
```

11. (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。

12. 選擇建立。

使用 Amazon EventBridge 排程器排程 Amazon ECS 任務

EventBridge 排程器是無伺服器排程器，可讓您從單一受管的中央服務建立、執行及管理任務。其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 排程器具有高度可自訂性，並透過 EventBridge 排程規則改善可擴展性，提供更廣泛的目標 API 操作和 AWS 服務。EventBridge 排程器提供下列排程，您可以在 EventBridge 排程器主控台中為您的任務設定這些排程：

- 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 EventBridge 排程器排程 Amazon ECS。

雖然您可以在 Amazon ECS 主控台中建立排程任務，但目前 EventBridge 排程器主控台提供更多功能。

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 [VPC 的子網路](#)，以及 [AWS 使用安全群組控制資源的流量](#)。
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](#)。
3. 若要使用容量提供者策略執行任務，容量提供者必須與叢集相關聯。

使用主控台建立新排程

1. 前往 <https://console.aws.amazon.com/scheduler/home> 開啟 Amazon EventBridge 排程器。
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. 在排程群組欄位中選擇排程群組。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。 如需 Cron 和 Rate 運算式的詳細資訊，請參閱 《Amazon EventBridge 排程器使用者指南》中的 EventBridge 排程器上的排程類型。 b. 對於彈性時段，選擇關閉可關閉此選項，或者也	

頻率	執行此作業...
	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. 選擇任務在叢集基礎結構中的分佈方式。展開運算選項區段，然後選擇下列其中一個選項

運算選項	步驟
容量提供者策略	a. 選擇容量提供者策略。 b. 選擇策略： • 若要使用預設的容量提供者策略，選擇使用叢集預設值。 • 若要使用自訂策略，請選擇使用自訂。然後，

運算選項	步驟	
	輸入基礎、容量提供者與權重。 對於 EC2，容量提供者是 Auto Scaling 群組。	
啟動類型	a. 選擇啟動類型。 b. 針對 啟動類型，選擇啟動類型。 c. 當指定 Fargate 時，在平台版本欄位中指定要使用的平台版本。	

- 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置放任務](#)。

- 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。
 - 在事件的最大保留時間中，輸入 EventBridge 排程器保留未處理事件的最大時數和分鐘數。
 - 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 排程器重新嘗試執行排程的次數上限。

最大值為重試 185 次。

設定好重試政策後，如果排程無法調用其目標，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 排程器儲存未交付事件的位置。

無效字母佇列 (DLQ) 選項	執行此作業...
不儲存	選擇無。
將事件存放在您建立排程的相同 AWS 帳戶 中	a. 選擇在 中選取 Amazon SQS 佇列 AWS 帳戶 做為 DLQ。 b. 選擇 Amazon SQS 佇列的 Amazon Resource Name (ARN)。
將事件存放在與您要建立排程 AWS 帳戶 的位置不同的 中	a. 選擇將其他 中的 Amazon SQS 佇列 AWS 帳戶 指定為 DLQ。 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 排程器會將範本化目標所需的必要許可與角色連接。

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 AWS CLI 來管理排程。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

若不再需要讓獨立任務持續執行，可以將其停止。Amazon ECS 主控台可讓您輕鬆停止一項或多項任務。

您無法重新啟動已停止的獨立任務。

如果想要停止服務，請參閱[使用主控台刪除 Amazon ECS 服務](#)。

停止獨立任務 (AWS 管理主控台)

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，選擇任務索引標籤。
5. 您可以使用篩選啟動類型清單，依啟動類型篩選任務。

要停止的任務	步驟
一個或多個	a. 選取任務，然後選擇停止、停止已選取。 b. 在停止任務確認頁面上，選擇停止

要停止的任務	步驟	
全部	 Important 如果您選擇使用主控台停止所有任務，Amazon ECS 會停止所有獨立任務以及屬於服務一部分的任務。因此，建議謹慎使用此選項。 - 選擇停止 > 全部停止。 - 在停止任務確認頁面上，輸入停止所有任務，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。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。

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。然後會發生下列情況。

- 如果替代任務的運作狀態為 `HEALTHY`，服務排程器會停止運作狀態不良的任務

- 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。如需詳細資訊，請參閱[Amazon ECS 服務限流邏輯](#)及[檢視 Amazon ECS 服務事件訊息](#)。

基礎結構運算選項

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。

如果想要在 Amazon ECS 受管執行個體上執行工作負載，必須使用容量提供者策略選項。

針對容量提供者策略，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，則系統會選取使用 FARGATE 容量提供者的自訂容量提供者策略。
- 如果叢集未定義預設容量提供者策略，但您已將一個或多個 Auto Scaling 群組容量提供者新增至叢集，則系統會選取使用自訂 (進階) 選項，並且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。
- 啟動類型便於 Amazon ECS 直接在 Fargate 或註冊至叢集的 EC2 執行個體上啟動任務。

如果想要在 Amazon ECS 受管執行個體上執行工作負載，必須使用容量提供者策略選項。

依預設，服務會在叢集 VPC 的子網路中啟動。

服務自動擴展

服務自動擴展是一項自動增加或減少 Amazon ECS 服務中所需任務數量的功能。Amazon ECS 利用 Application Auto Scaling 服務來提供此功能。

如需詳細資訊，請參閱[自動擴展您的 Amazon ECS 服務](#)。

服務負載平衡

在上託管的 Amazon ECS 服務 AWS Fargate 支援 Application Load Balancer、Network Load Balancer 和 Gateway Load Balancer。使用下表了解要使用的負載平衡器類型。

負載平衡器類型	在下列情況下使用
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

負載平衡器類型	在下列情況下使用
	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。 使用虛擬應用裝置，如防火牆、入侵偵測與預防系統，以及深層封包檢查系統。

如需詳細資訊，請參閱[使用負載平衡分佈 Amazon ECS 服務流量](#)。

互連服務

如果您需要應用程式連線至以 Amazon ECS 服務形式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來執行此動作：

- Service Connect – 允許使用簡短名稱與標準連接埠，實現與自動探索的服務間通訊。
- 服務探索 – 服務探索使用 Route 53 為服務建立命名空間，以便透過 DNS 進行探索。
- Amazon VPC Lattice – VPC Lattice 是一種全受管應用程式聯網服務，可用於連線、保護與監控多個帳戶與 VPC 中的服務。使用該服務需要付費。

如需詳細資訊，請參閱[互連 Amazon ECS 服務](#)。

Amazon ECS 服務部署控制器與策略

部署服務之前，請先確定其部署選項以及服務所使用的功能。

排程策略

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[複本排程策略](#)。
- 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[常駐程式排程策略](#)。

 Note

Fargate 任務不支援 DAEMON 排程策略。

複本排程策略

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 任務置放限制條件範例](#)。
- 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。

建議在使用 REPLICAS 策略時使用服務重新平衡功能，因為該功能有助於確保服務的高可用性。

常駐程式排程策略

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任務的任務置放限制條件，並會停止不符合置放限制條件的任務。使用此策略時，無需指定所需的任務數量、任務置放策略或使用服務自動擴展政策。

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。這表示常駐程式任務是執行個體上啟動的第一項任務，也是停止所有複本任務後停止的最後一項任務。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。

指定部署組態時，`maximumPercent` 參數的值必須為 100 (以百分比形式指定)，如果未設定，則會使用該預設值。`minimumHealthyPercent` 參數的預設值為 0 (以百分比形式指定)。

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。更新服務或設定任務定義的不同修訂版時，也會啟動新的部署。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 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 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。

部署控制器

部署控制器是決定服務任務部署方式的機制。有效選項為：

- ECS

建立使用 ECS 部署控制器的服務時，可以選擇下列部署策略：

- ROLLING：如果建立的服務使用的是滾動更新 (ROLLING) 部署策略，Amazon ECS 服務排程器會使用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。

滾動更新部署最適合下列案例：

- 逐步服務更新：您需要以遞增的方式更新服務，而不要一次讓整個服務離線。
- 有限資源需求：您希望避免同時執行兩個完整環境的額外資源成本 (藍/綠部署時所需)。
- 可接受的部署時間：應用程式可接受較長的部署程序，因為滾動更新會逐一取代任務。
- 無需即時復原：服務可接受復原程序需耗時幾分鐘完成，而非幾秒鐘就完成。
- 簡單的部署程序：您傾向採取直接的部署方法，省去管理多個環境、目標群組與接聽程式的麻煩。
- 無需負載平衡器：服務不使用或不需要負載平衡器、Application Load Balancer、Network Load Balancer 或 Service Connect (此為藍/綠部署所需)。
- 有狀態應用程式：應用程式處於難以執行兩個平行環境的狀態。
- 成本敏感度：您想要藉由在部署期間不執行重複的環境，盡量減少部署成本。

滾動更新是服務的預設部署策略，能夠在許多常見的應用程式案例中，於部署安全性與資源效率之間取得平衡。

- **BLUE_GREEN**：藍/綠部署策略 (BLUE_GREEN) 是一種發布方法，可透過執行兩個稱為藍色與綠色的相同生產環境來縮短停機時間並降低風險。藉助 Amazon ECS 藍/綠部署，您可以先驗證新的服務修訂版，再將生產流量導向這些修訂版。這種方法提供了一種更安全的變更部署方式，並且能在需要時快速復原。

Amazon ECS 藍/綠部署最適合下列案例：

- **服務驗證**：需要先驗證新的服務修訂版，再將生產流量導向這些修訂版時
- **零停機時間**：當服務需要零停機時間部署時
- **即時復原**：當您需要能夠在偵測到問題時快速復原時
- **負載平衡器需求**：當服務使用 Application Load Balancer、Network Load Balancer 或 Service Connect 時
- **LINEAR**：線性部署策略 (LINEAR) 會逐漸將流量從目前的生產環境轉移到新的環境，在指定的期間內以相等百分比遞增。透過 Amazon ECS 線性部署，您可以控制流量轉移的速度，並透過增加的生產流量來驗證新的服務修訂。

Amazon ECS 線性部署最適合下列案例：

- **逐步驗證**：當您想要隨著流量增加逐步驗證新的服務版本時
- **效能監控**：當您需要時間在部署期間監控指標和效能時
- **風險最小化**：當您想要透過逐步向生產流量公開新版本來將風險降至最低時
- **負載平衡器需求**：當服務使用 Application Load Balancer、Network Load Balancer 或 Service Connect 時
- **CANARY**：Canary 部署策略 (CANARY) 會先將一小部分的流量轉移到新的服務修訂版，然後在指定的時段之後一次轉移剩餘的流量。這可讓您在完全部署之前，使用使用者子集測試新版本。

Amazon ECS Canary 部署最適合下列案例：

- **功能測試**：當您想要在完全推出前測試一小部分使用者的新功能時
- **生產驗證**：當您需要使用實際生產流量驗證效能和功能時
- **爆量半徑控制**：如果在新版本中發現問題，則您想要將爆量半徑降至最低
- **負載平衡器需求**：當服務使用 Application Load Balancer、Network Load Balancer 或 Service Connect 時
- **外部**

使用第三方部署控制器。

- 藍/綠部署 (採用技術 AWS CodeDeploy)

CodeDeploy 會將應用程式的更新版本安裝為新的替代任務集，並將生產流量從原始應用程式任務集重新路由至替代任務集。原始任務集會在成功部署後終止。使用此部署控制器在傳送生產流量到服務的新部署之前驗證新部署。

Amazon ECS 部署失敗偵測

Amazon ECS 提供兩種偵測部署失敗的方法：

- 部署斷路器
- CloudWatch 警示

這兩種方法都可以設定為自動將失敗的部署復原至上一個已知的良好狀態。

考慮下列各項：

- 兩種方法都僅支援滾動更新部署與藍/綠部署類型。
- 使用這兩種方法時，任一方法都可能觸發部署失敗。
- 復原方法需要先前處於 COMPLETED 狀態的部署。
- 系統會針對部署狀態變更產生 EventBridge 事件。

Amazon ECS 部署斷路器如何偵測失敗

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。

當開始服務部署變更狀態時，Amazon ECS 會傳送服務部署狀態變更事件至 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](#)。建議您使用 SERVICE_DEPLOYMENT_FAILED 的 eventName 建立和監控 EventBridge 規則，以便您可以採取手動動作來啟動部署。如需詳細資訊，請參閱 Amazon EventBridge User Guide 中的[Getting started with EventBridge](#)。

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。這表示在部署達到

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。

建立服務時，排程器會追蹤兩個階段中啟動失敗的任務。

- 階段 1 – 排程器會監控任務，檢查其是否轉換為 RUNNING 狀態。
 - 成功 – 部署有機會轉換為 COMPLETED 狀態，因為有多項任務已轉換為 RUNNING 狀態。系統會略過失敗條件，且斷路器會移至階段 2。
 - 失敗 – 有連續任務未能轉換為 RUNNING 狀態，且部署可能會轉換為 FAILED 狀態。
- 階段 2 – 當至少有一項任務處於 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任務的運作狀態檢查。驗證的運作狀態檢查為 Elastic Load Balancing、AWS Cloud Map 服務運作狀態檢查和容器運作狀態檢查。
 - 成功 – 至少有一項處於執行中狀態的任務通過了運作狀態檢查。
 - 失敗 – 因運作狀態檢查失敗而被取代的任務達到了失敗閾值。

當您對服務使用部署斷路器方法時，請考量下列事項。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。
- 當您搭配 CloudWatch 選項使用部署斷路器 AWS CLI 時，必須使用 Amazon ECS 主控台或。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 範例顯示當部署斷路器與復原選項搭配使用時，如何建立 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。

部署斷路器的最小閾值為 3，最大閾值為 200。會使用下列公式中的值來判斷部署失敗。

$$\text{Minimum threshold} \leq 0.5 * \text{desired task count} \Rightarrow \text{maximum threshold}$$

當計算結果大於最小值 3，但小於最大值 200 時，失敗閾值會設定為計算閾值（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. 如果一項或多項任務處於 RUNNING 狀態，則會進入此階段。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	$3 \leq 0.5 * 1 \Rightarrow 200$	3 (計算值小於最小值)
25	$3 \leq 0.5 * 25 \Rightarrow 200$	13 (值會無條件進位)
400	$3 \leq 0.5 * 400 \Rightarrow 200$	200
800	$3 \leq 0.5 * 800 \Rightarrow 200$	200 (計算值大於最大值)

例如，當閾值為 3 時，斷路器會從失敗計數設定為 0 開始。當任務未能達到 RUNNING 狀態時，部署斷路器會在失敗計數上增加一。當失敗計數等於 3 時，部署將標記為 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，以便在其偵測到指定的 CloudWatch 警示已進入 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 範例顯示當部署警示與復原選項搭配使用時，如何建立 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
```

當您對服務使用 Amazon CloudWatch 警示方式時，請考量下列事項。

- 在生產流量轉移後，藍色與綠色服務修訂版同時執行的持續時間。Amazon ECS 會根據與部署相關聯的警示組態來計算此時段。您無法設定此值。
- `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 Elastic Container Service API 參考》中的 [CreateService](#)。
- `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 警示來偵測失敗，則任何一種方法都可以在符合任一方法的條件後立即初始化部署失敗。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。
- Amazon CloudWatch 警示僅支援使用滾動更新 (ECS) 部署控制器的 Amazon ECS 服務。
- 您可以使用 Amazon ECS 主控台或 AWS CLI 來設定此選項。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](#)。
- 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。對 DescribeAlarms 的呼叫會計入與您帳戶相關聯的 CloudWatch 服務配額。如果您有其他呼叫 AWS 的服務 DescribeAlarms，可能會對 Amazon ECS 輪詢警示造成影響。例如，如果其他服務發出的 DescribeAlarms 呼叫數量足以達到配額，則該服務會受到限流，Amazon ECS 也會受到限流，而且無法輪詢警示。如果在限流期間產生警示，Amazon ECS 可能會遺漏警示，且可能不會發生復原。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 使用者指南》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)。此行為可處理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 指標的詳細資訊，請參閱《Application Load Balancer 使用者指南》中的[Application Load Balancer 的 CloudWatch 指標](#)。
- 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의 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。如需有關 Amazon SQS 指標的詳細資訊，請參閱《Amazon Simple Queue Service 開發人員指南》中的[適用於 Amazon SQS 的 CloudWatch 指標](#)。

封裝時間

為服務部署使用復原選項時，Amazon ECS 會在目標服務修訂版部署完成後，多等待一段時間，然後才傳送 CloudWatch 警示。這段時間稱為封裝時間。此時間將於下列情況之後開始：

- 目標服務修訂版的所有任務都在執行中，且運作狀態良好
- 來源服務修訂版規模縮減至 0%

預設的封裝時間少於 5 分鐘。封裝時間過期後，服務部署會標示為完成。

您可以為滾動部署設定封裝時間。使用 CloudWatch 警示來偵測失敗時，如果變更了封裝時間，之後卻希望使用 Amazon ECS 預設值，則必須手動設定封裝時間。

Amazon ECS 服務部署的 lifecycle hook

部署開始後，會經歷多個生命週期階段。這些階段可能處於 IN_PROGRESS 或成功等狀態。您可以使用 lifecycle hook，這是 Amazon ECS 在指定生命週期階段代表您執行的 Lambda 函式。這些函式可以是下列其中一項：

- 非同步 API，可在 15 分鐘內驗證運作狀態檢查。
- 輪詢 API，會啟動另一個非同步程序來評估 lifecycle hook 完成情況。

函式執行完成後，必須傳回 hookStatus，部署才能繼續。如果未傳回 hookStatus，或函式執行失敗，部署將會復原。以下為 hookStatus 的值：

- SUCCEEDED – 部署會繼續進入下一個生命週期階段
- FAILED – 部署會復原至上次成功的部署。
- IN_PROGRESS – Amazon ECS 會在短時間內再次執行函式。預設時間間隔為 30 秒，但此值可透過在傳回 hookStatus 時同時傳回 callbackDelay 進行自訂。

下列範例說明如何傳回具有自訂復原延遲的 hookStatus。在此範例中，Amazon ECS 會在 60 秒 (而不是預設的 30 秒) 內重試此勾點：

```
{
  "hookStatus": "IN_PROGRESS",
  "callbackDelay": 60
}
```

發生復原時，Amazon ECS 會為下列生命週期階段執行 lifecycle hook：

- PRODUCTION_TRAFFIC_SHIFT
- TEST_TRAFFIC_SHIFT

生命週期承載

為 ECS 服務部署設定 lifecycle hook 時，Amazon ECS 會在部署程序的特定階段調用這些勾點。每個生命週期階段都會提供 JSON 承載，其中包含有關目前部署狀態的資訊。本文件主要介紹每個生命週期階段的承載結構。

常見的承載結構

所有生命週期階段承載都包含下列常見欄位：

- `serviceArn` – 服務的 Amazon Resource Name (ARN)。
- `targetServiceRevisionArn` – 正在部署的目標服務修訂版的 ARN。
- `testTrafficWeights` – 服務修訂版 ARN 對其對應測試流量權重百分比的映射。
- `productionTrafficWeights` – 服務修訂版 ARN 對其對應生產流量權重百分比的映射。

生命週期階段承載

RECONCILE_SERVICE

此階段發生在部署程序開始時 (即正在協調服務時)。下列顯示此生命週期階段的範例承載。

```
{
  "serviceArn": "arn:aws:ecs:us-west-2:1234567890:service/myCluster/myService",
  "targetServiceRevisionArn": "arn:aws:ecs:us-west-2:1234567890:service-revision/myCluster/myService/01275892",
  "testTrafficWeights": {
    "arn:aws:ecs:us-west-2:1234567890:service-revision/myCluster/myService/01275892": 100,
    "arn:aws:ecs:us-west-2:1234567890:service-revision/myCluster/myService/78652123": 0
  },
  "productionTrafficWeights": {
    "arn:aws:ecs:us-west-2:1234567890:service-revision/myCluster/myService/01275892": 100,
    "arn:aws:ecs:us-west-2:1234567890:service-revision/myCluster/myService/78652123": 0
  }
}
```

此階段的預期狀態：

- 主要任務集規模為 0%

PRE_SCALE_UP

此階段發生在新任務向上擴展之前。下列顯示此生命週期階段的範例承載。

```
{
  "serviceArn": "arn:aws:ecs:us-west-2:1234567890:service/myCluster/myService",
  "targetServiceRevisionArn": "arn:aws:ecs:us-west-2:1234567890:service-revision/
myCluster/myService/01275892",
  "testTrafficWeights": {},
  "productionTrafficWeights": {}
}
```

此階段的預期狀態：

- 綠色服務修訂版任務規模為 0%

POST_SCALE_UP

此階段發生在新任務向上擴展且運作狀態良好之後。下列顯示此生命週期階段的範例承載。

```
{
  "serviceArn": "arn:aws:ecs:us-west-2:1234567890:service/myCluster/myService",
  "targetServiceRevisionArn": "arn:aws:ecs:us-west-2:1234567890:service-revision/
myCluster/myService/01275892",
  "testTrafficWeights": {},
  "productionTrafficWeights": {}
}
```

此階段的預期狀態：

- 綠色服務修訂版任務規模為 100%
- 綠色服務修訂版中的任務運作狀態良好

TEST_TRAFFIC_SHIFT

此階段發生在測試流量在轉移到綠色服務修訂版任務時。

下列顯示此生命週期階段的範例承載。

```
{
  "serviceArn": "arn:aws:ecs:us-west-2:1234567890:service/myCluster/myService",
  "targetServiceRevisionArn": "arn:aws:ecs:us-west-2:1234567890:service-revision/
myCluster/myService/01275892",
  "testTrafficWeights": {
    "arn:aws:ecs:us-west-2:1234567890:service-revision/myCluster/myService/01275892":
100,
    "arn:aws:ecs:us-west-2:1234567890:service-revision/myCluster/myService/78652123": 0
  },
  "productionTrafficWeights": {}
}
```

此階段的預期狀態：

- 測試流量正在流向綠色服務修訂版任務。

POST_TEST_TRAFFIC_SHIFT

此階段發生在測試流量完全轉移到新任務之後。

下列顯示此生命週期階段的範例承載。

```
{
  "serviceArn": "arn:aws:ecs:us-west-2:1234567890:service/myCluster/myService",
  "targetServiceRevisionArn": "arn:aws:ecs:us-west-2:1234567890:service-revision/
myCluster/myService/01275892",
  "testTrafficWeights": {},
  "productionTrafficWeights": {}
}
```

此階段的預期狀態：

- 100% 的測試流量已流向綠色服務修訂版任務。

PRODUCTION_TRAFFIC_SHIFT

此階段發生在生產流量轉在移到綠色服務修訂版任務時。

```
{
  "serviceArn": "arn:aws:ecs:us-west-2:1234567890:service/myCluster/myService",
  "targetServiceRevisionArn": "arn:aws:ecs:us-west-2:1234567890:service-revision/
myCluster/myService/01275892",
  "testTrafficWeights": {},
  "productionTrafficWeights": {
    "arn:aws:ecs:us-west-2:1234567890:service-revision/myCluster/myService/01275892":
100,
    "arn:aws:ecs:us-west-2:1234567890:service-revision/myCluster/myService/78652123": 0
  }
}
```

此階段的預期狀態：

- 生產流量正在流向綠色服務修訂版。

POST_PRODUCTION_TRAFFIC_SHIFT

此階段發生在生產流量完全轉移到綠色服務修訂版任務之後。

```
{
  "serviceArn": "arn:aws:ecs:us-west-2:1234567890:service/myCluster/myService",
  "targetServiceRevisionArn": "arn:aws:ecs:us-west-2:1234567890:service-revision/
myCluster/myService/01275892",
  "testTrafficWeights": {},
  "productionTrafficWeights": {}
}
```

此階段的預期狀態：

- 100% 的生產流量已流向綠色服務修訂版任務。

生命週期階段類別

生命週期階段分為兩個類別：

1. 單次調用階段 – 這些階段在服務部署期間僅調用一次：
 - PRE_SCALE_UP
 - POST_SCALE_UP
 - POST_TEST_TRAFFIC_SHIFT
 - POST_PRODUCTION_TRAFFIC_SHIFT
2. 重複調用階段 – 這些階段在服務部署期間可能會多次調用，例如發生復原操作時：
 - TEST_TRAFFIC_SHIFT
 - PRODUCTION_TRAFFIC_SHIFT

Lifecycle hook 期間的部署狀態

Lifecycle hook 執行期間，所有生命週期階段的部署狀態都將是 IN_PROGRESS。

生命週期狀態	部署狀態
RECONCILE_SERVICE	IN_PROGRESS
PRE_SCALE_UP	IN_PROGRESS
POST_SCALE_UP	IN_PROGRESS
TEST_TRAFFIC_SHIFT	IN_PROGRESS
POST_TEST_TRAFFIC_SHIFT	IN_PROGRESS
PRODUCTION_TRAFFIC_SHIFT	IN_PROGRESS
POST_PRODUCTION_TRAFFIC_SHIFT	IN_PROGRESS

停止 Amazon ECS 服務部署

當斷路器或 CloudWatch 警示未偵測到失敗的部署時，可以手動停止部署。以下為可用的停止類型：

- 復原 – 此選項會將服務部署復原至先前的服務修訂版。

即使您未將服務部署設定為使用復原選項，仍可使用此選項。

您可以停止處於下列任何狀態的部署。如需有關服務部署狀態的詳細資訊，請參閱[使用 Amazon ECS 服務部署檢視服務歷史記錄](#)。

- PENDING – 服務部署會移至 ROLLBACK_REQUESTED 狀態，然後開始復原操作。
- IN_PROGRESS – 服務部署會移至 ROLLBACK_REQUESTED 狀態，然後開始復原操作。
- STOP_REQUESTED – 服務部署會繼續停止。
- ROLLBACK_REQUESTED – 服務部署會繼續復原操作。
- ROLLBACK_IN_PROGRESS – 服務部署會繼續復原操作。

程序

開始之前，請先設定檢視服務部署所需的許可。如需詳細資訊，請參閱[檢視 Amazon ECS 服務部署所需的許可](#)。

Amazon ECS Console

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「叢集詳細資訊」頁面上的服務區段中，選擇所需服務。

服務詳細資訊頁面隨即顯示。

4. 在服務詳細資訊頁面上，選擇部署。

部署頁面隨即顯示。

5. 在正在進行的部署下，選擇復原。然後在確認視窗中，選擇復原。

AWS CLI

1. 執行 `list-service-deployments` 以擷取服務部署 ARN。

將 *user-input* 取代為實際值。

```
aws ecs list-service-deployments --cluster cluster-name --service service-name
```

記下要停止的部署的 `serviceDeploymentArn`。

```
{
  "serviceDeployments": [
    {
      "serviceDeploymentArn": "arn:aws:ecs:us-west-2:123456789012:service-deployment/cluster-name/service-name/NCWGC2ZR-taawPAYrIaU5",
      "serviceArn": "arn:aws:ecs:us-west-2:123456789012:service/cluster-name/service-name",
      "clusterArn": "arn:aws:ecs:us-west-2:123456789012:cluster/cluster-name",
      "targetServiceRevisionArn": "arn:aws:ecs:us-west-2:123456789012:service-revision/cluster-name/service-name/4980306466373577095",
      "status": "SUCCESSFUL"
    }
  ]
}
```

2. 執行 `stop-service-deployments`。使用從 `list-service-deployments` 傳回的 `serviceDeploymentArn`。

將 *user-input* 取代為實際值。

```
aws ecs stop-service-deployment --service-deployment-arn
arn:aws:ecs:region:123456789012:service-deployment/cluster-name/service-name/NCWGC2ZR-taawPAYrIaU5 --stop-type ROLLBACK
```

後續步驟

決定需要對服務進行哪些變更，然後更新服務。如需詳細資訊，請參閱[更新 Amazon ECS 服務](#)。

透過取代任務來部署 Amazon ECS 服務

如果建立的服務使用的是滾動更新 (ECS) 部署類型，Amazon ECS 服務排程器會使用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。

Amazon ECS 使用下列參數來決定任務數量：

- `minimumHealthyPercent` 代表在滾動部署期間或容器執行個體耗盡時，應為服務執行且運作狀態良好的任務數量下限，以服務所需任務數量的百分比表示。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。

- `maximumPercent` 代表在滾動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上限，以服務所需任務數量的百分比表示。此值會向下捨去。例如，如果最大百分比為 200，而所需的任務計數為 4，則排程器可以在停止四個現有任務之前啟動四個新任務。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。

在滾動部署期間，當任務運作狀態不佳時，Amazon ECS 會取代它們，以維護您服務的 `minimumHealthyPercent` 並保護可用性。運作狀態不佳的任務會使用其所屬的相同服務修訂來取代。這可確保來源修訂版中運作狀態不佳的任務替換與目標修訂版中的任務失敗無關。當 `maximumPercent` 設定允許時，排程器會在停止運作狀態不佳的任務之前啟動替代任務。如果 `maximumPercent` 參數限制排程器先啟動取代任務，排程器會一次停止一個運作狀態不佳的任務，以在啟動取代任務之前釋放容量。

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。更新 `minimumHealthyPercent` 或 `maximumPercent` 值，然後再試一次。](#)

滾動部署有兩種方法，有助於快速識別服務部署何時失敗：

- [the section called “部署斷路器如何偵測失敗”](#)
- [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。同時使用兩種方法時，只要符合任一失敗方法的失敗條件，就會將部署設定為失敗。

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 警示 – 當您想要根據應用程式指標停止部署時，請使用此方法。

這兩種方法都支援復原至先前的服務修訂版。

容器映像解析

依預設，Amazon ECS 會將任務定義中指定的容器映像標籤解析為容器映像摘要。如果您建立的服務執行並維護單一任務，則該任務將用於建立任務中容器的映像摘要。如果您建立的服務執行並維護多項任務，則服務排程器在部署期間啟動的第一項任務將用於建立這些任務中容器的映像摘要。

如果三次或多次嘗試建立容器映像摘要失敗，則部署將在不進行映像摘要解析的情況下繼續。如果啟用了部署斷路器，則部署將額外標記為失敗並得到復原。

建立容器映像摘要之後，Amazon ECS 會使用摘要來啟動任何其他所需的任務，以及進行任何未來的服務更新。這會導致服務中的所有任務始終執行相同的容器映像，從而確保軟體的版本一致性。

您可以使用容器定義中的 `versionConsistency` 參數，為任務中的每個容器設定此行為。如需詳細資訊，請參閱[versionConsistency](#)。

Note

- 低於 1.31.0 的 Amazon ECS 代理程式版本不支援映像摘要解析。1.31.0 至 1.69.0 的代理程式版本僅支援解析推送至 Amazon ECR 儲存庫之映像的摘要。1.70.0 或更高版本的代理程式版本支援解析所有映像的摘要。
- 用於映像摘要解析的最低 Fargate Linux 平台版本為 1.3.0。用於映像摘要解析的最低 Fargate Windows 平台版本為 1.0.0。
- Amazon ECS 不會擷取 Amazon ECS 管理的邊車容器摘要，例如 Amazon GuardDuty 安全代理程式或 Service Connect Proxy。
- 若要在具有多項任務的服務中降低與容器映像解析相關聯的潛在延遲，請在 EC2 容器執行個體上執行 Amazon ECS 代理程式 1.83.0 或更高版本。若要避免潛在延遲，請在任務定義中指定容器映像摘要。
- 如果您建立所需任務計數為零的服務，則在您觸發另一次所需任務計數大於零的服務部署之前，Amazon ECS 無法建立容器摘要。
- 若要建立更新後的映像摘要，您可以強制執行新部署。更新後的摘要將用於啟動新任務，不會影響已在執行的任務。如需有關強制執行新部署的詳細資訊，請參閱 Amazon ECS API reference 中的 [forceNewDeployment](#)。
- 使用 EC2 容量提供者時，如果在初始部署期間沒有足夠的容量來啟動任務，軟體版本一致性可能會失敗。為確保即使容量有限也能維持版本一致性，請在任務定義容器組態中明確設定 `versionConsistency: "enabled"`，而不是依賴預設行為。這會導致 Amazon ECS 等待容量變為可用，然後再繼續部署。

Amazon ECS 服務參數最佳實務

為確保應用程式零停機時間，部署程序如下：

1. 啟動新的應用程式容器，同時讓現有容器保持執行狀態。
2. 檢查新容器是否正常運作。
3. 停止舊容器。

根據部署組態與叢集中可用、未預留的空間量，可能需要多次重複此程序才能完成以新任務取代所有舊任務的工作。

可以使用下列兩種服務組態選項來修改任務數量：

- `minimumHealthyPercent` : 100% (預設)

服務在部署期間必須保持在 `RUNNING` 狀態的任務數量下限。此值是無條件進位到最接近整數的 `desiredCount` 的百分比。此參數可讓您不使用額外的叢集容量進行部署。

- `maximumPercent` : 200% (預設)

服務在部署期間允許保持在 `RUNNING` 或 `PENDING` 狀態的任務數量上限。此值是無條件捨去到最接近整數的 `desiredCount` 的百分比。

範例：預設組態選項

假設下列服務具有六項任務，部署在總共可容納八項任務的叢集中。預設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. 目標是取代六項任務。
2. 排程器會啟動兩項新任務，因為預設設定要求必須有六項執行中任務。

現在有六項現有任務與兩項新任務。

3. 排程器會停止兩項現有任務。

現在有四項現有任務與兩項新任務。

4. 排程器會再啟動兩項新任務。

現在有四項現有任務與四項新任務。

5. 排程器會關閉兩項現有任務。

現在有兩項現有任務與四項新任務。

6. 排程器會再啟動兩項新任務。

現在有兩項現有任務與六項新任務

7. 排程器會關閉最後兩項現有任務。

現在有六項新任務。

在上述範例中，如果使用選項的預設值，則需要等待 2.5 分鐘，才能啟動每項新任務。此外，負載平衡器可能必須等待 5 分鐘，舊任務才能停止。

範例：修改 `minimumHealthyPercent`

您可以將 `minimumHealthyPercent` 值設定為 50%，以加速部署。

假設下列服務具有六項任務，部署在總共可容納八項任務的叢集中。部署程序如下：

1. 目標是取代六項任務。

2. 排程器會停止三項現有任務。

仍有三項現有任務在執行中，符合 `minimumHealthyPercent` 值。

3. 排程器會啟動五項新任務。

現在有三項現有任務與五項新任務。

4. 排程器會停止剩餘的三項現有任務。

現在有五項新任務

5. 排程器會啟動最後一項新任務。

現在有六項新任務。

範例：修改叢集可用空間

您也可以新增額外的可用空間，以便執行更多任務。

假設下列服務具有六項任務，部署在總共可容納十項任務的叢集中。部署程序如下：

1. 目標是取代現有任務。
2. 排程器會停止三項現有任務，

現在有三項現有任務。

3. 排程器會啟動六項新任務。

現在有三項現有任務與六項新任務

4. 排程器會停止三項現有任務。

現在有六項新任務。

建議

如果任務閒置一段時間且使用率不高，請針對服務組態選項使用下列值。

- `minimumHealthyPercent` : 50%
- `maximumPercent` : 200%

建立 Amazon ECS 滾動更新部署

建立服務，以在叢集中同時執行並維持指定數量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。

建立服務之前，請先決定下列組態參數：

- 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。

如果想要在 Amazon ECS 受管執行個體上執行工作負載，必須使用容量提供者策略選項。

- 啟動類型便於 Amazon ECS 直接在 Fargate 或註冊至叢集的 EC2 執行個體上啟動任務。

如果想要在 Amazon ECS 受管執行個體上執行工作負載，必須使用容量提供者策略選項。

- 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。
- 置放策略，預設任務置放策略會將任務均勻分佈到可用區域。

建議使用可用區域重新平衡，以協助確保服務的高可用性。如需詳細資訊，請參閱[在可用區域之間平衡 Amazon ECS 服務](#)。

- 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，則系統會選取使用 FARGATE 容量提供者的自訂容量提供者策略。
 - 如果叢集未定義預設容量提供者策略，但您已將一個或多個 Auto Scaling 群組容量提供者新增至叢集，則系統會選取使用自訂 (進階) 選項，並且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。
- 部署失敗偵測的預設選項是搭配失敗時復原選項使用 Amazon ECS 部署斷路器選項。

如需詳細資訊，請參閱[Amazon ECS 部署斷路器如何偵測失敗](#)。

- 決定是否希望 Amazon ECS 自動增加或減少服務中所需的任務數量。如需相關資訊，請參閱[自動擴展您的 Amazon ECS 服務](#)。
- 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](#)。
- 建立使用 Amazon ECS 斷路器的服務時，Amazon ECS 會建立服務部署與服務修訂版。這些資源可用於檢視服務歷史記錄的詳細資訊。如需詳細資訊，請參閱[使用 Amazon ECS 服務部署檢視服務歷史記錄](#)。

如需有關如何使用 建立服務的資訊 AWS CLI，請參閱《AWS Command Line Interface 參考[create-service](#)》中的。

如需有關如何使用 建立服務的資訊 AWS CloudFormation，請參閱AWS CloudFormation 《使用者指南[AWS::ECS::Service](#)》中的。

使用預設選項建立服務

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，選擇要建立服務的叢集。
4. 在 Services (服務) 索引標籤上，選擇 Create (建立)。

建立服務頁面隨即顯示。

5. 在服務詳細資訊下，執行下列動作：
 - a. 在任務定義欄位中，選擇要使用的任務定義系列與修訂版。
 - b. 針對 Service name (服務名稱)，輸入服務的名稱。
6. 若要使用 ECS Exec 來偵錯服務，請在對組態進行疑難排解下，選取開啟 ECS Exec。
7. 在部署組態下，執行下列動作：
 - 針對 Desired tasks (所需任務)，輸入要在服務中啟動並維護的任務數。
8. (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

若要使用定義的參數建立服務，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. 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)。	
任務定義	a. 在任務定義頁面上，選取任務定義旁的選項按鈕。 b. 在部署功能表中，選擇建立服務。	

建立服務頁面隨即顯示。

3. 在服務詳細資訊下，執行下列動作：
 - a. 在任務定義欄位中輸入要使用的任務定義。然後，在修訂版欄位中選擇要使用的修訂版。
 - b. 針對 Service name (服務名稱)，輸入服務的名稱。
4. 在現有叢集欄位中選擇叢集。

選擇建立叢集以在新叢集上執行任務

5. 選擇任務在叢集基礎結構中的分佈方式。在運算組態下選擇選項。

運算選項	步驟	
容量提供者策略	a. 在 Compute options (運算選項) 下，選擇 Capacity	

運算選項	步驟	
	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。	

運算選項	步驟	
啟動類型	- 在運算選項區段中，選取啟動類型。 - 針對 啟動類型，選擇啟動類型。 (選用) 使用 Fargate 時，在平台版本欄位中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。	

6. 若要使用 ECS Exec 來偵錯服務，請在對組態進行疑難排解下，選取開啟 ECS Exec。

7. 在部署組態下，執行下列動作：

- 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)。
- 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。
- 如果選擇複本，若要讓 Amazon ECS 監控可用區域間的任務分佈，並在出現不平衡時重新分佈，請在可用區域服務重新平衡下，選取可用區域服務重新平衡。
- 在運作狀態檢查寬限期欄位中輸入時間量 (以秒為單位)，這是服務排程器在任務首次啟動後忽略運作狀態不良的 Elastic Load Balancing、VPC Lattice 與容器運作狀態檢查的時間。如果您未指定運作狀態檢查寬限期間值，則會使用預設值 0。
- 決定服務的部署類型。展開部署選項區段，然後指定下列參數。

部署類型	步驟	
滾動更新	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	

部署類型	步驟
	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)。

- f. 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)。

在部署斷路器將部署設定為失敗狀態時，若要讓軟體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. 若要根據應用程式指標停止部署，請選取使用 CloudWatch 警示。然後，從 CloudWatch 警示名稱欄位中選擇警示。要建立新的警示，請前往 CloudWatch 主控台。

在 CloudWatch 警示將部署設定為失敗狀態時，若要讓軟體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。

8. 如果任務定義使用 awsvpc 網路模式，可展開聯網區段以指定自訂網路組態，然後執行下列動作：

- 針對 VPC，選擇要使用的 VPC。
- 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。
- 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。

- 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 在公有子網路中執行時，任務可以指派公有 IP 地址，以便他們有網際網路的路由。無法使用此欄位將公有 IP 指派給 EC2 任務。如需詳細資訊，請參閱 [Fargate 的 Amazon ECS 任務聯網選項](#)，以及為 [Amazon ECS 任務配置網路介面](#)。

9. (選用) 若要使用 Service Connect 與服務互連，請展開 Service Connect 區段，然後指定下列項目：

- a. 選取開啟 Service Connect。
- b. 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至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)。
- c. 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。這可以是在您的 中在相同 AWS 區域 中分別建立的命名空間，AWS 帳戶 或是使用 AWS Resource Access Manager () 與您的帳戶共用的相同區域中的命名空間 AWS RAM。如需共用 AWS Cloud Map 命名空間的詳細資訊，請參閱《AWS Cloud Map 開發人員指南》中的 [跨帳戶 AWS Cloud Map 命名空間共用](#)。
- d. (選用) 指定日誌組態。選取使用日誌收集。預設選項會將容器日誌傳送至 CloudWatch 日誌。其他日誌驅動程式選項是使用 AWS FireLens 設定。如需詳細資訊，請參閱 [將 Amazon ECS 日誌傳送至 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 – 將任務設定為將容器日誌傳送至 CloudWatch Logs。系統會提供預設日誌驅動程式選項，用來代表您建立 CloudWatch 日誌群組。若要指定不同的日誌群組名稱，請變更驅動程式選項值。
- Amazon Data Firehose – 將任務設定為將容器日誌傳送至 Firehose。系統會提供預設日誌驅動程式選項，用來將日誌傳送至 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 – 將任務設定為將容器日誌傳送至 Amazon Kinesis Data Streams。系統會提供預設日誌驅動程式選項，用來將日誌傳送至 Kinesis Data Streams 串流。若要指定不同的串流名稱，請變更驅動程式選項值。

- Amazon OpenSearch Service – 將任務設定為將容器日誌傳送至 OpenSearch Service 網域。務必提供日誌驅動程式選項。
 - Amazon S3：將任務設定為將容器日誌傳送至 Amazon S3 儲存貯體。系統會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。
- e. (選用) 若要啟用存取日誌，請遵循下列步驟：
- i. 展開存取日誌組態。針對格式，選擇 JSON 或 TEXT。
 - ii. 若要在存取日誌中包含查詢參數，請選取包含查詢參數。
10. (選用) 若要使用服務探索與服務互連，請展開服務探索區段，然後執行下列動作。
- a. 選取使用服務探索。
 - b. 若要使用新的命名空間，請在設定命名空間下選擇建立新的命名空間，然後提供命名空間名稱與描述。若要使用現有命名空間，請選擇選取現有的命名空間，然後選擇要使用的命名空間。
 - c. 提供服務探索服務資訊，例如服務的名稱與描述。
 - d. 若要讓 Amazon ECS 定期執行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。
 - e. 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 與 SRV 記錄，這取決於任務定義指定的網路模式。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。如果選擇 A，請跳到下一步驟。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)。
11. (選用) 若要使用 VPC Lattice，請展開 VPC Lattice 與服務互連，然後執行下列動作：
- a. 選取開啟 VPC Lattice
 - b. 在基礎結構角色欄位中，選擇對應的基礎結構角色。
- 如果尚未建立角色，請選擇建立基礎結構角色。

- c. 在目標群組下，選擇一個或多個目標群組。您需要選擇至少一個目標群組，最多可以選擇五個。選擇新增目標群組以新增額外的目標群組。針對選擇的每個目標群組，選擇連接埠名稱、通訊協定與連接埠。

若要刪除目標群組，請選擇移除。

 Note

- 若要新增現有的目標群組，需要使用 AWS CLI。如需如何使用 新增目標群組的指示 AWS CLI，請參閱《AWS Command Line Interface 參考》中的 [register-targets](#)。
- 雖然 VPC Lattice 服務可以有多個目標群組，但每個目標群組只能新增至一項服務。

- d. 若要完成 VPC Lattice 組態，請在接聽程式預設動作中或在 VPC Lattice 主控台的現有 VPC Lattice 服務的規則中包含新目標群組。如需詳細資訊，請參閱 [VPC Lattice 服務的接聽程式規則](#)。

12. (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。

使用此負載平衡器	執行此作業	
	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。 g. 在取消註冊延遲欄位中，輸入負載平衡器將目標狀態變更為 UNUSED 的秒數。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	

使用此負載平衡器	執行此作業	
	接狀態。預設使用根目錄 (/)。	
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. 在取消註冊延遲欄位中，輸入負載平衡器將目標狀態變更為 UNUSED 的秒數。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 (/)。	

13. (選用) 若要設定服務自動擴展功能，請展開服務自動擴展區段，然後指定下列參數。若要使用預測自動擴展功能 (該功能可查看過去從流量流程載入的資料)，請在建立服務後進行設定。如需詳細資訊，請參閱[使用歷史模式透過預測擴展來擴展 Amazon ECS 服務](#)。
- a. 若要使用服務自動擴展，請選取 Service auto scaling (服務自動擴展)。
 - b. 在任務數量下限欄位中，輸入供服務自動擴展功能使用的任務數量下限。所需的計數不會低於此計數。
 - c. 在任務數量上限中，輸入供服務自動擴展功能使用的任務數量上限。所需的計數不會高於此計數。
 - d. 選擇政策類型。在擴展政策類型下，選擇下列其中一個選項。

使用此政策類型	執行此作業	
目標追蹤	- 針對 Scaling policy type (擴展政策類型)，選擇 Target tracking (目標追蹤)。 - 針對 Policy name (政策名稱)，輸入政策的名稱。 - 針對 ECS service metric (ECS 服務指標)，選擇下列其中一項指標。 - ECSServiceAverageCPUUtilization – 服務的平均 CPU 使用率。 - ECSServiceAverageMemoryUtilization – 服務的平均記憶體使用率。 - ALBRequestCountPerTarget – Application Load Balancer 目標群組中每個目標的已完成請求數。 - 針對 Target value (目標值)，輸入服務為選取的指標保持的值。 - 在橫向擴充冷卻時間欄位中，輸入一個橫向擴充活動 (新增任務) 完成之後、另一個橫向擴充活動開始之前，必須經過的時間 (以秒為單位)。 - 在縮減冷卻時間欄位中，輸入一個縮減活動 (移除	

使用此政策類型	執行此作業	
	任務) 完成之後、另一個縮減活動開始之前，必須經過的時間 (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) 如果希望擴展政策在流量增加時進行橫向擴充，但不需要在流量減少時縮減，請選取關閉縮減。	

使用此政策類型	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：	

使用此政策類型	執行此作業	
	• 在動作欄位中，選取是為服務新增、移除還是設定特定的所需計數。 • 如果選擇新增或移除任務，請在值欄位中，輸入啟動擴展動作時要新增或移除的任務數量 (或現有任務的百分比)。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。選擇新增擴展動作，然後重複擴展動作步驟。 • 在冷卻時間欄位中，輸入等待上一個擴展活動生效的時間 (以秒為單位)。對於新增政策，這是橫向擴充活動結束	

使用此政策類型	執行此作業	
	之後，擴展政策封鎖縮減活動並限制一次可以橫向擴充多少項任務的時間。對於移除政策，這是一個縮減活動結束之後、另一個縮減活動開始之前必須經過的時間。	

14. (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置放任務](#)。

- AZ 平衡分散 - 在可用區域及可用區域中的容器執行個體之間分散任務。
- AZ 平衡 BinPack - 使用最低可用記憶體，在可用區域及可用區域中的容器執行個體之間分散任務。
- BinPack - 根據最低可用的 CPU 或記憶體數分散任務。
- 每個主機一個任務 - 在每個容器執行個體上最多置放一個服務的任務。
- 自訂 - 定義自己的任務置放策略。

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。

15. 如果任務使用的資料磁碟區與部署時的組態相容，則可透過展開磁碟區區段對磁碟區進行設定。

磁碟區名稱與磁碟區類型會在建立任務定義修訂版時設定，且無法在服務建立期間變更。若要更新磁碟區名稱與類型，必須建立新的任務定義修訂版，並使用新修訂版建立服務。

設定此磁碟區類型	執行此作業	
Amazon EBS	a. 在 EBS 磁碟區類型欄位中，選擇要連接至任務的 EBS 磁碟區類型。 b. 在大小 (GiB) 欄位中，輸入磁碟區大小的有效值 (以 GiB 為單位)。您可以指定磁碟區大小 (大小下限 1 GiB，大小上限 16,384 GiB)。除非提供快照 ID，否則需要此值。 c. 在 IOPS 欄位中，輸入磁碟區應提供的輸入/輸出操作次數 (IOPS) 上限。此值僅可針對 io1、io2 與 gp3 磁碟區類型設定。 d. 在輸送量 (MiB/s) 欄位中，輸入磁碟區應提供的輸送量 (以 MiBps 或 MiB/s 為單位)。此值僅可針對 gp3 磁碟區類型設定。 e. 在快照 ID 欄位中，若想從快照建立磁碟區，請選擇現有的 Amazon EBS 磁碟區快照或輸入快照的 ARN。您也可以不選擇或不輸入快照 ID，建立新的空白磁碟區。 f. 如果指定了快照 ID，則可指定磁碟區初始化速率 (MiB/s)。輸入一個介於 100 至 300 MiB/s 之間的值，該值將決定從使用快	

設定此磁碟區類型	執行此作業	
	照 ID 指定的快照在建立磁碟區時載入資料的速度。 g. 在檔案系統類型欄位中，選擇用於磁碟區上資料儲存與擷取的檔案系統類型。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。如果有檔案系統類型不相符，則任務將無法啟動。 h. 在基礎結構角色欄位中，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 <code>AmazonECSInfrastructureRolePolicyForVolumes</code> 受管政策連接至該角色，也可參照該政策建立並連接自己的政策，該政策需具有符合特定需求的許可。如需有關必要許可的詳細資訊，請參閱 Amazon ECS 基礎結構 IAM 角色。 i. 若想依預設設定使用 Amazon EBS 加密，請在加密欄位中選擇預設值。如果您的帳戶已依	

設定此磁碟區類型	執行此作業	
	預設設定加密，則會使用設定中指定的 AWS Key Management Service (AWS KMS) 金鑰來加密磁碟區。如果選擇預設，但未開啟 Amazon EBS 預設加密，則磁碟區將取消加密。 如果您選擇自訂，您可以為磁碟區加密指定 AWS KMS key 您選擇的。 如果選擇無，除非您已設定為預設加密，或者從加密快照建立磁碟區，否則磁碟區將不會加密。 j. 如果您已選擇自訂加密，則必須指定要使用 AWS KMS key 的。在 KMS 金鑰欄位中選擇 AWS KMS key 或輸入金鑰 ARN。如果您選擇使用對稱客戶受管金鑰來加密磁碟區，請確定您已在 AWS KMS key 政策中定義正確的許可。如需詳細資訊，請參閱 Amazon EBS 磁碟區的資料加密。 k. (選用) 在標籤下，您可以透過從任務定義或服務傳播標籤或提供自己的標籤，將標籤新增至 Amazon EBS 磁碟區。	

設定此磁碟區類型	執行此作業	
	如果要從任務定義傳播標籤，請在傳播標籤來源欄位中選擇任務定義。 如果要從服務傳播標籤，請在傳播標籤來源欄位中選擇服務。如果選擇不傳播，或者未選擇任何值，則不會傳播標籤。 如果要提供自己的標籤，請選擇新增標籤，然後為每個新增的標籤提供鍵與值。 如需有關標記 Amazon EBS 磁碟區的詳細資訊，請參閱標記 Amazon EBS 磁碟區。	

16. 若要使用 ECS Exec 來偵錯服務，請在對組態進行疑難排解下，選取開啟 ECS Exec。

17. (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。

18. 選擇建立。

後續步驟

下列是建立服務後的其他動作。

- 設定預測自動擴展功能，該功能可查看過去從流量流程載入的資料。如需詳細資訊，請參閱[使用歷史模式透過預測擴展來擴展 Amazon ECS 服務](#)。
- 追蹤部署，並檢視 Amazon ECS 斷路器服務的服務歷史記錄。如需詳細資訊，請參閱[使用 Amazon ECS 服務部署檢視服務歷史記錄](#)。

Amazon ECS 藍/綠部署

藍/綠部署是一種發行方法，可透過執行兩個稱為藍和綠的相同生產環境來減少停機時間和風險。藉助 Amazon ECS 藍/綠部署，您可以先驗證新的服務修訂版，再將生產流量導向這些修訂版。這種方法提供了一種更安全的變更部署方式，並且能在需要時快速復原。

優勢

使用藍/綠部署具有以下優勢：

- 在切換生產之前，透過使用生產流量進行測試來降低風險。您可以在將生產流量導向新部署之前，使用測試流量對部署進行驗證。
- 零停機時間部署。在整個部署程序中，生產環境仍然可用，以確保持續的服務可用性。
- 如果偵測到問題，可輕鬆復原。如果綠色部署發生問題，您可以快速還原至藍色部署，而不會造成長時間的服務中斷。
- 受控的測試環境。綠色環境提供隔離的空間，可在完全部署之前使用實際流量模式測試新功能。
- 可預測的部署程序。具有定義的生命週期階段的結構化方法，可讓部署更加一致且可靠。
- 透過 lifecycle hook 實現自動化驗證。您可以在部署的各個階段實作自動化測試，以驗證功能。

術語

下列是 Amazon ECS 藍/綠部署術語：

- 封裝時間 – 在生產流量轉移後，藍色與綠色服務修訂版同時執行的持續時間。
- 藍色部署 – 要取代的目前生產服務修訂版。
- 綠色部署 – 要部署的新服務修訂版。

- 生命週期階段 – 部署操作中的一系列事件，例如「生產流量轉移後」。
- Lifecycle hook – 在特定生命週期階段驗證部署的 Lambda 函式。
- 接聽程式 – 一種 Elastic Load Balancing 資源，用於透過已設定的通訊協定與連接埠檢查連線請求。您為接聽程式定義的規則決定了 Amazon ECS 如何將請求路由至已註冊的目標。
- 規則 – 與接聽程式相關聯的 Elastic Load Balancing 資源。規則定義了請求的路由方式，包含動作、條件與優先順序。
- 目標群組 – 一種 Elastic Load Balancing 資源，用於將請求路由至一個或多個已註冊的目標 (例如 EC2 執行個體)。當您建立接聽程式時，可以為其預設動作指定一個目標群組。流量會轉送至接聽程式規則中指定的目標群組。
- 流量轉移 – Amazon ECS 用於將流量從藍色部署轉移到綠色部署的程序。對於 Amazon ECS 藍/綠部署，所有流量都會一次從藍色服務轉移到綠色服務。

考量事項

選擇部署類型時，請考量下列事項：

- 資源用量 – 藍/綠部署會暫時同時執行藍色與綠色服務修訂版，這可能會在部署期間使資源用量加倍。
- 部署監控 – 藍/綠部署會提供更詳細的部署狀態資訊，便於您監控部署程序的每個階段。
- 復原 – 藍/綠部署會在偵測到問題時更輕鬆地復原至先前的版本，因為藍色修訂版會持續執行，直到封裝時間過期為止。
- Network Load Balancer 生命週期掛鉤：如果您使用 Network Load Balancer 進行藍/綠部署，則 TEST_TRAFFIC_SHIFT 和 PRODUCTION_TRAFFIC_SHIFT 生命週期階段會有額外的 10 分鐘。這是因為 Amazon ECS 可確保轉移流量是安全的。

Amazon ECS 藍/綠服務部署工作流程

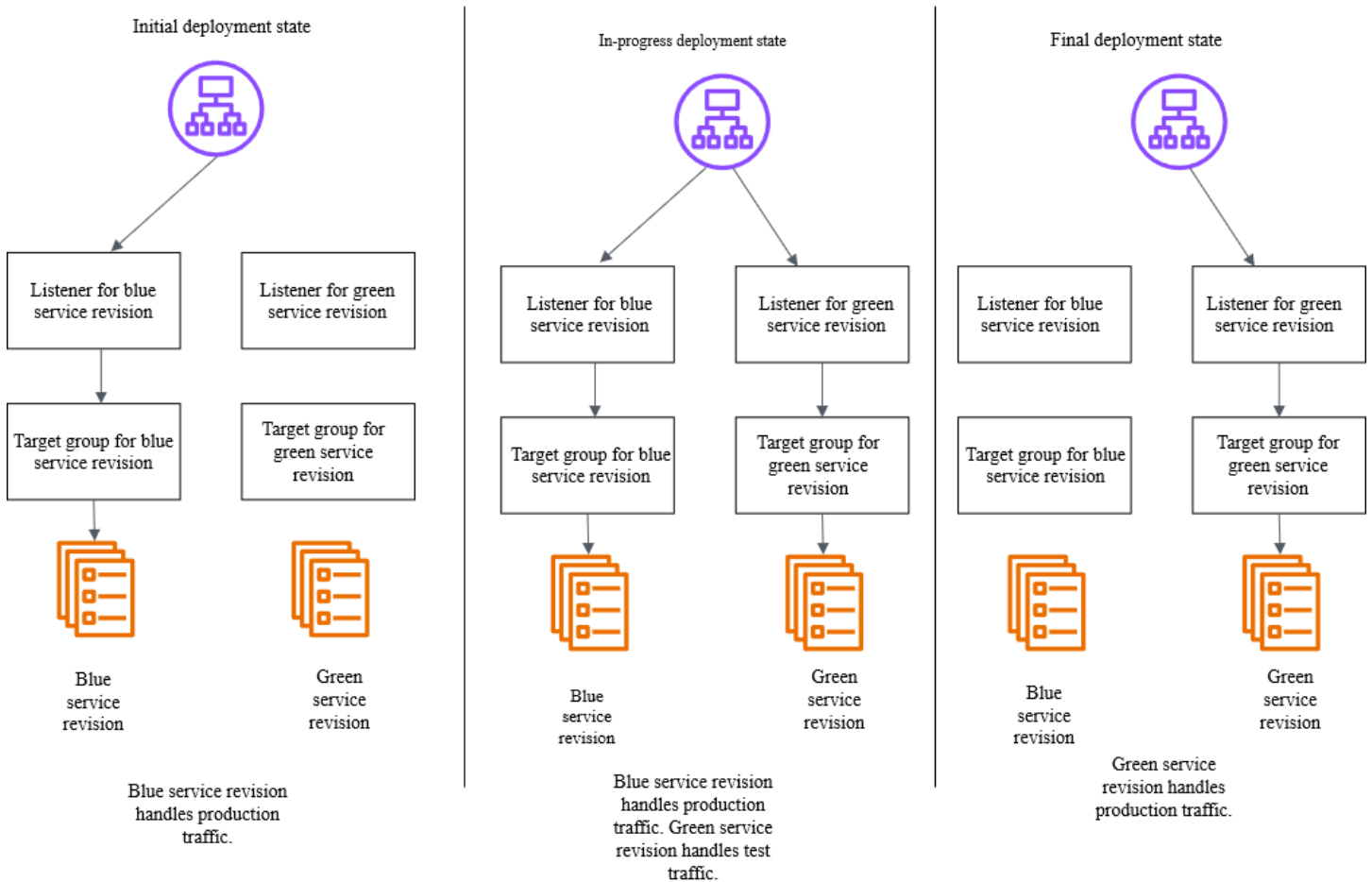
Amazon ECS 藍/綠部署程序採用結構化方法，包含六個不同階段，以確保安全可靠的應用程式更新。每個階段都有特定用途，以驗證應用程式，並將其從目前版本 (藍色) 轉換為新版本 (綠色)。

1. 準備階段 – 在現有的藍色環境旁建立綠色環境。這包括佈建新的服務修訂版以及準備目標群組。
2. 部署階段：將新的服務修訂版部署到綠色環境。Amazon ECS 會使用更新後的服務修訂版啟動新任務，同時藍色環境會繼續處理生產流量。
3. 測試階段：使用測試流量路由驗證綠色環境。Application Load Balancer 會將測試請求導向綠色環境，同時生產流量仍保留在藍色環境中。

4. 流量轉移階段：根據設定的部署策略，將生產流量從藍色環境轉移到綠色環境。此階段包含監控與驗證檢查點。
5. 監控階段：在封裝時間段內監控應用程式運作狀態、效能指標與警示狀態。偵測到問題時，會啟動復原操作。
6. 完成階段：視組態而定，透過終止藍色環境或者為潛在復原案例維持該環境來完成最終部署。

工作流程

下圖說明完整的藍/綠部署工作流程，顯示了 Amazon ECS 與 Application Load Balancer 之間的互動：



增強型部署工作流程包含下列詳細步驟：

1. 初始狀態：藍色服務 (目前生產環境) 處理 100% 的生產流量。Application Load Balancer 具有一個接聽程式，其規則會將所有請求路由至包含運作狀態良好之藍色任務的藍色目標群組。
2. 綠色環境佈建：Amazon ECS 會使用更新後的任務定義建立新任務。這些任務已向新的綠色目標群組註冊，但最初不會接收任何流量。

3. 運作狀態檢查驗證：Application Load Balancer 會對綠色任務執行運作狀態檢查。只有在綠色任務通過運作狀態檢查時，部署才會進入下一階段。
4. 測試流量路由：如果已設定，Application Load Balancer 的接聽程式規則會將特定流量模式 (例如具有測試標頭的請求) 路由至綠色環境進行驗證，而生產流量仍保留在藍色環境中。這由處理生產流量的同一接聽程式控制，並根據請求屬性使用不同的規則。
5. 生產流量轉移：根據部署組態，流量會從藍色環境轉移到綠色環境。在 ECS 藍/綠部署中，這是即時 (一次全部) 轉移，其中 100% 的流量會從藍色環境移至綠色環境。Application Load Balancer 搭配接聽程式規則使用單一接聽程式，根據權重控制藍色與綠色目標群組之間的流量分佈。
6. 監控與驗證：在整個流量轉移期間，Amazon ECS 會監控 CloudWatch 指標、警示狀態與部署運作狀態。如果偵測到問題，就會觸發自動復原。
7. 封裝時間段：在生產流量轉移後，藍色與綠色服務修訂版同時執行的持續時間。
8. 藍色環境終止：在成功的流量轉移與驗證之後，藍色環境會終止以釋放叢集資源，或者會保留以實現快速復原功能。
9. 最終狀態：綠色環境會成為新的生產環境，處理 100% 的流量。部署會標記為成功。

部署生命週期階段

藍/綠部署程序會經歷不同的生命週期階段 (部署操作中的一系列事件，例如「生產流量轉移後」)，每個事件都有特定的責任與驗證檢查點。了解這些階段有助於監控部署進度並對問題進行有效的疑難排解。

每個生命週期階段最多可持續 24 小時。建議將該值保持在 24 小時標記以內。這是因為非同步程序需要時間來觸發勾點。系統逾時、部署失敗，然後在階段達到 24 小時後啟動轉返。CloudFormation 部署有額外的逾時限制。雖然 24 小時階段限制仍然有效，但對整個部署 CloudFormation 強制執行 36 小時限制。部署 CloudFormation 失敗，然後在程序未在 36 小時內完成時啟動轉返。

生命週期階段	Description	是否將此階段用於 lifecycle hook ?
RECONCILE_SERVICE	僅當啟動新的服務部署且存在多個處於 ACTIVE 狀態的服務修訂版時，此階段才會發生。	是
PRE_SCALE_UP	綠色服務修訂版尚未啟動。藍色服務修訂版正在處理 100% 的生產流量。沒有測試流量。	是

生命週期階段	Description	是否將此階段用於 lifecycle hook ?
SCALE_UP	綠色服務修訂版向上擴展到 100% 並啟動新任務的時間。此時綠色服務修訂版不處理任何流量。	否
POST_SCALE_UP	綠色服務修訂版已啟動。藍色服務修訂版正在處理 100% 的生產流量。沒有測試流量。	是
TEST_TRAFFIC_SHIFT	藍色與綠色服務修訂版均在執行中。藍色服務修訂版處理 100% 的生產流量。綠色服務修訂版正在從 0% 遷移至 100% 的測試流量。	是
POST_TEST_TRAFFIC_SHIFT	測試流量轉移已完成。綠色服務修訂版處理 100% 的生產流量。	是
PRODUCTION_TRAFFIC_SHIFT	生產流量正在轉移至綠色服務修訂版。綠色服務修訂版正在從 0% 遷移至 100% 的生產流量。	是
POST_PRODUCTION_TRAFFIC_SHIFT	生產流量轉移已完成。	是
BAKE_TIME	藍色與綠色服務修訂版同時執行的持續時間。	否
CLEAN_UP	藍色服務修訂已完全縮減至 0 項執行中任務。在此階段之後，綠色服務修訂版現在是生產服務修訂版。	否

每個生命週期階段都包含內建的驗證檢查點，必須通過這些檢查點才能進入下一階段。如果有任何驗證失敗，部署可以自動復原，以維持服務可用性與可靠性。

當您使用 Lambda 函式時，該函式必須在 15 分鐘內完成工作或傳回 IN_PROGRESS。您可以使用 `callbackDelaySeconds` 來延遲對 Lambda 的呼叫。如需詳細資訊，請參閱 GitHub 上 [sample-amazon-ecs-blue-green-deployment-patterns](#) 中的 [app.py function](#)。

Amazon ECS 藍/綠部署所需資源

若要搭配受管流量轉移使用藍/綠部署，服務必須使用下列其中一項功能：

- Elastic Load Balancing
- Service Connect

未使用服務探索、Service Connect、VPC Lattice 或 Elastic Load Balancing 的服務也可以使用藍/綠部署，但不會獲得任何受管流量轉移優勢。

下列清單提供了設定 Amazon ECS 藍/綠部署所需資源的高階概觀：

- 服務會使用 Application Load Balancer、Network Load Balancer 或 Service Connect。設定相應的資源。
 - Application Load Balancer – 如需詳細資訊，請參閱[藍/綠、線性和金絲雀部署的 Application Load Balancer 資源](#)。
 - Network Load Balancer – 如需詳細資訊，請參閱[Amazon ECS 藍/綠、線性和 Canary 部署的 Network Load Balancer 資源](#)。
 - Service Connect – 如需詳細資訊，請參閱[Amazon ECS 藍色/綠色、線性和 Canary 部署的 Service Connect 資源](#)。
- 將服務部署控制器設定為 ECS。
- 在服務定義中，將部署策略設定為 blue/green。
- 或者設定其他參數，例如：
 - 新部署的封裝時間
 - 自動復原的 CloudWatch 警示
 - 用於測試的部署 lifecycle hook (這些是在指定部署階段執行的 Lambda 函式)

最佳實務

請遵循以下最佳實務，以成功實現 Amazon ECS 藍/綠部署：

- 設定相應的運作狀態檢查，以準確反映應用程式的運作狀態。

- 設定封裝時間，以對綠色部署進行充分測試。
- 實作 CloudWatch 警示，以自動偵測問題並觸發復原。
- 使用 lifecycle hook，在每個部署階段執行自動化測試。
- 確保您的應用程式可以同時處理藍色和綠色服務修訂。
- 規劃足夠的叢集容量，以在部署期間處理這兩個服務修訂。
- 在生產環境中實作復原程序之前，先對這些程序進行測試。

藍/綠、線性和金絲雀部署的 Application Load Balancer 資源

若要搭配 Amazon ECS 藍/綠部署使用 Application Load Balancer，需要設定特定資源，以允許藍色與綠色服務修訂版之間的流量路由。

目標群組

若要搭配 Elastic Load Balancing 使用藍/綠部署，需要建立兩個目標群組：

- 用於藍色服務修訂版的主要目標群組 (目前生產流量)
- 用於綠色服務修訂版的替代目標群組 (新版本)

兩個目標群組都應使用下列設定進行設定：

- 目標類型：IP (適用於 Fargate 或使用 awsvpc 網路模式的 EC2)
- 通訊協定：HTTP (或應用程式使用的通訊協定)
- 連接埠：應用程式監聽的連接埠 (對於 HTTP 通常為 80)
- VPC：與 Amazon ECS 任務相同的 VPC
- 運作狀態檢查設定：設定為正確檢查應用程式的運作狀態

在藍/綠部署期間，Amazon ECS 會根據部署階段，自動向相應的目標群組註冊任務。

Example 為 Application Load Balancer 建立目標群組

下列 CLI 命令會建立兩個目標群組，用於藍/綠部署中的 Application Load Balancer：

```
aws elbv2 create-target-group \  
  --name blue-target-group \  
  --protocol HTTP \  
  --ports 80
```

```
--port 80 \  
--vpc-id vpc-abcd1234 \  
--target-type ip \  
--health-check-path / \  
--health-check-protocol HTTP \  
--health-check-interval-seconds 30 \  
--health-check-timeout-seconds 5 \  
--healthy-threshold-count 2 \  
--unhealthy-threshold-count 2  
  
aws elbv2 create-target-group \  
--name green-target-group \  
--protocol HTTP \  
--port 80 \  
--vpc-id vpc-abcd1234 \  
--target-type ip \  
--health-check-path / \  
--health-check-protocol HTTP \  
--health-check-interval-seconds 30 \  
--health-check-timeout-seconds 5 \  
--healthy-threshold-count 2 \  
--unhealthy-threshold-count 2
```

Application Load Balancer

您需要建立包含下列組態的 Application Load Balancer：

- 結構描述：面向網際網路或內部，視要求而定
- IP 位址類型：IPv4
- VPC：與 Amazon ECS 任務相同的 VPC
- 子網路：至少兩個位於不同可用區域的子網路
- 安全群組：允許接聽程式連接埠流量的安全群組

連接至 Application Load Balancer 的安全群組必須具有傳出規則，允許流量傳送至連接至 Amazon ECS 任務的安全群組。

Example 建立 Application Load Balancer

下列 CLI 命令會建立用於藍綠部署的 Application Load Balancer：

```
aws elbv2 create-load-balancer \  

```



```
--name my-application-load-balancer \  
--type application \  
--security-groups sg-abcd1234 \  
--subnets subnet-12345678 subnet-87654321
```

接聽程式和規則

對於藍/綠部署，您需要在 Application Load Balancer 上設定接聽程式：

- 生產接聽程式：處理生產流量 (通常位於連接埠 80 或 443)
 - 最初將流量轉送至主要目標群組 (藍色服務修訂版)
 - 部署之後，將流量轉送至替代目標群組 (綠色服務修訂版)
- 測試接聽程式 (選用)：處理測試流量，以在轉移生產流量之前驗證綠色服務修訂版
 - 可以在不同的連接埠上設定 (例如 8080 或 8443)
 - 在測試期間將流量轉送至替代目標群組 (綠色服務修訂版)

在藍/綠部署期間，Amazon ECS 會根據部署階段自動更新接聽程式規則，將流量路由至相應的目標群組。

Example 建立生產接聽程式

下列 CLI 命令會在連接埠 80 上建立生產接聽程式，將流量轉送至主要 (藍色) 目標群組：

```
aws elbv2 create-listener \  
  --load-balancer-arn arn:aws:elasticloadbalancing:region:123456789012:loadbalancer/  
app/my-application-load-balancer/abcdef123456 \  
  --protocol HTTP \  
  --port 80 \  
  --default-actions  
    Type=forward,TargetGroupArn=arn:aws:elasticloadbalancing:region:123456789012:targetgroup/  
blue-target-group/abcdef123456
```

Example 建立測試接聽程式

下列 CLI 命令會在連接埠 8080 上建立測試接聽程式，將流量轉送至替代 (綠色) 目標群組：

```
aws elbv2 create-listener \  
  --load-balancer-arn arn:aws:elasticloadbalancing:region:123456789012:loadbalancer/  
app/my-application-load-balancer/abcdef123456 \  
  --protocol HTTP \  
  --port 8080 \  
  --default-actions  
    Type=forward,TargetGroupArn=arn:aws:elasticloadbalancing:region:123456789012:targetgroup/  
green-target-group/abcdef123456
```



```
--protocol HTTP \  
--port 8080 \  
--default-actions  
Type=forward,TargetGroupArn=arn:aws:elasticloadbalancing:region:123456789012:targetgroup/  
green-target-group/ghijkl789012
```

Example為以路徑為基礎的路由建立接聽程式規則

下列 CLI 命令會建立規則，將特定路徑的流量轉送至綠色目標群組進行測試：

```
aws elbv2 create-rule \  
--listener-arn arn:aws:elasticloadbalancing:region:123456789012:listener/app/my-  
application-load-balancer/abcdef123456/ghijkl789012 \  
--priority 10 \  
--conditions Field=path-pattern,Values='/test/*' \  
--actions  
Type=forward,TargetGroupArn=arn:aws:elasticloadbalancing:region:123456789012:targetgroup/  
green-target-group/ghijkl789012
```

Example為以標頭為基礎的路由建立接聽程式規則

下列 CLI 命令會建立規則，將包含特定標頭的流量轉送至綠色目標群組進行測試：

```
aws elbv2 create-rule \  
--listener-arn arn:aws:elasticloadbalancing:region:123456789012:listener/app/my-  
application-load-balancer/abcdef123456/ghijkl789012 \  
--priority 20 \  
--conditions Field=http-header,HttpHeaderConfig='{Name=X-  
Environment,Values=[test]}' \  
--actions  
Type=forward,TargetGroupArn=arn:aws:elasticloadbalancing:region:123456789012:targetgroup/  
green-target-group/ghijkl789012
```

服務組態

您必須擁有許可，才能允許 Amazon ECS 代表您管理叢集中的負載平衡器資源。如需詳細資訊，請參閱[負載平衡器的 Amazon ECS 基礎結構 IAM 角色](#)。

為搭配 Elastic Load Balancing 使用的藍/綠部署建立或更新 Amazon ECS 服務時，需要指定下列組態。

將 *user-input* 取代為實際值。

此組態中的關鍵元件包括：

- `targetGroupArn`：主要目標群組的 ARN (藍色服務修訂版)。
- `alternateTargetGroupArn`：替代目標群組的 ARN (綠色服務修訂版)。
- `productionListenerRule`：用於生產流量的接聽程式規則 ARN。
- `roleArn`：允許 Amazon ECS 管理 Elastic Load Balancing 資源的角色 ARN。
- `strategy`：設定為 `BLUE_GREEN` 以啟用藍/綠部署。
- `bakeTimeInMinutes`：在生產流量轉移後，藍色與綠色服務修訂版同時執行的持續時間。
- `TestListenerRule`：用於測試流量的接聽程式規則 ARN。這是選擇性的參數。

```
{
  "loadBalancers": [
    {
      "targetGroupArn":
        "arn:aws:elasticloadbalancing:region:123456789012:targetgroup/primary-target-group/
        abcdef123456",
      "containerName": "container-name",
      "containerPort": 80,
      "advancedConfiguration": {
        "alternateTargetGroupArn":
          "arn:aws:elasticloadbalancing:region:account-id:targetgroup/alternate-target-group/
          ghijkl789012",
        "productionListenerRule": "arn:aws:elasticloadbalancing:region:account-
        id:listener-rule/app/load-balancer-name/abcdef123456/listener/ghijkl789012/rule/
        mnopqr345678",
        "roleArn": "arn:aws:iam::123456789012:role/ecs-elb-role"
      }
    }
  ],
  "deploymentConfiguration": {
    "strategy": "BLUE_GREEN",
    "maximumPercent": 200,
    "minimumHealthyPercent": 100,
    "bakeTimeInMinutes": 5
  }
}
```

部署期間的流量流程

在搭配 Elastic Load Balancing 使用藍/綠部署期間，流量會流經系統，如下所示：

1. 初始狀態：所有生產流量都會路由至主要目標群組 (藍色服務修訂版)。
2. 綠色服務修訂版部署：Amazon ECS 會部署新任務，並向替代目標群組註冊新任務。
3. 測試流量：如果已設定測試接聽程式，則測試流量會路由至替代目標群組，以驗證綠色服務修訂版。
4. 生產流量轉移：Amazon ECS 會更新生產接聽程式規則，將流量路由至替代目標群組 (綠色服務修訂版)。
5. 封裝時間：在生產流量轉移後，藍色與綠色服務修訂版同時執行的持續時間。
6. 完成：成功部署後，藍色服務修訂版會終止。

如果在部署期間偵測到問題，Amazon ECS 可以透過將流量路由回主要目標群組 (藍色服務修訂版) 實現自動復原。

Amazon ECS 藍/綠、線性和 Canary 部署的 Network Load Balancer 資源

若要搭配 Amazon ECS 藍/綠部署使用 Network Load Balancer，需要設定特定資源，以啟用藍色與綠色服務修訂版之間的流量路由。本節說明必要的元件及其組態。

如果組態包含 Network Load Balancer，Amazon ECS 會向下列生命週期階段新增 10 分鐘延遲：

- TEST_TRAFFIC_SHIFT
- PRODUCTION_TRAFFIC_SHIFT

此延遲旨在應對 Network Load Balancer 的計時問題，這些問題可能導致設定的流量權重與資料平面中的實際流量路由不相符。

目標群組

對於搭配 Network Load Balancer 使用的藍/綠部署，您需要建立兩個目標群組：

- 用於藍色服務修訂版的主要目標群組 (目前生產流量)
- 用於綠色服務修訂版的替代目標群組 (新服務修訂版)

兩個目標群組都應使用下列設定進行設定：

- 目標類型：ip (適用於 Fargate 或使用 awsvpc 網路模式的 EC2)
- 通訊協定：TCP (或應用程式使用的通訊協定)

- 連接埠：應用程式監聽的連接埠 (對於 HTTP 通常為 80)
- VPC：與 Amazon ECS 任務相同的 VPC
- 運作狀態檢查設定：設定為正確檢查應用程式的運作狀態

對於 TCP 運作狀態檢查，Network Load Balancer 會與目標建立 TCP 連線。如果連線成功，則目標會視為運作狀態良好。

對於 HTTP/HTTPS 運作狀態檢查，Network Load Balancer 會將 HTTP/HTTPS 請求傳送至目標，並驗證回應。

在藍/綠部署期間，Amazon ECS 會根據部署階段，自動向相應的目標群組註冊任務。

Example 為 Network Load Balancer 建立目標群組

下列 AWS CLI 命令會建立兩個目標群組，以便在藍/綠部署中與 Network Load Balancer 搭配使用：

```
aws elbv2 create-target-group \  
  --name blue-target-group \  
  --protocol TCP \  
  --port 80 \  
  --vpc-id vpc-abcd1234 \  
  --target-type ip \  
  --health-check-protocol TCP  
  
aws elbv2 create-target-group \  
  --name green-target-group \  
  --protocol TCP \  
  --port 80 \  
  --vpc-id vpc-abcd1234 \  
  --target-type ip \  
  --health-check-protocol TCP
```

Network Load Balancer

您需要建立包含下列組態的 Network Load Balancer：

- 結構描述：面向網際網路或內部，視要求而定
- IP 位址類型：IPv4
- VPC：與 Amazon ECS 任務相同的 VPC
- 子網路：至少兩個位於不同可用區域的子網路

與 Application Load Balancer 不同，Network Load Balancer 會在傳輸層 (第 4 層) 運作，並且不會使用安全群組。反之，您需要確保與 Amazon ECS 任務相關聯的安全群組允許來自接聽程式連接埠上 Network Load Balancer 的流量。

Example 建立 Network Load Balancer

下列 AWS CLI 命令會建立 Network Load Balancer 以用於藍/綠部署：

```
aws elbv2 create-load-balancer \  
  --name my-network-load-balancer \  
  --type network \  
  --subnets subnet-12345678 subnet-87654321
```

搭配藍/綠部署使用 NLB 的考量

搭配藍/綠部署使用 Network Load Balancer 時，請考量下列事項：

- 第 4 層運作：Network Load Balancer 會在傳輸層 (第 4 層) 運作，且不會檢查應用程式層 (第 7 層) 的內容。這表示您無法使用 HTTP 標頭或路徑作出路由決策。
- 運作狀態檢查：Network Load Balancer 運作狀態檢查僅限於 TCP、HTTP 或 HTTPS 通訊協定。對於 TCP 運作狀態檢查，Network Load Balancer 僅會驗證是否可以建立連線。
- 連線保留：Network Load Balancer 會保留用戶端的來源 IP 位址，這對於安全與記錄用途非常有用。
- 靜態 IP 位址：Network Load Balancer 會為每個子網路提供靜態 IP 位址，這在需要列入允許清單或用戶端需要連線至固定 IP 位址時非常有用。
- 測試流量：由於 Network Load Balancer 不支援以內容為基礎的路由，測試流量必須傳送至與生產流量不同的連接埠。

接聽程式和規則

對於搭配 Network Load Balancer 使用的藍/綠部署，您需要設定接聽程式：

- 生產接聽程式：處理生產流量 (通常位於連接埠 80 或 443)
 - 最初將流量轉送至主要目標群組 (藍色服務修訂版)
 - 部署之後，將流量轉送至替代目標群組 (綠色服務修訂版)
- 測試接聽程式 (選用)：處理測試流量，以在轉移生產流量之前驗證綠色服務修訂版
 - 可以在不同的連接埠上設定 (例如 8080 或 8443)
 - 在測試期間將流量轉送至替代目標群組 (綠色服務修訂版)

與 Application Load Balancer 不同，Network Load Balancer 不支援以內容為基礎的路由規則。反之，流量會根據接聽程式連接埠與通訊協定進行路由。

下列 AWS CLI 命令會建立 Network Load Balancer 的生產和測試接聽程式：

將 *user-input* 取代為實際值。

```
aws elbv2 create-listener \  
  --load-balancer-arn arn:aws:elasticloadbalancing:region:123456789012:loadbalancer/net/my-network-lb/1234567890123456 \  
  --protocol TCP \  
  --port 80 \  
  --default-actions Type=forward,  
  TargetGroupArn=arn:aws:elasticloadbalancing:region:123456789012:targetgroup/blue-target-group/1234567890123456  
  
aws elbv2 create-listener \  
  --load-balancer-arn arn:aws:elasticloadbalancing:region:123456789012:loadbalancer/net/my-network-lb/1234567890123456 \  
  --protocol TCP \  
  --port 8080 \  
  --default-actions Type=forward,  
  TargetGroupArn=arn:aws:elasticloadbalancing:region:123456789012:targetgroup/green-target-group/1234567890123456
```

服務組態

您必須擁有許可，才能允許 Amazon ECS 代表您管理叢集中的負載平衡器資源。如需詳細資訊，請參閱[負載平衡器的 Amazon ECS 基礎結構 IAM 角色](#)。

為搭配 Network Load Balancer 使用的藍/綠部署建立或更新 Amazon ECS 服務時，需要指定下列組態：

將 *user-input* 取代為實際值。

此組態中的關鍵元件包括：

- targetGroupArn：主要目標群組的 ARN (藍色服務修訂版)
- alternateTargetGroupArn：替代目標群組的 ARN (綠色服務修訂版)
- productionListenerRule：用於生產流量的接聽程式 ARN
- testListenerRule：(選用) 用於測試流量的接聽程式 ARN
- roleArn：允許 Amazon ECS 管理 Network Load Balancer 資源的角色 ARN

- `strategy`：設定為 `BLUE_GREEN` 以啟用藍/綠部署
- `bakeTimeInMinutes`：生產流量轉移後，藍色和綠色服務修訂同時執行的持續時間

```
{
  "loadBalancers": [
    {
      "targetGroupArn":
        "arn:aws:elasticloadbalancing:region:123456789012:targetgroup/blue-target-
        group/1234567890123456",
      "containerName": "container-name",
      "containerPort": 80,
      "advancedConfiguration": {
        "alternateTargetGroupArn":
          "arn:aws:elasticloadbalancing:region:123456789012:targetgroup/green-target-
          group/1234567890123456",
        "productionListenerRule":
          "arn:aws:elasticloadbalancing:region:123456789012:listener/net/my-network-
          lb/1234567890123456/1234567890123456",
        "testListenerRule":
          "arn:aws:elasticloadbalancing:region:123456789012:listener/net/my-network-
          lb/1234567890123456/2345678901234567",
        "roleArn": "arn:aws:iam::123456789012:role/ecs-nlb-role"
      }
    }
  ],
  "deploymentConfiguration": {
    "strategy": "BLUE_GREEN",
    "maximumPercent": 200,
    "minimumHealthyPercent": 100,
    "bakeTimeInMinutes": 5
  }
}
```

部署期間的流量流程

在搭配 Network Load Balancer 使用藍/綠部署期間，流量會流經系統，如下所示：

1. 初始狀態：所有生產流量都會路由至主要目標群組 (藍色服務修訂版)。
2. 綠色服務修訂版部署：Amazon ECS 會部署新任務，並向替代目標群組註冊新任務。
3. 測試流量：如果已設定測試接聽程式，則測試流量會路由至替代目標群組，以驗證綠色服務修訂版。

4. 生產流量轉移：Amazon ECS 會更新生產接聽程式，將流量路由至替代目標群組 (綠色服務修訂版)。
5. 封裝時間：在生產流量轉移後，藍色與綠色服務修訂版同時執行的持續時間。
6. 完成：成功部署後，藍色服務修訂版會終止。

如果在部署期間偵測到問題，Amazon ECS 可以透過將流量路由回主要目標群組 (藍色服務修訂版) 實現自動復原。

Amazon ECS 藍色/綠色、線性和 Canary 部署的 Service Connect 資源

搭配藍/綠部署使用 Service Connect 時，您需要設定特定元件，才能啟用藍色與綠色服務修訂版之間適當的流量路由。本節說明必要的元件及其組態。

架構概觀

Service Connect 透過自動注入 Amazon ECS 任務的受管邊車代理，建置服務探索與服務網格功能。這些代理處理路由決策、重試和指標收集，同時 AWS Cloud Map 提供服務登錄後端。當您在啟用 Service Connect 的情況下部署服務時，服務會自行註冊 AWS Cloud Map，而用戶端服務會透過命名空間探索該服務。

在標準 Service Connect 實作中，用戶端服務會連線至邏輯服務名稱，而邊車代理則會處理路由至實際服務執行個體的作業。在藍/綠部署中，此模型會透過 `testTrafficRules` 組態擴展為包含測試流量路由。

在藍/綠部署期間，下列關鍵元件可搭配運作：

- Service Connect Proxy：服務之間的所有流量都會通過 Service Connect Proxy，這會根據組態作出路由決策。
- AWS Cloud Map 註冊：藍色和綠色部署都會向註冊 AWS Cloud Map，但綠色部署最初會註冊為「測試」端點。
- 測試流量路由：Service Connect 組態中的 `testTrafficRules` 決定了如何識別測試流量並將其路由至綠色部署。這是透過以標頭為基礎的路由實現，請求中的特定 HTTP 標頭會將流量導向測試修訂版。依預設，當未指定自訂規則時，Service Connect 會識別以 HTTP 為基礎的通訊協定的 `x-amzn-ecs-blue-green-test` 標頭。
- 用戶端組態：命名空間中的所有用戶端都會自動接收生產與測試路由，但只有符合測試規則的請求才會傳送至綠色部署。

這種方法的強大之處在於，它處理了轉移期間複雜的服務探索問題。隨著流量從藍色部署轉移到綠色部署，所有連線與探索機制都會自動更新。無需單獨更新 DNS 記錄、重新設定負載平衡器或部署服務探索變更，因為服務網格會處理所有工作。

流量路由與測試

Service Connect 為藍/綠部署提供進階的流量路由功能，包括以標頭為基礎的路由與用於測試案例的用戶端別名組態。

測試流量標頭規則

在藍/綠部署期間，您可以設定測試流量標頭規則，將特定請求路由至綠色 (新) 服務修訂版以供測試之用。這可讓您在完成部署之前，使用受控流量驗證新版本。

Service Connect 使用以標頭為基礎的路由來識別測試流量。依預設，當未指定自訂規則時，Service Connect 會識別以 HTTP 為基礎的通訊協定的 `x-amzn-ecs-blue-green-test` 標頭。當請求中出現此標頭時，Service Connect Proxy 會自動將請求路由至綠色部署進行測試。

測試流量標頭規則可讓您：

- 將具有特定標頭的請求路由至綠服務修訂版
- 使用流量子集測試新功能
- 在流量完全切換之前驗證服務行為
- 實作 Canary 測試策略
- 在類似生產的環境中執行整合測試

以標頭為基礎的路由機制可與現有應用程式架構無縫搭配運作。用戶端服務無需了解藍綠部署程序，只需在傳送測試請求時包含相應的標頭，Service Connect Proxy 就會自動處理路由邏輯。

如需有關設定測試流量標頭規則的詳細資訊，請參閱 Amazon Elastic Container Service API Reference 中的 [ServiceConnectTestTrafficHeaderRules](#)。

標頭比對規則

標頭比對規則定義了藍/綠部署期間路由測試流量的準則。您可以設定多個比對條件，以精確控制哪些請求要路由至綠服務修訂版。

標頭比對支援：

- 精確標頭值比對

- 檢查標頭是否存在
- 模式型比對
- 多個標頭組合

範例使用案例包括將具有特定使用者代理字串、API 版本或功能旗標的請求路由至綠服務，以進行測試。

如需有關標頭比對組態的詳細資訊，請參閱 Amazon Elastic Container Service API Reference 中的 [ServiceConnectTestTrafficHeaderMatchRules](#)。

用於藍/綠部署的用戶端別名

用戶端別名可在藍/綠部署期間為服務提供穩定的 DNS 端點。這些別名可在藍色與綠色服務修訂版之間實現無縫流量路由，而無需用戶端應用程式變更其連線端點。

在藍/綠部署期間，用戶端別名會：

- 為用戶端連線維持一致的 DNS 名稱
- 啟用服務修訂版之間的自動流量切換
- 支援漸進式流量遷移策略
- 透過將流量重新導向至藍色修訂版，提供復原功能

您可以為不同的連接埠或通訊協定設定多個用戶端別名，讓複雜的服務架構在部署期間維持連線能力。

如需有關用戶端別名組態的詳細資訊，請參閱 Amazon Elastic Container Service API Reference 中的 [ServiceConnectClientAlias](#)。

流量路由最佳實務

使用 Service Connect 實作藍/綠部署的流量路由時，請考量下列最佳實務：

- 從標頭型測試開始著手：先使用測試流量標頭規則透過受控流量來驗證綠服務，再切換所有流量。
- 設定運作狀態檢查：確保藍色與綠色服務均已設定相應的運作狀態檢查，防止將流量路由至運作狀態不良的執行個體。
- 監控服務指標：在部署期間追蹤兩個服務修訂版的關鍵績效指標，以及早識別問題。
- 規劃復原策略：設定用戶端別名與路由規則，以便在偵測到問題時能夠快速復原至藍色服務。
- 測試標頭比對邏輯：先在非生產環境中驗證您的標頭比對規則，再將規則套用至生產部署。

Service Connect 藍/綠部署工作流程

了解 Service Connect 如何管理藍/綠部署程序，可協助有效地實作部署並對其進行疑難排解。下列工作流程展示了不同元件如何在部署的每個階段互動。

部署階段

Service Connect 藍/綠部署會經歷數個不同的階段：

1. 初始狀態：藍色服務處理 100% 的生產流量。命名空間中的所有用戶端服務會透過 Service Connect 中設定的邏輯服務名稱連線至藍色服務。
2. Green Service Registration：當綠色部署開始時，它會向註冊 AWS Cloud Map 為「測試」端點。用戶端服務中的 Service Connect Proxy 會自動接收生產與測試路由組態。
3. 測試流量路由：包含測試流量標頭 (例如 `x-amzn-ecs-blue-green-test`) 的請求會由 Service Connect Proxy 自動路由至綠色服務。生產流量會繼續流向藍色服務。
4. 流量轉移準備：成功測試後，部署程序會準備進行生產流量轉移。藍色與綠色服務都會保持註冊狀態且運作狀態良好。
5. 生產流量轉移：Service Connect 組態會更新，將生產流量路由至綠色服務。此動作會自動發生，無需更新用戶端服務或變更 DNS。
6. 封裝時間段：在生產流量轉移後，藍色與綠色服務修訂版同時執行的持續時間。
7. 藍色服務取消註冊：在成功的流量轉移和驗證之後，藍色服務會從取消註冊 AWS Cloud Map 並終止，完成部署。

Service Connect Proxy 行為

Service Connect Proxy 在管理藍/綠部署期間的流量方面起著重要作用。了解其行為可協助您設計有效的測試與部署策略。

藍/綠部署期間的關鍵代理行為：

- 自動路由探索：代理會自動探索來自的生產和測試路由 AWS Cloud Map，而不需要重新啟動應用程式或變更組態。
- 以標頭為基礎的路由：代理會檢查傳入的請求標頭，並根據設定的測試流量規則將流量路由至相應的服務修訂版。
- 運作狀態檢查整合：代理僅會將流量路由至運作狀態良好的服務執行個體，自動從路由集區中排除運作狀態不良的任務。
- 重試與斷路器：代理會提供內建的重試邏輯與斷路器功能，從而改善部署期間的恢復能力。

- 指標收集：代理會收集藍色與綠色服務的詳細指標，以便在部署期間進行全面監控。

服務探索更新

使用 Service Connect 進行藍/綠部署的其中一個主要優勢是自動處理服務探索更新。傳統的藍/綠部署通常需要複雜的 DNS 更新或負載平衡器重新設定，但 Service Connect 會透明地管理這些變更。

在部署期間，Service Connect 會處理：

- 命名空間更新：Service Connect 命名空間會自動包含藍色與綠色服務端點，並具有相應的路由規則。
- 用戶端組態：命名空間中的所有用戶端服務會自動接收更新後的路由資訊，無需重新啟動或重新部署。
- 逐步轉移：服務探索更新會逐步且安全地進行，確保不會中斷正在進行的請求。
- 復原支援：如果需要復原，Service Connect 可以快速還原服務探索組態，將流量路由回藍色服務。

建立 Amazon ECS 藍/綠部署

透過使用 Amazon ECS 藍/綠部署，您可以進行服務變更並加以測試，然後在生產環境中實作這些變更。

先決條件

開始藍/綠部署之前，請執行下列動作。

1. 設定相應的許可。
 - 如需有關 Elastic Load Balancing 許可的資訊，請參閱[負載平衡器的 Amazon ECS 基礎結構 IAM 角色](#)。
 - 如需有關 Lambda 許可的資訊，請參閱[Amazon ECS 藍/綠部署中 Lambda 函式所需的許可](#)
2. Amazon ECS 藍/綠部署要求服務使用下列其中一項功能：設定相應的資源。
 - Application Load Balancer – 如需詳細資訊，請參閱[藍/綠、線性和金絲雀部署的 Application Load Balancer 資源](#)。
 - Network Load Balancer – 如需詳細資訊，請參閱[Amazon ECS 藍/綠、線性和 Canary 部署的 Network Load Balancer 資源](#)。
 - Service Connect – 如需詳細資訊，請參閱[Amazon ECS 藍色/綠色、線性和 Canary 部署的 Service Connect 資源](#)。

3. 決定是否要執行生命週期階段的 Lambda 函數。

- PRE_SCALE_UP
- POST_SCALE_UP
- TEST_TRAFFIC_SHIFT
- POST_TEST_TRAFFIC_SHIFT
- PRODUCTION_TRAFFIC_SHIFT
- POST_PRODUCTION_TRAFFIC_SHIFT

如需詳細資訊，請參閱 AWS Lambda Developer Guide 中的 [Create a Lambda function with the console](#)。

程序

您可以使用 主控台或 AWS CLI 來建立 Amazon ECS 藍/綠服務。

Console

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。 「叢集詳細資訊」頁面隨即顯示。 b. 在 Services (服務) 標籤上，選擇 Create (建立)。	
任務定義	a. 在任務定義頁面中選取任務定義。 b. 從部署功能表中選擇建立服務。	

建立服務頁面隨即顯示。

3. 在服務詳細資訊下，執行下列動作：
 - a. 在任務定義系列欄位中選擇要使用的任務定義。然後，在任務定義修訂版欄位中輸入要使用的修訂版。
 - b. 針對 Service name (服務名稱)，輸入服務的名稱。
4. 若要在現有叢集中執行服務，請在現有叢集欄位中選擇叢集。若要在新叢集中執行服務，請選擇建立叢集
5. 選擇任務在叢集基礎結構中的分佈方式。在運算組態下選擇選項。

運算選項	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。	

運算選項	步驟	
	 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	- 在運算選項區段中，選取啟動類型。 - 針對 啟動類型，選擇啟動類型。 (選用) 當指定 Fargate 時，請在平台版本欄位中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。	

6. 在部署組態下，執行下列動作：
 - 在服務類型欄位中選擇複本。
 - 針對 Desired tasks (所需任務)，輸入要在服務中啟動並維護的任務數。
 - 若要讓 Amazon ECS 監控可用區域間的任務分佈，並在出現不平衡時重新分佈，請在可用區域服務重新平衡下，選取可用區域服務重新平衡。
 - 在運作狀態檢查寬限期欄位中，輸入時間 (以秒為單位)，這是服務排程器在任務首次啟動後忽略運作狀態不良的 Elastic Load Balancing、VPC Lattice 與容器運作狀態檢查的時間。如果您未指定運作狀態檢查寬限期間值，則會使用預設值 0。
7.
 - 在封裝時間欄位中，輸入在藍色修訂版終止之前，藍色與綠色服務修訂版將同時執行的分鐘數。這為驗證與測試提供了時間。
 - (選用) 執行要在部署的特定階段執行的 Lambda 函式。在部署 lifecycle hook 下，選取要執行 lifecycle hook 的階段。

若要新增 lifecycle hook：

- a. 選擇新增。
 - b. 在 Lambda 函式欄位中，輸入函式名稱或 ARN。
 - c. 在角色欄位中，選取具有調用 Lambda 函式許可的 IAM 角色。
 - d. 在生命週期階段欄位中，選取 Lambda 函式應執行的階段。
8. 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)。

在部署斷路器將部署設定為失敗狀態時，若要讓軟體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. 若要根據應用程式指標停止部署，請選取使用 CloudWatch 警示。然後，從 CloudWatch 警示名稱欄位中選擇警示。要建立新的警示，請前往 CloudWatch 主控台。

在 CloudWatch 警示將部署設定為失敗狀態時，若要讓軟體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。
9. (選用) 若要使用 Service Connect 與服務互連，請展開 Service Connect 區段，然後指定下列項目：
 - a. 選取開啟 Service Connect。
 - b. 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至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)。
 - c. 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。這可以是在您的 中在相同 AWS 區域 中分別建立的命名空間，AWS 帳戶 或是使用 AWS Resource Access Manager () 與您的帳戶共用的相同區域中的命名空間 AWS RAM。如需共用 AWS Cloud Map 命名空間的詳細資訊，請參閱《AWS Cloud Map 開發人員指南》中的[跨帳戶 AWS Cloud Map 命名空間共用](#)。
 - d. (選用) 設定藍/綠部署的測試流量標頭規則。在測試流量路由下，指定下列項目：
 - i. 選取啟用測試流量標頭規則，以在測試期間將特定請求路由至綠色服務修訂版。
 - ii. 在標頭比對規則欄位中，設定路由測試流量的條件：

- 標頭名稱：輸入要比對的 HTTP 標頭名稱 (例如 X-Test-Version 或 User-Agent)。
- 比對類型：選擇比對條件：
 - 完全相符：標頭值與指定值完全相符的路由請求
 - 標頭存在：包含指定標頭的路由請求，無論其值為何
 - 模式比對：標頭值與指定模式相符的路由請求
- 標頭值 (如果使用「完全相符」或「模式相符」)：輸入要比對的值或模式。

您可以新增多個標頭比對規則來建立複雜的路由邏輯。符合任何已設定規則的請求將路由至綠色服務修訂版進行測試。

- iii. 選擇新增標頭規則以設定其他標頭比對條件。

 Note

測試流量標頭規則讓您可以在完成完整部署之前，使用受控流量驗證新功能。這允許您使用特定請求 (例如來自內部測試工具或 Beta 版使用者的請求) 測試綠色服務修訂版，同時維持流向藍色服務修訂版的正常流量流程。

- e. (選用) 指定日誌組態。選取使用日誌收集。預設選項會將容器日誌傳送至 CloudWatch 日誌。其他日誌驅動程式選項是使用 AWS FireLens 設定。如需詳細資訊，請參閱[將 Amazon ECS 日誌傳送至 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 – 將任務設定為將容器日誌傳送至 CloudWatch Logs。系統會提供預設日誌驅動程式選項，用來代表您建立 CloudWatch 日誌群組。若要指定不同的日誌群組名稱，請變更驅動程式選項值。
- Amazon Data Firehose – 將任務設定為將容器日誌傳送至 Firehose。系統會提供預設日誌驅動程式選項，用來將日誌傳送至 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 – 將任務設定為將容器日誌傳送至 Amazon Kinesis Data Streams。系統會提供預設日誌驅動程式選項，用來將日誌傳送至 Kinesis Data Streams 串流。若要指定不同的串流名稱，請變更驅動程式選項值。

- Amazon OpenSearch Service – 將任務設定為將容器日誌傳送至 OpenSearch Service 網域。務必提供日誌驅動程式選項。
- Amazon S3：將任務設定為將容器日誌傳送至 Amazon S3 儲存貯體。系統會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。

10. (選用) 設定藍/綠部署的負載平衡。

Elastic Load Balancing 類型	步驟
Application Load Balancer	- 在負載平衡器類型欄位中選擇 Application Load Balancer。 - 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。 - 在容器欄位中選擇託管服務的容器。 - 針對 Load balancer name (負載平衡器名稱)，輸入不重複的名稱。 - 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。 - 在生產規則欄位中，輸入規則的評估順序與路徑模式。

Elastic Load Balancing 類型	步驟	
	此規則適用於您的生產服務修訂流量。 - 在測試規則欄位中，輸入規則的評估順序與路徑模式。 此規則適用於您的測試服務修訂流量。 f. 在目標群組欄位中設定下列項目： - 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。 - 在通訊協定欄位中，選擇 Application Load Balancer 向其路由請求之目標群組的通訊協定。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。 - 針對取消註冊延遲，輸入負載平衡器將目標狀態變更為 的秒數UNUSED。預設為 300 秒。 - 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	

Elastic Load Balancing 類型	步驟	
	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 (/)。 在替代群組名稱欄位中，輸入測試 (綠色) 服務修訂版的目標群組名稱。	

Elastic Load Balancing 類型	步驟	
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. 在生產接聽程式欄位中，選擇生產接聽程式連接埠與生產接聽程式通訊協定。 這是生產服務修訂流量的接聽程式。 e. 在測試接聽程式欄位中，選擇測試接聽程式連接埠與測試接聽程式通訊協定。 這是測試服務修訂流量的接聽程式。 f. 在目標群組欄位中設定下列項目： 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。	

Elastic Load Balancing 類型	步驟	
	• 在通訊協定欄位中，選擇 Network Load Balancer 向其路由請求之目標群組的通訊協定。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。 • 針對取消註冊延遲，輸入負載平衡器將目標狀態變更為 的秒數UNUSED。預設為 300 秒。 • 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 (/)。 • 在替代群組名稱欄位中，輸入測試 (綠色) 服務修訂版的目標群組名稱。	

11. (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。

12. 選擇建立。

AWS CLI

1. 建立名為 service-definition.json 且具有下列內容的檔案。

將 *user-input* 取代為實際值。

```
{
  "serviceName": "myBlueGreenService",
  "cluster": "arn:aws:ecs:us-west-2:123456789012:cluster/sample-fargate-cluster",
  "taskDefinition": "sample-fargate:1",
  "desiredCount": 5,
  "launchType": "FARGATE",
  "networkConfiguration": {
    "awsvpcConfiguration": {
      "subnets": [
        "subnet-09ce6e74c116a2299",
        "subnet-00bb3bd7a73526788",
        "subnet-0048a611aaec65477"
      ],
      "securityGroups": [
        "sg-09d45005497daa123"
      ],
      "assignPublicIp": "ENABLED"
    }
  },
  "deploymentController": {
    "type": "ECS"
  }
}
```

```

},
"deploymentConfiguration": {
  "strategy": "BLUE_GREEN",
  "maximumPercent": 200,
  "minimumHealthyPercent": 100,
  "bakeTimeInMinutes": 2,
  "alarms": {
    "alarmNames": [
      "myAlarm"
    ],
    "rollback": true,
    "enable": true
  },
  "lifecycleHooks": [
    {
      "hookTargetArn": "arn:aws:lambda:us-  
west-2:7123456789012:function:checkExample",
      "roleArn": "arn:aws:iam::123456789012:role/ECSLifecycleHookInvoke",
      "lifecycleStages": [
        "PRE_SCALE_UP"
      ]
    }
  ]
},
"loadBalancers": [
  {
    "targetGroupArn": "arn:aws:elasticloadbalancing:us-  
west-2:123456789012:targetgroup/blue-target-group/54402ff563af1197",
    "containerName": "fargate-app",
    "containerPort": 80,
    "advancedConfiguration": {
      "alternateTargetGroupArn": "arn:aws:elasticloadbalancing:us-  
west-2:123456789012:targetgroup/green-target-group/cad10a56f5843199",
      "productionListenerRule": "arn:aws:elasticloadbalancing:us-  
west-2:123456789012:listener-rule/app/my-blue-green-  
demo/32e0e4f946c3c05b/9cfa8c482e204f7d/831dbaf72edb911",
      "roleArn": "arn:aws:iam::123456789012:role/LoadBalancerManagementforECS"
    }
  }
]
}

```

2. 執行 create-service。

將 `user-input` 取代為實際值。

```
aws ecs create-service --cli-input-json file://service-definition.json
```

或者，您可以使用下列範例來建立具有負載平衡器組態的藍/綠部署服務：

```
aws ecs create-service \
  --cluster "arn:aws:ecs:us-west-2:123456789012:cluster/MyCluster" \
  --service-name "blue-green-example-service" \
  --task-definition "nginxServer:1" \
  --launch-type "FARGATE" \
  --network-configuration
  "awsvpcConfiguration={subnets=[subnet-12345,subnet-67890,subnet-abcdef,subnet-
  fedcba],securityGroups=[sg-12345],assignPublicIp=ENABLED}" \
  --desired-count 3 \
  --deployment-controller "type=ECS" \
  --deployment-configuration
  "strategy=BLUE_GREEN,maximumPercent=200,minimunHealthyPercent=100,bakeTimeInMinutes=0" \
  --load-balancers "targetGroupArn=arn:aws:elasticloadbalancing:us-
  west-2:123456789012:targetgroup/MyBGtg1/
  abcdef1234567890,containerName=nginx,containerPort=80,advancedConfiguration={alternateTa
  west-2:123456789012:targetgroup/
  MyBGtg2/0987654321fedcba,productionListenerRule=arn:aws:elasticloadbalancing:us-
  west-2:123456789012:listener-rule/app/
  MyLB/1234567890abcdef/1234567890abcdef,roleArn=arn:aws:iam::123456789012:role/
  ELBManagementRole}"
```

後續步驟

- 更新服務以開始部署。如需詳細資訊，請參閱[更新 Amazon ECS 服務](#)。
- 監控部署程序，確保其遵循藍/綠模式：
 - 綠色服務修訂版已建立並向上擴展
 - 測試流量會路由至綠色修訂版 (若已設定)
 - 生產流量會轉移至綠色修訂版
 - 封裝時間過期後，藍色修訂版會終止

對 Amazon ECS 藍/綠部署進行疑難排解

下列內容提供搭配 Amazon ECS 使用藍/綠部署時可能遇到的常見問題的解決方案。藍/綠部署錯誤可能會發生在下列階段：

- 同步路徑：在回應 CreateService 或 UpdateService API 呼叫時立即出現的錯誤。
- 非同步路徑：出現在 DescribeServiceDeployments 的 statusReason 欄位中並導致部署復原的錯誤

Tip

您可以[Amazon ECS MCP 伺服器](#)搭配 AI 助理使用 來監控部署，並使用自然語言疑難排解部署問題。

負載平衡器組態問題

負載平衡器組態是 Amazon ECS 中藍/綠部署的關鍵元件。正確設定接聽程式規則、目標群組與負載平衡器類型，對於成功部署至關重要。本節涵蓋可能導致藍/綠部署失敗的常見負載平衡器組態問題。

對負載平衡器問題進行疑難排解時，務必了解接聽程式規則與目標群組之間的關係。在藍/綠部署中：

- 生產接聽程式規則會將流量導向目前作用中的 (藍色) 服務修訂版
- 測試接聽程式規則可用於在轉移生產流量之前驗證新的 (綠色) 服務修訂版
- 目標群組用於註冊每個服務修訂版中的容器執行個體
- 部署期間，流量會透過調整接聽程式規則中目標群組的權重，逐漸從藍色服務修訂版轉移至綠色服務修訂版

接聽程式規則組態錯誤

下列問題與藍/綠部署的接聽程式規則組態不正確有關。

使用 Application Load Balancer 接聽程式 ARN 而非接聽程式規則 ARN

```
錯誤訊息：productionListenerRule has an invalid ARN format.  
Must be RuleArn for ALB or ListenerArn for NLB. Got:  
arn:aws:elasticloadbalancing:us-west-2:123456789012:listener/app/my-alb/  
abc123/def456
```

解決方案：使用 Application Load Balancer 時，必須為 `productionListenerRule` 與 `testListenerRule` 指定接聽程式規則 ARN，而不是接聽程式 ARN。對於 Network Load Balancer，必須使用接聽程式 ARN。

如需有關如何查找接聽程式 ARN 的資訊，請參閱 Application Load Balancer User Guide 中的 [Listeners for your Application Load Balancers](#)。規則的 ARN 格式為 `arn:aws:elasticloadbalancing:region:account-id:listener-rule/app/...`。

為生產與測試接聽程式使用相同的規則

錯誤訊息：The following rules cannot be used as both production and test listener rules: `arn:aws:elasticloadbalancing:us-west-2:123456789012:listener-rule/app/my-alb/abc123/def456/ghi789`

解決方案：生產流量與測試流量必須使用不同的接聽程式規則。為路由至測試目標群組的測試流量建立單獨的接聽程式規則。

目標群組未與接聽程式規則建立關聯

錯誤訊息：Service deployment rolled back because of invalid networking configuration: Target group `arn:aws:elasticloadbalancing:us-west-2:123456789012:targetgroup/myAlternateTG/abc123` is not associated with either `productionListenerRule` or `testListenerRule`.

解決方案：主要目標群組與替代目標群組都必須與生產接聽程式規則或測試接聽程式規則建立關聯。更新負載平衡器組態，確保兩個目標群組皆與接聽程式規則正確建立關聯。

Application Load Balancer 缺少測試接聽程式規則

錯誤訊息：For Application LoadBalancer, `testListenerRule` is required when `productionListenerRule` is not associated with both `targetGroup` and `alternateTargetGroup`

解決方案：使用 Application Load Balancer 時，如果兩個目標群組都未與生產接聽程式規則建立關聯，則必須指定測試接聽程式規則。將 `testListenerRule` 新增至組態，並確保兩個目標群組皆與生產或測試接聽程式規則建立關聯。如需詳細資訊，請參閱 Application Load Balancer User Guide 中的 [Listeners for your Application Load Balancers](#)。

目標群組組態錯誤

下列問題與藍/綠部署的目標群組組態不正確有關。

接聽程式規則中存在多個具有流量的目標群組

錯誤訊息：Service deployment rolled back because of invalid networking configuration. productionListenerRule arn:aws:elasticloadbalancing:us-west-2:123456789012:listener-rule/app/my-alb/abc123/def456/ghi789 should have exactly one target group serving traffic but found 2 target groups which are serving traffic

解決方案：在開始藍/綠部署之前，請確保接聽程式規則中只有一個目標群組正在接收流量 (具有非零權重)。更新接聽程式規則組態，將不應接收流量的任何目標群組的權重設定為零。

負載平衡器項目中的重複目標群組

錯誤訊息：Duplicate targetGroupArn found: arn:aws:elasticloadbalancing:us-west-2:123456789012:targetgroup/myecs-targetgroup/abc123

解決方案：每個目標群組 ARN 在服務定義的所有負載平衡器項目中必須是唯一的。檢閱組態，並確保為每個負載平衡器項目使用不同的目標群組。

生產接聽程式規則中存在未預期的目標群組

錯誤訊息：Service deployment rolled back because of invalid networking configuration. Production listener rule is forwarding traffic to unexpected target group arn:aws:elasticloadbalancing:us-west-2:123456789012:targetgroup/random-nlb-tg/abc123. Expected traffic to be forwarded to either targetGroupArn: arn:aws:elasticloadbalancing:us-west-2:123456789012:targetgroup/nlb-targetgroup/def456 or alternateTargetGroupArn: arn:aws:elasticloadbalancing:us-west-2:123456789012:targetgroup/nlb-tg-alternate/ghi789

解決方案：生產接聽程式規則正在將流量轉送至服務定義中未指定的目標群組。確保將接聽程式規則設定為僅將流量轉送至服務定義中指定的目標群組。

如需詳細資訊，請參閱 Application Load Balancer User Guide 中的 [forward actions](#)。

負載平衡器類型組態錯誤

下列問題與藍/綠部署的負載平衡器類型組態不正確有關。

混合使用 Classic Load Balancer 與 Application Load Balancer 或 Network Load Balancer 組態

錯誤訊息：All loadBalancers must be strictly either ELBv1 (defining loadBalancerName) or ELBv2 (defining targetGroupArn)

Note

Classic Load Balancer 是 Elastic Load Balancing 的上一代負載平衡器。建議遷移至目前一代的負載平衡器。如需詳細資訊，請參閱 [Migrate your Classic Load Balancer](#)。

解決方案：。使用所有 Classic Load Balancer 或所有 Application Load Balancer 與 Network Load Balancer。

對於 Application Load Balancer 與 Network Load Balancer，僅指定 targetGroupArn 欄位。

搭配 Classic Load Balancer 使用進階組態

錯誤訊息：advancedConfiguration field is not allowed with ELBv1 loadBalancers

解決方案：只有 Application Load Balancer 與 Network Load Balancer 支援藍/綠部署的進階組態。如果使用 Classic Load Balancer (透過 loadBalancerName 指定)，則無法使用 advancedConfiguration 欄位。請切換到 Application Load Balancer，或移除 advancedConfiguration 欄位。

負載平衡器之間的進階組態不一致

錯誤訊息：Either all or none of the provided loadBalancers must have advancedConfiguration defined

解決方案：如果使用多個負載平衡器，則必須為所有負載平衡器定義 advancedConfiguration，或者都不定義。更新組態，確保所有負載平衡器項目的一致性。

藍/綠部署缺少進階組態

錯誤訊息：advancedConfiguration field is required for all loadBalancers when using a non-ROLLING deployment strategy

解決方案：搭配 Application Load Balancer 使用藍/綠部署策略時，必須為所有負載平衡器項目指定 advancedConfiguration 欄位。將所需的 advancedConfiguration 新增至負載平衡器組態。

許可問題

下列問題與藍/綠部署的許可不足有關。

基礎結構角色缺少信任政策

錯誤訊息：Service deployment rolled back because of invalid networking configuration. ECS was unable to manage the ELB resources due to missing permissions on ECS Infrastructure Role 'arn:aws:iam::123456789012:role/Admin'.

解決方案：指定用於管理負載平衡器資源的 IAM 角色沒有正確的信任政策。更新角色的信任政策，以允許服務擔任角色。信任政策必須包含：

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "ecs.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

負載平衡器角色缺少讀取許可

錯誤訊息：service myService failed to describe target health on target-group myTargetGroup with (error User: arn:aws:sts::123456789012:assumed-role/myELBRole/ecs-service-scheduler is not authorized to perform: elasticloadbalancing:DescribeTargetHealth because no identity-based policy allows the elasticloadbalancing:DescribeTargetHealth action)

解決方案：用於管理負載平衡器資源的 IAM 角色沒有讀取目標運作狀態資訊的許可。將 elasticloadbalancing:DescribeTargetHealth 許可新增至角色的政策。如需有關 Elastic Load Balancing 許可的資訊，請參閱[負載平衡器的 Amazon ECS 基礎結構 IAM 角色](#)。

負載平衡器角色缺少寫入許可

錯誤訊息：service myService failed to register targets in target-group myTargetGroup with (error User: arn:aws:sts::123456789012:assumed-role/myELBRole/ecs-service-scheduler is not authorized to perform: elasticloadbalancing:RegisterTargets on resource: arn:aws:elasticloadbalancing:us-west-2:123456789012:targetgroup/myTargetGroup/abc123 because no identity-based policy allows the elasticloadbalancing:RegisterTargets action)

解決方案：用於管理負載平衡器資源的 IAM 角色沒有註冊目標的許可。將 elasticloadbalancing:RegisterTargets 許可新增至角色的政策。如需有關 Elastic Load Balancing 許可的資訊，請參閱[負載平衡器的 Amazon ECS 基礎結構 IAM 角色](#)。

缺少修改接聽程式規則的許可

錯誤訊息：Service deployment rolled back because TEST_TRAFFIC_SHIFT lifecycle hook(s) failed. User: arn:aws:sts::123456789012:assumed-role/myELBRole/ECSNetworkingWithELB is not authorized to perform: elasticloadbalancing:ModifyListener on resource: arn:aws:elasticloadbalancing:us-west-2:123456789012:listener/app/my-alb/abc123/def456 because no identity-based policy allows the elasticloadbalancing:ModifyListener action

解決方案：用於管理負載平衡器資源的 IAM 角色沒有修改接聽程式的許可。將 elasticloadbalancing:ModifyListener 許可新增至角色的政策。如需有關 Elastic Load Balancing 許可的資訊，請參閱[負載平衡器的 Amazon ECS 基礎結構 IAM 角色](#)。

對於藍/綠部署，建議將 AmazonECS-ServiceLinkedRolePolicy 受管政策連接至基礎結構角色，該角色包含管理負載平衡器資源的所有必要許可。

Lifecycle hook 問題

下列問題與藍/綠部署中的 lifecycle hook 有關。

Lambda 勾點角色上的信任政策不正確

錯誤訊息：Service deployment rolled back because TEST_TRAFFIC_SHIFT lifecycle hook(s) failed. ECS was unable to assume role arn:aws:iam::123456789012:role/Admin

解決方案：為 Lambda lifecycle hook 指定的 IAM 角色沒有正確的信任政策。更新角色的信任政策，以允許服務擔任角色。信任政策必須包含：

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "ecs.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

Lambda 勾點傳回 FAILED 狀態

錯誤訊息：Service deployment rolled back because TEST_TRAFFIC_SHIFT lifecycle hook(s) failed. Lifecycle hook target arn:aws:lambda:us-west-2:123456789012:function:myHook returned FAILED status.

解決方案：指定為 lifecycle hook 的 Lambda 函式傳回了 FAILED 狀態。檢查 Amazon CloudWatch Logs 中的 Lambda 函式日誌以確定失敗原因，並更新函式以正確處理部署事件。

缺少調用 Lambda 函式的許可

錯誤訊息：Service deployment rolled back because TEST_TRAFFIC_SHIFT lifecycle hook(s) failed. ECS was unable to invoke hook target arn:aws:lambda:us-west-2:123456789012:function:myHook due to User: arn:aws:sts::123456789012:assumed-role/myLambdaRole/ECS-Lambda-Execution is not authorized to perform: lambda:InvokeFunction on resource: arn:aws:lambda:us-west-2:123456789012:function:myHook because no identity-based policy allows the lambda:InvokeFunction action

解決方案：用於 Lambda lifecycle hook 的 IAM 角色沒有調用 Lambda 函式的許可。將針對特定 Lambda 函式 ARN 的 lambda:InvokeFunction 許可新增至角色政策。如需有關 Lambda 許可的資訊，請參閱 [Amazon ECS 藍/綠部署中 Lambda 函式所需的許可](#)。

Lambda 函式逾時或回應無效

錯誤訊息：Service deployment rolled back because TEST_TRAFFIC_SHIFT lifecycle hook(s) failed. ECS was unable to parse the response from arn:aws:lambda:us-west-2:123456789012:function:myHook due to HookStatus must not be null

解決方案：Lambda 函式逾時或傳回了無效的回應。確保 Lambda 函式傳回有效的回應，其中 hookStatus 欄位設定為 SUCCEEDED 或 FAILED。此外，檢查 Lambda 函式逾時是否已針對驗證邏輯適當設定。如需詳細資訊，請參閱[Amazon ECS 服務部署的 lifecycle hook](#)。

有效 Lambda 回應範例：

```
{
  "hookStatus": "SUCCEEDED",
  "reason": "Validation passed"
}
```

Amazon ECS 線性部署

線性部署會逐漸將流量從舊服務修訂版轉移到新的服務修訂版，隨著時間的推移以相等遞增，允許您監控每個步驟，然後再繼續進行下一個步驟。透過 Amazon ECS 線性部署，控制流量轉移的速度，並隨著生產流量的增加來驗證新的服務修訂。此方法提供一種可控制的方式來部署變更，並能夠監控每個增量的效能。

線性部署中涉及的資源

以下是 Amazon ECS 線性部署中涉及的資源：

- 流量轉移 - Amazon ECS 用來轉移生產流量的程序。對於 Amazon ECS 線性部署，流量會以相等百分比增量轉移，每個增量之間的可設定等待時間。
- 步驟百分比 - 線性部署期間每個增量移動的流量百分比。此欄位採用 Double 作為值，有效值介於 3.0 到 100.0 之間。
- 步驟製作時間 - 在線性部署期間，每個流量轉移增量之間的等待時間。有效值為 0 - 1440 分鐘。
- 部署製作時間 - 在將所有生產流量轉移到新服務修訂版之後，Amazon ECS 會在終止舊服務修訂版之前等待的時間，以分鐘為單位。這是生產流量轉移後，藍色和綠色服務修訂同時執行的持續時間。
- 生命週期階段 - 部署操作中的一系列事件，例如「生產流量轉移後」。

- 生命週期掛鉤 - 在特定生命週期階段執行的 Lambda 函數。您可以建立可驗證部署的 函數。為每個生產流量轉移步驟調用為 PRODUCTION_TRAFFIC_SHIFT 設定的 Lambda 函數或生命週期掛鉤。
- 目標群組 - 一種 Elastic Load Balancing 資源，用於將請求路由至一個或多個已註冊的目標 (例如 EC2 執行個體)。當您建立接聽程式時，可以為其預設動作指定一個目標群組。流量會轉送至接聽程式規則中指定的目標群組。
- 接聽程式 - 一種 Elastic Load Balancing 資源，用於透過已設定的通訊協定與連接埠檢查連線請求。您為接聽程式定義的規則決定了 Amazon ECS 如何將請求路由至已註冊的目標。
- 規則 - 與接聽程式相關聯的 Elastic Load Balancing 資源。規則定義了請求的路由方式，包含動作、條件與優先順序。

考量事項

選擇部署類型時，請考量下列事項：

- 資源用量：線性部署會同時暫時執行藍色和綠色服務修訂，這可能會在部署期間將您的資源用量加倍。
- 部署監控：線性部署提供詳細的部署狀態資訊，可讓您監控部署程序的每個階段和每個流量轉移增量。
- 回復：線性部署可讓您在偵測到問題時更輕鬆地回復到先前的版本，因為藍色修訂會持續執行，直到製作時間過期為止。
- 漸進式驗證：線性部署可讓您使用增加的生產流量來驗證新修訂版，從而提高部署的可信度。
- 部署持續時間：由於步驟之間的增量流量轉移和等待時間，線性部署的完成時間比all-at-once部署更長。

線性部署的運作方式

Amazon ECS 線性部署程序遵循具有六個不同階段的結構化方法，以確保安全可靠的應用程式更新。每個階段都有特定用途，以驗證應用程式，並將其從目前版本 (藍色) 轉換為新版本 (綠色)。

1. 準備階段 - 在現有的藍色環境旁建立綠色環境。
2. 部署階段：將新的服務修訂版部署到綠色環境。Amazon ECS 會使用更新後的服務修訂版啟動新任務，同時藍色環境會繼續處理生產流量。
3. 測試階段：使用測試流量路由驗證綠色環境。Application Load Balancer 會將測試請求導向綠色環境，同時生產流量仍保留在藍色環境中。

4. 線性流量轉移階段：根據您設定的部署策略，以相等的百分比增量逐漸將生產流量從藍色轉移到綠色。
5. 監控階段：在封裝時間段內監控應用程式運作狀態、效能指標與警示狀態。偵測到問題時，會啟動復原操作。
6. 完成階段：終止藍色環境以完成部署。

線性流量轉移階段遵循下列步驟：

- 初始 - 部署從 100% 路由到藍色（目前）服務修訂版的流量開始。綠色（新）服務修訂最初會接收測試流量，但不會接收生產流量。
- 增量流量轉移 - 流量會以相等百分比增量逐漸從藍色轉換為綠色。例如，使用 10.0% 步驟組態時，流量轉移會如下所示：
 - 步驟 1：10.0% 到綠色，90.0% 到藍色
 - 步驟 2：20.0% 到綠色，80.0% 到藍色
 - 步驟 3：30.0% 到綠色，70.0% 到藍色
 - 因此，直到 100% 達到綠色
- 步驟製作時間 - 在每個流量轉移增量之間，部署會等待可設定的持續時間（步驟製作時間），以允許在流量負載增加的情況下監控和驗證新修訂版的效能。請注意，當流量轉移 100.0% 時，會略過最後一個步驟的製作時間。
- 生命週期關聯 - 選用 Lambda 函數可以在部署期間的不同生命週期階段執行，以執行自動驗證、監控或自訂邏輯。為每個生產流量轉移步驟調用為 `PRODUCTION_TRAFFIC_SHIFT` 設定的 Lambda 函數或生命週期掛鉤。

部署生命週期階段

線性部署程序會經過不同的生命週期階段，每個階段都有特定的責任和驗證檢查點。了解這些階段有助於監控部署進度並對問題進行有效的疑難排解。

每個生命週期階段最多可持續 24 小時，此外，`PRODUCTION_TRAFFIC_SHIFT` 中的每個流量轉移步驟最多可持續 24 小時。建議將該值保持在 24 小時標記以內。這是因為非同步程序需要時間來觸發勾點。系統逾時、部署失敗，然後在階段達到 24 小時後啟動復原。

CloudFormation 部署有額外的逾時限制。雖然 24 小時階段限制仍然有效，但對整個部署 CloudFormation 強制執行 36 小時限制。部署 CloudFormation 失敗，然後在程序未在 36 小時內完成時啟動轉返。

生命週期階段	Description	生命週期掛鉤支援
RECONCILE_SERVICE	僅當啟動新的服務部署且存在多個處於 ACTIVE 狀態的服務修訂版時，此階段才會發生。	是
PRE_SCALE_UP	綠色服務修訂版尚未啟動。藍色服務修訂版正在處理 100% 的生產流量。沒有測試流量。	是
SCALE_UP	綠色服務修訂版向上擴展到 100% 並啟動新任務的時間。此時綠色服務修訂版不處理任何流量。	否
POST_SCALE_UP	綠色服務修訂版已啟動。藍色服務修訂版正在處理 100% 的生產流量。沒有測試流量。	是
TEST_TRAFFIC_SHIFT	藍色與綠色服務修訂版均在執行中。藍色服務修訂版處理 100% 的生產流量。綠色服務修訂版正在從 0% 遷移至 100% 的測試流量。	是
POST_TEST_TRAFFIC_SHIFT	測試流量轉移已完成。綠色服務修訂版處理 100% 的生產流量。	是
PRODUCTION_TRAFFIC_SHIFT	流量會從藍色逐漸轉換為綠色，並以相等百分比遞增，直到綠色接收到 100% 的流量。每個流量轉移都會調用具有 24 小時逾時的生命週期關聯。	
POST_PRODUCTION_TRAFFIC_SHIFT	生產流量轉移已完成。	是
BAKE_TIME	藍色與綠色服務修訂版同時執行的持續時間。	否
CLEAN_UP	藍色服務修訂已完全縮減至 0 項執行中任務。在此階段之後，綠色服務修訂版現在是生產服務修訂版。	否

Amazon ECS 線性部署所需的資源

若要搭配受管流量轉移使用線性部署，您的服務必須使用下列其中一項功能：

- Elastic Load Balancing
- Service Connect

以下清單提供您需要為 Amazon ECS 線性部署設定哪些項目的高階概觀：

- 服務會使用 Application Load Balancer、Network Load Balancer 或 Service Connect。設定相應的資源。
 - Application Load Balancer – 如需詳細資訊，請參閱[藍/綠、線性和金絲雀部署的 Application Load Balancer 資源](#)。
 - Network Load Balancer – 如需詳細資訊，請參閱[Amazon ECS 藍/綠、線性和 Canary 部署的 Network Load Balancer 資源](#)。
 - Service Connect – 如需詳細資訊，請參閱[Amazon ECS 藍色/綠色、線性和 Canary 部署的 Service Connect 資源](#)。
- 將服務部署控制器設定為 ECS。
- 在服務定義中，將部署策略設定為 linear。
- 或者設定其他參數，例如：
 - 新部署的封裝時間
 - 每個增量中要轉移的流量百分比。
 - 每個流量轉移增量之間等待的持續時間，以分鐘為單位。
 - 自動復原的 CloudWatch 警示
 - 部署生命週期掛鉤（這些是在指定的部署階段執行的 Lambda 函數，例如 BEFORE_INSTALL、Production_TRAFFIC_SHIFT 或 POST_PRODUCTION_TRAFFIC_SHIFT)

最佳實務

針對成功的 Amazon ECS 線性部署，請遵循下列最佳實務：

- 確保您的應用程式可以同時處理兩個服務修訂。
- 規劃足夠的叢集容量，以在部署期間處理這兩個服務修訂。
- 在生產環境中實作復原程序之前，請先測試復原程序。
- 設定相應的運作狀態檢查，以準確反映應用程式的運作狀態。
- 設定可充分測試新服務修訂的製作時間。
- 實作 CloudWatch 警示，以自動偵測問題並觸發復原。

- 選擇步驟百分比和製作時間，以平衡部署速度與驗證需求。
- 針對關鍵應用程式使用較小的步驟百分比 (5-10%)，將風險暴露降至最低。
- 為需要時間暖機或穩定的應用程式設定較長的步驟製作時間。
- 實作 CloudWatch 警示，以自動偵測問題，並以任何流量增量觸發轉返。
- 在每次流量轉移期間密切監控應用程式指標，以提早偵測效能降低。
- 確保您的應用程式可以同時處理兩個服務修訂。
- 在生產環境中實作復原程序之前，以不同的流量百分比測試復原程序。

建立 Amazon ECS 線性部署

透過使用 Amazon ECS 線性部署，您可以在指定的時間間隔內以相同的增量逐步轉移流量。這會在部署程序的每個步驟提供受控驗證。

先決條件

開始線性部署之前，請執行下列操作。

1. 設定相應的許可。
 - 如需有關 Elastic Load Balancing 許可的資訊，請參閱[負載平衡器的 Amazon ECS 基礎結構 IAM 角色](#)。
 - 如需有關 Lambda 許可的資訊，請參閱[Amazon ECS 藍/綠部署中 Lambda 函式所需的許可](#)。
2. Amazon ECS 線性部署需要您的服務使用下列其中一項功能：設定適當的資源。
 - Application Load Balancer – 如需詳細資訊，請參閱[藍/綠、線性和金絲雀部署的 Application Load Balancer 資源](#)。
 - Network Load Balancer – 如需詳細資訊，請參閱[Amazon ECS 藍/綠、線性和 Canary 部署的 Network Load Balancer 資源](#)。
 - Service Connect – 如需詳細資訊，請參閱[Amazon ECS 藍色/綠色、線性和 Canary 部署的 Service Connect 資源](#)。

程序

您可以使用 主控台或 AWS CLI 來建立 Amazon ECS 線性部署服務。

Console

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	- 在叢集頁面，選取您要在哪個叢集中建立服務。 「叢集詳細資訊」頁面隨即顯示。 - 在 Services (服務) 標籤上，選擇 Create (建立)。	
任務定義	- 在任務定義頁面中選取任務定義。 - 從部署功能表中選擇建立服務。	

建立服務頁面隨即顯示。

- 在服務詳細資訊下，執行下列動作：
 - 在任務定義系列欄位中選擇要使用的任務定義。然後，在任務定義修訂版欄位中輸入要使用的修訂版。
 - 針對 Service name (服務名稱)，輸入服務的名稱。
- 若要在現有叢集中執行服務，請在現有叢集欄位中選擇叢集。若要在新叢集中執行服務，請選擇建立叢集
- 選擇任務在叢集基礎結構中的分佈方式。在運算組態下選擇選項。

運算選項	步驟	
容量提供者策略	- 在 Compute options (運算選項) 下，選擇 Capacity provider strategy (容量提供者策略)。 - 選擇策略：	

運算選項	步驟	
	- 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	- 在運算選項區段中，選取啟動類型。 - 針對 啟動類型，選擇啟動類型。 (選用) 當指定 Fargate 時，請在平台版本欄位中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。	

6. 在部署組態下，執行下列動作：

- a. 在服務類型欄位中選擇複本。
 - b. 針對 Desired tasks (所需任務)，輸入要在服務中啟動並維護的任務數。
 - c. 若要讓 Amazon ECS 監控可用區域間的任務分佈，並在出現不平衡時重新分佈，請在可用區域服務重新平衡下，選取可用區域服務重新平衡。
 - d. 在運作狀態檢查寬限期欄位中，輸入時間 (以秒為單位)，這是服務排程器在任務首次啟動後忽略運作狀態不良的 Elastic Load Balancing、VPC Lattice 與容器運作狀態檢查的時間。如果您未指定運作狀態檢查寬限期間值，則會使用預設值 0。
7. 在部署組態下，設定線性部署設定：
- a. 針對部署策略，選擇線性。
 - b. 對於流量增量百分比，輸入每個增量要轉移的流量百分比（例如，10% 以 10% 增量轉移流量）。
 - c. 對於增量之間的時間，輸入每個流量轉移增量之間等待的時間，以分鐘為單位。
 - d. 針對製作時間，輸入藍色和綠色服務修訂在終止藍色修訂之前，在最終流量轉移後同時執行的分鐘數。
 - e. (選用) 執行要在部署的特定階段執行的 Lambda 函式。在部署 lifecycle hook 下，選取要執行 lifecycle hook 的階段。

若要新增 lifecycle hook：

- a. 選擇新增。
 - b. 在 Lambda 函式欄位中，輸入函式名稱或 ARN。
 - c. 在角色欄位中，選取具有調用 Lambda 函式許可的 IAM 角色。
 - d. 在生命週期階段欄位中，選取 Lambda 函式應執行的階段。
8. 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)。

在部署斷路器將部署設定為失敗狀態時，若要讓軟體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. 若要根據應用程式指標停止部署，請選取使用 CloudWatch 警示。然後，從 CloudWatch 警示名稱欄位中選擇警示。要建立新的警示，請前往 CloudWatch 主控台。

在 CloudWatch 警示將部署設定為失敗狀態時，若要讓軟體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。

9. (選用) 若要使用 Service Connect 與服務互連，請展開 Service Connect 區段，然後指定下列項目：
 - a. 選取開啟 Service Connect。
 - b. 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至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)。
 - c. 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。這可以是在您的 中在相同 AWS 區域 中分別建立的命名空間，AWS 帳戶 或是使用 AWS Resource Access Manager () 與您的帳戶共用的相同區域中的命名空間AWS RAM。如需共用 AWS Cloud Map 命名空間的詳細資訊，請參閱《AWS Cloud Map 開發人員指南》中的[跨帳戶 AWS Cloud Map 命名空間共用](#)。
 - d. (選用) 設定線性部署的測試流量標頭規則。在測試流量路由下，指定下列項目：
 - i. 選取啟用測試流量標頭規則，以在測試期間將特定請求路由至綠色服務修訂版。
 - ii. 在標頭比對規則欄位中，設定路由測試流量的條件：
 - 標頭名稱：輸入要比對的 HTTP 標頭名稱 (例如 X-Test-Version 或 User-Agent)。
 - 比對類型：選擇比對條件：
 - 完全相符：標頭值與指定值完全相符的路由請求
 - 標頭存在：包含指定標頭的路由請求，無論其值為何
 - 模式比對：標頭值與指定模式相符的路由請求
 - 標頭值 (如果使用「完全相符」或「模式相符」)：輸入要比對的值或模式。
 - iii. 選擇新增標頭規則以設定其他標頭比對條件。

您可以新增多個標頭比對規則來建立複雜的路由邏輯。符合任何已設定規則的請求將路由至綠色服務修訂版進行測試。

Note

測試流量標頭規則讓您可以在完成完整部署之前，使用受控流量驗證新功能。這允許您使用特定請求 (例如來自內部測試工具或 Beta 版使用者的請求) 測試綠色服務修訂版，同時維持流向藍色服務修訂版的正常流量流程。

- e. (選用) 指定日誌組態。選取使用日誌收集。預設選項會將容器日誌傳送至 CloudWatch 日誌。其他日誌驅動程式選項是使用 AWS FireLens 設定。如需詳細資訊，請參閱[將 Amazon ECS 日誌傳送至 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 – 將任務設定為將容器日誌傳送至 CloudWatch Logs。系統會提供預設日誌驅動程式選項，用來代表您建立 CloudWatch 日誌群組。若要指定不同的日誌群組名稱，請變更驅動程式選項值。
- Amazon Data Firehose – 將任務設定為將容器日誌傳送至 Firehose。系統會提供預設日誌驅動程式選項，用來將日誌傳送至 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 – 將任務設定為將容器日誌傳送至 Amazon Kinesis Data Streams。系統會提供預設日誌驅動程式選項，用來將日誌傳送至 Kinesis Data Streams 串流。若要指定不同的串流名稱，請變更驅動程式選項值。
- Amazon OpenSearch Service – 將任務設定為將容器日誌傳送至 OpenSearch Service 網域。務必提供日誌驅動程式選項。
- Amazon S3：將任務設定為將容器日誌傳送至 Amazon S3 儲存貯體。系統會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。

10. (選用) 設定線性部署的負載平衡。

Elastic Load Balancing 類型	步驟
Application Load Balancer	- 在負載平衡器類型欄位中選擇 Application Load Balancer。 - 選擇 Create a new load balancer (建立新的負載平衡器) 以建立新

Elastic Load Balancing 類型	步驟	
	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. 在容器欄位中選擇託管服務的容器。 d. 針對 Load balancer name (負載平衡器名稱)，輸入不重複的名稱。 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。 - 在生產規則欄位中，輸入規則的評估順序與路徑模式。 此規則適用於您的生產服務修訂流量。 - 在測試規則欄位中，輸入規則的評估順序與路徑模式。 此規則適用於您的測試服務修訂流量。 f. 在目標群組欄位中設定下列項目： - 針對 Target group name (目標群組名	

Elastic Load Balancing 類型	步驟	
	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。 • 在通訊協定欄位中, 選擇 Application Load Balancer 向其路由請求之目標群組的通訊協定。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。 • 針對取消註冊延遲, 輸入負載平衡器將目標狀態變更為 的秒數UNUSED。預設為 300 秒。 • 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 (/)。 • 針對替代群組名稱, 輸入測試服務修訂版目標群組的群組名稱。	

Elastic Load Balancing 類型	步驟	
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. 在生產接聽程式欄位中 , 選擇生產接聽程式連接埠與生產接聽程式通訊協定。 這是生產服務修訂流量的接聽程式。 e. 在測試接聽程式欄位中 , 選擇測試接聽程式連接埠與測試接聽程式通訊協定。 這是測試服務修訂流量的接聽程式。 f. 在目標群組欄位中設定下列項目 : 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。	

Elastic Load Balancing 類型	步驟	
	• 在通訊協定欄位中，選擇 Network Load Balancer 向其路由請求之目標群組的通訊協定。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。 • 針對取消註冊延遲，輸入負載平衡器將目標狀態變更為 的秒數UNUSED。預設為 300 秒。 • 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 (/)。 • 在替代群組名稱欄位中，輸入測試 (綠色) 服務修訂版的目標群組名稱。	

11. (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。

12. 選擇建立。

AWS CLI

1. 建立名為 linear-service-definition.json 且具有下列內容的檔案。

將 *user-input* 取代為實際值。

```
{
  "serviceName": "myLinearService",
  "cluster": "arn:aws:ecs:us-west-2:123456789012:cluster/sample-fargate-cluster",
  "taskDefinition": "sample-fargate:1",
  "desiredCount": 5,
  "launchType": "FARGATE",
  "networkConfiguration": {
    "awsvpcConfiguration": {
      "subnets": [
        "subnet-09ce6e74c116a2299",
        "subnet-00bb3bd7a73526788",
        "subnet-0048a611aaec65477"
      ],
      "securityGroups": [
        "sg-09d45005497daa123"
      ],
      "assignPublicIp": "ENABLED"
    }
  },
  "deploymentController": {
    "type": "ECS"
  }
}
```



```

    },
    "deploymentConfiguration": {
      "strategy": "LINEAR",
      "maximumPercent": 200,
      "minimumHealthyPercent": 100,
      "linearConfiguration": {
        "stepPercentage": 10.0,
        "stepBakeTimeInMinutes": 6
      },
      "bakeTimeInMinutes": 10,
      "alarms": {
        "alarmNames": [
          "myAlarm"
        ],
        "rollback": true,
        "enable": true
      }
    },
    "loadBalancers": [
      {
        "targetGroupArn": "arn:aws:elasticloadbalancing:us-
west-2:123456789012:targetgroup/blue-target-group/54402ff563af1197",
        "containerName": "fargate-app",
        "containerPort": 80,
        "advancedConfiguration": {
          "alternateTargetGroupArn": "arn:aws:elasticloadbalancing:us-
west-2:123456789012:targetgroup/green-target-group/cad10a56f5843199",
          "productionListenerRule": "arn:aws:elasticloadbalancing:us-
west-2:123456789012:listener-rule/app/my-linear-
demo/32e0e4f946c3c05b/9cfa8c482e204f7d/831dbaf72edb911",
          "roleArn": "arn:aws:iam::123456789012:role/LoadBalancerManagementforECS"
        }
      }
    ]
  }
}

```

2. 執行 create-service。

```
aws ecs create-service --cli-input-json file://linear-service-definition.json
```

後續步驟

- 更新服務以開始部署。如需詳細資訊，請參閱[更新 Amazon ECS 服務](#)。
- 監控部署程序，以確保其遵循線性模式：
 - 綠色服務修訂版已建立並向上擴展
 - 流量會以指定的間隔遞增轉移
 - 每個增量在下一次輪班之前等待指定的間隔
 - 轉移所有流量後，製作時間開始
 - 封裝時間過期後，藍色修訂版會終止

Amazon ECS Canary 部署

Canary 部署會先將一小部分的流量路由到新修訂版以進行初始測試，然後在 Canary 階段成功完成後一次轉移所有剩餘的流量。透過 Amazon ECS Canary 部署，使用實際使用者流量驗證新的服務修訂，同時將暴露風險降至最低。此方法提供一種可控制的方式來部署變更，並能夠監控效能，並在偵測到問題時快速復原。

Canary 部署中涉及的資源

以下是 Amazon ECS Canary 部署中涉及的資源：

- 流量轉移 - Amazon ECS 用來轉移生產流量的程序。對於 Amazon ECS Canary 部署，流量會分兩個階段轉移：先到 Canary 百分比，然後完成部署。
- Canary 百分比 - 在評估期間路由至新版本的流量百分比。
- Canary 製作時間 - 在繼續完整部署之前監控 Canary 版本的持續時間。
- 部署製作時間 - 在將所有生產流量轉移到新服務修訂版之後，Amazon ECS 會在終止舊服務修訂版之前等待的時間，以分鐘為單位。這是生產流量轉移後，藍色和綠色服務修訂同時執行的持續時間。
- 生命週期階段 - 部署操作中的一系列事件，例如「生產流量轉移後」。
- 生命週期掛鉤 - 在特定生命週期階段執行的 Lambda 函數。您可以建立驗證部署的函數。
- 目標群組 - 一種 Elastic Load Balancing 資源，用於將請求路由至一個或多個已註冊的目標 (例如 EC2 執行個體)。當您建立接聽程式時，可以為其預設動作指定一個目標群組。流量會轉送至接聽程式規則中指定的目標群組。
- 接聽程式 - 一種 Elastic Load Balancing 資源，用於透過已設定的通訊協定與連接埠檢查連線請求。您為接聽程式定義的規則決定了 Amazon ECS 如何將請求路由至已註冊的目標。

- 規則 – 與接聽程式相關聯的 Elastic Load Balancing 資源。規則定義了請求的路由方式，包含動作、條件與優先順序。

考量事項

選擇部署類型時，請考量下列事項：

- 資源用量：Canary 部署會在評估期間同時執行原始任務集和 Canary 任務集，進而增加資源用量。
- 流量：確保 Canary 百分比產生足夠的流量，以便對新版本進行有意義的驗證。
- 監控複雜性：Canary 部署需要同時監控和比較兩個不同版本之間的指標。
- 回復速度：Canary 部署可透過將流量移回原始任務集來啟用快速回復。
- 風險緩解：Canary 部署透過將暴露限制在少數百分比的使用者，提供絕佳的風險緩解能力。
- 部署持續時間：Canary 部署包含延長整體部署時間但提供驗證機會的評估期間。

Canary 部署的運作方式

Amazon ECS Canary 部署程序遵循具有六個不同階段的結構化方法，以確保安全可靠的應用程式更新。每個階段都有特定用途，以驗證應用程式，並將其從目前版本 (藍色) 轉換為新版本 (綠色)。

1. 準備階段 – 在現有的藍色環境旁建立綠色環境。
2. 部署階段：將新的服務修訂版部署到綠色環境。Amazon ECS 會使用更新後的服務修訂版啟動新任務，同時藍色環境會繼續處理生產流量。
3. 測試階段：使用測試流量路由驗證綠色環境。Application Load Balancer 會將測試請求導向綠色環境，同時生產流量仍保留在藍色環境中。
4. Canary 流量轉移階段：在 Canary 階段將流量設定百分比轉移至新的綠色服務修訂版本，接著將 100.0% 的流量轉移至 Green 服務修訂版本
5. 監控階段：在封裝時間段內監控應用程式運作狀態、效能指標與警示狀態。偵測到問題時，會啟動復原操作。
6. 完成階段：終止藍色環境以完成部署。

Canary 流量轉移階段遵循下列步驟：

- 初始 - 部署從 100% 路由至藍色 (目前) 服務修訂版的流量開始。綠色 (新) 服務修訂最初會接收測試流量，但不會接收生產流量。
- Canary 流量轉移 - 這是兩步驟流量轉移策略。

- 步驟 1：10.0% 到綠色，90.0% 到藍色
- 步驟 2：100.0% 到綠色，0.0% 到藍色
- Canary 製作時間 - 在 Canary 流量轉移後等待可設定的持續時間 (Canary 製作時間)，以允許在流量負載增加時監控和驗證新修訂版的效能。
- 生命週期關聯 - 選用 Lambda 函數可以在部署期間的不同生命週期階段執行，以執行自動驗證、監控或自訂邏輯。為每個生產流量轉移步驟調用為 `PRODUCTION_TRAFFIC_SHIFT` 設定的 Lambda 函數或生命週期掛鉤。

部署生命週期階段

Canary 部署程序會經過不同的生命週期階段，每個階段都有特定的責任和驗證檢查點。了解這些階段有助於監控部署進度並對問題進行有效的疑難排解。

每個生命週期階段最多可持續 24 小時，此外，`PRODUCTION_TRAFFIC_SHIFT` 中的每個流量轉移步驟最多可持續 24 小時。建議將該值保持在 24 小時標記以內。這是因為非同步程序需要時間來觸發勾點。系統逾時、部署失敗，然後在階段達到 24 小時後啟動轉返。

CloudFormation 部署有額外的逾時限制。雖然 24 小時階段限制仍然有效，但會 CloudFormation 強制執行整個部署的 36 小時限制。會 CloudFormation 失敗部署，然後在程序未在 36 小時內完成時啟動轉返。

生命週期階段

生命週期階段	Description	生命週期掛鉤支援
RECONCILE_SERVICE	僅當啟動新的服務部署且存在多個處於 ACTIVE 狀態的服務修訂版時，此階段才會發生。	是
PRE_SCALE_UP	綠色服務修訂版尚未啟動。藍色服務修訂版正在處理 100% 的生產流量。沒有測試流量。	是
SCALE_UP	綠色服務修訂版向上擴展到 100% 並啟動新任務的時間。此時綠色服務修訂版不處理任何流量。	否
POST_SCALE_UP	綠色服務修訂版已啟動。藍色服務修訂版正在處理 100% 的生產流量。沒有測試流量。	是

生命週期階段	Description	生命週期掛鉤支援
TEST_TRAFFIC_SHIFT	藍色與綠色服務修訂版均在執行中。藍色服務修訂版處理 100% 的生產流量。綠色服務修訂版正在從 0% 遷移至 100% 的測試流量。	是
POST_TEST_TRAFFIC_SHIFT	測試流量轉移已完成。綠色服務修訂版處理 100% 的生產流量。	是
PRODUCTION_TRAFFIC_SHIFT	Canary 生產流量會路由至綠色修訂版，並使用 24 小時逾時叫用生命週期掛鉤。第二個步驟會將剩餘的生產流量轉移到綠色修訂版。	是
POST_PRODUCTION_TRAFFIC_SHIFT	生產流量轉移已完成。	是
BAKE_TIME	藍色與綠色服務修訂版同時執行的持續時間。	否
CLEAN_UP	藍色服務修訂已完全縮減至 0 項執行中任務。在此階段之後，綠色服務修訂版現在是生產服務修訂版。	否

組態參數

Canary 部署需要下列組態參數：

- Canary 百分比 - 在 Canary 階段期間路由至新服務修訂的流量百分比。這允許使用受控的生產流量子集進行測試。
- Canary 製作時間 - 在 Canary 階段期間等待的持續時間，然後再將剩餘的流量轉移到新的服務修訂版。這提供了監控和驗證新版本的時間。

流量管理

Canary 部署使用負載平衡器目標群組來管理流量分佈：

- 原始目標群組 - 包含目前穩定版本的任務，並接收大部分流量。
- Canary 目標群組 - 包含新版本的任務，並接收一小部分的流量進行測試。

- 加權路由 - 負載平衡器使用加權路由規則，根據設定的 Canary 百分比在目標群組之間分配流量。

監控和驗證

有效的 Canary 部署依賴於全面監控：

- 運作狀態檢查 - 在接收流量之前，兩個任務集都必須通過運作狀態檢查。
- 指標比較 - 比較原始版本和 Canary 版本之間的關鍵效能指標，例如回應時間、錯誤率和輸送量。
- 自動化轉返 - 如果 Canary 版本顯示效能降低，將 CloudWatch 警示設定為自動觸發轉返。
- 手動驗證 - 在繼續之前，使用評估期間手動檢閱日誌、指標和使用者意見回饋。

Canary 部署的最佳實務

遵循這些最佳實務，以確保使用 服務成功部署 Canary。

選擇適當的流量百分比

選取 Canary 流量百分比時，請考慮下列因素：

- 啟動小型 - 從 5-10% 的流量開始，在發生問題時將影響降至最低。
- 考慮應用程式關鍵性 - 對關鍵任務應用程式使用較小的百分比，對較不重要的服務使用較大的百分比。
- 流量帳戶 - 確保 Canary 百分比產生足夠的流量以進行有意義的驗證。

設定適當的評估期間

根據下列考量設定評估期間：

- 允許足夠的時間 - 設定足夠長的評估期間以擷取有意義的效能資料，通常為 10-30 分鐘。
- 考慮流量模式 - 考慮應用程式的流量模式和尖峰使用時間。
- 平衡速度和安全性 - 較長的評估期間提供更多資料，但部署速度緩慢。

實作全方位監控

設定監控以追蹤 Canary 部署效能：

- 關鍵指標 - 監控兩個任務集的回應時間、錯誤率、輸送量和資源使用率。
- 警示型回復 - 設定 CloudWatch 警示，以在指標超過閾值時自動觸發回復。

- 比較分析 - 設定儀表板來side-by-side比較原始版本和 Canary 版本之間的指標。
- 業務指標 - 包括業務特定的指標，例如轉換率或使用參與度以及技術指標。

規劃復原策略

使用以下策略準備潛在的回復案例：

- 自動化轉返 - 根據運作狀態檢查和效能指標設定自動轉返觸發條件。
- 手動轉返程序 - 當自動觸發未擷取所有問題時，記錄手動轉返的清除程序。
- 回復測試 - 定期測試回復程序，以確保在需要時正常運作。

在部署之前徹底驗證

在繼續 Canary 部署之前，請確保徹底驗證：

- 部署前測試 - 在 Canary 部署之前，徹底測試預備環境中的變更。
- 運作狀態檢查組態 - 確保運作狀態檢查準確反映應用程式準備程度和功能。
- 相依性驗證 - 確認新版本與下游和上游服務相容。
- 資料一致性 - 確保資料庫結構描述變更和資料遷移回溯相容。

協調團隊參與

在 Canary 部署期間確保有效的團隊協調：

- 部署時段 - 當團隊可以監控和回應時，在上班時間排程 Canary 部署。
- 通訊管道 - 為部署狀態和問題升級建立明確的通訊管道。
- 角色指派 - 定義監控、決策和復原執行的角色和責任。

Amazon ECS Canary 部署所需的資源

若要搭配受管流量轉移使用 Canary 部署，您的服務必須使用下列其中一項功能：

- Elastic Load Balancing
- Service Connect

以下清單提供您需要為 Amazon ECS Canary 部署設定哪些項目的高階概觀：

- 服務會使用 Application Load Balancer、Network Load Balancer 或 Service Connect。設定相應的資源。
- Application Load Balancer – 如需詳細資訊，請參閱[藍/綠、線性和金絲雀部署的 Application Load Balancer 資源](#)。
- Network Load Balancer – 如需詳細資訊，請參閱[Amazon ECS 藍/綠、線性和 Canary 部署的 Network Load Balancer 資源](#)。
- Service Connect – 如需詳細資訊，請參閱[Amazon ECS 藍色/綠色、線性和 Canary 部署的 Service Connect 資源](#)。
- 將服務部署控制器設定為 ECS。
- 在服務定義中，將部署策略設定為 canary。
- 或者設定其他參數，例如：
 - 新部署的封裝時間
 - 在 Canary 階段期間路由至新服務修訂的流量百分比。
 - 在 Canary 階段等待的持續時間，然後再將剩餘的流量轉移到新的服務修訂版。
 - 自動復原的 CloudWatch 警示
 - 部署生命週期掛鉤（這些是在指定部署階段執行的 Lambda 函數）

最佳實務

針對成功的 Amazon ECS 程式庫部署，請遵循下列最佳實務：

- 確保您的應用程式可以同時處理兩個服務修訂。
- 規劃足夠的叢集容量，以在部署期間處理這兩個服務修訂。
- 在生產環境中實作復原程序之前，請先測試復原程序。
- 設定相應的運作狀態檢查，以準確反映應用程式的運作狀態。
- 設定封裝時間，以對綠色部署進行充分測試。
- 實作 CloudWatch 警示，以自動偵測問題並觸發復原。
- 使用 lifecycle hook，在每個部署階段執行自動化測試。
- 從小 Canary 百分比 (5-10%) 開始，在發生問題時將影響降至最低。
- 設定適當的評估期間，以允許足夠的時間進行有意義的效能資料收集。
- 使用 CloudWatch 警示實作自動化轉返觸發的全面監控。
- 設定可準確反映應用程式整備和功能的運作狀態檢查。
- 在評估期間監控技術指標（回應時間、錯誤率）和業務指標。

- 確保您的應用程式可以處理流量分割，而不會發生工作階段或狀態問題。
- 規劃轉返程序並定期測試，以確保它們在需要時有效。
- 在團隊可以監控和回應的上班時間排程 Canary 部署。
- 在 Canary 部署之前，在預備環境中徹底驗證變更。
- 記錄手動介入和回復決策的明確程序。

建立 Amazon ECS Canary 部署

透過使用 Amazon ECS Canary 部署，您可以將一小部分的流量轉移到新的服務修訂版（「Canary」）。驗證部署，然後在指定的間隔之後一次轉移剩餘的流量。此方法可讓您在完全部署之前，以最低風險測試新功能。

先決條件

在啟動 Canary 部署之前，請執行下列操作。

1. 設定相應的許可。
 - 如需有關 Elastic Load Balancing 許可的資訊，請參閱[負載平衡器的 Amazon ECS 基礎結構 IAM 角色](#)。
 - 如需有關 Lambda 許可的資訊，請參閱[Amazon ECS 藍/綠部署中 Lambda 函式所需的許可](#)。
2. Amazon ECS Canary 部署要求您的服務使用下列其中一項功能：設定適當的資源。
 - Application Load Balancer – 如需詳細資訊，請參閱[藍/綠、線性和金絲雀部署的 Application Load Balancer 資源](#)。
 - Network Load Balancer – 如需詳細資訊，請參閱[Amazon ECS 藍/綠、線性和 Canary 部署的 Network Load Balancer 資源](#)。
 - Service Connect – 如需詳細資訊，請參閱[Amazon ECS 藍色/綠色、線性和 Canary 部署的 Service Connect 資源](#)。

程序

您可以使用 主控台或 AWS CLI 來建立 Amazon ECS Canary 部署服務。

Console

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	- 在叢集頁面，選取您要在哪個叢集中建立服務。 「叢集詳細資訊」頁面隨即顯示。 - 在 Services (服務) 標籤上，選擇 Create (建立)。	
任務定義	- 在任務定義頁面中選取任務定義。 - 從部署功能表中選擇建立服務。	

建立服務頁面隨即顯示。

- 在服務詳細資訊下，執行下列動作：
 - 在任務定義系列欄位中選擇要使用的任務定義。然後，在任務定義修訂版欄位中輸入要使用的修訂版。
 - 針對 Service name (服務名稱)，輸入服務的名稱。
- 若要在現有叢集中執行服務，請在現有叢集欄位中選擇叢集。若要在新叢集中執行服務，請選擇建立叢集
- 選擇任務在叢集基礎結構中的分佈方式。在運算組態下選擇選項。

運算選項	步驟	
容量提供者策略	- 在 Compute options (運算選項) 下，選擇 Capacity provider strategy (容量提供者策略)。 - 選擇策略：	

運算選項	步驟	
	- 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	- 在運算選項區段中，選取啟動類型。 - 針對 啟動類型，選擇啟動類型。 (選用) 當指定 Fargate 時，請在平台版本欄位中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。	

6. 在部署組態下，執行下列動作：

- a. 在服務類型欄位中選擇複本。
- b. 針對 Desired tasks (所需任務)，輸入要在服務中啟動並維護的任務數。
- c. 若要讓 Amazon ECS 監控可用區域間的任務分佈，並在出現不平衡時重新分佈，請在可用區域服務重新平衡下，選取可用區域服務重新平衡。
- d. 在運作狀態檢查寬限期欄位中，輸入時間 (以秒為單位)，這是服務排程器在任務首次啟動後忽略運作狀態不良的 Elastic Load Balancing、VPC Lattice 與容器運作狀態檢查的時間。如果您未指定運作狀態檢查寬限期間值，則會使用預設值 0。

7. 在部署組態下，設定 Canary 部署設定：

- a. 針對部署策略，選擇 Canary。
- b. 針對 Canary 百分比，輸入要在第一階段轉移到綠色服務修訂的流量百分比（例如，初始 Canary 流量為 10%）。
- c. 針對 Canary 製作時間，輸入在將剩餘的流量轉移到綠色服務修訂版之前等待的時間，以分鐘為單位。
- d. 針對製作時間，輸入藍色和綠色服務修訂在終止藍色修訂之前，在最終流量轉移後同時執行的分鐘數。
- e. (選用) 執行要在部署的特定階段執行的 Lambda 函式。在部署 lifecycle hook 下，選取要執行 lifecycle hook 的階段。

若要新增 lifecycle hook：

- a. 選擇新增。
- b. 在 Lambda 函式欄位中，輸入函式名稱或 ARN。
- c. 在角色欄位中，選取具有調用 Lambda 函式許可的 IAM 角色。
- d. 在生命週期階段欄位中，選取 Lambda 函式應執行的階段。

8. 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)。

在部署斷路器將部署設定為失敗狀態時，若要讓軟體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. 若要根據應用程式指標停止部署，請選取使用 CloudWatch 警示。然後，從 CloudWatch 警示名稱欄位中選擇警示。要建立新的警示，請前往 CloudWatch 主控台。

在 CloudWatch 警示將部署設定為失敗狀態時，若要讓軟體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。

9. (選用) 若要使用 Service Connect 與服務互連，請展開 Service Connect 區段，然後指定下列項目：
 - a. 選取開啟 Service Connect。
 - b. 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至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)。
 - c. 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。這可以是 AWS 區域 在您的相同 中分別建立的命名空間，AWS 帳戶 或是使用 AWS Resource Access Manager () 與您的帳戶共用的相同區域中的命名空間AWS RAM。如需共用 AWS Cloud Map 命名空間的詳細資訊，請參閱《AWS Cloud Map 開發人員指南》中的[跨帳戶 AWS Cloud Map 命名空間共用](#)。
 - d. (選用) 設定 Canary 部署的測試流量標頭規則。在測試流量路由下，指定下列項目：
 - i. 選取啟用測試流量標頭規則，以在測試期間將特定請求路由至綠色服務修訂版。
 - ii. 在標頭比對規則欄位中，設定路由測試流量的條件：
 - 標頭名稱：輸入要比對的 HTTP 標頭名稱 (例如 X-Test-Version 或 User-Agent)。
 - 比對類型：選擇比對條件：
 - 完全相符：標頭值與指定值完全相符的路由請求
 - 標頭存在：包含指定標頭的路由請求，無論其值為何
 - 模式比對：標頭值與指定模式相符的路由請求
 - 標頭值 (如果使用「完全相符」或「模式相符」)：輸入要比對的值或模式。
 - iii. 選擇新增標頭規則以設定其他標頭比對條件。

您可以新增多個標頭比對規則來建立複雜的路由邏輯。符合任何已設定規則的請求將路由至綠色服務修訂版進行測試。

Note

測試流量標頭規則讓您可以在完成完整部署之前，使用受控流量驗證新功能。這允許您使用特定請求 (例如來自內部測試工具或 Beta 版使用者的請求) 測試綠色服務修訂版，同時維持流向藍色服務修訂版的正常流量流程。

- e. (選用) 指定日誌組態。選取使用日誌收集。預設選項會將容器日誌傳送至 CloudWatch 日誌。其他日誌驅動程式選項是使用 AWS FireLens 設定。如需詳細資訊，請參閱[將 Amazon ECS 日誌傳送至 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 – 將任務設定為將容器日誌傳送至 CloudWatch Logs。系統會提供預設日誌驅動程式選項，用來代表您建立 CloudWatch 日誌群組。若要指定不同的日誌群組名稱，請變更驅動程式選項值。
- Amazon Data Firehose – 將任務設定為將容器日誌傳送至 Firehose。系統會提供預設日誌驅動程式選項，用來將日誌傳送至 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 – 將任務設定為將容器日誌傳送至 Amazon Kinesis Data Streams。系統會提供預設日誌驅動程式選項，用來將日誌傳送至 Kinesis Data Streams 串流。若要指定不同的串流名稱，請變更驅動程式選項值。
- Amazon OpenSearch Service – 將任務設定為將容器日誌傳送至 OpenSearch Service 網域。務必提供日誌驅動程式選項。
- Amazon S3：將任務設定為將容器日誌傳送至 Amazon S3 儲存貯體。系統會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。

10. (選用) 設定 Canary 部署的負載平衡。

Elastic Load Balancing 類型	步驟
Application Load Balancer	- 在負載平衡器類型欄位中選擇 Application Load Balancer。 - 選擇 Create a new load balancer (建立新的負載平衡器) 以建立新

Elastic Load Balancing 類型	步驟	
	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. 在容器欄位中選擇託管服務的容器。 d. 針對 Load balancer name (負載平衡器名稱)，輸入不重複的名稱。 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。 - 在生產規則欄位中，輸入規則的評估順序與路徑模式。 此規則適用於生產 (藍色) 服務修訂版流量。 - 在測試規則欄位中，輸入規則的評估順序與路徑模式。 此規則適用於測試 (綠色) 服務修訂版流量。 f. 在目標群組欄位中設定下列項目： - 針對 Target group name (目標群組名	

Elastic Load Balancing 類型	步驟	
	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。 • 在通訊協定欄位中, 選擇 Application Load Balancer 向其路由請求之目標群組的通訊協定。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。 • 針對取消註冊延遲, 輸入負載平衡器將目標狀態變更為 的秒數UNUSED。預設為 300 秒。 • 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 (/)。 • 針對替代群組名稱, 輸入測試服務修訂版目標群組的群組名稱。	

Elastic Load Balancing 類型	步驟	
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. 在生產接聽程式欄位中，選擇生產接聽程式連接埠與生產接聽程式通訊協定。 這是生產服務修訂流量的接聽程式。 e. 在測試接聽程式欄位中，選擇測試接聽程式連接埠與測試接聽程式通訊協定。 這是測試服務修訂流量的接聽程式。 f. 在目標群組欄位中設定下列項目： 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。	

Elastic Load Balancing 類型	步驟	
	• 在通訊協定欄位中，選擇 Network Load Balancer 向其路由請求之目標群組的通訊協定。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。 • 針對取消註冊延遲，輸入負載平衡器將目標狀態變更為 的秒數UNUSED。預設為 300 秒。 • 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 (/)。 • 在替代群組名稱欄位中，輸入測試 (綠色) 服務修訂版的目標群組名稱。	

11. (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。

12. 選擇建立。

AWS CLI

1. 建立名為 canary-service-definition.json 且具有下列內容的檔案。

將 *user-input* 取代為實際值。

```
{
  "serviceName": "myCanaryService",
  "cluster": "arn:aws:ecs:us-west-2:123456789012:cluster/sample-fargate-cluster",
  "taskDefinition": "sample-fargate:1",
  "desiredCount": 5,
  "launchType": "FARGATE",
  "networkConfiguration": {
    "awsvpcConfiguration": {
      "subnets": [
        "subnet-09ce6e74c116a2299",
        "subnet-00bb3bd7a73526788",
        "subnet-0048a611aaec65477"
      ],
      "securityGroups": [
        "sg-09d45005497daa123"
      ],
      "assignPublicIp": "ENABLED"
    }
  },
  "deploymentController": {
    "type": "ECS"
  }
}
```

```

    },
    "deploymentConfiguration": {
      "strategy": "CANARY",
      "maximumPercent": 200,
      "minimumHealthyPercent": 100,
      "canaryConfiguration" : {
        "canaryPercent" : 5.0,
        "canaryBakeTime" : 10
      },
      "bakeTimeInMinutes": 10,
      "alarms": {
        "alarmNames": [
          "myAlarm"
        ],
        "rollback": true,
        "enable": true
      }
    },
    "loadBalancers": [
      {
        "targetGroupArn": "arn:aws:elasticloadbalancing:us-
west-2:123456789012:targetgroup/blue-target-group/54402ff563af1197",
        "containerName": "fargate-app",
        "containerPort": 80,
        "advancedConfiguration": {
          "alternateTargetGroupArn": "arn:aws:elasticloadbalancing:us-
west-2:123456789012:targetgroup/green-target-group/cad10a56f5843199",
          "productionListenerRule": "arn:aws:elasticloadbalancing:us-
west-2:123456789012:listener-rule/app/my-canary-
demo/32e0e4f946c3c05b/9cfa8c482e204f7d/831dbaf72edb911",
          "roleArn": "arn:aws:iam::123456789012:role/LoadBalancerManagementforECS"
        }
      }
    ]
  }
}

```

2. 執行 create-service。

```
aws ecs create-service --cli-input-json file://canary-service-definition.json
```

後續步驟

設定 Canary 部署之後，請完成下列步驟：

- 更新服務以開始部署。如需詳細資訊，請參閱[更新 Amazon ECS 服務](#)。
- 監控部署程序，以確保其遵循 Canary 模式：
 - 綠色服務修訂版已建立並向上擴展
 - 一小部分的流量 (Canary) 會轉移到綠色修訂版
 - 系統會等待指定的 Canary 間隔
 - 剩餘的流量會一次全部轉移到綠色修訂版
 - 封裝時間過期後，藍色修訂版會終止

部署術語

下列術語用於 Amazon ECS 部署文件：

藍綠部署

與現有環境（藍色）一起建立新環境（綠色）的部署策略，然後在驗證後將流量從藍色切換到綠色。

Canary 部署

一種部署策略，可將一小部分的流量路由到新版本，同時將大多數流量維持在穩定版本上進行驗證。

線性部署

一種部署策略，隨著時間的推移，逐漸將流量從舊版本轉移到新版本。

滾動部署

一種部署策略，可將舊版本的執行個體一次取代為新版本的執行個體。

任務集

部署期間在服務內執行相同任務定義的任務集合。

目標群組

部署期間從負載平衡器接收流量的目標邏輯分組。

部署控制器

用來部署新版本服務的方法，例如 Amazon ECS、CodeDeploy 或外部控制器。

轉返

在部署期間偵測到問題時，還原至舊版應用程式的程序。

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。如果需要更新運算選項、平台版本、負載平衡器詳細資訊、網路組態或任務定義，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 或 EXTERNAL 部署控制器類型不支援 DAEMON 排程策略。
- 應用程式自動擴展功能與 Amazon ECS 的整合僅支援將 Amazon ECS 服務作為目標。Amazon ECS 支援的可擴展維度為 `ecs:service:DesiredCount` (Amazon ECS 服務的任務計數)。應用程式自動擴展功能與 Amazon ECS 任務集之間沒有直接整合。Amazon ECS 任務集會根據 Amazon ECS 服務的 `DesiredCount` 計算 `ComputedDesiredCount`。

外部部署工作流程

下列是在 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。最多允許 255 個字母（大寫和小寫）、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](#)。

更新服務時，此參數不會觸發新的服務部署。

`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。

enableECSTags

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[將標籤用於計費](#)。

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。

更新服務時，此參數不會觸發新的服務部署。

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)。若使用的是 Fargate，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": ""
    }
  ],
  "enableECSTags": 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`。

更新服務時，此參數會觸發新的服務部署。

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)。

更新服務時，此參數會觸發新的服務部署。

AWS Fargate 平台版本用於參考 Fargate 任務基礎設施的特定執行期環境。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 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 聯網的詳細資訊，請參閱 [Fargate 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 [使用服務探索以利用 DNS 名稱連接 Amazon ECS 服務](#)。

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。

適用於 Amazon ECS 的 CodeDeploy 藍/綠部署

建議使用 Amazon ECS 藍/綠部署。如需詳細資訊，請參閱[建立 Amazon ECS 藍/綠部署](#)。

藍/綠部署類型使用由 CodeDeploy 控制的藍/綠部署模式。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 User Guide 中的 [What Is CodeDeploy](#)。在部署之前，驗證 Amazon ECS 服務的狀態

在藍/綠部署期間，有三種流量轉移方式：

- **Canary**：流量以兩個增量轉移。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**：流量以每個增量之間的相等分鐘數按同等增量轉移。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，以及在每個增量之間的分鐘數。
- **一次全部**：所有流量從原始任務集一次全部轉移至更新後的任務集。

以下是當在服務使用藍/綠部署類型時，Amazon ECS 使用的 CodeDeploy 的元件：

CodeDeploy 應用程式

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程式資訊
- 部署復原策略
- 流量重新路由設定
- 原始修訂版終止設定
- Deployment configuration (部署組態)

- 可以設定為停止部署的 CloudWatch 警示組態
- 通知的 SNS 或 CloudWatch Events 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 CodeDeploy 在部署期間將生產流量路由到您替代任務集的方式。以下預先定義的線性與 Canary 部署組態可供使用。您也可以建立自訂定義的線性與 Canary 部署部署。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.ECSAllAtOnce：將所有流量一次轉移至更新的 Amazon ECS 容器
- CodeDeployDefault.ECSLinear10PercentEvery1Minutes：每分鐘轉移 10% 的流量，直到轉移所有流量為止。
- CodeDeployDefault.ECSLinear10PercentEvery3Minutes：每 3 分鐘轉移 10% 的流量，直到轉移所有流量為止。
- CodeDeployDefault.ECSCanary10Percent5Minutes：在第一次增量中轉移 10% 的流量。剩餘的 90% 會在五分鐘之後部署。
- CodeDeployDefault.ECSCanary10Percent15Minutes：在第一次增量中轉移 10% 的流量。剩餘的 90% 會在 15 分鐘後部署。

修訂

修訂是 CodeDeploy 應用程式規格檔案 (AppSpec 檔案)。在 AppSpec 檔案中，您會指定任務定義的完整 ARN 和建立新部署時要路由流量到其中的替代任務集的容器和連接埠。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。如果已在服務定義中更新網路組態或平台版本，您還必須在 AppSpec 檔案中指定這些詳細資訊。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。如需詳細資訊，請參閱 AWS CodeDeploy 使用者指南中的[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，CodeDeploy 會在部署期間將您的測試流量路由到替代任務集。
- 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。
 - 若服務在擴展而部署開始，綠任務集被建立，CodeDeploy 將最長等待一個小時並且不會轉移任何流量，直至綠任務集達到穩定狀態。
 - 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。若服務未在 5 分鐘內達到穩定狀態，CodeDeploy 將停止部署並將其標記為失敗。
- 使用 Fargate 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 (或綠任務集)。如果您將測試接聽程式新增至負載平衡器，CodeDeploy 會將您的測試流量路由到替代任務集。此時您可以執行任何驗證測試。然後，CodeDeploy 會根據部署群組的流量重新路由設定，將生產流量從原始任務集重新路由到替代任務集。

所需的 IAM 許可

透過結合 Amazon ECS 與 CodeDeploy API 可實現藍/綠部署。使用者必須擁有這些服務的適當許可，才能在或 AWS 管理主控台或 AWS CLI SDKs 中使用 Amazon ECS 藍/綠部署。

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[AmazonECS_FullAccess](#)。

- `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`

 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 需要許可來呼叫 Amazon ECS API、修改 Elastic Load Balancing、叫用 Lambda 函數和說明 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](#)。

將 CodeDeploy 藍/綠部署遷移至 Amazon ECS 藍/綠部署

CodeDeploy 藍/綠部署與 Amazon ECS 藍/綠部署提供類似的功能，但兩者在設定與管理方式上有所不同。

CodeDeploy 藍/綠部署概觀

使用 CodeDeploy 建立 Amazon ECS 服務時，您需要：

1. 建立具有生產接聽程式與測試接聽程式 (選用) 的負載平衡器。每個接聽程式都會設定單一 (預設) 規則，將所有流量路由至單一目標群組 (主要目標群組)。
2. 建立 Amazon ECS 服務，設定為使用該接聽程式與目標群組，並將 deploymentController 類型設定為 CODE_DEPLOY。建立服務後，會建立已向指定目標群組註冊的 (藍色) 任務集。
3. 建立 CodeDeploy 部署群組 (作為 CodeDeploy 應用程式的一部分)，並使用 Amazon ECS 叢集、服務名稱、負載平衡器接聽程式、兩個目標群組 (用於生產接聽程式規則的主要目標群組，以及用於替代任務的次要目標群組)、服務角色 (向 CodeDeploy 授予操作 Amazon ECS 與 Elastic Load Balancing 資源的許可) 以及用於控制部署行為之各種參數的詳細資訊對該部署群組進行設定。

使用 CodeDeploy 時，會指定 CodeDeploy 應用程式名稱、部署群組名稱與 AppSpec 檔案 (該檔案提供有關新修訂版與選用 lifecycle hook 的詳細資訊)，透過 CreateDeployment() 部署新版本的服務。CodeDeploy 部署會建立替代 (綠色) 任務集，並向次要目標群組註冊其任務。當運作狀態良好時，即可進行測試 (選用) 與生產。在這兩種情況下，會透過將個別接聽程式規則變更為指向與綠色任務集關聯的次要目標群組來實現重新路由。透過將生產接聽程式規則變更回主要目標群組來實現復原。

Amazon ECS 藍/綠部署概觀

使用 Amazon ECS 藍/綠部署時，部署組態是 Amazon ECS 服務本身的一部分：

1. 必須預先設定負載平衡器生產接聽程式，並使其具備的規則包含兩個權重分別為 1 與 0 的目標群組。
2. 您需要指定下列資源，或更新服務資源：
 - 此接聽程式規則的 ARN
 - 兩個目標群組
 - IAM 角色，用於授予 Amazon ECS 呼叫 Elastic Load Balancing API 的許可
 - 選用 IAM 角色，用於執行 Lambda 函式

- 將 `deploymentController` 類型設定為 `ECS`，並將 `deploymentConfiguration.strategy` 設定為 `BLUE_GREEN`。這會建立 (藍色) 服務部署，並向主要目標群組註冊其任務。

使用 Amazon ECS 藍/綠部署時，會透過呼叫 `Amazon ECS UpdateService()` 建立新的服務修訂版，並傳遞新修訂版的詳細資訊。服務部署會建立新的 (綠色) 服務修訂版任務，並向次要目標群組註冊任務。Amazon ECS 透過切換接聽程式規則上的權重來處理重新路由與復原操作。

關鍵實作差異

雖然這兩種方法都會建立初始任務集，但基礎實作有所不同：

- CodeDeploy 使用任務集，而 Amazon ECS 使用服務修訂版。Amazon ECS 任務集是較舊的建構模組，已由 Amazon ECS 服務修訂版與部署取代。後者提供了對部署程序以及服務部署與服務修訂版歷史記錄的更高可見性。
- 對於 CodeDeploy，`lifecycle hook` 會指定為提供給 `CreateDeployment()` 之 `AppSpec` 檔案的一部分。這表示勾點可以在不同的部署之間變更。對於 Amazon ECS 藍/綠部署，勾點會指定為服務組態的一部分，進行任何更新都需要呼叫 `UpdateService()`。
- CodeDeploy 與 Amazon ECS 藍/綠部署都使用 `Lambda` 進行勾點實作，但預期的輸入與輸出有所不同。

對於 CodeDeploy，函式必須呼叫 `PutLifecycleEventHookExecutionStatus()` 才能傳回勾點狀態，狀態可以是 `SUCCEEDED` 或 `FAILED`。對於 Amazon ECS，`Lambda` 回應用於指示勾點狀態。

- CodeDeploy 會調用每個勾點作為一次性呼叫，且預期會在一小時內傳回最終執行狀態。Amazon ECS 勾點則更具彈性，因為它們可以傳回 `IN_PROGRESS` 指標，這表示必須反復重新調用勾點，直到狀態為 `SUCCEEDED` 或 `FAILED`。如需詳細資訊，請參閱 [Amazon ECS 服務部署的 lifecycle hook](#)。

遷移方法

從 CodeDeploy 藍/綠部署遷移到 Amazon ECS 藍/綠部署主要有三種方法。每種方法在複雜性、風險、復原功能與潛在停機時間方面都有不同的特性。

重複使用用於 CodeDeploy 的相同 Elastic Load Balancing 資源

您可以更新現有的 Amazon ECS 服務，以搭配藍/綠部署策略使用 Amazon ECS 部署控制器，而不是 CodeDeploy 部署控制器。採取此方法時，請考量下列事項：

- 遷移程序更為簡單，因為您正在更新現有的 Amazon ECS 服務部署控制器與部署策略。
- 正確設定與遷移時，不會有停機時間。
- 復原需要您還原服務修訂版。
- 風險較高，因為沒有平行的藍/綠組態。

您可以使用與用於 CodeDeploy 的相同負載平衡器接聽程式及目標群組。如果您使用的是 CloudFormation，請參閱 [將 CloudFormation CodeDeploy 藍/綠部署範本遷移至 Amazon ECS 藍/綠 CloudFormation 範本](#)。

1. 修改生產/測試接聽程式的預設規則，以包含替代目標群組，並將主要目標群組的權重設定為 1，將替代目標群組設定為 0。

對於 CodeDeploy，必須為連接至服務的負載平衡器接聽程式設定單一 (預設) 規則，將所有流量路由至單一目標群組。對於 Amazon ECS 藍/綠部署，負載平衡器接聽程式必須預先設定規則，其中包含兩個具有權重的目標群組。主要目標群組的權重必須為 1，替代目標群組的權重必須為 0。

2. 透過呼叫 UpdateService API 並將參數 deploymentController 設定為 ECS，將參數 deploymentStrategy 設定為 BLUE_GREEN，來更新現有的 Amazon ECS 服務。指定目標群組、替代目標群組、生產接聽程式以及選用的測試接聽程式的 ARN。
3. 驗證服務是否如預期般運作。
4. 刪除此 Amazon ECS 服務的 CodeDeploy 設定，因為您現在使用的是 Amazon ECS 藍/綠部署。

使用現有負載平衡器的新服務

此方法使用藍/綠策略進行遷移。

採取此方法時，請考量下列事項：

- 干擾最少。僅在 Elastic Load Balancing 連接埠交換期間發生。
 - 復原需要還原 Elastic Load Balancing 連接埠交換。
 - 風險較低，因為存在平行組態。因此，您可以在流量轉移之前進行測試。
1. 保持 CodeDeploy 設定的接聽程式、目標群組與 Amazon ECS 服務不變，以便您可以視需要輕鬆復原至此設定。
 2. 在現有負載平衡器下建立新的目標群組與新的接聽程式 (使用與原始接聽程式不同的連接埠)。然後，建立一個符合現有 Amazon ECS 服務的新 Amazon ECS 服務，不同之處在於使用 ECS 作為部署控制器，BLUE_GREEN 作為部署策略，並傳遞新目標群組與新接聽程式規則的 ARN。

3. 透過向服務手動傳送 HTTP 流量來驗證新設定。如果一切順利，請交換原始接聽程式與新接聽程式的連接埠，將流量路由至新設定。
4. 驗證新設定，如果一切繼續如預期般運作，則刪除 CodeDeploy 設定。

使用新負載平衡器的新服務

與前一種方法類似，此方法使用藍/綠策略進行遷移。關鍵差異在於從 CodeDeploy 設定切換到 Amazon ECS 藍/綠設定發生於負載平衡器上方的反向代理層。反向代理層的範例實作是 Route 53 與 CloudFront。

此方法適用於已經擁有此反向代理層的客戶，並且所有與服務的通訊都透過該代理層進行的情況 (例如，負載平衡器層級沒有直接通訊)。

採取此方法時，請考量下列事項：

- 這需要反向代理層。
 - 遷移程序更為複雜，因為您需要更新現有的 Amazon ECS 服務部署控制器與部署策略。
 - 干擾最少。僅在 Elastic Load Balancing 連接埠交換期間發生。
 - 復原需要您反轉代理組態變更。
 - 風險較低，因為存在平行組態。因此，您可以在流量轉移之前進行測試。
1. 請勿完全修改現有的 CodeDeploy 設定 (負載平衡器、接聽程式、目標群組、Amazon ECS 服務與 CodeDeploy 部署群組)。
 2. 建立針對 Amazon ECS 藍/綠部署設定的新負載平衡器、目標群組與接聽程式。

設定相應的資源。

- Application Load Balancer – 如需詳細資訊，請參閱[藍/綠、線性和金絲雀部署的 Application Load Balancer 資源](#)。
 - Network Load Balancer – 如需詳細資訊，請參閱[Amazon ECS 藍/綠、線性和 Canary 部署的 Network Load Balancer 資源](#)。
3. 建立新服務，將 ECS 作為部署控制器，BLUE_GREEN 作為部署策略，並指向新的負載平衡器資源。
 4. 透過新的負載平衡器測試新設定，以驗證設定。
 5. 更新反向代理組態，將流量路由至新的負載平衡器。

6. 觀測新的服務修訂版，如果一切繼續如預期般運作，則刪除 CodeDeploy 設定。

後續步驟

遷移至 Amazon ECS 藍/綠部署之後：

- 更新部署指令碼與 CI/CD 管道，以使用 Amazon ECS UpdateService API 而不是 CodeDeploy CreateDeployment API。
- 更新監控與提醒，以追蹤 Amazon ECS 服務部署，而不是 CodeDeploy 部署。
- 請考慮實作新部署程序的自動化測試，確保其如預期般運作。

從 CodeDeploy 藍/綠部署遷移至 Amazon ECS 藍/綠服務部署

透過使用 Amazon ECS 藍/綠部署，您可以進行服務變更並加以測試，然後在生產環境中實作這些變更。

必須為 Amazon ECS 藍/綠部署建立新的 lifecycle hook。

先決條件

開始藍/綠部署之前，請執行下列動作。

1. 將 Amazon ECS CodeDeploy IAM 角色取代為下列許可。
 - 如需有關 Elastic Load Balancing 許可的資訊，請參閱[負載平衡器的 Amazon ECS 基礎結構 IAM 角色](#)。
 - 如需有關 Lambda 許可的資訊，請參閱[Amazon ECS 藍/綠部署中 Lambda 函式所需的許可](#)。
2. 關閉 CodeDeploy 自動化。如需詳細資訊，請參閱 CodeDeploy User Guide 中的[Working with deployment groups in CodeDeploy](#)。
3. 確定擁有 CodeDeploy 藍/綠部署中的下列資訊。您可以重複使用這些資訊進行 Amazon ECS 藍/綠部署：
 - 生產目標群組
 - 生產接聽程式
 - 生產規則
 - 測試目標群組

這是綠色服務修訂版的目標群組，

4. 確保 Application Load Balancer 目標群組與接聽程式規則正確建立關聯：

- 如果您未使用測試接聽程式，則兩個目標群組 (生產與測試) 都必須與生產接聽程式規則建立關聯。
- 如果您使用的是測試接聽程式，一個目標群組必須連結至生產接聽程式規則，另一個目標群組則必須連結至測試接聽程式規則。

如果不符合此需求，服務部署將會失敗，並出現下列錯誤：Service deployment rolled back because of invalid networking configuration. Both targetGroup and alternateTargetGroup must be associated with the productionListenerRule or testListenerRule.

5. 確認服務沒有正在進行的服務部署。如需詳細資訊，請參閱[使用 Amazon ECS 服務部署檢視服務歷史記錄](#)。
6. Amazon ECS 藍/綠部署要求服務使用下列其中一項功能：設定相應的資源。
 - Application Load Balancer – 如需詳細資訊，請參閱[藍/綠、線性和金絲雀部署的 Application Load Balancer 資源](#)。
 - Network Load Balancer – 如需詳細資訊，請參閱[Amazon ECS 藍/綠、線性和 Canary 部署的 Network Load Balancer 資源](#)。
 - Service Connect – 如需詳細資訊，請參閱[Amazon ECS 藍色/綠色、線性和 Canary 部署的 Service Connect 資源](#)。
7. 決定是否要為 Amazon ECS 藍/綠部署中的生命週期階段執行 Lambda 函式。
 - 向上擴展前
 - 向上擴展後
 - 測試流量轉移
 - 測試流量轉移後
 - 生產流量轉移
 - 生產流量轉移後

為每個生命週期階段建立 Lambda 函式。如需詳細資訊，請參閱 AWS Lambda Developer Guide 中的 [Create a Lambda function with the console](#)。

如需有關更新服務部署控制器的詳細資訊，請參閱[更新 Amazon ECS 服務參數](#)。

程序

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。

「叢集詳細資訊」頁面隨即顯示。

3. 從服務索引標籤中選擇服務。

服務詳細資訊頁面隨即顯示。

4. 在橫幅中，選擇更新部署控制器類型。

遷移部署控制器類型頁面隨即顯示。

5. 展開新增區段，然後指定下列參數。
 - a. 在部署控制器類型欄位中選擇 ECS。
 - b. 在部署策略欄位中選擇藍/綠部署。
 - c. 在封裝時間欄位中，輸入藍色與綠色服務修訂版同時執行的時間。
 - d. 若要為生命週期階段執行 Lambda 函式，請在部署 lifecycle hook 下為每個唯一的 Lambda 函式執行下列動作：
 - i. 選擇新增。

針對每個要執行的唯一函式重複此步驟。
 - ii. 在 Lambda 函式欄位中輸入函式名稱。
 - iii. 在角色欄位中，選擇您在先決條件中建立的具有藍/綠部署許可的角色。

如需詳細資訊，請參閱[Amazon ECS 藍/綠部署中 Lambda 函式所需的許可](#)。
 - iv. 在生命週期階段欄位中選取 Lambda 函式執行的階段。
 - v. (選用) 在勾點詳細資訊欄位中，輸入提供勾點相關資訊的鍵值對。
6. 展開負載平衡區段，並設定下列項目：

- a. 在角色欄位中，選擇您在先決條件中建立的具有藍/綠部署許可的角色。

如需詳細資訊，請參閱[Amazon ECS 藍/綠部署中 Lambda 函式所需的許可](#)。

- b. 在接聽程式欄位中，從 CodeDeploy 藍/綠部署中選擇生產接聽程式。
 - c. 在生產規則欄位中，從 CodeDeploy 藍/綠部署中選擇生產規則。
 - d. 在測試規則欄位中，從 CodeDeploy 藍/綠部署中選擇測試規則。
 - e. 在目標群組欄位中，從 CodeDeploy 藍/綠部署中選擇生產目標群組。
 - f. 在替代目標群組欄位中，從 CodeDeploy 藍/綠部署中選擇測試目標群組。

7. 選擇更新。

後續步驟

- 更新服務以開始部署。如需詳細資訊，請參閱[更新 Amazon ECS 服務](#)。
- 監控部署程序，確保其遵循藍/綠模式：
 - 綠色服務修訂版已建立並向上擴展
 - 測試流量會路由至綠色修訂版 (若已設定)
 - 生產流量會轉移至綠色修訂版
 - 封裝時間過期後，藍色修訂版會終止

將 CloudFormation CodeDeploy 藍/綠部署範本遷移至 Amazon ECS 藍/綠 CloudFormation 範本

將針對 Amazon ECS 服務使用 CodeDeploy 藍/綠部署的 CloudFormation 範本遷移至使用原生 Amazon ECS 藍/綠部署策略的範本。遷移遵循「重複使用用於 CodeDeploy 的相同 Elastic Load Balancing 資源」方法。如需詳細資訊，請參閱[the section called “將 CodeDeploy 藍/綠部署遷移至 Amazon ECS 藍/綠部署”](#)。

來源範本

此範本使用 `AWS::CodeDeployBlueGreen` 轉換與 `AWS::CodeDeploy::BlueGreen` 勾點為 Amazon ECS 服務實作藍/綠部署。

來源

這是使用 CodeDeploy 藍/綠部署的完整 CloudFormation 範本。如需詳細資訊，請參閱 AWS CloudFormation 《使用者指南》中的[藍/綠部署範本範例](#)：

```
{
  "AWSTemplateFormatVersion": "2010-09-09",
  "Parameters": {
    "Vpc": {
      "Type": "AWS::EC2::VPC::Id"
    },
    "Subnet1": {
      "Type": "AWS::EC2::Subnet::Id"
    },
    "Subnet2": {
      "Type": "AWS::EC2::Subnet::Id"
    }
  },
  "Transform": [
```



```

    "AWS::CodeDeployBlueGreen"
  ],
  "Hooks": {
    "CodeDeployBlueGreenHook": {
      "Type": "AWS::CodeDeploy::BlueGreen",
      "Properties": {
        "TrafficRoutingConfig": {
          "Type": "TimeBasedCanary",
          "TimeBasedCanary": {
            "StepPercentage": 15,
            "BakeTimeMins": 5
          }
        },
        "Applications": [
          {
            "Target": {
              "Type": "AWS::ECS::Service",
              "LogicalID": "ECSDemoService"
            },
            "ECSAttributes": {
              "TaskDefinitions": [
                "BlueTaskDefinition",
                "GreenTaskDefinition"
              ],
              "TaskSets": [
                "BlueTaskSet",
                "GreenTaskSet"
              ],
              "TrafficRouting": {
                "ProdTrafficRoute": {
                  "Type": "AWS::ElasticLoadBalancingV2::Listener",
                  "LogicalID": "ALBListenerProdTraffic"
                },
                "TargetGroups": [
                  "ALBTargetGroupBlue",
                  "ALBTargetGroupGreen"
                ]
              }
            }
          }
        ]
      }
    }
  }
},

```

```

"Resources": {
  "ExampleSecurityGroup": {
    "Type": "AWS::EC2::SecurityGroup",
    "Properties": {
      "GroupDescription": "Security group for ec2 access",
      "VpcId": {"Ref": "Vpc"},
      "SecurityGroupIngress": [
        {
          "IpProtocol": "tcp",
          "FromPort": 80,
          "ToPort": 80,
          "CidrIp": "0.0.0.0/0"
        },
        {
          "IpProtocol": "tcp",
          "FromPort": 8080,
          "ToPort": 8080,
          "CidrIp": "0.0.0.0/0"
        },
        {
          "IpProtocol": "tcp",
          "FromPort": 22,
          "ToPort": 22,
          "CidrIp": "0.0.0.0/0"
        }
      ]
    }
  },
  "ALBTargetGroupBlue": {
    "Type": "AWS::ElasticLoadBalancingV2::TargetGroup",
    "Properties": {
      "HealthCheckIntervalSeconds": 5,
      "HealthCheckPath": "/",
      "HealthCheckPort": "80",
      "HealthCheckProtocol": "HTTP",
      "HealthCheckTimeoutSeconds": 2,
      "HealthyThresholdCount": 2,
      "Matcher": {
        "HttpCode": "200"
      },
      "Port": 80,
      "Protocol": "HTTP",
      "Tags": [
        {

```

```

        "Key": "Group",
        "Value": "Example"
    }
],
"TargetType": "ip",
"UnhealthyThresholdCount": 4,
"VpcId": {"Ref": "Vpc"}
}
},
"ALBTargetGroupGreen": {
    "Type": "AWS::ElasticLoadBalancingV2::TargetGroup",
    "Properties": {
        "HealthCheckIntervalSeconds": 5,
        "HealthCheckPath": "/",
        "HealthCheckPort": "80",
        "HealthCheckProtocol": "HTTP",
        "HealthCheckTimeoutSeconds": 2,
        "HealthyThresholdCount": 2,
        "Matcher": {
            "HttpCode": "200"
        },
        "Port": 80,
        "Protocol": "HTTP",
        "Tags": [
            {
                "Key": "Group",
                "Value": "Example"
            }
        ],
        "TargetType": "ip",
        "UnhealthyThresholdCount": 4,
        "VpcId": {"Ref": "Vpc"}
    }
},
"ExampleALB": {
    "Type": "AWS::ElasticLoadBalancingV2::LoadBalancer",
    "Properties": {
        "Scheme": "internet-facing",
        "SecurityGroups": [
            {"Ref": "ExampleSecurityGroup"}
        ],
        "Subnets": [
            {"Ref": "Subnet1"},
            {"Ref": "Subnet2"}
        ]
    }
}

```

```

    ],
    "Tags": [
      {
        "Key": "Group",
        "Value": "Example"
      }
    ],
    "Type": "application",
    "IpAddressType": "ipv4"
  }
},
"ALBListenerProdTraffic": {
  "Type": "AWS::ElasticLoadBalancingV2::Listener",
  "Properties": {
    "DefaultActions": [
      {
        "Type": "forward",
        "ForwardConfig": {
          "TargetGroups": [
            {
              "TargetGroupArn": {"Ref": "ALBTargetGroupBlue"},
              "Weight": 1
            }
          ]
        }
      }
    ],
    "LoadBalancerArn": {"Ref": "ExampleALB"},
    "Port": 80,
    "Protocol": "HTTP"
  }
},
"ALBListenerProdRule": {
  "Type": "AWS::ElasticLoadBalancingV2::ListenerRule",
  "Properties": {
    "Actions": [
      {
        "Type": "forward",
        "ForwardConfig": {
          "TargetGroups": [
            {
              "TargetGroupArn": {"Ref": "ALBTargetGroupBlue"},
              "Weight": 1
            }
          ]
        }
      }
    ]
  }
}

```

```

        ]
      }
    }
  ],
  "Conditions": [
    {
      "Field": "http-header",
      "HTTPHeaderConfig": {
        "HTTPHeaderName": "User-Agent",
        "Values": [
          "Mozilla"
        ]
      }
    }
  ],
  "ListenerArn": {"Ref": "ALBListenerProdTraffic"},
  "Priority": 1
}
},
"ECSTaskExecutionRole": {
  "Type": "AWS::IAM::Role",
  "Properties": {
    "AssumeRolePolicyDocument": {
      "Version": "2012-10-17",
      "Statement": [
        {
          "Sid": "",
          "Effect": "Allow",
          "Principal": {
            "Service": "ecs-tasks.amazonaws.com"
          },
          "Action": "sts:AssumeRole"
        }
      ]
    }
  },
  "ManagedPolicyArns": [
    "arn:aws:iam::aws:policy/service-role/AmazonECSTaskExecutionRolePolicy"
  ]
},
"BlueTaskDefinition": {
  "Type": "AWS::ECS::TaskDefinition",
  "Properties": {
    "ExecutionRoleArn": {"Fn::GetAtt": ["ECSTaskExecutionRole", "Arn"]},

```

```
"ContainerDefinitions": [
  {
    "Name": "DemoApp",
    "Image": "nginxdemos/hello:latest",
    "Essential": true,
    "PortMappings": [
      {
        "HostPort": 80,
        "Protocol": "tcp",
        "ContainerPort": 80
      }
    ]
  }
],
"RequiresCompatibilities": [
  "FARGATE"
],
"NetworkMode": "awsvpc",
"Cpu": "256",
"Memory": "512",
"Family": "ecs-demo"
},
"ECSDemoCluster": {
  "Type": "AWS::ECS::Cluster",
  "Properties": {}
},
"ECSDemoService": {
  "Type": "AWS::ECS::Service",
  "Properties": {
    "Cluster": {"Ref": "ECSDemoCluster"},
    "DesiredCount": 1,
    "DeploymentController": {
      "Type": "EXTERNAL"
    }
  }
},
"BlueTaskSet": {
  "Type": "AWS::ECS::TaskSet",
  "Properties": {
    "Cluster": {"Ref": "ECSDemoCluster"},
    "LaunchType": "FARGATE",
    "NetworkConfiguration": {
      "AwsVpcConfiguration": {
```

```

        "AssignPublicIp": "ENABLED",
        "SecurityGroups": [
            {"Ref": "ExampleSecurityGroup"}
        ],
        "Subnets": [
            {"Ref": "Subnet1"},
            {"Ref": "Subnet2"}
        ]
    },
    "PlatformVersion": "1.4.0",
    "Scale": {
        "Unit": "PERCENT",
        "Value": 100
    },
    "Service": {"Ref": "ECSDemoService"},
    "TaskDefinition": {"Ref": "BlueTaskDefinition"},
    "LoadBalancers": [
        {
            "ContainerName": "DemoApp",
            "ContainerPort": 80,
            "TargetGroupArn": {"Ref": "ALBTargetGroupBlue"}
        }
    ]
},
"PrimaryTaskSet": {
    "Type": "AWS::ECS::PrimaryTaskSet",
    "Properties": {
        "Cluster": {"Ref": "ECSDemoCluster"},
        "Service": {"Ref": "ECSDemoService"},
        "TaskSetId": {"Fn::GetAtt": ["BlueTaskSet", "Id"]}
    }
}
}
}
}

```

移轉步驟

移除 CodeDeploy 特定資源

不再需要下列屬性：

- `AWS::CodeDeployBlueGreen` 轉換

- CodeDeployBlueGreenHook 勾點
- GreenTaskDefinition 與 GreenTaskSet 資源 (這些資源將由 Amazon ECS 管理)
- PrimaryTaskSet 資源 (Amazon ECS 將在內部管理任務集)

重新設定負載平衡器接聽程式

修改 ALBListenerProdTraffic 資源以使用具有兩個目標群組的轉送動作：

```
{
  "DefaultActions": [
    {
      "Type": "forward",
      "ForwardConfig": {
        "TargetGroups": [
          {
            "TargetGroupArn": {"Ref": "ALBTargetGroupBlue"},
            "Weight": 1
          },
          {
            "TargetGroupArn": {"Ref": "ALBTargetGroupGreen"},
            "Weight": 0
          }
        ]
      }
    }
  ]
}
```

更新部署屬性

更新並新增下列項目：

- 將 DeploymentController 屬性從 EXTERNAL 變更為 ECS。
- 新增 Strategy 屬性，並將其設定為 BLUE_GREEN。
- 新增 BakeTimeInMinutes 屬性。

```
{
  "DeploymentConfiguration": {
    "MaximumPercent": 200,
    "MinimumHealthyPercent": 100,
    "DeploymentCircuitBreaker": {
```



```

    "Enable": true,
    "Rollback": true
  },
  "BakeTimeInMinutes": 5,
  "Strategy": "BLUE_GREEN"
}
}

```

- 將負載平衡器組態新增至服務：

```

{
  "LoadBalancers": [
    {
      "ContainerName": "DemoApp",
      "ContainerPort": 80,
      "TargetGroupArn": {"Ref": "ALBTargetGroupBlue"},
      "AdvancedConfiguration": {
        "AlternateTargetGroupArn": {"Ref": "ALBTargetGroupGreen"},
        "ProductionListenerRule": {"Ref": "ALBListenerProdRule"},
        "RoleArn": {"Fn::GetAtt": ["ECSInfrastructureRoleForLoadBalancers", "Arn"]}
      }
    }
  ]
}

```

- 將任務定義參考新增至服務：

```

{
  "TaskDefinition": {"Ref": "BlueTaskDefinition"}
}

```

建立 AmazonECSInfrastructureRolePolicyForLoadBalancers 角色

新增允許 Amazon ECS 管理負載平衡器資源的新 IAM 角色。如需詳細資訊，請參閱[the section called “負載平衡器的基礎結構 IAM 角色”](#)

測試建議

1. 將遷移後的範本部署至非生產環境。
2. 確認服務已使用初始組態正確部署。
3. 透過更新任務定義並觀測藍/綠部署程序來測試部署。

4. 確認流量在藍色與綠色部署之間正確轉移。
5. 透過強制執行部署失敗來測試復原功能。

遷移後的範本

最終範本

這是使用 Amazon ECS 藍/綠部署的完整 CloudFormation 範本：

```
{
  "AWSTemplateFormatVersion": "2010-09-09",
  "Parameters": {
    "Vpc": {
      "Type": "AWS::EC2::VPC::Id"
    },
    "Subnet1": {
      "Type": "AWS::EC2::Subnet::Id"
    },
    "Subnet2": {
      "Type": "AWS::EC2::Subnet::Id"
    }
  },
  "Resources": {
    "ExampleSecurityGroup": {
      "Type": "AWS::EC2::SecurityGroup",
      "Properties": {
        "GroupDescription": "Security group for ec2 access",
        "VpcId": {"Ref": "Vpc"},
        "SecurityGroupIngress": [
          {
            "IpProtocol": "tcp",
            "FromPort": 80,
            "ToPort": 80,
            "CidrIp": "0.0.0.0/0"
          },
          {
            "IpProtocol": "tcp",
            "FromPort": 8080,
            "ToPort": 8080,
            "CidrIp": "0.0.0.0/0"
          },
          {
            "IpProtocol": "tcp",
```

```

        "FromPort": 22,
        "ToPort": 22,
        "CidrIp": "0.0.0.0/0"
    }
]
}
},
"ALBTargetGroupBlue": {
    "Type": "AWS::ElasticLoadBalancingV2::TargetGroup",
    "Properties": {
        "HealthCheckIntervalSeconds": 5,
        "HealthCheckPath": "/",
        "HealthCheckPort": "80",
        "HealthCheckProtocol": "HTTP",
        "HealthCheckTimeoutSeconds": 2,
        "HealthyThresholdCount": 2,
        "Matcher": {
            "HttpCode": "200"
        },
        "Port": 80,
        "Protocol": "HTTP",
        "Tags": [
            {
                "Key": "Group",
                "Value": "Example"
            }
        ],
        "TargetType": "ip",
        "UnhealthyThresholdCount": 4,
        "VpcId": {"Ref": "Vpc"}
    }
},
"ALBTargetGroupGreen": {
    "Type": "AWS::ElasticLoadBalancingV2::TargetGroup",
    "Properties": {
        "HealthCheckIntervalSeconds": 5,
        "HealthCheckPath": "/",
        "HealthCheckPort": "80",
        "HealthCheckProtocol": "HTTP",
        "HealthCheckTimeoutSeconds": 2,
        "HealthyThresholdCount": 2,
        "Matcher": {
            "HttpCode": "200"
        },
    },

```

```

    "Port": 80,
    "Protocol": "HTTP",
    "Tags": [
      {
        "Key": "Group",
        "Value": "Example"
      }
    ],
    "TargetType": "ip",
    "UnhealthyThresholdCount": 4,
    "VpcId": {"Ref": "Vpc"}
  }
},
"ExampleALB": {
  "Type": "AWS::ElasticLoadBalancingV2::LoadBalancer",
  "Properties": {
    "Scheme": "internet-facing",
    "SecurityGroups": [
      {"Ref": "ExampleSecurityGroup"}
    ],
    "Subnets": [
      {"Ref": "Subnet1"},
      {"Ref": "Subnet2"}
    ],
    "Tags": [
      {
        "Key": "Group",
        "Value": "Example"
      }
    ],
    "Type": "application",
    "IpAddressType": "ipv4"
  }
},
"ALBListenerProdTraffic": {
  "Type": "AWS::ElasticLoadBalancingV2::Listener",
  "Properties": {
    "DefaultActions": [
      {
        "Type": "forward",
        "ForwardConfig": {
          "TargetGroups": [
            {
              "TargetGroupArn": {"Ref": "ALBTargetGroupBlue"},

```

```

        "Weight": 1
      },
      {
        "TargetGroupArn": {"Ref": "ALBTargetGroupGreen"},
        "Weight": 0
      }
    ]
  }
},
"LoadBalancerArn": {"Ref": "ExampleALB"},
"Port": 80,
"Protocol": "HTTP"
}
},
"ALBListenerProdRule": {
  "Type": "AWS::ElasticLoadBalancingV2::ListenerRule",
  "Properties": {
    "Actions": [
      {
        "Type": "forward",
        "ForwardConfig": {
          "TargetGroups": [
            {
              "TargetGroupArn": {"Ref": "ALBTargetGroupBlue"},
              "Weight": 1
            },
            {
              "TargetGroupArn": {"Ref": "ALBTargetGroupGreen"},
              "Weight": 0
            }
          ]
        }
      }
    ]
  },
  "Conditions": [
    {
      "Field": "http-header",
      "HTTPHeaderConfig": {
        "HttpHeaderName": "User-Agent",
        "Values": [
          "Mozilla"
        ]
      }
    }
  ]
}
}

```

```

    }
  ],
  "ListenerArn": {"Ref": "ALBListenerProdTraffic"},
  "Priority": 1
}
},
"ECSTaskExecutionRole": {
  "Type": "AWS::IAM::Role",
  "Properties": {
    "AssumeRolePolicyDocument": {
      "Version": "2012-10-17",
      "Statement": [
        {
          "Sid": "",
          "Effect": "Allow",
          "Principal": {
            "Service": "ecs-tasks.amazonaws.com"
          },
          "Action": "sts:AssumeRole"
        }
      ]
    },
    "ManagedPolicyArns": [
      "arn:aws:iam::aws:policy/service-role/AmazonECSTaskExecutionRolePolicy"
    ]
  }
},
"ECSInfrastructureRoleForLoadBalancers": {
  "Type": "AWS::IAM::Role",
  "Properties": {
    "AssumeRolePolicyDocument": {
      "Version": "2012-10-17",
      "Statement": [
        {
          "Sid": "AllowAccessToECSForInfrastructureManagement",
          "Effect": "Allow",
          "Principal": {
            "Service": "ecs.amazonaws.com"
          },
          "Action": "sts:AssumeRole"
        }
      ]
    },
    "ManagedPolicyArns": [

```

```

        "arn:aws:iam::aws:policy/AmazonECSInfrastructureRolePolicyForLoadBalancers"
    ]
}
},
"BlueTaskDefinition": {
    "Type": "AWS::ECS::TaskDefinition",
    "Properties": {
        "ExecutionRoleArn": {"Fn::GetAtt": ["ECSTaskExecutionRole", "Arn"]},
        "ContainerDefinitions": [
            {
                "Name": "DemoApp",
                "Image": "nginxdemos/hello:latest",
                "Essential": true,
                "PortMappings": [
                    {
                        "HostPort": 80,
                        "Protocol": "tcp",
                        "ContainerPort": 80
                    }
                ]
            }
        ]
    },
    "RequiresCompatibilities": [
        "FARGATE"
    ],
    "NetworkMode": "awsvpc",
    "Cpu": "256",
    "Memory": "512",
    "Family": "ecs-demo"
}
},
"ECSDemoCluster": {
    "Type": "AWS::ECS::Cluster",
    "Properties": {}
},
"ECSDemoService": {
    "Type": "AWS::ECS::Service",
    "Properties": {
        "Cluster": {"Ref": "ECSDemoCluster"},
        "DesiredCount": 1,
        "DeploymentController": {
            "Type": "ECS"
        },
        "DeploymentConfiguration": {

```

```
"MaximumPercent": 200,  
  "MinimumHealthyPercent": 100,  
  "DeploymentCircuitBreaker": {  
    "Enable": true,  
    "Rollback": true  
  },  
  "BakeTimeInMinutes": 5,  
  "Strategy": "BLUE_GREEN"  
},  
"NetworkConfiguration": {  
  "AwsVpcConfiguration": {  
    "AssignPublicIp": "ENABLED",  
    "SecurityGroups": [  
      {"Ref": "ExampleSecurityGroup"}  
    ],  
    "Subnets": [  
      {"Ref": "Subnet1"},  
      {"Ref": "Subnet2"}  
    ]  
  }  
},  
"LaunchType": "FARGATE",  
"PlatformVersion": "1.4.0",  
"TaskDefinition": {"Ref": "BlueTaskDefinition"},  
"LoadBalancers": [  
  {  
    "ContainerName": "DemoApp",  
    "ContainerPort": 80,  
    "TargetGroupArn": {"Ref": "ALBTARGETGROUPBLUE"},  
    "AdvancedConfiguration": {  
      "AlternateTargetGroupArn": {"Ref": "ALBTARGETGROUPGREEN"},  
      "ProductionListenerRule": {"Ref": "ALBLISTENERPRODRULE"},  
      "RoleArn": {"Fn::GetAtt": ["ECSInfrastructureRoleForLoadBalancers",  
"Arn"]}  
    }  
  }  
]  
}  
}
```


從 CodeDeploy 藍/綠部署遷移至 Amazon ECS 滾動更新服務部署

您可以將服務部署從 CodeDeploy 藍/綠部署遷移至 Amazon ECS 滾動更新部署。這將使您擺脫 CodeDeploy 相依性，轉而使用整合式部署。

Amazon ECS 服務排程器會將目前執行的任務取代為新任務。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。

先決條件

開始藍/綠部署之前，請執行下列動作。

1. 您不再需要 Amazon ECS CodeDeploy IAM 角色。
2. 關閉 CodeDeploy 自動化。如需詳細資訊，請參閱 CodeDeploy User Guide 中的 [Working with deployment groups in CodeDeploy](#)。
3. 確認服務沒有正在進行的服務部署。如需詳細資訊，請參閱[使用 Amazon ECS 服務部署檢視服務歷史記錄](#)。

如需有關更新服務部署控制器的詳細資訊，請參閱[更新 Amazon ECS 服務參數](#)。

程序

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。

「叢集詳細資訊」頁面隨即顯示。

3. 從服務索引標籤中選擇服務。

服務詳細資訊頁面隨即顯示。

4. 在橫幅中，選擇遷移。

更新部署組態頁面隨即顯示。

5. 展開部署選項區段，然後指定下列參數。
 - a. 在部署控制器類型欄位中選擇 ECS。
 - b. 在部署策略欄位中選擇滾動更新。
 - c. 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](#)。

- d. 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)。
6. 展開負載平衡區段，然後設定下列項目：
 - a. 在角色欄位中，選擇您在先決條件中建立的具有藍/綠部署許可的角色。
如需詳細資訊，請參閱[Amazon ECS 藍/綠部署中 Lambda 函式所需的許可](#)。
 - b. 在接聽程式欄位中，從 CodeDeploy 藍/綠部署中選擇生產接聽程式。
 - c. 在目標群組欄位中，從 CodeDeploy 藍/綠部署中選擇生產目標群組。
 7. 選擇更新。

後續步驟

您必須更新服務，變更才會生效。如需詳細資訊，請參閱[更新 Amazon ECS 服務](#)。

更新 Amazon ECS 部署策略

Amazon ECS 支援多種部署策略來更新服務。您可以根據應用程式需求在這些策略之間遷移。本主題說明如何在滾動部署與藍/綠部署之間遷移。

了解 Amazon ECS 部署策略

在部署策略之間進行遷移之前，務必了解每種策略的運作方式及其主要差異：

滾動部署

在滾動部署中，Amazon ECS 會將應用程式目前執行的版本取代為新版本。服務排程器會使用運作狀態百分比下限與運作狀態百分比上限參數來決定部署策略。

滾動部署更容易設定，但對部署程序與流量路由的控制較少。

藍/綠部署

在藍/綠部署中，Amazon ECS 會與現有版本 (藍色) 一起建立新的服務版本 (綠色)。這可讓您在將生產流量路由至新版本之前先驗證新版本。

藍/綠部署提供對部署程序的更多控制，包括流量轉移、測試與復原功能。

最佳實務

在部署策略之間進行遷移時，請遵循下列最佳實務：

- 在非生產環境中測試：一律先在非生產環境中測試更新，然後再將變更套用至生產服務。
- 制定復原計畫：制定復原計畫，以防更新無法如預期般運作。
- 在轉移期間進行監控：在遷移期間與遷移之後密切監控服務，確保其繼續正常運作。
- 更新文件：更新部署文件以反映新的部署策略。
- 考量流量影響：了解更新可能如何影響服務的流量，並據此進行規劃。

將部署策略從滾動更新更新為 Amazon ECS 藍/綠部署

若要在生產環境中實作服務變更之前，先進行變更與測試，可以從滾動更新部署遷移至 Amazon ECS 藍/綠部署。

先決條件

在將服務從滾動部署遷移至藍/綠部署之前，請確定您已滿足下列條件：

- 等待任何目前的部署完成。
- 現有 Amazon ECS 服務使用滾動部署策略。
- 如果您有多個服務修訂版正在處理流量，Amazon ECS 會嘗試在遷移期間將流量合併為單一修訂版。如果失敗，您可能需要在遷移之前手動更新服務，以使用單一修訂版。
- 設定相應的許可。
 - 如需有關 Elastic Load Balancing 許可的資訊，請參閱[負載平衡器的 Amazon ECS 基礎結構 IAM 角色](#)。
 - 如需有關 Lambda 許可的資訊，請參閱[Amazon ECS 藍/綠部署中 Lambda 函式所需的許可](#)。
- 根據組態，您需要執行下列其中一項動作：
 - 如果服務使用 Elastic Load Balancing，請使用新的 `advancedConfiguration` 更新服務並啟動滾動部署。
 - 如果服務使用 Service Connect，請更新服務並啟動滾動部署。
 - 如果服務同時使用 Elastic Load Balancing 與 Service Connect，請執行上述兩個步驟 (可以使用單一 UpdateService 請求)。
 - 如果服務不使用上述任何一項，則無需執行其他操作。
- Amazon ECS 藍/綠部署要求服務使用下列其中一項功能。設定相應的資源。
 - Application Load Balancer – 如需詳細資訊，請參閱[藍/綠、線性和金絲雀部署的 Application Load Balancer 資源](#)。

- Network Load Balancer – 如需詳細資訊，請參閱[Amazon ECS 藍/綠、線性和 Canary 部署的 Network Load Balancer 資源](#)。
- Service Connect – 如需詳細資訊，請參閱[Amazon ECS 藍色/綠色、線性和 Canary 部署的 Service Connect 資源](#)。

程序

1. 在 <https://console.aws.amazon.com/ecs/v2> 開啟 Amazon ECS 主控台。
2. 在導覽窗格中，選擇叢集。
3. 在叢集頁面上，選擇包含要遷移之服務的叢集。

「叢集詳細資訊」頁面隨即顯示。

4. 在叢集詳細資訊頁面中，選擇服務索引標籤。
5. 選擇服務，然後選擇更新。

「更新服務」頁面隨即顯示

6. 展開部署選項區段，然後執行下列動作：
7. 在部署策略欄位中選擇藍/綠部署。
8. 進行藍/綠部署設定：
 - a. 在封裝時間欄位中，輸入在藍色修訂版終止之前，藍色與綠色服務修訂版將同時執行的分鐘數。

這為驗證與測試提供了時間。

- b. (選用) 設定要在部署的特定階段執行的 Lambda 函式。在部署 lifecycle hook 下，為下列階段設定 Lambda 函式：
 - 向上擴展前：在向上擴展綠色服務修訂版之前執行
 - 向上擴展後：在向上擴展綠色服務修訂版之後執行
 - 測試流量轉移：在測試流量路由至綠色服務修訂版期間執行
 - 測試流量轉移後：在測試流量路由至綠色服務修訂版之後執行
 - 生產流量轉移：在生產流量路由至綠色服務修訂版期間執行
 - 生產流量轉移後：在生產流量路由至綠色服務修訂版之後執行

若要新增 lifecycle hook：

- a. 選擇新增。
 - b. 在 Lambda 函式欄位中，輸入函式名稱或 ARN。
 - c. 在角色欄位中，選擇具有調用 Lambda 函式之許可的 IAM 角色。
 - d. 在生命週期階段欄位中，選取 Lambda 函式應執行的階段。
 - e. 選用：在勾點詳細資訊欄位中，輸入鍵值對，為勾點提供其他資訊。
9. 進行負載平衡器設定：
- a. 在負載平衡下，確認服務已設定為使用負載平衡器。
 - b. 在目標群組欄位中，選擇用於生產 (藍色) 環境的主要目標群組。
 - c. 在替代目標群組欄位中，選擇用於測試 (綠色) 環境的目標群組。
 - d. 在生產接聽程式規則欄位中，選擇用於路由生產流量的接聽程式規則。
 - e. 選用：在測試接聽程式規則欄位中，選擇用於將測試流量路由至綠色環境的接聽程式規則。
 - f. 在角色欄位中，選擇允許 Amazon ECS 管理負載平衡器的 IAM 角色。
10. 檢閱組態變更，然後選擇更新。

後續步驟

- 更新服務以開始部署。如需詳細資訊，請參閱[更新 Amazon ECS 服務](#)。
- 監控部署程序，確保其遵循藍/綠模式：
 - 綠色服務修訂版已建立並向上擴展
 - 測試流量會路由至綠色修訂版 (若已設定)
 - 生產流量會轉移至綠色修訂版
 - 封裝時間過期後，藍色修訂版會終止

將部署策略從 Amazon ECS 藍/綠部署更新為滾動更新

您可以將藍/綠部署遷移至滾動更新部署。

遷移至滾動部署時，請記住下列考量事項：

- 流量處理：使用滾動部署時，新任務會在通過運作狀態檢查後立即開始接收流量。與藍/綠部署不同，此方式沒有單獨的測試階段。
- 資源效率：滾動部署通常會使用比藍/綠部署更少的資源，因為這種部署會逐漸取代任務，而不是建立完整的重複環境。

- 復原複雜性：與藍/綠部署相比，滾動部署使復原操作更為複雜。如果需要復原，必須使用先前的任務定義啟動新的部署。
- 部署速度：滾動部署可能需要比藍/綠部署更長的時間才能完成，特別是對於具有許多任務的服務。
- 負載平衡器組態：現有的負載平衡器組態仍然適用於滾動部署，但流量轉移行為會有所不同。

先決條件

在將服務從藍/綠部署遷移至滾動部署之前，請確定您已滿足下列條件：

- 現有 Amazon ECS 服務使用藍/綠部署策略
- 服務沒有正在進行的部署 (等待任何目前部署完成)
- 清楚了解服務在滾動部署下的行為方式

Note

如果服務有正在進行的部署，則無法將其遷移至滾動部署。等待任何目前的部署完成，然後再繼續操作。

遷移程序

請遵循下列步驟，將 Amazon ECS 服務從藍/綠部署遷移至滾動部署：

1. 在 <https://console.aws.amazon.com/ecs/v2> 開啟 Amazon ECS 主控台。
2. 在導覽窗格中，選擇叢集。
3. 在叢集頁面上，選擇包含要遷移之服務的叢集。
4. 在叢集詳細資訊頁面中，選擇服務索引標籤。
5. 選取要遷移的服務，然後選擇更新。
6. 在更新服務頁面上，導覽至部署選項區段，並視需要展開該區段。
7. 在部署策略欄位中選擇滾動更新。
8. 對滾動部署進行設定：
 - a. 在運作狀態百分比下限欄位中，輸入部署期間必須保持在 RUNNING 狀態的任務百分比下限。此值指定為服務所需任務數量的百分比。

- b. 在百分比上限欄位中，輸入部署期間允許處於 RUNNING 或 PENDING 狀態的任務百分比上限。此值指定為服務所需任務數量的百分比。
9. 選用：在部署失敗偵測下，設定 Amazon ECS 如何偵測與處理部署失敗：
 - a. 若要啟用部署斷路器，請選擇使用部署斷路器。
 - b. 若要自動復原失敗的部署，請選擇失敗時復原。
10. 檢閱組態變更，然後選擇更新以儲存變更，並將服務遷移至滾動部署。

Amazon ECS 會更新服務組態，以使用滾動部署策略。下次更新服務時，將會使用滾動部署程序。

Note

從藍/綠部署遷移至滾動部署時，Amazon ECS 會透過下列方式處理轉移：

1. 識別正在處理流量的目前作用中服務修訂版。
2. 維護現有的負載平衡器組態，但變更新部署的處理方式。
3. 為未來的滾動部署準備服務。

後續步驟

- 更新服務以開始部署。如需詳細資訊，請參閱[更新 Amazon ECS 服務](#)。

對 Amazon ECS 部署策略更新進行疑難排解

本節提供遷移部署策略時可能遇到的常見問題的解決方案。

多個服務修訂版或任務集

下列問題與部署時存在多個服務修訂版有關。

更新 ECS 部署控制器時存在多項任務集

錯誤訊息：Updating the deployment controller is not supported when there are multiple tasksets in the service. Please ensure your service has only one taskset and try again.

解決方案：嘗試變更具有多個作用中任務集之服務的部署控制器類型時，會發生此錯誤。若要為 CODE_DEPLOY 或 EXTERNAL 部署控制器解決此問題，請執行下列動作：

1. 檢查目前的任務集：

```
aws ecs describe-services --cluster your-cluster-name --services your-service-name --query "services[0].taskSets"
```

2. 等待任何進行中的部署完成。
3. 強制執行新部署以清理任務集：

```
aws ecs update-service --cluster your-cluster-name --service your-service-name --force-new-deployment
```

4. 如有必要，手動刪除額外的任務集：

```
aws ecs delete-task-set --cluster your-cluster-name --service your-service-name --task-set task-set-id
```

5. 在僅剩一個任務集後，重試更新部署控制器。

如需詳細資訊，請參閱[the section called “服務部署控制器與策略”](#)。

更新 ECS 部署控制器時缺少主要任務集

錯誤訊息：Updating the deployment controller requires a primary taskset in the service. Please ensure your service has a primary taskset and try again.

解決方案：嘗試變更沒有主要任務集之服務的部署控制器類型時，會發生此錯誤。若要解決此問題：

1. 驗證服務狀態與任務集。如果服務中存在任務集，則應標記為 ACTIVE。

```
aws ecs describe-services --cluster your-cluster-name --services your-service-name --query "services[0].taskSets[*].status[id]"
```

如果沒有處於 ACTIVE 狀態的任務集，請遷移部署。如需詳細資訊，請參閱[遷移方法](#)。

2. 如果服務沒有執行中的任務，請更新服務，以部署至少一項任務：

```
aws ecs update-service-primary-task-set --cluster your-cluster-name --service your-service-name --primary-task-set your-taskset-id
```


這會將服務中 (先前為 ACTIVE 的) 任務集標記為 PRIMARY 狀態。

3. 等待任務達到穩定的執行狀態。您可以使用下列命令檢查狀態：

```
aws ecs describe-services --cluster your-cluster-name --services your-service-name --query "services[0].deployments"
```

4. 在服務的主要任務集存在執行中任務後，重試更新部署控制器。

如需詳細資訊，請參閱[the section called “服務部署控制器與策略”](#)。

部署失敗偵測類型與部署控制器不相符

下列問題和部署失敗偵測類型與部署控制器不相符有關。

使用非 ECS 控制器的部署斷路器

錯誤訊息：Deployment circuit breaker feature is only supported with ECS deployment controller. Update to ECS deployment controller and try again.

解決方案：嘗試在未使用 ECS 部署控制器的服務上啟用部署斷路器功能時，會發生此錯誤。部署斷路器僅與 ECS 部署控制器相容。

1. 檢視服務目前的部署控制器：

```
aws ecs describe-services --cluster your-cluster-name --services your-service-name --query "services[0].deploymentController"
```

2. 更新服務以使用 ECS 部署控制器：

```
aws ecs update-service --cluster your-cluster-name --service your-service-name --deployment-controller type=ECS
```

3. 服務使用 ECS 部署控制器後，啟用部署斷路器：

```
aws ecs update-service --cluster your-cluster-name --service your-service-name --deployment-configuration "deploymentCircuitBreaker={enable=true,rollback=true}"
```

如需詳細資訊，請參閱[the section called “部署斷路器如何偵測失敗”](#)。

使用非 ECS 控制器的警示型復原

錯誤訊息：Alarm based rollback feature is only supported with ECS deployment controller. Update to ECS deployment controller and try again.

解決方案：嘗試在未使用 ECS 部署控制器的服務上設定警示型復原時，會發生此錯誤。警示型復原功能僅與 ECS 部署控制器相容。

1. 檢視服務目前的部署控制器：

```
aws ecs describe-services --cluster your-cluster-name --services your-service-name --query "services[0].deploymentController"
```

2. 更新服務以使用 ECS 部署控制器：

```
aws ecs update-service --cluster your-cluster-name --service your-service-name --deployment-controller type=ECS
```

3. 服務使用 ECS 部署控制器後，設定警示型復原：

```
aws ecs update-service --cluster your-cluster-name --services your-service-name --deployment-configuration "alarms={alarmNames=[your-alarm-name],enable=true,rollback=true}"
```

如需詳細資訊，請參閱[the section called “CloudWatch 警示如何偵測部署失敗”](#)。

Service Connect 與部署控制器不相符

下列問題和 Service Connect 與部署控制器不相符有關。

使用 Service Connect 的 EXTERNAL 控制器

錯誤訊息：The EXTERNAL deployment controller type is not supported for services using Service Connect.

解決方案：嘗試搭配已啟用 Service Connect 的服務使用 EXTERNAL 部署控制器時，會發生此錯誤。EXTERNAL 控制器與 Service Connect 不相容。

1. 檢查服務是否啟用了 Service Connect：

```
aws ecs describe-services --cluster your-cluster-name --services your-service-name --query "services[0].serviceConnectConfiguration"
```

2. 如果需要使用 EXTERNAL 部署控制器，請更新服務以停用 Service Connect：

```
aws ecs update-service --cluster your-cluster-name --service your-service-name --service-connect-configuration "{}"
```

3. 或者，如果必須使用 Service Connect，請改用 ECS 部署控制器：

```
aws ecs update-service --cluster your-cluster-name --service your-service-name --deployment-controller type=ECS
```

如需詳細資訊，請參閱[the section called “服務部署控制器與策略”](#)。

使用非 ECS 控制器的 Service Connect

錯誤訊息：Service Connect feature is only supported with ECS (rolling update) deployment controller. Update to ECS deployment controller and try again.

解決方案：嘗試在未使用 ECS 部署控制器的服務上啟用 Service Connect 時，會發生此錯誤。Service Connect 功能僅與 ECS 部署控制器相容。

1. 檢視服務目前的部署控制器：

```
aws ecs describe-services --cluster your-cluster-name --services your-service-name --query "services[0].deploymentController"
```

2. 更新服務以使用 ECS 部署控制器：

```
aws ecs update-service --cluster your-cluster-name --service your-service-name --deployment-controller type=ECS
```

3. 服務使用 ECS 部署控制器後，即可啟用 Service Connect：

```
aws ecs update-service --cluster your-cluster-name --service your-service-name --service-connect-configuration "enabled=true,namespace=your-namespace"
```

如需詳細資訊，請參閱[the section called “服務部署控制器與策略”](#)。

控制器類型與排程策略不相符

下列問題和控制器類型與排程策略不相符有關。

使用 DAEMON 排程策略的 CODE_DEPLOY 控制器

錯誤訊息：The CODE_DEPLOY deployment controller type is not supported for services using the DAEMON scheduling strategy.

解決方案：嘗試搭配使用了 DAEMON 排程策略的服務使用 CODE_DEPLOY 部署控制器時，會發生此錯誤。CODE_DEPLOY 控制器僅與 REPLICA 排程策略相容。

1. 檢查服務目前的排程策略：

```
aws ecs describe-services --cluster your-cluster-name --services your-service-name --query "services[0].schedulingStrategy"
```

2. 如果需要使用藍/綠部署，請將服務變更為使用 REPLICA 排程策略：

```
aws ecs update-service --cluster your-cluster-name --service your-service-name --scheduling-strategy REPLICA
```

3. 或者，如果必須使用 DAEMON 排程策略，請改用 ECS 部署控制器：

```
aws ecs update-service --cluster your-cluster-name --service your-service-name --deployment-controller type=ECS
```

如需詳細資訊，請參閱[the section called “服務部署控制器與策略”](#)。

使用 DAEMON 排程策略的 EXTERNAL 控制器

錯誤訊息：The EXTERNAL deployment controller type is not supported for services using the DAEMON scheduling strategy.

解決方案：嘗試搭配使用了 DAEMON 排程策略的 ECS 服務使用 EXTERNAL 部署控制器時，會發生此錯誤。EXTERNAL 控制器僅與 REPLICA 排程策略相容。

1. 檢查服務目前的排程策略：

```
aws ecs describe-services --cluster your-cluster-name --services your-service-name --query "services[0].schedulingStrategy"
```

2. 如果需要使用 EXTERNAL 部署控制器，請將服務變更為使用 REPLICAS 排程策略：

```
aws ecs update-service --cluster your-cluster-name --service your-service-name --scheduling-strategy REPLICAS
```

3. 或者，如果必須使用 DAEMON 排程策略，請改用 ECS 部署控制器：

```
aws ecs update-service --cluster your-cluster-name --service your-service-name --deployment-controller type=ECS
```

如需詳細資訊，請參閱[the section called “服務部署控制器與策略”](#)。

使用外部啟動類型的服務登錄檔

錯誤訊息：Service registries are not supported for external launch type.

解決方案：嘗試為使用 EXTERNAL 啟動類型的服務設定服務探索 (服務登錄檔) 時，會發生此錯誤。服務探索與 EXTERNAL 啟動類型不相容。

1. 檢查服務目前的啟動類型：

```
aws ecs describe-services --cluster your-cluster-name --services your-service-name --query "services[0].launchType"
```

2. 如果需要使用服務探索，請將服務變更為使用 EC2 或 FARGATE 啟動類型：

```
aws ecs update-service --cluster your-cluster-name --service your-service-name --launch-type FARGATE
```

3. 或者，如果必須使用 EXTERNAL 啟動類型，請移除服務登錄檔組態：

```
aws ecs update-service --cluster your-cluster-name --service your-service-name --service-registries "[]"
```

如需詳細資訊，請參閱[the section called “服務部署控制器與策略”](#)。

還原部署控制器更新

如果決定要返回先前的部署控制器，可以執行下列其中一項動作：

- 如果您使用 CloudFormation，則可以使用先前的範本來建立新的堆疊。如需詳細資訊，請參閱 CloudFormation User Guide 中的 [Create a stack from](#)。
- 如果您使用 Amazon ECS 主控台或 AWS CLI，則可以更新服務。如需詳細資訊，請參閱 [the section called “更新服務”](#)。

如果您使用 `update-service` 命令，請使用 `--deployment-controller` 選項，並將其設定為先前的部署控制器。

使用 Amazon ECS 服務部署檢視服務歷史記錄

服務部署可用於全面檢視部署。服務部署提供下列服務相關資訊：

- 目前部署的工作負載組態 (來源服務修訂版)
- 正在部署的工作負載組態 (目標服務修訂版)
- 部署狀態
- 斷路器偵測到的失敗任務數量
- 處於警示狀態的 CloudWatch 警示
- 服務部署開始與完成的時間
- 已發生之復原的詳細資訊

如需有關服務部署屬性的資訊，請參閱 [Amazon ECS 服務部署中包含的屬性](#)。

服務部署是唯讀的，每個都有唯一的 ID。

服務部署分為三個階段：

階段	定義	相關狀態
待定	已建立服務部署，但尚未啟動	待定
持續性	服務部署正在進行中	• IN_PROGRESS • STOP_REQUESTED • ROLLBACK_REQUESTED

階段	定義	相關狀態
		ROLLBACK_IN_PROGRESS
已完成	服務部署已完成 (成功或失敗)	- SUCCESSFUL - 已停止 - ROLLBACK_SUCCESSFUL - ROLLBACK_FAILED

您可以使用服務部署來了解服務的生命週期，並判斷是否需要採取任何動作。例如，如果發生復原，您可能需要調查服務部署並查看服務事件。

您可以使用主控台、API 與 AWS CLI，檢視 2024 年 10 月 25 日當天或之後建立之部署的最近 90 天歷史記錄。

您可以停止尚未完成的部署。如需詳細資訊，請參閱[停止 Amazon ECS 服務部署](#)。

服務部署生命週期

發生下列任何動作時，Amazon ECS 會自動建立新的服務部署：

- 使用者建立服務。
- 使用者更新服務並使用「強制執行新部署」選項。
- 使用者更新一個或多個需要部署的服務屬性。

在部署進行期間，Amazon ECS 會更新下列服務部署屬性，以反映服務部署進度：

- 狀態
- 執行中任務數量

服務修訂版中指示的執行中任務數量，可能不等於實際的執行中任務數量。此數字代表部署完成時的執行中任務數量。例如，若您啟動獨立於服務部署的任務，則這些任務不會包含在服務修訂版的執行中任務計數中。

- 斷路器失敗偵測：
 - 啟動失敗的任務數量
- CloudWatch 警示失敗偵測

- 處於作用中狀態的警示
- 復原資訊：
 - 啟動時間
 - 復原原因
 - 用於復原之服務修訂版的 ARN
- 狀態原因

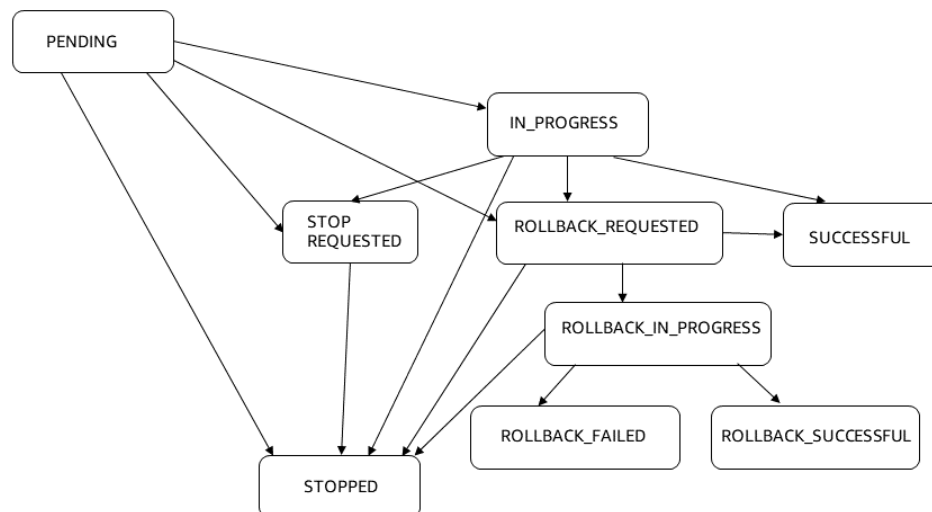
當您刪除服務時，Amazon ECS 會刪除服務部署。

服務部署狀態

服務部署從 PENDING 狀態啟動。

下圖展示了在 PENDING 狀態之後可能發生的服務部署狀態：

IN_PROGRESS、ROLLBACK_REQUESTED、SUCCESSFUL、STOP_REQUESTED、ROLLBACK_IN_PROGRESS 與 STOPPED。



下列資訊提供有關服務部署狀態的詳細資訊：

- PENDING：已建立服務部署，但尚未啟動。

該狀態可以移至 IN_PROGRESS、ROLLBACK_REQUESTED、STOP_REQUESTED 或 STOPPED。

- IN_PROGRESS：服務部署正在進行中。

該狀態可以移至

SUCCESSFUL、STOP_REQUESTED、ROLLBACK_REQUESTED、ROLLBACK_IN_PROGRESS 與 STOPPED。

- STOP_REQUESTED：發生下列任何情況時，服務部署狀態會移至 STOP_REQUESTED：
 - 使用者啟動新的服務部署。
 - 失敗偵測機制 (斷路器或警示類型) 未使用復原選項，且服務未達到 SUCCESSFUL 狀態。

該狀態會移至 STOPPED。

- ROLLBACK_REQUESTED：當使用者透過主控台、API 或 CLI 請求復原時，服務部署狀態會移至 ROLLBACK_REQUESTED。

該狀態可以移至 SUCCESSFUL、ROLLBACK_IN_PROGRESS 與 STOPPED。

- SUCCESSFUL：當服務部署成功完成後，服務部署狀態會移至 SUCCESSFUL。
- ROLLBACK_IN_PROGRESS：當失敗偵測機制 (斷路器或警示型) 使用了復原選項且服務失敗時，服務部署狀態會移至 ROLLBACK_IN_PROGRESS。

該狀態可以移至 ROLLBACK_SUCCESSFUL 或 ROLLBACK_FAILED。

Amazon ECS 服務部署中包含的屬性

服務部署中包含下列屬性。

屬性	Description
服務部署 ARN	服務部署的 ARN。
服務 ARN	此服務部署所屬服務的 ARN。
叢集 ARN	託管服務的叢集 ARN。
服務部署建立時間	服務部署建立的時間。
服務部署開始時間	服務部署開始的時間。

屬性	Description	
服務部署完成時間	服務部署完成的時間。	
服務部署停止時間	服務部署停止的時間。	
服務部署更新時間	服務部署上次更新的時間。	
來源服務修訂版	目前執行中的服務修訂版。 如需有關包含的屬性的資訊，請參閱 Amazon ECS 服務修訂版中包含的屬性。	
Deployment configuration (部署組態)	部署參數，包括斷路器組態、用於判斷的警示。 - 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)。 - 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上限 (佔服務所需任務數量的百分比)。 - 斷路器組態。 - 用於失敗偵測的警示。	
目標服務修訂版	要部署的服務修訂版。 部署成功完成後，目標服務修訂版即為執行中的服務修訂版。	

屬性	Description	
服務部署狀態	服務部署狀態。 有效值為 PENDING、SUCCESSFUL、STOPPED、STOP_REQUESTED、STOP_IN_PROGRESS、IN_PROGRESS、ROLLBACK_IN_PROGRESS、ROLLBACK_SUCCESSFUL 與 ROLLBACK_FAILED。	
服務部署狀態資訊	有關服務部署為何處於目前狀態的資訊。例如，斷路器偵測到失敗。	
復原資訊	部署失敗時，服務部署使用的復原選項。	
服務部署斷路器選項	用於判斷服務部署失敗的斷路器。	
服務部署的 CloudWatch 警示	用於判斷服務部署失敗的 CloudWatch 警示。	

檢視 Amazon ECS 服務部署所需的許可

當您遵循授予最低權限的最佳實務時，必須新增額外許可，才能在主控台中檢視服務部署。

您需要存取下列動作：

- ListServiceDeployments
- DescribeServiceDeployments
- DescribeServiceRevisions

您需要存取下列資源：

- 服務
- 服務部署
- 服務修訂版

下列範例政策包含必要的許可，並將動作限制於指定的服務。

將 `account`、`cluster-name` 與 `service-name` 取代為實際值。

JSON

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ecs:ListServiceDeployments",
        "ecs:DescribeServiceDeployments",
        "ecs:DescribeServiceRevisions"
      ],
      "Resource": [
        "arn:aws:ecs:us-east-1:123456789012:service/cluster-name/service-name",
        "arn:aws:ecs:us-east-1:123456789012:service-deployment/cluster-name/service-name/*",
        "arn:aws:ecs:us-east-1:123456789012:service-revision/cluster-name/service-name/*"
      ]
    }
  ]
}
```

檢視 Amazon ECS 服務部署

您可以檢視 2024 年 10 月 25 日當天或之後建立的部署的最近 90 天歷史記錄。服務部署可以處於下列任何狀態：

- 進行中
- 待定
- 已完成

您可以使用此資訊來判斷是否需要更新服務的部署方式或服務修訂版。如需有關包含的屬性的資訊，請參閱 [Amazon ECS 服務部署中包含的屬性](#)。

開始之前，請先設定檢視服務部署所需的許可。如需詳細資訊，請參閱[檢視 Amazon ECS 服務部署所需的許可](#)。

Amazon ECS Console

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「叢集詳細資訊」頁面上的服務區段中，選擇所需服務。

服務詳細資訊頁面隨即顯示。

4. 在服務詳細資訊頁面上，選擇部署。
5. 選擇要檢視的服務部署。

檢視此部署類型的服務部署	執行此作業	
持續部署	在正在進行的部署下，選擇部署 ID。	
上次部署	在上次部署下，選擇部署 ID。	
已完成的部署	在服務部署下，選擇部署 ID。	

服務部署詳細資訊頁面隨即顯示。

6. (選用) 比較服務修訂版以檢視差異。

在服務修訂版下，選擇比較修訂版，然後選取 2 個要比較的修訂版。

服務修訂版會並排顯示，並反白顯示差異。

AWS CLI

1. 執行 `list-service-deployments` 以擷取服務部署 ARN。

將變數取代為您的值。

```
aws ecs list-service-deployments --cluster cluster-name --service service-name
```

記下要檢視之部署的 serviceDeploymentArn。

```
{
  "serviceDeployments": [
    {
      "serviceDeploymentArn": "arn:aws:ecs:us-west-2:123456789012:service-deployment/example/sd-example/NCWGC2ZR-taawPAYrIaU5",
      "serviceArn": "arn:aws:ecs:us-west-2:123456789012:service/example/sd-example",
      "clusterArn": "arn:aws:ecs:us-west-2:123456789012:cluster/example",
      "targetServiceRevisionArn": "arn:aws:ecs:us-west-2:123456789012:service-revision/example/sd-example/4980306466373577095",
      "status": "SUCCESSFUL"
    }
  ]
}
```

2. 執行 describe-service-deployments。使用從 list-service-deployments 傳回的 serviceDeploymentArn。

將變數取代為您的值。

```
aws ecs describe-service-deployments --service-deployment-arns
arn:aws:ecs:region:123456789012:service-deployment/cluster-name/service-name/NCWGC2ZR-taawPAYrIaU5
```

後續步驟

您可以在部署中檢視服務修訂版的詳細資訊。如需詳細資訊，請參閱[檢視 Amazon ECS 服務修訂版詳細資訊](#)

Amazon ECS 服務修訂版

服務修訂版包含 Amazon ECS 嘗試部署的工作負載組態記錄。每當您建立或部署服務時，Amazon ECS 就會自動建立並擷取您嘗試在服務修訂版中部署的組態。

服務修訂版為唯讀且具有唯一識別碼。如需有關包含的屬性的資訊，請參閱 [Amazon ECS 服務修訂版中包含的屬性](#)。

服務修訂版提供下列優勢：

- 在服務部署期間，您可以將目前部署的服務修訂版 (來源) 與正在部署的服務修訂版 (目標) 進行比較。
- 當您為服務部署使用復原選項時，Amazon ECS 會自動將服務部署復原至上次成功部署的服務修訂版。
- 服務修訂版會將工作負載組態的記錄集中在一個資源中。

服務修訂版生命週期

當您建立服務，或更新啟動部署的服務屬性時，Amazon ECS 會自動建立新的服務修訂版。

Amazon ECS 不會為復原操作建立新的服務修訂版。Amazon ECS 會使用上次成功的服務修訂版進行復原。

服務修訂版不可變。

您刪除服務時，Amazon ECS 會刪除服務修訂版。

您可以使用主控台、API 與 CLI，檢視 2024 年 10 月 25 日當天或之後建立的服務修訂版。

Amazon ECS 服務修訂版中包含的屬性

服務修訂版中包含下列屬性。

資源	Description
服務 ARN	用於識別用戶端的 ARN。
叢集 ARN	託管服務的叢集 ARN。
任務定義 ARN	用於服務任務的任務定義 ARN。
服務登錄檔	用於服務探索之服務登錄檔的詳細資訊。

資源	Description	
	• 容器名稱 容器名稱值用於您的服務探索服務。 • 容器連接埠 連接埠值用於您的服務探索服務。此值也已在任務定義中指定。 • 連接埠 如果您的服務探索服務指定了 SRV 記錄，則使用該連接埠值。 • 登錄檔 ARN 服務登錄檔的 Amazon Resource Name (ARN)。	
容量提供者	容量提供者策略詳細資訊。 • 容量提供者名稱 用於服務的容量提供者名稱。 • 容量提供者權重 應使用指定容量提供者之已啟動任務總數的相對百分比。 • 容量提供者基礎 在指定容量提供者上執行的任務數量下限。	

資源	Description	
容器映像	有關容器映像的詳細資訊。 • 容器名稱 • 映像摘要 • 容器映像	
聯網	服務的網路組態。 • 是否具有公有 IP 位址 • 任務在其中執行的子網路 • 控制服務流量的安全群組	
啟動類型	用於服務的運算選項。	
Fargate 特定屬性	使用 Fargate 時，此為有關 Fargate 版本的資訊。 • 平台版本與平台系列 • 暫時性儲存 • 為部署配置的暫時性儲存量。 • 用於儲存加密的 KMS 金鑰。	

資源	Description	
部署時設定的 Amazon EBS 磁碟區	在任務定義中指定為啟動時設定的磁碟區的組態。 - 加密 表示磁碟區是否加密。 - 檔案系統類型。 磁碟區的 Linux 檔案系統類型。 - IOPS 每秒 I/O 操作的次數 (IOPS)。 - KMS 金鑰 - IAM 角色 要與此磁碟區建立關聯的 IAM 角色 ARN。 - 大小 磁碟區的大小 (GiB)。 - 快照 ID 磁碟區快照 ID。 - 標籤規格 磁碟區的標籤組態。 - 輸送量 為磁碟區佈建的輸送量。 - 磁碟區類型 磁碟區類型。	

資源	Description	
Service Connect	Service Connect 組態。 - 指出 Service Connect 是否正在使用 - 命名空間 - 互連服務清單 - Service Connect 記錄組態	
服務負載平衡器	路由服務流量的負載平衡器。 - 名稱 - 容器名稱 - 容器連接埠 - 目標群組 ARN	
執行時期監控	指示是否開啟執行時期監控。	
Creation (建立) 日期	服務修訂版建立的日期。	
VPC Lattice	服務修訂版的 VPC Lattice 組態。	

檢視 Amazon ECS 服務修訂版詳細資訊

您可以檢視 2024 年 10 月 25 日當天或之後建立的下列服務修訂版類型的相關資訊：

- 來源 – 目前部署的工作負載組態
- 目標 – 正在部署的工作負載組態

Amazon ECS Console

- 開啟主控台，網址為 <https://console.aws.amazon.com/ecs/v2>。
- 在叢集頁面上，選擇叢集。
- 在「叢集詳細資訊」頁面上的服務區段中，選擇所需服務。

服務詳細資訊頁面隨即顯示。

4. 在服務詳細資訊頁面上，選擇部署。
5. 選擇要檢視的服務修訂版。

檢視此部署類型的服務修訂版	執行此作業	
持續部署	在正在進行的部署下，執行下列動作： • 若要檢視來源服務修訂版，請在服務修訂版下，選擇來源修訂版類型的服務修訂版 ID。 • 若要檢視目標服務修訂版，請在服務修訂版下，選擇目標修訂版類型的服務修訂版 ID。	
上次部署	在上次部署下，選擇目標服務修訂版。	
已完成的部署	在服務部署下，執行下列動作： • 在目標服務修訂版下，選擇 ID。	

AWS CLI

1. 執行 `describe-service-deployments` 以擷取服務修訂版 ARN。

將變數取代為您的值。

```
aws ecs describe-service-deployments --service-deployment-arns
arn:aws:ecs:region:account-id:service/cluster-name/service-name/NCWGC2ZR-
taawPAYrIaU5
```

記下 `sourceServiceRevisions` 或 `targetServiceRevisions` 的 arn。

```
{
  "serviceDeployments": [
    {
      "serviceDeploymentArn": "arn:aws:ecs:us-west-2:123456789012:service-
deployment/example/sd-example/NCWGC2ZR-taawPAYrIaU5",
      "serviceArn": "arn:aws:ecs:us-west-2:123456789012:service/example/
sd-example",
      "clusterArn": "arn:aws:ecs:us-west-2:123456789012:cluster/example",
      "updatedAt": "2024-09-10T16:49:35.572000+00:00",
      "sourceServiceRevision": {
        "arn": "arn:aws:ecs:us-west-2:123456789012:service-revision/
example/sd-example/4980306466373578954",
        "requestedTaskCount": 0,
        "runningTaskCount": 0,
        "pendingTaskCount": 0
      },
      "targetServiceRevision": {
        "arn": "arn:aws:ecs:us-west-2:123456789012:service-revision/
example/sd-example/4980306466373577095",
        "requestedTaskCount": 0,
        "runningTaskCount": 0,
        "pendingTaskCount": 0
      },
      "status": "IN_PROGRESS",
      "deploymentConfiguration": {
        "deploymentCircuitBreaker": {
          "enable": false,
          "rollback": false
        },
        "maximumPercent": 200,
        "minimumHealthyPercent": 100
      }
    }
  ],
  "failures": []
}
```

```
}
```

2. 執行 `describe-service-revisions`。使用從 `describe-service-deployments` 傳回的 `arn`。

將變數取代為您的值。

```
aws ecs describe-service-revisions --service-revision-arns
arn:aws:ecs:region:123456789012:service-revision/cluster-name/service-
name/4980306466373577095
```

在可用區域之間平衡 Amazon ECS 服務

自 2025 年 9 月 5 日起，Amazon ECS 會為所有符合功能資格的服務啟用可用區域重新平衡。當可用區域分佈是第一項任務置放策略，或者沒有置放策略時，服務即符合資格。

為了協助讓應用程式達到高可用性，建議您將多任務服務設定為在多個可用區域中執行。對於將第一個置放策略指定為可用區域分散的服務，AWS 會盡最大努力跨可用區域平均分配服務任務。

但是，在某個可用區域中執行的任務數目，有時可能會與其他可用區域中執行的任務數目不相同，例如在可用區域中斷之後。若要解決此任務不平衡情況，您可以啟用可用區域重新平衡功能。

Amazon ECS 會利用可用區域重新平衡功能，持續監控您每項服務的可用區域之間的任務分佈。當 Amazon ECS 偵測到任務分佈不均勻時，便會自動採取行動來重新平衡可用區域之間的工作負載。這包括在任務最少的可用區域中啟動新任務，以及終止超載的可用區域中的任務。

這種重新分佈可確保不會有單一可用區域成為失敗點，有助於維持容器化應用程式的整體可用性。有了自動重新平衡程序，就不需手動介入操作，且能加快事件後復原的時間。

下列是可用區域重新平衡程序的概觀：

1. Amazon ECS 會在服務達到穩定狀態後開始監控服務，並查看每個可用區域中的執行中任務數量。
2. 當 Amazon ECS 偵測到每個可用區域中執行的任務數量不平衡時，會執行下列操作：
 - 傳送指示可用區域重新平衡即將啟動的服務事件。
 - 在執行中任務數量最少的可用區域中啟動任務
 - 在執行中任務數量最多的可用區域中停止任務。
 - 排程器會等待新啟動的任務變為 HEALTHY 與 RUNNING，然後在過度擴展的可用區域中停止任務。

- 傳送具有可用區域重新平衡結果的服務事件。

Amazon ECS 如何偵測任務分佈不均問題

Amazon ECS 透過將服務所需的任務計數除以設定的可用區域數量，來判斷每個可用區域中的執行中任務數量是否不平衡。如果所需任務計數不能整除，Amazon ECS 會將剩餘的任務均勻分佈到設定的可用區域。每個可用區域必須有至少一項任務。

例如，假設 Amazon ECS 服務所需計數為針對兩個可用區域設定的兩項任務。在這種情況下，所需任務計數可以整除。平衡分佈應為每個可用區域一項任務。如果可用區域 1 中有兩項任務，可用區域 2 中有零項任務，則 Amazon ECS 將透過在可用區域 2 中啟動任務，然後在可用區域 1 中停止任務來啟動重新平衡。

現在，假設 Amazon ECS 服務所需計數為針對兩個可用區域設定的三項任務。在這種情況下，所需任務計數不能整除。平衡分佈應為可用區域 1 中一項任務，可用區域 2 中兩項任務，因為每個可用區域有至少一項任務，且剩餘的任務會置放在可用區域 2 中。

假設 Amazon ECS 服務所需計數為針對三個可用區域設定的五項任務。在這種情況下，所需任務計數不能整除。平衡分佈應為可用區域 1 中一項任務，可用區域 2 與 3 中各兩項任務。在確保每個可用區域都有一項任務後，剩餘的兩項任務會均勻分佈到各可用區域。

設定可用區域重新平衡的考量

若要設定可用區域重新平衡，請考量下列事項：

- 可用區域重新平衡支援 Fargate 與 EC2 容量提供者，並在 Amazon ECS 受管執行個體上受支援。對於 Fargate，Amazon ECS 會自動在可用區域中重新分佈任務，以維持平衡。對於 EC2 容量提供者，Amazon ECS 會盡力重新平衡現有容器執行個體間的任務，同時遵循您定義的置放策略與限制條件。但是，在重新平衡程序中，Amazon ECS 無法在使用率不足的可用區域中啟動新的執行個體，這會將重新平衡限制在現有容器執行個體的範圍內。
- 可用區域重新平衡適用於下列組態：
 - 使用 Replica 策略的服務
 - 將可用區域分佈指定為第一項任務置放策略，或未指定置放策略的服務。
- 您無法搭配符合下列任一條件的服務使用可用區域重新平衡：
 - 使用 Daemon 策略
 - 使用 EXTERNAL 啟動類型 (ECS Anywhere)
 - 對 maximumPercent 值使用 100%

- 使用 Classic Load Balancer
- 使用 `attribute:ecs.availability-zone` 作為任務置放限制條件

使用可用區域重新平衡的置放策略與置放限制條件

置放策略決定了 Amazon ECS 如何選取容器執行個體與可用區域來終止任務置放。任務置放限制條件是判斷是否允許任務在特定容器執行個體上執行的規則。

對於 EC2，您可以搭配可用區域重新平衡使用置放策略與置放限制條件。不過，為了讓可用區域重新平衡正常運作，可用區域分佈置放策略必須是第一項指定的策略。

可用區域重新平衡與各種置放策略組合相容。例如，您可以建立一項策略，先將任務均勻分佈到各個可用區域，然後根據每個可用區域內的記憶體對任務進行分箱封裝。在這種情況下，可用區域重新平衡能夠運作，因為會先指定可用區域分佈策略。

請務必注意，如果置放策略陣列中的第一項策略不是可用區域分佈元件，可用區域重新平衡將無法運作。此需求可確保任務分佈的主要重點是維持可用區域之間的平衡，這對高可用性至關重要。

如需有關任務置放策略與限制條件的詳細資訊，請參閱 [Amazon ECS 如何在容器執行個體上置放任務](#)。

使用 Fargate 的任務不支援任務置放策略與限制條件。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。

下列範例策略會將任務均勻分佈到各個可用區域，然後根據每個可用區域內的記憶體對任務進行分箱封裝。由於 `spread` 策略是第一項策略，可用區域重新平衡與該服務相容。

```
"placementStrategy": [  
  {  
    "field": "attribute:ecs.availability-zone",  
    "type": "spread"  
  },  
  {  
    "field": "memory",  
    "type": "binpack"  
  }  
]
```

開啟可用區域重新平衡功能

您需要為新服務與現有服務啟用可用區域重新平衡功能。

您可以使用主控台 AWS CLI、APIs 和 來啟用和停用可用區域重新平衡 CloudFormation。

AvailabilityZoneRebalancing 的預設行為在建立與更新請求之間有所不同：

- 對於建立服務請求，如果未為 AvailabilityZoneRebalancing 指定值，Amazon ECS 會將該值預設為 ENABLED。
- 對於更新服務請求，如果未為 AvailabilityZoneRebalancing 指定值，Amazon ECS 預設會使用現有服務的 AvailabilityZoneRebalancing 值。如果服務從未設定 AvailabilityZoneRebalancing 值，Amazon ECS 會將其視為 DISABLED。

服務類型	API	主控台	CLI
現有	UpdateService	更新 Amazon ECS 服務	update-service
新增	CreateService	建立 Amazon ECS 滾動更新部署	create-service

下列範例說明如何在建立新服務時啟用服務重新平衡：

```
aws ecs create-service \
  --cluster my-cluster \
  --service-name my-service \
  --task-definition my-task-definition:1 \
  --desired-count 6 \
  --availability-zone-rebalancing ENABLED
```

追蹤 Amazon ECS 可用區域重新平衡

您可以透過主控台或呼叫 `describe-services` 來驗證服務是否已啟用可用區域重新平衡。下列範例可用來查看 CLI 的狀態。

回應將為 ENABLED 或 DISABLED。

```
aws ecs describe-services \
  --services service-name \
  --cluster cluster-name \
  --query services[0].availabilityZoneRebalancing
```

服務事件

Amazon ECS 會傳送服務動作事件，協助您了解可用區域重新平衡生命週期。

事件	案例	Type	進一步了解
SERVICE_REBALANCING_STARTED	Amazon ECS 啟動可用區域重新平衡操作	INFO	service (service-name) is not AZ balanced with number-tasks tasks in Availability Zone 1 , number-tasks tasks in Availability Zone 2 , and number-tasks tasks in Availability Zone 3 . AZ Rebalancing in progress.
SERVICE_REBALANCING_COMPLETED	可用區域重新平衡操作完成	INFO	service (service-name) is AZ balanced with number-tasks tasks in Availability Zone 1 , number-tasks tasks in Availability Zone 2 , and number-tasks tasks in Availability Zone 3 .
TASKS_STARTED	Amazon ECS 在可用區域重新平衡操作中成功啟動任務	INFO	service-name has started number-tasks tasks in Availability

事件	案例	Type	進一步了解
			ity Zone to AZ Rebalance: task-ids.
TASKS_STOPPED	Amazon ECS 在可用區域重新平衡操作中成功停止任務	INFO	service-name has stopped number-tasks running tasks in Availability Zone due to AZ rebalancing: task-id.
SERVICE_TASK_PLACEMENT_FAILURE	Amazon ECS 在可用區域重新平衡操作中啟動任務失敗	ERROR	對於 EC2，請參閱service (service-name) is unable to place a task in Availability Zone because no container instance met all of its requirements. 對於 Fargate，請參閱service (service-name) is unable to place a task in Availability Zone.

事件	案例	Type	進一步了解
TASKSET_SCALE_IN_FAILURE_BY_TASK_PROTECTION	由於任務保護正在使用中，因此會封鎖可用區域重新平衡操作。	INFO	service (<i>service-name</i>) was unable to AZ Rebalance because <i>task-set-name</i> was unable to scale in due to <i>reason</i>.
SERVICE_REBALANCING_STOPPED	可用區域重新平衡操作停止。Amazon ECS 會傳送提供詳細資訊的額外事件。	INFO	service (<i>service-name</i>) stopped AZ Rebalancing.

任務狀態變更事件

Amazon ECS 會針對在重新平衡程序中啟動的每項任務傳送任務狀態變更事件 (START)。

Amazon ECS 會針對在重新平衡程序中停止的每項任務傳送任務狀態變更事件 (STOPPED)。原因設定為 Availability-zone rebalancing initiated by (deployment ecs-svc/*deployment-id*)。

如需有關事件的詳細資訊，請參閱 [Amazon ECS 任務狀態變更事件](#)。

對服務重新平衡進行疑難排解

如果您在重新平衡服務時遇到問題，請考慮下列疑難排解步驟：

重新平衡未啟動

請驗證：

- 服務已啟用服務重新平衡
- 服務使用支援組態 (請參閱 [the section called “設定可用區域重新平衡的考量”](#))
- 服務已達到穩定狀態

重新平衡期間任務置放失敗

如果您看到 SERVICE_TASK_PLACEMENT_FAILURE 事件：

- 對於 EC2：檢查目標可用區域中是否有可用的容器執行個體
- 對於 Fargate：檢查是否存在資源限制條件或服務配額限制了任務置放
- 檢閱任務置放限制條件，確保它們不會阻止適當的任務分佈

重新平衡意外停止

如果您看到 `SERVICE_REBALANCING_STOPPED` 事件：

- 檢查是否存在可能封鎖操作的任務保護
- 尋找可能中斷重新平衡的並行服務部署
- 檢閱服務事件，以取得有關為何停止重新平衡的其他資訊

服務重新平衡的最佳實務

遵循以下最佳實務以充分利用服務重新平衡：

- 監控重新平衡操作 – 設定 CloudWatch 警示來監控與重新平衡相關的服務事件，以便快速識別任何問題。
- 考慮效能影響 – 重新平衡操作可能會暫時增加資源使用量，因為新任務會在舊任務停止之前啟動。
- 策略性使用任務保護 – 如果有關鍵任務不應在重新平衡期間終止，請考慮使用任務保護。
- 制定 EC2 容量計畫 – 對於 EC2，請確保所有可用區域中擁有足夠的容器執行個體，能夠支援重新平衡有效運作。
- 測試重新平衡行為 – 在生產環境中依賴重新平衡之前，請在非生產環境中測試服務在重新平衡操作期間的行為。

使用負載平衡分佈 Amazon ECS 服務流量

服務可以選擇性地設定為使用 Elastic Load Balancing，以將流量均勻分佈到服務中的各項任務。

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。

在上託管的 Amazon ECS 服務 AWS Fargate 支援 Application Load Balancer、Network Load Balancer 和 Gateway Load Balancer。使用下表了解要使用的負載平衡器類型。

負載平衡器類型	在下列情況下使用	
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。	

負載平衡器類型	在下列情況下使用
	使用虛擬應用裝置，如防火牆、入侵偵測與預防系統，以及深層封包檢查系統。

除非服務需要僅 Network Load Balancer 或 Gateway Load Balancer 才能使用的功能，否則建議您針對 Amazon ECS 服務使用 Application Load Balancer，以利用這些最新功能。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 最佳化負載平衡器運作狀態檢查參數

負載平衡器僅會將請求路由至其所屬可用區域中運作狀態良好的目標。每個目標都會註冊至目標群組。負載平衡器會使用目標群組的運作狀態檢查設定，檢查每個目標的運作狀態。目標註冊後，必須通過一次運作狀態檢查，才會視為運作狀態良好。Amazon ECS 會監控負載平衡器。負載平衡器會定期向 Amazon ECS 容器傳送運作狀態檢查。Amazon ECS 代理程式會監控並等待負載平衡器報告容器運作狀態。代理程式必須執行此操作，才會將容器視為運作狀態良好。

有兩個 Elastic Load Balancing 運作狀態檢查參數會影響部署速度：

- 運作狀態檢查間隔：決定個別容器各運作狀態檢查之間的約略間隔時間 (以秒為單位)。依預設，負載平衡器會每 30 秒檢查一次。

此參數名稱為：

- Elastic Load Balancing API 中的 `HealthCheckIntervalSeconds`
- Amazon EC2 主控台上的時間間隔
- 運作良好閾值計數：決定在將運作狀態不良的容器視為運作狀態良好之前，所需的連續成功的運作狀態檢查次數。依預設，負載平衡器需要五次通過運作狀態檢查，才會報告目標容器運作狀態良好。

此參數名稱為：

- Elastic Load Balancing API 中的 `HealthyThresholdCount`
- Amazon EC2 主控台上的運作良好閾值

重要提示：對於新註冊的目標，無論運作良好閾值計數設定為何，只需一次成功的運作狀態檢查，即可將目標視為運作狀態良好。運作良好閾值計數僅適用於目標從運作狀態不良轉換回運作狀態良好的情況。

使用預設設定時，如果目標運作狀態不良，然後復原，則判定容器運作狀態的總時間為 2 分鐘 30 秒 (30 seconds * 5 = 150 seconds)。

如果服務在 10 秒內啟動並穩定下來，您可以加速運作狀態檢查程序。若要加速程序，請縮短運作狀態檢查間隔並減少運作良好閾值計數。

- 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) 或運作良好閾值 (Amazon EC2 主控台名稱) : 2

使用此設定後，運作狀態檢查程序僅需 10 秒，而預設值需要 2 分鐘 30 秒。

如需有關 Elastic Load Balancing 運作狀態檢查參數的詳細資訊，請參閱 Elastic Load Balancing User Guide 中的 [Health checks for your target groups](#)。

針對 Amazon ECS 最佳化負載平衡器連接耗盡參數

為了實現最佳化，用戶端會維持與容器服務的持續連線。這允許該用戶端的後續請求重複使用現有連線。若要停止流向容器的流量，需要通知負載平衡器。負載平衡器會定期檢查用戶端是否關閉持續連線。Amazon ECS 會監控負載平衡器，並等待負載平衡器報告持續連線已關閉 (目標處於 UNUSED 狀態)。

負載平衡器等待將目標移至 UNUSED 狀態的時間即為取消註冊延遲。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)

如果服務回應時間低於 1 秒，請將參數設定為下列值，使負載平衡器僅等待 5 秒就中斷用戶端與後端服務之間的連線：

- deregistration_delay.timeout_seconds : 5

 Note

如果服務涉及長效請求(例如慢速檔案上傳或串流連線)，請勿將此值設定為 5 秒。

SIGTERM 回應能力

Amazon ECS 會先傳送停止訊號給任務，以通知應用程式需要完成並關閉。此訊號可以使用 `STOPSIGNAL` 指令在您的容器映像中定義，並預設為 `SIGTERM`。然後，Amazon ECS 會傳送 `SIGKILL` 訊息。當應用程式忽略 `SIGTERM` 時，Amazon ECS 服務必須等待一段時間才能傳送 `SIGKILL` 訊號來終止程序。

Amazon ECS 等待傳送 `SIGKILL` 訊息的時間取決於下列 Amazon ECS 代理程式選項：

- `ECS_CONTAINER_STOP_TIMEOUT` : 30 (預設值)

如需有關容器代理程式參數的詳細資訊，請參閱 GitHub 上的 [Amazon ECS Container Agent](#)。

若要縮短等待期間，請將 Amazon ECS 代理程式參數設定為下列值：

- `ECS_CONTAINER_STOP_TIMEOUT`: 2

如果應用程式需要超過 1 秒的時間，請將該值乘以 2，並將該數字用作設定值。

在這種情況下，Amazon ECS 會等待 2 秒讓容器關閉，如果應用程式沒有停止，Amazon ECS 會傳送 `SIGKILL` 訊息。

您也可以修改應用程式程式碼來捕捉 `SIGTERM` 訊號並對其作出反應。以下是 JavaScript 中的範例：

```
process.on('SIGTERM', function() {  
  server.close();  
})
```

此程式碼會導致 HTTP 伺服器停止監聽任何新請求、完成回應所有傳輸中請求，然後 Node.js 程序會終止，因為事件迴圈沒有執行任何動作。有鑑於此，如果程序僅需 500 毫秒就能完成其傳輸中請求，它就會提早終止，而無需等待停止逾時並收到 `SIGKILL`。

搭配 Amazon ECS 使用 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。任務啟動時，NGINX 容器會以執行個體 ID 與連接埠組合向 Application Load Balancer 註冊，而流量將分佈到與該容器對應的執行個體 ID 與連接埠。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](#)。

如需有關設定參數以加速部署的最佳實務資訊，請參閱：

- [針對 Amazon ECS 最佳化負載平衡器運作狀態檢查參數](#)
- [針對 Amazon ECS 最佳化負載平衡器連接耗盡參數](#)

搭配 Amazon ECS 使用 Application Load Balancer 時，請考量下列事項：

- 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](#)。
- 對於純 IPv6 組態的服務，必須將 Application Load Balancer 的目標群組 IP 位址類型設定為 `dualstack` 或 `dualstack-without-public-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。
- 如果服務需要存取多個負載平衡連接埠 (例如適用於 HTTP/HTTPS 服務的連接埠 80 與連接埠 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 管理主控台變更。您可以使用 AWS Copilot AWS CloudFormation AWS CLI 或 SDK，僅修改 ECS 滾動部署控制器的負載平衡器組態，而不是 AWS CodeDeploy 藍/綠或外部。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。如需有關如何修改組態的資訊，請參閱 Amazon Elastic Container Service API 參考中的 [UpdateService](#)。

- 如果服務任務未能通過負載平衡器運作狀態檢查條件，則會停止並重新啟動任務。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[對 Amazon ECS 中的服務負載平衡器進行疑難排解](#)。
- 使用 instance 目標類型時，任務與負載平衡器必須位於同一 VPC 中。使用 ip 目標類型時，支援跨 VPC 連線。
- 針對每項服務使用唯一目標群組。

針對多項服務使用同一目標群組可能會在服務部署期間導致問題。

- 您必須指定與 Application Load Balancer 相關聯的目標群組。

如需有關如何建立 Application Load Balancer 的資訊，請參閱 Application Load Balancers 中的 [Create an Application Load Balancer](#)

搭配 Amazon ECS 使用 Network Load Balancer

Network Load Balancer 會在傳輸層做出路由決策 (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](#)。

如需有關設定參數以加速部署的最佳實務資訊，請參閱：

- [針對 Amazon ECS 最佳化負載平衡器運作狀態檢查參數](#)
- [針對 Amazon ECS 最佳化負載平衡器連接耗盡參數](#)

搭配 Amazon ECS 使用 Network Load Balancer 時，請考量下列事項：

- 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](#)。
- 一項服務最多只能連接五個目標群組。

- 對於純 IPv6 組態的服務，必須將 Network Load Balancer 的目標群組 IP 位址類型設定為 `dualstack`。
- 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。
- 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 管理主控台變更。您可以使用 AWS Copilot AWS CloudFormation AWS CLI 或 SDK，僅修改 ECS 滾動部署控制器的負載平衡器組態，而不是 AWS CodeDeploy 藍/綠或外部。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。如需有關如何修改組態的資訊，請參閱 Amazon Elastic Container Service API 參考中的 [UpdateService](#)。
- 如果服務任務未能通過負載平衡器運作狀態檢查條件，則會停止並重新啟動任務。此程序會持續到您的服務達到所需的執行任務數量為止。
- 當您使用以 IP 位址作為目標設定的且關閉了用戶端 IP 保留功能的 Gateway Load Balancer 時，請求會視為來自 Gateway Load Balancer 私有 IP 位址。也就是說，當您在目標安全群組中允許傳入請求與運作狀態檢查時，Gateway Load Balancer 後端的服務就會立即對外開放。
- 對於 Fargate 任務，必須使用平台版本 1.4.0 (Linux) 或 1.0.0 (Windows)。
- 如果您遇上啟用負載平衡器功能之服務的問題，請參閱[對 Amazon ECS 中的服務負載平衡器進行疑難排解](#)。
- 使用 `instance` 目標類型時，任務與負載平衡器必須位於同一 VPC 中。使用 `ip` 目標類型時，支援跨 VPC 連線。
- Network Load Balancer 用戶端 IP 位址保留與 Fargate 目標相容。
- 針對每項服務使用唯一目標群組。

針對多項服務使用同一目標群組可能會在服務部署期間導致問題。

- 您必須指定與 Network Load Balancer 相關聯的目標群組。

如需有關如何建立 Network Load Balancer 的資訊，請參閱 Network Load Balancers 中的 [Create a Network Load Balancer](#)

⚠ Important

若服務的任務定義採用 Fargate 所需的 awsvpc 網路模式，則您必須選擇 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。

如果執行個體類型如下：

C1、CC1、CC2、CG1、CG2、CR1、G1、G2、H1、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。

搭配 Amazon ECS 使用 Gateway 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。任務啟動時，NGINX 容器會以執行個體 ID 與連接埠組合向 Gateway Load Balancer 註冊，而流量將分佈到與該容器對應的執行個體 ID 與連接埠。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 Gateway Load Balancers 中的 [What is a Gateway Load Balancer](#)。

如需有關設定參數以加速部署的最佳實務資訊，請參閱：

- [針對 Amazon ECS 最佳化負載平衡器運作狀態檢查參數](#)
- [針對 Amazon ECS 最佳化負載平衡器連接耗盡參數](#)

搭配 Amazon ECS 使用 Gateway Load Balancer 時，請考量下列事項：

- 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](#)。
- 對於純 IPv6 組態的服務，必須將 Gateway Load Balancer 的目標群組 IP 位址類型設定為 dualstack。
- 對於任務使用 awsvpc 以外網路模式的服務，不支援 Gateway Load Balancer。
- 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 管理主控台變更。您可以使用 AWS Copilot AWS CloudFormation AWS CLI 或 SDK，僅修改 ECS 滾動部署控制器的負載平衡器組態，而不是 AWS

CodeDeploy 藍/綠或外部。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。如需有關如何修改組態的資訊，請參閱 Amazon Elastic Container Service API 參考 中的 [UpdateService](#)。

- 如果服務任務未能通過負載平衡器運作狀態檢查條件，則會停止並重新啟動任務。此程序會持續到您的服務達到所需的執行任務數量為止。
- 當您使用以 IP 位址作為目標設定的 Gateway Load Balancer 時，請求會視為來自 Gateway Load Balancer 私有 IP 位址。也就是說，當您在目標安全群組中允許傳入請求與運作狀態檢查時，Gateway Load Balancer 後端的服務就會立即對外開放。
- 對於 Fargate 任務，必須使用平台版本 1.4.0 (Linux) 或 1.0.0 (Windows)。
- 如果您遇上啟用負載平衡器功能之服務的問題，請參閱[對 Amazon ECS 中的服務負載平衡器進行疑難排解](#)。
- 使用 instance 目標類型時，任務與負載平衡器必須位於同一 VPC 中。使用 ip 目標類型時，支援跨 VPC 連線。
- 針對每項服務使用唯一目標群組。

針對多項服務使用同一目標群組可能會在服務部署期間導致問題。

- 您必須指定與 Gateway Load Balancer 相關聯的目標群組。

如需有關如何建立 Gateway Load Balancer 的資訊，請參閱 Gateway Load Balancers 中的 [Getting started with Gateway Load Balancers](#)

Important

若服務的任務定義採用 Fargate 所需的 awsvpc 網路模式，則您必須選擇 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。

透過 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、SDK 或 CloudFormation 範本建立服務。建立服務之後，您可以使用 AWS 管理主控台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 (ECS) 部署控制器類型。這可以是 Amazon ECS 原生/藍綠部署或滾動更新部署。
- 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 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](#)。

具有可供內部與外部流量使用的個別負載平衡器

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
  }
]
```

公開相同容器中的多個連接埠

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  
}  
]
```

公開多個容器中的連接埠

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 服務中所需任務數量的功能。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 會發佈具有您的服務平均 CPU 和記憶體用量的 CloudWatch 指標。如需詳細資訊，請參閱 [Amazon ECS 服務使用率指標](#)。您可以使用這些指標和其他 CloudWatch 指標來向外擴展服務 (新增更多任務) 來處理尖峰時段的高需求量，以及將服務向內縮減 (執行較少任務) 以減少低使用率期間的成本。

Amazon ECS Service Auto Scaling 支援下列自動擴展類型：

- [使用目標指標擴展 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 CloudWatch 警示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。
- [使用排程動作來擴展 Amazon ECS 服務](#)：根據日期與時間增加或減少服務執行的任務數量。
- [使用歷史模式透過預測擴展來擴展 Amazon ECS 服務](#)：根據歷史負載資料分析增加或減少服務執行的任務數量，以偵測每日或每週流量流程模式。

考量事項

使用擴展策略時，考慮下列事項：

- Amazon ECS 每隔 1 分鐘會將指標傳送至 CloudWatch。在叢集和服務將指標傳送至 CloudWatch 前，指標皆無法使用，而且您也無法建立不存在之指標的 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。
- 如果設定的服務所需計數低於其容量值下限，且警示啟動橫向擴充活動，服務自動擴展功能會將所需計數擴展到容量值下限，然後視需要根據與警示相關聯的擴展政策來持續橫向擴充。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。
- 如果設定的服務所需計數高於其容量值上限，且警示啟動縮減活動，服務自動擴展功能會將所需計數橫向擴充到容量值上限，然後視需要根據與警示相關聯的擴展政策來持續縮減。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。此行為不適用於使用外部部署控制器的 Amazon ECS 服務。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。

如需有關服務自動擴展最佳實務的詳細資訊，請參閱[最佳化 Amazon ECS 服務自動擴展](#)。

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。此行為不適用於使用外部部署控制器的 Amazon ECS 服務。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](#)。

使用目標指標擴展 Amazon ECS 服務

使用目標追蹤擴展政策，您可以選取指標並設定目標值。Amazon ECS Service Auto Scaling 會建立和管理可控制擴展政策的 CloudWatch 警示，並根據指標和目標值來計算擴展調整。擴展政策會視需要新增或移除服務任務，讓指標保持在或接近指定的目標值。除了將指標保持在接近目標值之外，目標追蹤擴展政策也會根據由於負載模式波動而導致的指標波動進行調整，並將服務中執行的任務數量快速波動降至最低。

目標追蹤政策無需手動定義 CloudWatch 警示和擴展調整內容，Amazon ECS 會根據您設定的目標自動處理此問題。

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。此行為不適用於使用外部部署控制器的 Amazon ECS 服務。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 CloudWatch 警示，Service Auto Scaling 管理它們用於目標追蹤擴展政策。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](#)。

為 Amazon ECS 服務自動擴展功能建立目標追蹤擴展政策

建立目標追蹤擴展政策，以便 Amazon ECS 自動增加或減少服務中所需的任務計數。目標追蹤根據目標指標值運作。

主控台

1. 除了建立與更新服務所需的標準 IAM 許可外，您還需要其他許可。如需詳細資訊，請參閱[Amazon ECS 服務自動擴展所需的 IAM 許可](#)。
2. 決定用於政策的指標。下列指標可供使用：

- ECSServiceAverageCPUUtilization – 服務應使用的平均 CPU 使用率。
- ECSServiceAverageMemoryUtilization – 服務應使用的平均記憶體使用率。
- ALBRequestCountPerTarget – 任務理想情況下，任務每分鐘應接收的平均請求數。

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

4. 在叢集頁面上，選擇叢集。

5. 在「叢集詳細資訊」頁面上的服務區段中選擇服務。

服務資訊頁面隨即顯示。

6. 選擇設定任務數量。

7. 在 Amazon ECS 服務任務計數下，選擇使用自動擴展。

任務計數區段隨即顯示。

- a. 在任務數量下限欄位中，輸入供服務自動擴展功能使用的任務數量下限。所需的計數不會低於此計數。
- b. 在上限欄位中，輸入供服務自動擴展功能使用的任務數量上限。所需的計數不會高於此計數。
- c. 選擇儲存。

政策頁面隨即顯示。

8. 選擇建立擴展政策。

建立政策頁面隨即顯示。

9. 針對 Scaling policy type (擴展政策類型)，選擇 Target tracking (目標追蹤)。

10. 針對 Policy name (政策名稱)，輸入政策的名稱。

11. 在指標類型欄位中，從選項清單中選擇指標。

12. 在目標使用率欄位中，輸入 Amazon ECS 應維護之任務的百分比目標值。服務自動擴展功能會橫向擴充容量，直到平均使用率達到目標使用率，或達到所指定的任務數量上限。

13. 在其他設定下，執行下列動作

- a. 在縮減冷卻時間欄位中，輸入一個縮減活動完成之後、另一個縮減活動開始之前的時間 (以秒為單位)。
- b. 在橫向擴充冷卻時間欄位中，輸入等待上一個橫向擴充活動生效的時間 (以秒為單位)。
- c. 若要僅建立橫向擴充政策，請選取停用縮減。

14. 選擇建立擴展政策。

AWS CLI

1. 使用 [register-scalable-target](#) 命令，將 Amazon ECS 服務註冊為可擴展性目標。
2. 使用 [put-scaling-policy](#) 命令，來建立擴展政策。

根據 CloudWatch 警示使用預先定義的增量來擴展 Amazon ECS 服務

如果使用步進擴展政策，您可以建立及管理觸發擴展程序的 CloudWatch 警示。違反警示時，Amazon ECS 會啟動與該警示相關聯的擴展政策。步進擴展政策會根據一組調整 (稱為步進調整) 來擴展任務。調整幅度會依流量達到警示閾值的程度而有所不同。

- 如果違規程度超過第一個閾值，Amazon ECS 會套用第一個步進調整。
- 如果違規程度超過第二個閾值，Amazon ECS 會套用第二個步進調整，以此類推。

強烈建議使用目標追蹤擴展政策，根據平均 CPU 使用率或每個目標的平均請求計數等指標進行擴展。容量增加時減少的指標與容量減少時增加的指標，可用於使用目標追蹤依比例橫向擴充或縮減任務數量。這有助於確保 Amazon ECS 緊密地遵循應用程式的需求曲線。

為 Amazon ECS 服務自動擴展建立步進擴展政策

建立步進擴展政策，以便 Amazon ECS 自動增加或減少服務中所需的任務數量。步進擴展根據一組擴展調整 (稱為步進調整) 來執行，這些調整會依警示違規程度而變動。

主控台

1. 除了建立與更新服務所需的標準 IAM 許可外，您還需要其他許可。如需詳細資訊，請參閱[Amazon ECS 服務自動擴展所需的 IAM 許可](#)。
2. 決定用於政策的指標。下列指標可供使用：
 - ECSServiceAverageCPUUtilization – 服務應使用的平均 CPU 使用率。
 - ECSServiceAverageMemoryUtilization – 服務應使用的平均記憶體使用率。
 - ALBRequestCountPerTarget – 任務理想情況下，任務每分鐘應接收的平均請求數。
3. 為指標建立 CloudWatch 警示。如需詳細資訊，請參閱 Amazon CloudWatch 使用者指南中的[建立以靜態閾值為基礎的 CloudWatch 警示](#)。
4. 開啟主控台，網址為 <https://console.aws.amazon.com/ecs/v2>。
5. 在叢集頁面上，選擇叢集。

6. 在「叢集詳細資訊」頁面上的服務區段中選擇服務。

服務資訊頁面隨即顯示。

7. 選擇設定任務數量。
8. 在 Amazon ECS 服務任務計數下，選擇使用自動擴展。

任務計數區段隨即顯示。

- a. 在任務數量下限欄位中，輸入供服務自動擴展功能使用的任務數量下限。所需的計數不會低於此計數。
- b. 在上限欄位中，輸入供服務自動擴展功能使用的任務數量上限。所需的計數不會高於此計數。
- c. 選擇儲存。

政策頁面隨即顯示。

9. 選擇建立擴展政策。

建立政策頁面隨即顯示。

10. 在擴展政策類型欄位中選擇步進擴展。
11. 設定橫向擴充屬性。在新增任務的步驟下，執行下列動作：

- a. 針對 Policy name (政策名稱)，輸入政策的名稱。
- b. 在 CloudWatch 警示名稱欄位中選擇 CloudWatch 警示。
- c. 在指標彙總類型欄位中，選擇如何將所選指標與定義的閾值進行比較。
- d. 在調整類型欄位中，選擇是根據任務數量的變更進行調整，還是根據任務百分比的變更進行調整。
- e. 在要採取的動作欄位中，輸入要採取的動作值。

選擇新增步驟以新增其他動作。

12. 設定縮減屬性。在移除任務的步驟下，執行下列動作：

- a. 針對 Policy name (政策名稱)，輸入政策的名稱。
- b. 在 CloudWatch 警示名稱欄位中選擇 CloudWatch 警示。
- c. 在指標彙總類型欄位中，選擇如何將所選指標與定義的閾值進行比較。
- d. 在調整類型欄位中，選擇是根據任務數量的變更進行調整，還是根據任務百分比的變更進行調整。
- e. 在要採取的動作欄位中，輸入要採取的動作值。

選擇新增步驟以新增其他動作。

13. 在冷卻時間欄位中，輸入等待上一個擴展活動生效的時間 (以秒為單位)。對於新增政策，這是橫向擴充活動結束之後，擴展政策封鎖縮減活動並限制一次可以橫向擴充多少項任務的時間。對於移除政策，這是一個縮減活動結束之後、另一個縮減活動開始之前必須經過的時間。
14. 選擇建立擴展政策。

AWS CLI

1. 使用 [register-scalable-target](#) 命令，將 Amazon ECS 服務註冊為可擴展性目標。
2. 使用 [put-scaling-policy](#) 命令，來建立擴展政策。

使用排程動作來擴展 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任務數量，針對應用程式設定自動擴展。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。應用程式將擁有足夠數量的任務來處理週間的流量高峰，但在其他時段不會過度佈建任務數量。

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任務數量。這有助於確保您擁有足夠數量的任務來處理應用程式負載。當應用程式擴展以滿足需求時，目前的容量必須處於排程動作所設定的任務數量上下限之內。

可以使用 AWS CLI 設定排程擴展。如需有關排程擴展的詳細資訊，請參閱 Application Auto Scaling User Guide 中的 [Scheduled Scaling](#)。

為 Amazon ECS 服務自動擴展功能建立排程動作

您建立排程動作時，Amazon ECS 會根據日期與時間增加或減少服務執行的任務數量。

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「叢集詳細資訊」頁面上的服務區段中，選擇所需服務。

服務資訊頁面隨即顯示。

4. 選擇服務自動擴展。

服務自動擴展頁面隨即顯示。

5. 如果您尚未設定服務自動擴展功能，請選擇設定任務數量。

Amazon ECS 服務任務計數區段隨即顯示。

在 Amazon ECS 服務任務計數下，選擇使用服務自動擴展來調整服務所需的任務計數。

任務計數區段隨即顯示。

- a. 在任務數量下限欄位中，輸入供服務自動擴展功能使用的任務數量下限。所需的計數不會低於此計數。
- b. 在上限欄位中，輸入供服務自動擴展功能使用的任務數量上限。所需的計數不會高於此計數。
- c. 選擇儲存。

政策頁面隨即顯示。

6. 選擇排程動作，然後選擇建立。

建立排程動作頁面隨即顯示。

7. 在名稱欄位中輸入唯一的名稱。

8. 對於 Time zone (時區)，選擇時區。

所有列出的時區都來自 IANA 時區資料庫。如需詳細資訊，請參閱 [List of tz database time zones](#)。

9. 在開始時間欄位中，輸入動作開始的日期與時間。

如果選擇週期性排程，開始時間會定義週期性序列中第一個排程作業的執行時間。

10. 針對 Recurrence (週期)，選擇其中一個可用選項。

- 若要依週期性排程進行擴展，請選擇 Amazon ECS 執行排程動作的頻率。
 - 如果選擇以速率開始的選項，則會為您建立 Cron 表達式。
 - 如果選擇 Cron，請輸入指定何時執行動作的 Cron 表達式。
- 若僅擴展一次，請選擇一次。

11. 在任務調整下，執行下列動作：

- 在下限欄位中，輸入服務應執行的任務數量下限。
- 在上限欄位中，輸入服務應執行的任務數量上限。

12. 選擇建立排程動作。

CLI

使用 設定服務的排程擴展政策 AWS CLI ，如下所示。將每個#####替換為自己的資訊。

範例：僅擴展一次

搭配 `--start-time "YYYY-MM-DDThh:mm:ssZ"` 以及 `--MinCapacity` 與 `--MaxCapacity` 選項之一或兩者，使用下列 [put-scheduled-action](#) 命令。

```
aws application-autoscaling put-scheduled-action --service-namespace ecs \
  --resource-id service/my-cluster/my-service \
  --scheduled-action-name my-one-time-schedule \
  --start-time 2021-01-30T12:00:00 \
  --scalable-target-action MinCapacity=3,MaxCapacity=10
```

範例：依據週期性排程排定擴展

使用下列 [put-scheduled-action](#) 命令。將 *user input* 取代為實際值。

```
aws application-autoscaling put-scheduled-action --service-namespace ecs \
  --resource-id service/my-cluster/my-service \
  --scheduled-action-name my-recurring-action \
  --schedule "rate(5 hours)" \
  --start-time 2021-01-30T12:00:00 \
  --end-time 2021-01-31T22:00:00 \
  --scalable-target-action MinCapacity=3,MaxCapacity=10
```

指定的週期性排程會根據 UTC 時區執行。若要指定不同的時區，請包含 `--time-zone` 選項與 IANA 時區的名稱，如下列範例所示。

```
--time-zone "America/New_York"
```

如需詳細資訊，請參閱 [List of tz database time zones](#)。

使用歷史模式透過預測擴展來擴展 Amazon ECS 服務

預測性擴展會檢視過去從流量流程載入的資料，以分析每日或每週模式。然後利用此分析來預測未來需求，並根據需要主動增加服務中的任務。

預測性自動擴展在下列情況中最为有用。

- 周期性流量 – 在常規工作時間資源使用量增加，而在夜間與週末資源使用量減少。
- 重複間歇工作負載模式 – 例如批次處理、測試或週期性資料分析。
- 需要長時間才能初始化的應用程式 – 這可能會在橫向擴充事件期間影響應用程式效能，造成明顯延遲。

如果應用程式需要長時間才能初始化，並且流量以常規模式增加，您應該考慮使用預測性擴展。此功能透過為預測負載主動增加任務數量來協助您更快地擴展，而不是僅使用動態擴展政策，例如目標追蹤或步進擴展。透過協助您避免出現過度佈建任務數量的可能性，預測性擴展也可能會為您節省成本。

例如，考量應用程式在營業時間內具有高使用率而在夜間具有低使用率。在每個工作日開始時，預測性擴展可以在第一次流量湧入之前橫向擴充任務。在從較低的使用率期間到較高的使用率期間時，這可協助您的應用程式維持高可用性和效能。您不必等待動態擴展來對不斷變化的流量做出反應。您也無需花時間檢閱應用程式的負載模式並嘗試使用排程擴展來排程適量的任務。

預測性擴展是一項服務層級功能，可獨立於基礎運算容量 (例如 EC2 或 Fargate) 的擴展來擴展服務任務。對於 Fargate，會根據任務需求 AWS 管理和自動擴展基礎容量。對於 EC2 容量，您可以使用 Auto Scaling 群組容量提供者，根據任務的擴展需求自動擴展基礎 EC2 執行個體。

目錄

- [預測性擴展在 Amazon ECS 中的運作方式](#)
- [為 Amazon ECS 服務自動擴展建立預測性擴展政策](#)
- [為 Amazon ECS 評估預測性擴展政策](#)
- [使用排程動作覆寫 Amazon ECS 的預測值](#)
- [使用自訂指標的 Amazon ECS 進階預測性擴展政策](#)

預測性擴展在 Amazon ECS 中的運作方式

您可以在此處了解使用預測性擴展的考量、其運作方式以及相關限制。

使用預測性擴展的考量

- 您需要確保預測性擴展適合工作負載。可以透過在僅預測模式下設定擴展政策對此進行檢查，並查看主控台的建議。在開始使用預測性擴展之前，您應該先評估預測與建議。
- 預測性擴展需要至少 24 小時的歷史資料才能開始預測。可用的歷史資料越多，預測效果越好，兩週的歷史資料最為理想。您刪除 Amazon ECS 服務並建立新服務時，還需要等待 24 小時，預測性擴

展才能產生新的預測。加快此程序的一種方法是使用自訂指標，以彙總新舊 Amazon ECS 服務的指標。

- 選擇能準確代表應用程式完整負載，且是應用程式最重要擴展方面的負載指標。
- 結合預測性擴展的動態擴展可協助您密切遵循應用程式的需求，因此您可以在低落期間進行縮減，並在流量意外增加期間進行橫向擴充。當多個擴展政策處於作用中狀態時，每個政策會獨立決定所需的任務數量，並將所需任務數量設定為其中的上限。
- 您可以使用預測性擴展搭配動態擴展政策，例如目標追蹤或步進擴展，以便應用程式根據即時模式與歷史模式進行擴展。預測性擴展本身不會縮減任務。
- 如果您在呼叫 `register-scalable-target` API 時使用自訂角色，可能會收到錯誤，指出預測性擴展政策只能在啟用 SLR 的情況下運作。在這種情況下，您應該再次呼叫 `register-scalable-target`，但不使用 `role-arn`。在註冊可擴展的目標時，請使用 SLR，並呼叫 `put-scaling-policy` API。

預測擴展的運作方式

您可以透過建立指定要監控與分析之 CloudWatch 指標的預測性擴展政策，來使用預測性擴展。預測性擴展需要至少 24 小時的資料，才能開始預測未來的值。

建立政策後，預測性擴展會開始分析過去最多 14 天內的指標資料，以識別模式。此分析用於產生接下來 48 小時每小時的需求預測。最新的 CloudWatch 資料用於每六小時更新一次預測。隨著新資料傳入，預測性擴展會不斷提高未來預測的準確性。

當您首次啟用預測性擴展時，它會以僅預測模式執行。它會在此模式下產生預測，但不會根據這些預測擴展 Amazon ECS 服務。這表示您可以評估預測的準確性與適用性。您可以透過使用 `GetPredictiveScalingForecast` API 操作或 AWS 管理主控台來檢視預測資料。

您決定開始使用預測性擴展時，請將擴展政策切換至預測與擴展模式。在此模式下會發生下列情況。

依預設，Amazon ECS 服務會在每小時開始時根據該小時的預測進行擴展。您可以透過在 `PutScalingPolicy` API 操作中使用 `SchedulingBufferTime` 屬性，選擇提早開始。這使得新任務在預測需求之前啟動，並給予其時間開機並準備好處理流量。

任務數量上限

您註冊 Amazon ECS 服務進行擴展時，需要定義每項服務可以啟動的任務數量上限。依預設，當設定擴展政策時，它們無法增加高於其上限的任務數量。

或者，如果預測接近或超過 Amazon ECS 服務的任務數量上限，您可以允許自動增加服務的任務數量上限。

⚠ Warning

允許自動增加任務數量上限時請小心謹慎。如果沒有監控與管理增加的任務數量上限，這可能會導致啟動的任務數量超出預期。增加的任務數量上限會成為 Amazon ECS 服務新的常規任務數量上限，直到您進行手動更新。任務數量上限不會自動減少回原始上限。

支援的 區域

- 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)
- 非洲 (開普敦)
- 亞太地區 (香港)
- 亞太地區 (雅加達)
- 亞太地區 (孟買)
- 亞太區域 (大阪)
- 亞太區域 (首爾)
- 亞太區域 (新加坡)
- 亞太地區 (雪梨)
- 亞太區域 (東京)
- 加拿大 (中部)
- 中國 (北京)
- 中國 (寧夏)
- 歐洲 (法蘭克福)
- 歐洲 (愛爾蘭)
- 歐洲 (倫敦)
- 歐洲 (米蘭)
- 歐洲 (巴黎)
- 歐洲 (斯德哥爾摩)
- 中東 (巴林)

- 南美洲 (聖保羅)
- AWS GovCloud (美國東部)
- AWS GovCloud (美國西部)

為 Amazon ECS 服務自動擴展建立預測性擴展政策

建立預測性擴展政策，以便 Amazon ECS 根據歷史資料增加或減少服務執行的任務數量。

 Note

新服務需要提供至少 24 小時的資料，才能產生預測結果。

主控台

1. 除了建立與更新服務所需的標準 IAM 許可外，您還需要其他許可。如需詳細資訊，請參閱[Amazon ECS 服務自動擴展所需的 IAM 許可](#)。
2. 決定用於政策的指標。下列指標可供使用：
 - ECSServiceAverageCPUUtilization – 服務應使用的平均 CPU 使用率。
 - ECSServiceAverageMemoryUtilization – 服務應使用的平均記憶體使用率。
 - ALBRequestCountPerTarget – 任務理想情況下，任務每分鐘應接收的平均請求數。

您也可以使用自訂指標。需要定義下列值：

- 負載 – 可準確代表應用程式完整負載，且是應用程式最重要擴展方面的指標。
 - 擴展指標 – 最能預測多少使用率最適合應用程式的指標。
3. 開啟主控台，網址為 <https://console.aws.amazon.com/ecs/v2>。
 4. 在叢集頁面上，選擇叢集。
 5. 在「叢集詳細資訊」頁面上的服務區段中，選擇所需服務。

服務資訊頁面隨即顯示。

6. 選擇服務自動擴展，然後選擇設定任務數量。
7. 在 Amazon ECS 服務任務計數下，選擇使用自動擴展。

任務計數區段隨即顯示。

- a. 在任務數量下限欄位中，輸入供服務自動擴展功能使用的任務數量下限。所需的計數不會低於此計數。
- b. 在上限欄位中，輸入供服務自動擴展功能使用的任務數量上限。所需的計數不會高於此計數。
- c. 選擇儲存。

政策頁面隨即顯示。

8. 選擇建立擴展政策。

建立政策頁面隨即顯示。

9. 在擴展政策類型欄位中選擇預測性擴展。
10. 針對 Policy name (政策名稱)，輸入政策的名稱。
11. 在指標對欄位中，從選項清單中選擇指標。

如果選擇了 Application Load Balancer request count per target (每個目標的 Application Load Balancer 請求計數)，則在 Target group (目標群組) 中選擇目標群組。只有將 Application Load Balancer 目標群組連接至服務後，才支援每個目標的 Application Load Balancer 請求計數。

如果選擇了自訂指標對，則從負載指標與擴展指標清單中選擇個別指標。

12. 在目標使用率欄位中，輸入 Amazon ECS 應維護之任務的百分比目標值。服務自動擴展功能會橫向擴充容量，直到平均使用率達到目標使用率，或達到所指定的任務數量上限。
13. 選擇建立擴展政策。

AWS CLI

使用 AWS CLI 設定 Amazon ECS 服務的預測擴展政策，如下所示。將每個#####替換為自己的資訊。

如需有關可指定之 CloudWatch 指標的詳細資訊，請參閱 Amazon EC2 Auto Scaling API Reference 中的 [PredictiveScalingMetricSpecification](#)。

範例 1：具有預先定義記憶體體的預測性擴展政策。

以下是具有預先定義記憶體組態的政策範例。

```
cat policy.json
{
```

```

    "MetricSpecifications": [
      {
        "TargetValue": 40,
        "PredefinedMetricPairSpecification": {
          "PredefinedMetricType": "ECSServiceMemoryUtilization"
        }
      }
    ],
    "SchedulingBufferTime": 3600,
    "MaxCapacityBreachBehavior": "HonorMaxCapacity",
    "Mode": "ForecastOnly"
  }

```

下列範例示範了透過執行帶有指定組態檔案的 [put-scaling-policy](#) 命令來建立政策。

```

aws application-autoscaling put-scaling-policy \
--service-namespace ecs \
--region us-east-1 \
--policy-name predictive-scaling-policy-example \
--resource-id service/MyCluster/test \
--policy-type PredictiveScaling \
--scalable-dimension ecs:service:DesiredCount \
--predictive-scaling-policy-configuration file://policy.json

```

如果成功，此命令會傳回政策的 ARN。

```

{
  "PolicyARN": "arn:aws:autoscaling:us-
east-1:012345678912:scalingPolicy:d1d72dfe-5fd3-464f-83cf-824f16cb88b7:resource/ecs/
service/MyCluster/test:policyName/predictive-scaling-policy-example",
  "Alarms": []
}

```

範例 2：具有預先定義 CPU 的預測性擴展政策。

以下是具有預先定義 CPU 組態的政策範例。

```

cat policy.json
{
  "MetricSpecifications": [
    {
      "TargetValue": 0.00000004,
      "PredefinedMetricPairSpecification": {

```



```

        "PredefinedMetricType": "ECSServiceCPUUtilization"
    }
}
],
"SchedulingBufferTime": 3600,
"MaxCapacityBreachBehavior": "HonorMaxCapacity",
"Mode": "ForecastOnly"
}

```

下列範例示範了透過執行帶有指定組態檔案的 [put-scaling-policy](#) 命令來建立政策。

```

aws aas put-scaling-policy \
--service-namespace ecs \
--region us-east-1 \
--policy-name predictive-scaling-policy-example \
--resource-id service/MyCluster/test \
--policy-type PredictiveScaling \
--scalable-dimension ecs:service:DesiredCount \
--predictive-scaling-policy-configuration file://policy.json

```

如果成功，此命令會傳回政策的 ARN。

```

{
  "PolicyARN": "arn:aws:autoscaling:us-east-1:012345678912:scalingPolicy:d1d72dfe-5fd3-464f-83cf-824f16cb88b7:resource/ecs/service/MyCluster/test:policyName/predictive-scaling-policy-example",
  "Alarms": []
}

```

為 Amazon ECS 評估預測性擴展政策

使用預測性擴展政策來擴展服務之前，請先在 Amazon ECS 主控台中檢閱建議及政策的其他資料。這很重要，因為您不希望預測擴展政策在您知道其預測準確之前擴展實際容量。

如果是新服務，需要 24 小時才能建立第一個預測。

AWS 建立預測時，會使用歷史資料。如果服務尚無足夠的最新歷史資料，預測性擴展可能會使用從目前可用的歷史彙總資料建立的彙總資料來暫時回填預測。預測會在政策建立日期前的兩週內回填。

檢視您的預測擴展建議

為了獲得有效的分析，服務自動擴展功能應具有至少兩個可進行比較的預測性擴展政策。(不過，您仍然可以檢閱單一政策的問題清單。) 建立多個政策時，您可以根據使用不同指標的政策，評估使用一個

指標的政策。您也可以評估不同目標值和指標組合的影響。建立預測性擴展政策之後，Amazon ECS 會立即開始評估哪些政策可以更好地擴展群組。

在 Amazon ECS 主控台中檢視建議

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「叢集詳細資訊」頁面上的服務區段中，選擇所需服務。

服務資訊頁面隨即顯示。

4. 選擇服務自動擴展。
5. 選擇預測性擴展政策，然後選擇動作、預測性擴展、檢視建議。

您可以檢視政策的詳細資訊以及我們的建議。該建議會告訴您使用預測擴展政策的結果是否優於不使用它。

如果您不確定預測擴展政策是否適合您的群組，請檢閱可用性影響和成本影響欄，以選擇正確的政策。每一欄的資訊都會說明政策的影響。

- 可用性影響：說明政策是否會佈建足夠的任務來處理工作負載，以避免對可用性造成負面影響 (相較於不使用該政策)。
- 成本影響：說明政策是否不會過度佈建任務，以避免對成本造成負面影響 (相較於不使用該政策)。如果過度佈建比較嚴重，服務就會出現使用率過低或閒置情況，這只會增加成本影響。

如果您有多個政策，則以較低成本提供最多可用性優勢的政策名稱旁會顯示最佳預測標籤。可用性影響會獲得更多加權。

6. (選用) 若要選取建議結果所需的期間，請從評估期間下拉式清單中選擇您偏好的值：2 天、1 週或 2 週。根據預設，評估期間是最近兩週。較長的評估期間會為建議結果提供更多資料點。不過，如果負載模式發生變更 (例如在一段異常需求期間後)，新增更多資料點可能無法改善結果。在這種情況下，您可以查看最新資料以獲得更有針對性的建議。

Note

只會針對處於僅預測模式的政策產生建議。當政策在整個評估期間都處於僅預測模式時，建議功能的效果會更好。如果您在預測和擴展模式中啟動政策，並於稍後將其切換至僅預測模式，則該政策的問題清單可能會有偏差。這是因為該政策已經為實際容量做出了貢獻。

檢閱預測擴展監控圖表

在主控台中，您可以檢閱前幾天、前幾週或前幾個月的預測，以視覺化方式呈現政策在一段時間內的表現。在決定是否允許政策擴展實際任務數量時，您也可以使用這些資訊來評估預測的準確性。

在 Amazon ECS 主控台中檢閱預測性擴展監控圖表

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「叢集詳細資訊」頁面上的服務區段中，選擇所需服務。

服務資訊頁面隨即顯示。

4. 選擇服務自動擴展。
5. 選擇預測性擴展政策，然後選擇動作、預測性擴展、檢視圖表。
6. 在監控區段中，您可以根據實際值檢視政策在過去和未來的負載和容量預測。負載圖表會顯示所選負載指標的負載預測與實際值。容量圖表會顯示政策預測的任務數量。它還包括實際啟動的任務數量。垂直線會將歷史值與未來預測隔開。建立政策後，這些圖表很快就可以使用。
7. (選用) 若要變更圖表中顯示的歷史資料量，請從頁面頂端的評估期間下拉式清單中選擇您偏好的值。評估期間不會以任何方式轉換此頁面上的資料。它只會變更顯示的歷史資料量。

比較負載圖表中的資料

每條水平線代表每間隔一小時報告的一組不同資料點：

1. 實際觀察到的負載會使用所選負載指標的 SUM 統計資料來顯示過去的每小時總負載。
2. 政策預測的負載會顯示每小時的負載預測。此預測是基於前兩週的實際負載觀察結果。

比較容量圖表中的資料

每條水平線代表每間隔一小時報告的一組不同資料點：

1. 實際觀察到的任務數量會顯示 Amazon ECS 服務過去的實際容量，這取決於其他擴展政策與所選時段內有效的群組大小下限。
2. 政策預測的容量會顯示政策處於預測和擴展模式時，可預期在每小時開始時獲得的基準容量。
3. 推斷的所需任務數量會顯示服務中的理想任務數量，以將擴展指標維持在所選目標值。
4. 任務數量下限會顯示服務中的任務數量下限。

5. 容量上限會顯示服務中的任務數量上限。

為了計算推斷的所需容量，我們首先假設以指定的目標值平均使用每項任務。實際上，並不會平均使用任務數量。但是，假設使用率均勻地分佈在任務之間，我們就可以對所需容量進行可能的估算。然後，所需任務數量的計算結果會與用於預測性擴展政策的擴展指標成反比。換句話說，隨著任務數量增加，擴展指標會以相同的速率減少。例如，如果任務數量加倍，擴展指標必定會減半。

推斷的所需容量公式：

$$\text{sum of } (\text{actualServiceUnits} * \text{scalingMetricValue}) / (\text{targetUtilization})$$

例如，我們使用特定一小時的 `actualServiceUnits` (10) 和 `scalingMetricValue` (30)。然後，我們會使用您在預測擴展政策中指定的 `targetUtilization` (60)，並計算同一小時內推斷的所需容量。這會傳回值 5。這表示 5 是維持容量與擴展指標目標值正好成反比所需的推斷容量。

Note

您可以使用各種控制桿來調整和改善應用程式的成本節省效益和可用性。

- 您可以針對基準容量使用預測擴展，並使用動態擴展來處理額外的容量。動態擴展會與預測擴展分開運作，可根據目前的使用率進行縮減和擴增。首先，Amazon ECS 會為每個非排程擴展政策計算建議的任務數量。然後，它會根據提供最多任務數量的政策進行擴展。
- 為了在負載減少時允許進行縮減，服務應一律至少具有一個動態擴展政策，並啟用縮減部分。
- 您可以確保您的最小和最大容量沒有太大限制，以提高擴展效能。如果政策的建議任務數量不在容量下限與上限範圍內，將無法進行縮減與橫向擴充。

使用 CloudWatch 監控 Amazon ECS 的預測性擴展指標

您可以使用 Amazon CloudWatch 監控資料以進行預測性擴展。預測性擴展政策會收集用於預測未來負載的資料。收集的資料會定期自動儲存在 CloudWatch 中，並可用於視覺化政策隨時間執行的表現。您還可以建立 CloudWatch 警示，以便在效能指標變更超出您定義的限制時通知您。

視覺化歷史預測資料

您可以在 CloudWatch 中檢視預測性擴展政策的負載預測資料，這些資料有助於在單一圖表中根據其他 CloudWatch 指標視覺化預測。透過檢視更廣泛的時間範圍，還可以看到隨時間變化的趨勢。您可以存取長達 15 個月的歷史指標，以更加了解政策的執行狀況。

使用 CloudWatch 主控台檢視歷史預測資料

1. 透過 <https://console.aws.amazon.com/cloudwatch/> 開啟 CloudWatch 主控台。
2. 在導覽窗格中，選擇 Metrics (指標)，然後選擇 All metrics (所有指標)。
3. 選擇應用程式自動擴展指標命名空間。
4. 選擇預測性擴展負載預測。
5. 在搜尋欄位中，輸入預測性擴展政策的名稱或者 Amazon ECS 服務群組的名稱，然後按 Enter 篩選結果。
6. 若要將指標圖形化，請勾選指標旁的核取方塊。若要變更圖形的名稱，請選擇鉛筆圖示。若要變更時間範圍，請選取一個預先定義的值，或選擇 custom (自訂)。如需詳細資訊，請參閱《Amazon CloudWatch 使用者指南》中的[建立指標圖表](#)。
7. 若要變更統計數字，請選擇 Graphed metrics (圖表化指標) 索引標籤。選擇欄位標題或個別的值，然後選擇不同的統計資料。儘管您可以為每個指標選擇任何統計資料，但並非所有統計資料都適用於 PredictiveScalingLoadForecast 指標。例如，Average (平均值)、Minimum (最小值) 和 Maximum (最大值) 統計資料有用，但是 Sum (總和) 統計資料無用。
8. 若要將其他指標新增到圖表，請在 Browse (瀏覽) 下，選擇 All (所有)，找到特定指標，然後選取旁邊的核取方塊。您最多可新增 10 個指標。
9. (選用) 若要將圖表新增至 CloudWatch 儀表板，請選擇 Actions (動作)、Add to dashboard (新增至儀表板)。

使用指標數學建立準確度指標

利用指標數學，可查詢多個 CloudWatch 指標，並使用數學表達式根據這些指標來建立新的時間序列。您可以在 CloudWatch 主控台視覺化產生的時間序列，並將其新增至儀表板。如需有關指標數學的詳細資訊，請參閱《Amazon CloudWatch 使用者指南》中的[使用指標數學](#)。

使用指標數學，您可以透過不同的方式繪製服務自動擴展功能為預測性擴展而產生的資料圖表。這可協助您監控一段時間內的政策績效，並協助您瞭解是否可以改善指標組合。

例如，您可以使用指標數學表達式來監控[平均絕對誤差百分比](#) (MAPE)。MAPE 指標有助於監控預測值與指定預測期間觀察到的實際值之間的差異。MAPE 值的變化可指示政策的績效是否隨著應用程式性質的變化在一段時間內而降低。MAPE 的增加表示預測值與實際值之間的差距更大。

範例：指標數學表達式

如果要開始使用此類圖表，您可以建立類似於以下範例中所示的指標數學表達式。

`MetricDataQueries` 有一個指標資料查詢結構陣列,而不是單一的指標。`MetricDataQueries` 中的每個項目都會取得指標或執行數學表達式。第一項, `e1`, 是數學表達式。指定的表達式將 `ReturnData` 參數設定為 `true`, 最終產生單一時間序列。對於所有其他指標, `ReturnData` 值為 `false`。

在此範例中, 指定的表達式會使用實際值和預測值作為輸入值, 並傳回新指標 (MAPE)。m1 是包含實際負載值的 CloudWatch 指標 (假設 CPU 使用率是針對名為 `my-predictive-scaling-policy` 的政策最初指定的負載指標)。m2 是包含預測負載值的 CloudWatch 指標。MAPE 指標的數學語法如下：

Average of $(\text{abs}((\text{Actual} - \text{Forecast})/(\text{Actual})))$

視覺化您的準確度指標並設定警示

若要視覺化準確度指標資料, 請選取 CloudWatch 主控台中的 Metrics (指標) 索引標籤。您可以從那裡繪製資料圖表。如需詳細資訊, 請參閱《Amazon CloudWatch 使用者指南》中的[將數學表達式新增到 CloudWatch 圖表](#)。

您也可以 Metrics (指標) 區段中, 對您監控的指標設定警示。在 Graphed metrics (圖表化指標) 索引標籤上, 選取 Actions (動作) 資料欄下的 Create alarm (建立警示) 圖示。Create alarm (建立警示) 圖示表示為一個小鐘。如需詳細資訊與通知選項, 請參閱 Amazon CloudWatch User Guide 中的[Creating a CloudWatch alarm based on a metric math expression](#) 與 [Notifying users on alarm changes](#)。

或者, 您可以使用 [GetMetricData](#) 和 [PutMetricAlarm](#) 利用指標數學來執行計算, 並根據輸出建立警示。

使用排程動作覆寫 Amazon ECS 的預測值

有時候, 您可能會有未來應用程式需求的其他資訊, 但預測計算無法考量該資訊。例如, 預測計算可能會低估即將到來的行銷活動所需的任務。您可以使用排程動作, 在未來時段暫時覆寫預測。排程動作可以週期性執行, 或在一次性需求波動的特定日期與時間執行。

例如, 您可以建立任務數量高於預測值的排程動作。在執行時期, Amazon ECS 會更新服務中的任務數量下限。由於預測性擴展會針對任務數量進行最佳化, 因此會接受任務數量下限高於預測值的排程動作。這可防止任務數量低於預期。若要停止複寫預測, 請使用第二個排程動作, 讓任務數量下限恢復至其原始設定。

下列程序概述在未來時段覆寫預測的步驟。

主題

- [步驟 1：\(選用\) 分析時間序列資料](#)
- [步驟 2：建立兩個排程動作](#)

Important

本主題假設您正嘗試覆寫預測，以擴展至高於預測值的容量。如果需要暫時減少任務數量，而不受預測性擴展政策的干擾，請改用僅預測模式。在僅預測模式下，預測性擴展將繼續產生預測，但不會自動增加任務數量。然後，您可以監控資源使用率，並視需要手動減少任務數量。

步驟 1：(選用) 分析時間序列資料

從分析預測時間序列資料開始。這是選用步驟，但是如果您想要了解預測的詳細資訊，這會很有幫助。

1. 擷取預測

建立預測之後，您可以查詢預測中的特定時段。查詢的目標是取得特定時段的時間序列資料的完整檢視。

查詢最多可包含未來兩天的預測資料。如果已經使用預測擴展一段時間，您也可以存取過去的預測資料。不過，開始和結束時間之間的最大持續時間是 30 天。

若要使用 [get-predictive-scaling-forecast](#) AWS CLI 命令取得預測，請在 命令中提供下列參數：

- 在 `resource-id` 參數中輸入叢集名稱。
- 在 `--policy-name` 參數中輸入政策名稱。
- 在 `--start-time` 參數中輸入開始時間，以便僅將指定時間之時或之後的預測資料傳回。
- 在 `--end-time` 參數中輸入結束時間，以便僅將指定時間之前的預測資料傳回。

```
aws application-autoscaling get-predictive-scaling-forecast \
  --service-namespace ecs \
  --resource-id service/MyCluster/test \
  --policy-name cpu40-predictive-scaling-policy \
  --scalable-dimension ecs:service:DesiredCount \
  --start-time "2021-05-19T17:00:00Z" \
  --end-time "2021-05-19T23:00:00Z"
```

如果成功，此命令會傳回類似下方範例的資料。

```
{
  "LoadForecast": [
    {
      "Timestamps": [
        "2021-05-19T17:00:00+00:00",
        "2021-05-19T18:00:00+00:00",
        "2021-05-19T19:00:00+00:00",
        "2021-05-19T20:00:00+00:00",
        "2021-05-19T21:00:00+00:00",
        "2021-05-19T22:00:00+00:00",
        "2021-05-19T23:00:00+00:00"
      ],
      "Values": [
        153.0655799339254,
        128.8288551285919,
        107.1179447150675,
        197.3601844551528,
        626.4039934516954,
        596.9441277518481,
        677.9675713779869
      ],
      "MetricSpecification": {
        "TargetValue": 40.0,
        "PredefinedMetricPairSpecification": {
          "PredefinedMetricType": "ASGCPUUtilization"
        }
      }
    }
  ],
  "CapacityForecast": {
    "Timestamps": [
      "2021-05-19T17:00:00+00:00",
      "2021-05-19T18:00:00+00:00",
      "2021-05-19T19:00:00+00:00",
      "2021-05-19T20:00:00+00:00",
      "2021-05-19T21:00:00+00:00",
      "2021-05-19T22:00:00+00:00",
      "2021-05-19T23:00:00+00:00"
    ],
    "Values": [
      2.0,
      2.0,
      2.0,

```



```
        2.0,  
        4.0,  
        4.0,  
        4.0  
    ]  
  },  
  "UpdateTime": "2021-05-19T01:52:50.118000+00:00"  
}
```

回應包含兩種預測：LoadForecast 和 CapacityForecast。LoadForecast 顯示每小時負載預測。CapacityForecast 顯示每小時處理預測負載時所需的容量預測值，同時維持 TargetValue 為 40.0 (40% 的 CPU 平均使用率)。

2. 確定目標時段

確定應發生一次性需求波動時的一個小時或數個小時。請記住，預測中顯示的日期和時間為 UTC 格式。

步驟 2：建立兩個排程動作

接下來，在應用程式具有高於預測的負載時，為特定時段建立兩個排程動作。舉例來說，如果行銷活動會在特定時段為網站帶來流量，您可以排程一次性動作，在它開始時更新最小容量。然後，排程另一個動作，以便在事件結束時將最小容量恢復至原始設定。

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「叢集詳細資訊」頁面上的服務區段中選擇服務。

服務資訊頁面隨即顯示。

4. 選擇服務自動擴展。

政策頁面隨即顯示。

5. 選擇排程動作，然後選擇建立。

建立排程動作頁面隨即顯示。

6. 在名稱欄位中輸入唯一的名稱。
7. 對於 Time zone (時區)，選擇時區。

所有列出的時區都來自 IANA 時區資料庫。如需詳細資訊，請參閱 [List of tz database time zones](#)。

8. 在開始時間欄位中，輸入動作開始的日期與時間。
9. 針對 Recurrence (重複)，選擇 Once (一次)。
10. 在任務調整下的「下限」欄位中，輸入小於或等於任務數量上限的值。
11. 選擇建立排程動作。

政策頁面隨即顯示。

12. 設定第二個排程動作，以便在事件結束時將任務數量下限恢復至原始設定。僅當為下限設定的值低於預測值時，預測性擴展才能擴展任務數量。

為一次性事件建立兩個排程動作 (AWS CLI)

若要使用 AWS CLI 建立排程動作，請使用 [put-scheduled-update-group-action](#) 命令。

例如，我們定義一個排程，在 5 月 19 日下午 5 點維持三個執行個體的最小容量，持續 8 小時。下列命令顯示如何實作此案例。

第一個 [put-scheduled-update-group-action](#) 命令會指示 Amazon EC2 Auto Scaling 在 2021 年 5 月 19 日下午 5 點 (UTC) 更新 Auto Scaling 群組的最小容量。

```
aws autoscaling put-scheduled-update-group-action --scheduled-action-name my-event-start \  
  --auto-scaling-group-name my-asg --start-time "2021-05-19T17:00:00Z" --minimum-  
  capacity 3
```

第二個命令指示 Amazon EC2 Auto Scaling 在 2021 年 5 月 20 日上午 1 點 (UTC)，將群組的最小容量設定為 1。

```
aws autoscaling put-scheduled-update-group-action --scheduled-action-name my-event-end \  
  --auto-scaling-group-name my-asg --start-time "2021-05-20T01:00:00Z" --minimum-  
  capacity 1
```

再將這些排程動作新增到 Auto Scaling 群組後，Amazon EC2 Auto Scaling 會執行下列動作：

- 在 2021 年 5 月 19 日下午 5 點 (UTC)，第一個排程動作會執行。如果群組目前擁有少於 3 個執行個體，群組則會擴增至 3 個執行個體。在此時間和接下來的八個小時內，如果預測容量高於實際容量，或者如果動態擴展政策生效，則 Amazon EC2 Auto Scaling 可以繼續擴增。
- 在 2021 年 5 月 20 日上午 1 點 (UTC)，第二個排程動作會執行。這會在事件結束時將最小容量恢復至原始設定。

根據週期性排程擴展

若要每週覆寫相同時段的預測，請建立兩個排程動作，並使用 Cron 表達式提供時間與日期邏輯。

Cron 表達式格式由 5 個以空格分隔的欄位組成：[分鐘] [小時] [一個月的第幾日] [一年的第幾個月] [一週的第幾日]。欄位可以包含任何允許的數值，包括特殊字元。

例如，以下 Cron 表達式會在每周二上午 6:30 執行動作。使用星號作為萬用字元，以比對欄位的所有數值。

```
30 6 * * 2
```

另請參閱

如需有關如何管理排程動作的詳細資訊，請參閱[使用排程動作來擴展 Amazon ECS 服務](#)。

使用自訂指標的 Amazon ECS 進階預測性擴展政策

您可以在預測性擴展政策中使用預先定義或自訂的指標。當預先定義的指標 (例如 CPU、記憶體等) 不足以充分描述應用程式負載時，自訂指標會非常有用。

使用自訂指標建立預測性擴展政策時，您可以指定由 AWS 提供的其他 CloudWatch 指標。或者，您也可以指定自行定義與發布的指標。您也可以使用指標數學來彙總現有指標，並將其轉換為 AWS 不會自動追蹤的新時間序列。例如，透過計算新的總和或平均值 (稱為彙總) 來合併資料中的值。產生的資料稱為彙總。

下一節包含如何建構政策的 JSON 結構的最佳實務和範例。

先決條件

若要在預測擴展政策中新增自訂指標，您必須擁有 `cloudwatch:GetMetricData` 許可。

若要指定自己的指標，而不是 AWS 提供的指標，您必須先將指標發佈至 CloudWatch。如需詳細資訊，請參閱《Amazon CloudWatch 使用者指南》中的[發佈自訂指標](#)。

如果您發佈自己的指標，應確保以至少五分鐘的頻率發佈資料點。系統會根據需要的時段長度從 CloudWatch 擷取資料點。例如，負載指標規範使用每小時指標來衡量應用程式的負載。CloudWatch 使用您發佈的指標資料，透過將時間戳記落於同一小時週期內的所有資料點彙總，為任何一小時週期提供單一資料值。

最佳實務

以下最佳實務可協助您更有效地使用自訂指標：

- 對於負載指標規格，最實用的指標是以 Auto Scaling 群組作為整體來表示負載的指標。
- 對於擴展指標規格，最實用的擴展指標是每項任務指標的平均輸送量或使用率。
- 目標使用率必須與擴展指標的類型相符。例如，對於使用 CPU 使用率的政策組態，這是一個目標百分比。
- 如果不遵循這些建議，則時間序列的預測未來值可能不正確。要驗證資料是否正確，您可以在主控台中檢視預測值。或者，在建立預測性擴展政策後，檢查 [GetPredictiveScalingForecast](#) API 呼叫傳回的 LoadForecast 物件。
- 強烈建議在僅預測模式下設定預測性擴展，以便在預測性擴展開始主動擴展之前評估預測。

限制

- 您可以在一個指標規範中查詢最多 10 個指標的資料點。
- 在此限制之下，一個表達式計為一個指標。

對使用自訂指標的預測性擴展政策進行疑難排解

如果使用自訂指標時出現問題，建議您執行以下操作：

- 如果您在使用搜尋表達式時在藍/綠部署中遇到問題，請確定所建立的搜尋表達式是在查找部分相符，而不是完全相符。此外，還應檢查查詢是否僅尋找在特定應用程式中執行的 Auto Scaling 群組。如需搜尋表達式語法的詳細資訊，請參閱《Amazon CloudWatch 使用者指南》中的 [CloudWatch 搜尋表達式語法](#)。
- [put-scaling-policy](#) 命令會在您建立擴展政策時驗證表達式。但是，此命令可能無法識別偵測到之錯誤的確切原因。若要修正這些問題，請對您收到之 [get-metric-data](#) 命令請求回應中的錯誤進行疑難排解。您也可以從 CloudWatch 主控台對表達式進行疑難排解。
- 如果 MetricDataQueries 自己指定了 SEARCH() 函數 (在無需 SUM() 等數學函數的狀況下)，則您必須為 ReturnData 指定 false。這是因為搜尋表達式可能會傳回多個時間序列，並且基於表達式的指標規範只能傳回一個時間序列。

- 搜尋表達式中涉及的所有指標均應具有相同的解析度。

透過 Amazon ECS 為預測性擴展自訂指標建構 JSON

下一節包含如何設定預測擴展以查詢來自 CloudWatch 的資料的範例。設定此選項有兩種不同的方法，而您選擇的方法會影響您用來建構預測擴展政策的 JSON 的格式。使用指標數學時，JSON 的格式會根據所執行的指標數學而進一步變化。

1. 若要建立直接從 AWS 您發佈至 CloudWatch 的其他 CloudWatch 指標取得資料的政策，請參閱 [使用 搭配自訂負載和擴展指標的預測擴展政策範例 AWS CLI](#)。

使用 搭配自訂負載和擴展指標的預測擴展政策範例 AWS CLI

若要使用 建立具有自訂負載和擴展指標的預測擴展政策 AWS CLI，請將 的引數存放在名為 `--predictive-scaling-configuration` 的 JSON 檔案中 `config.json`。

您可以藉由將以下範例中的可替換值換成您的指標和目標使用率的值，開始新增自訂指標。

```
{
  "MetricSpecifications": [
    {
      "TargetValue": 50,
      "CustomizedScalingMetricSpecification": {
        "MetricDataQueries": [
          {
            "Id": "scaling_metric",
            "MetricStat": {
              "Metric": {
                "MetricName": "MyUtilizationMetric",
                "Namespace": "MyNameSpace",
                "Dimensions": [
                  {
                    "Name": "MyOptionalMetricDimensionName",
                    "Value": "MyOptionalMetricDimensionValue"
                  }
                ]
              },
              "Stat": "Average"
            }
          ]
        }
      }
    }
  ]
}
```

```

    },
    "CustomizedLoadMetricSpecification": {
      "MetricDataQueries": [
        {
          "Id": "load_metric",
          "MetricStat": {
            "Metric": {
              "MetricName": "MyLoadMetric",
              "Namespace": "MyNameSpace",
              "Dimensions": [
                {
                  "Name": "MyOptionalMetricDimensionName",
                  "Value": "MyOptionalMetricDimensionValue"
                }
              ]
            },
            "Stat": "Sum"
          }
        }
      ]
    }
  ]
}

```

如需詳細資訊，請參閱《Amazon EC2 Auto Scaling API 參考》中的 [MetricDataQuery](#)。

Note

以下是一些其他資源，可協助您尋找 CloudWatch 指標的指標名稱、命名空間、維度和統計資料：

- 如需 AWS 服務可用指標的相關資訊，請參閱《Amazon [AWS CloudWatch 使用者指南](#)》中的 [發佈 CloudWatch 指標](#) 的服務。Amazon CloudWatch
- 若要使用取得 CloudWatch 指標的確切指標名稱、命名空間和維度（如適用）AWS CLI，請參閱 [list-metrics](#)。

若要建立此政策，執行使用 JSON 檔案作為輸入 [put-scaling-policy](#) 命令，如以下範例所示。

```
aws application-autoscaling put-scaling-policy --policy-name my-predictive-scaling-policy \
```

```
--auto-scaling-group-name my-asg --policy-type PredictiveScaling \  
--predictive-scaling-configuration file://config.json
```

如果成功，此命令會傳回政策的 Amazon Resource Name (ARN)。

```
{  
  "PolicyARN": "arn:aws:autoscaling:region:account-id:scalingPolicy:2f4f5048-d8a8-4d14-  
b13a-d1905620f345:autoScalingGroupName/my-asg:policyName/my-predictive-scaling-policy",  
  "Alarms": []  
}
```

使用指標數學表達式

下一節將提供有關在政策中搭配預測性擴展政策使用指標數學的資訊。

了解指標數學

如果您只想彙總現有指標資料，則 CloudWatch 指標數學可以節省將另一個指標發佈至 CloudWatch 的工作量和成本。您可以使用 AWS 提供的任何指標，也可以使用您在應用程式中定義的指標。

如需詳細資訊，請參閱《Amazon CloudWatch 使用者指南》中的[使用指標數學](#)。

如果選擇在預測擴展政策中使用指標數學表達式，則請考慮以下幾點：

- 指標數學運算會使用指標名稱、命名空間和指標維度鍵/值對之唯一組合的資料點。
- 您可以使用任何算術運算子 (+-*/^)、統計函數 (如 AVG 或 SUM) 或 CloudWatch 支援的其他函數。
- 您可以在數學表達式的公式中同時使用其他數學表達式的指標和結果。
- 您的指標數學表達式可以由不同的彙總組成。但是，最終彙總結果的最佳實務是將 Average 用於擴展指標，Sum 用於負載指標。
- 在指標規範中使用的任何表達式都必須最終傳回單一的時間序列。

若要使用指標數學，請執行以下操作：

- 選擇一或多個 CloudWatch 指標。然後，建立表達式。如需詳細資訊，請參閱《Amazon CloudWatch 使用者指南》中的[使用指標數學](#)。
- 透過使用 CloudWatch 主控台或 CloudWatch [GetMetricData](#) API 驗證指標數學表達式是否有效。

使用指標數學組合指標的預測擴展政策範例 (AWS CLI)

有時，您可能需要首先以某種方式處理其資料，而不是直接指定指標。例如，您可能有一個從 Amazon SQS 佇列中提取工作的應用程式，並且您可能想要使用佇列中的項目數量作為預測擴展的條件。佇列中的訊息數量並不僅僅定義所需的執行個體數量。因此，需要更多的工作來建立可用於計算每個執行個體待處理項目的指標。

以下是此案例的預測擴展政策範例。它指定了基於 Amazon SQS `ApproximateNumberOfMessagesVisible` 指標的擴展和負載指標，即可從佇列中擷取的訊息數量。它還使用 Amazon EC2 Auto Scaling `GroupInServiceInstances` 指標和數學表達式來計算每個執行個體的待處理項目，以擴展指標。

```
aws application-autoscaling put-scaling-policy --policy-name my-sqs-custom-metrics-policy \
--policy-type PredictiveScaling \
--predictive-scaling-configuration file://config.json \
--service-namespace ecs \
--resource-id service/MyCluster/test \
"MetricSpecifications": [
  {
    "TargetValue": 100,
    "CustomizedScalingMetricSpecification": {
      "MetricDataQueries": [
        {
          "Label": "Get the queue size (the number of messages waiting to be
processed)",
          "Id": "queue_size",
          "MetricStat": {
            "Metric": {
              "MetricName": "ApproximateNumberOfMessagesVisible",
              "Namespace": "AWS/SQS",
              "Dimensions": [
                {
                  "Name": "QueueName",
                  "Value": "my-queue"
                }
              ]
            },
            "Stat": "Sum"
          },
          "ReturnData": false
        },
        {
```



```

    "Label": "Get the group size (the number of running instances)",
    "Id": "running_capacity",
    "MetricStat": {
      "Metric": {
        "MetricName": "GroupInServiceInstances",
        "Namespace": "AWS/AutoScaling",
        "Dimensions": [
          {
            "Name": "AutoScalingGroupName",
            "Value": "my-asg"
          }
        ]
      },
      "Stat": "Sum"
    },
    "ReturnData": false
  },
  {
    "Label": "Calculate the backlog per instance",
    "Id": "scaling_metric",
    "Expression": "queue_size / running_capacity",
    "ReturnData": true
  }
],
},
"CustomizedLoadMetricSpecification": {
  "MetricDataQueries": [
    {
      "Id": "load_metric",
      "MetricStat": {
        "Metric": {
          "MetricName": "ApproximateNumberOfMessagesVisible",
          "Namespace": "AWS/SQS",
          "Dimensions": [
            {
              "Name": "QueueName",
              "Value": "my-queue"
            }
          ]
        },
        "Stat": "Sum"
      },
      "ReturnData": true
    }
  ]
}

```

```
    ]
  }
}
]
```

該範例會傳回政策的 ARN。

```
{
  "PolicyARN": "arn:aws:autoscaling:region:account-id:scalingPolicy:2f4f5048-d8a8-4d14-b13a-d1905620f345:autoScalingGroupName/my-asg:policyName/my-sqs-custom-metrics-policy",
  "Alarms": []
}
```

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 “透過負載平衡分佈服務流量”](#)。

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

建議使用 Service Connect，其可提供 Amazon ECS 組態，用於服務探索、連線與流量監控。透過 Service Connect，應用程式可以使用簡短名稱與標準連接埠，連線至同一叢集、其他叢集中的 Amazon ECS 服務，包括同一 AWS 區域內跨 VPC 的服務。

當您使用 Service Connect 時，Amazon ECS 將管理服務探索的所有環節：建立可被探索的名稱、在任務啟動與停止時動態管理每個任務的項目、在每個已設定為可探索這些名稱的任務中執行代理程式。應用程式可透過 DNS 名稱的標準功能查詢這些名稱並建立連線。若應用程式已具備此能力，則您無需修改應用程式即可使用 Service Connect。

您需在每個服務與任務定義內提供完整的組態。Amazon ECS 會在每次服務部署時管理此組態的變更，確保統一部署中的所有任務行為一致。例如，使用 DNS 作為服務探索機制時，常見問題在於遷移過程難以控制。若變更 DNS 名稱指向的新替代 IP 位址，可能需要等待最長 TTL 時間，所有用戶端才會開始使用新服務。而透過 Service Connect，用戶端部署會透過取代用戶端任務來更新組態。

您可以設定部署斷路器及其他部署組態，使 Service Connect 的變更與任何其他部署一樣受到同等管控。

如需詳細資訊，請參閱[使用 Service Connect 以利用簡稱連線 Amazon ECS 服務](#)。

- Amazon ECS 服務探索

服務間通訊的另一種方法是使用服務探索進行直接通訊。透過此方法，您可使用 AWS Cloud Map 服務探索與 Amazon ECS 的整合功能。使用服務探索，Amazon ECS 會將啟動任務的清單同步至 AWS Cloud Map，這會維護 DNS 主機名稱，以解析該特定服務中一或多個任務的內部 IP 地址。Amazon VPC 中的其他服務可使用此 DNS 主機名稱，透過容器的內部 IP 位址直接向另一個容器傳送流量。

這種服務間通訊方法的延遲較低。容器之間沒有額外的元件。流量會從一個容器直接傳送至另一個容器。

此方法適用於採用 awsvpc 網路模式的場景，該模式下每個任務都有專屬的唯一 IP 位址。大多數軟體僅支援使用 DNS A 記錄，這些記錄會直接解析為 IP 位址。使用 awsvpc 網路模式時，每個任務的 IP 位址都是 A 記錄。不過，如果您使用 bridge 網路模式，多個容器可能會共用同一個 IP 位址。此外，動態連接埠映射會導致容器在該單一 IP 位址上隨機指派連接埠號碼。至此，僅靠 A 記錄已不足以滿足服務探索需求。您必須同時使用 SRV 記錄。這種類型的記錄可以追蹤 IP 位址與連接埠號碼，但要求對應用程式進行相應設定。您使用的某些預先建置的應用程式可能不支援 SRV 記錄。

awsvpc 網路模式的另一項優勢，是每個服務都會擁有唯一的安全群組。您可以設定此安全群組，使其僅允許需要與該服務通訊的特定上游服務建立連入連線。

使用服務探索實現服務間直接通訊的主要劣勢在於，您必須實作額外邏輯來處理重試機制與連線失敗問題。DNS 記錄具有存留時間 (TTL) 設定，可控制記錄的快取存留時長。DNS 記錄的更新與快取失效需要一定時間，才能讓應用程式取得最新版本的 DNS 記錄。因此，應用程式最終可能會將 DNS 記錄解析至已不存在的另一個容器。應用程式需要處理重試，並具有忽略不良後端的邏輯。

如需詳細資訊，請參閱[使用服務探索以利用 DNS 名稱連接 Amazon ECS 服務](#)

- Amazon VPC Lattice

Amazon VPC Lattice 是一種受管應用程式聯網服務，Amazon ECS 客戶可用來觀察、保護和監控跨 AWS 運算服務、VPCs 和帳戶建置的應用程式，而無需修改其程式碼。

VPC Lattice 使用目標群組，這是運算資源的集合。這些目標會執行您的應用程式或服務，可以是 Amazon EC2 執行個體、IP 地址、Lambda 函數和 Application Load Balancer。透過將其 Amazon ECS 服務與 VPC Lattice 目標群組建立關聯，客戶現在可以在 VPC Lattice 中將 Amazon ECS 任務

啟用為 IP 目標。當已註冊服務的任務啟動時，Amazon ECS 會自動將任務註冊到 VPC Lattice 目標群組。

如需詳細資訊，請參閱[使用 Amazon VPC Lattice 連接、觀察和保護您的 Amazon ECS 服務](#)。

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。

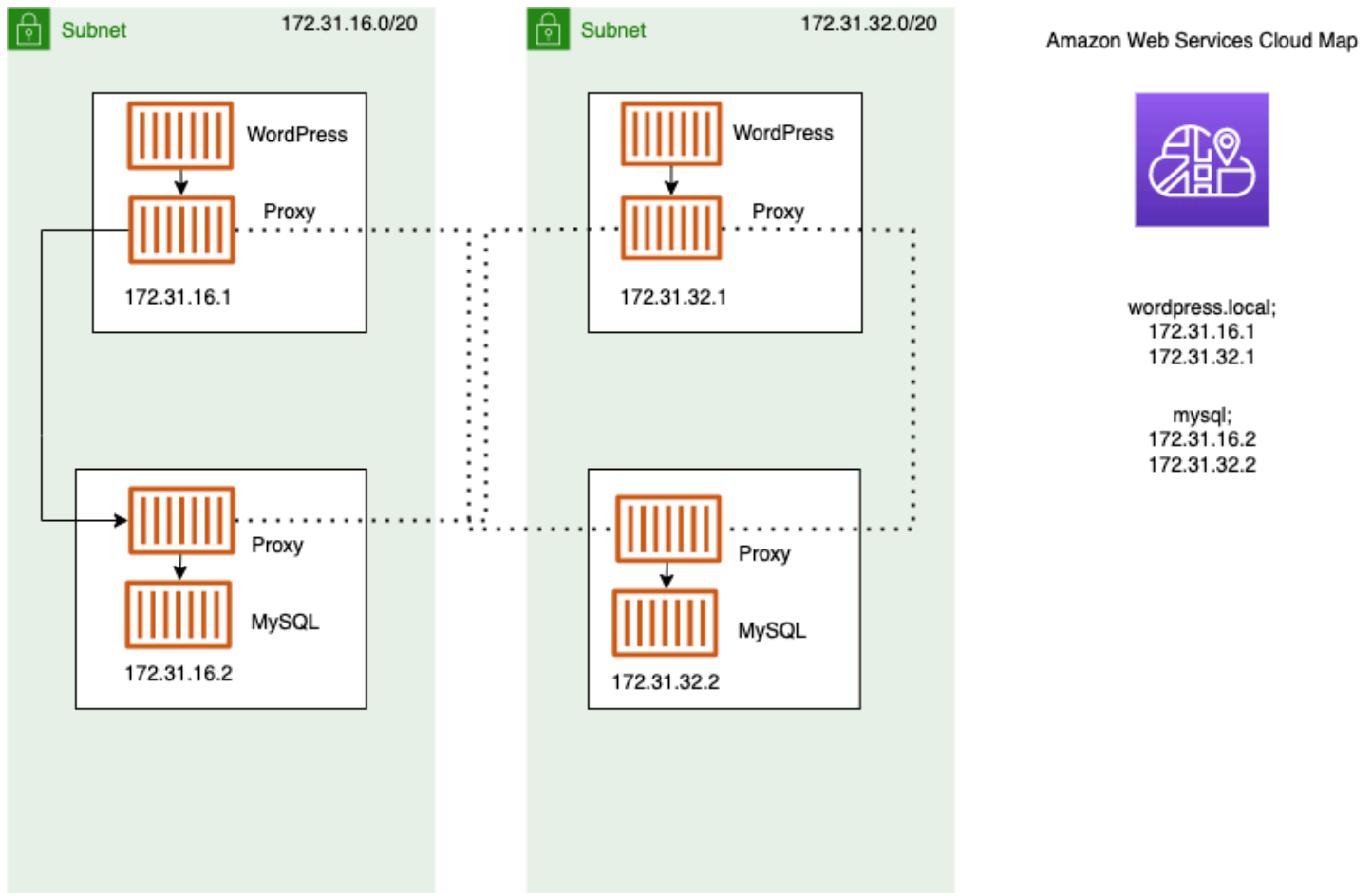
互連線選項	橋接	awsipc	主機
服務探索	是的，但用戶端需要知道在 DNS 中不含 hostPort 的 SRV 記錄。	是	是的，但用戶端需要知道在 DNS 中不含 hostPort 的 SRV 記錄。
Service Connect	是	是	否
VPC Lattice	是	是	是

使用 Service Connect 以利用簡稱連線 Amazon ECS 服務

Amazon ECS Service Connect 提供服務對服務通訊的管理作為 Amazon ECS 組態。其會在 Amazon ECS 中同時建置服務探索和服務網格。這會提供：您透過服務部署管理的每個服務內的完整組態、統一參考未根據 VPC DNS 組態之命名空間內的服務的方式，以及可監控所有應用程式的標準化指標和日誌。Service Connect 僅讓服務互連。

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。一種用戶端服務，在每個子網路中執行具有 1 項任務的 WordPress。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 到 MySQL 的連線。例如，從具有 IP 地址 172.31.16.1 任務中的 WordPress 容器內執行的 `mysql --host=mysql` CLI 命令。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 任務中的 Proxy 會在異常值偵測中使用循環配置負載平衡，以及任何先前的失敗資訊，以挑選要連接的 MySQL 任務。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 Proxy 都會回報在 Amazon ECS 和 Amazon CloudWatch 主控台中以圖形顯示的連線效能，讓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。此名稱用於建立 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) 中的單一系列，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 Amazon Elastic Container Service API Reference (《Amazon Elastic Container Service API 參考》) 中的 [ServiceConnectService](#)。

命名空間

用於 Service Connect 之 AWS Cloud Map 命名空間的簡短名稱或完整 Amazon Resource Name (ARN)。命名空間必須與 Amazon ECS 服務和叢集位於相同的 AWS 區域中。中的命名空間類型 AWS Cloud Map 不會影響 Service Connect。命名空間可以是與中可用 AWS 帳戶之 using AWS Resource Access Manager (AWS RAM) AWS RAM 共用 AWS 區域的命名空間。如需有關共用命名空間的詳細資訊，請參閱 AWS Cloud Map Developer Guide 中的 [Cross-account AWS Cloud Map namespace sharing](#)。

Service Connect 使用 AWS Cloud Map 命名空間做為彼此通訊的 Amazon ECS 任務邏輯分組。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佈到同一 AWS 區域內的不

同 Amazon ECS 叢集。如果命名空間是共用命名空間，則服務可以分佈到命名空間擁有者與命名空間取用者 AWS 帳戶。您可以根據任何條件隨意組織服務。

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至同一命名空間中的其他用戶端-伺服器服務，但仍需設定用戶端組態。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。服務執行的每項任務都會在任務中執行代理容器。

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任務
- 未針對 Service Connect 設定的任務
- Amazon ECS 以外的其他應用程式

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。

若要讓這些應用程式解析 Amazon ECS 任務的 IP 位址，則需使用其他互連方法。

主題

- [定價](#)
- [Amazon ECS Service Connect 元件](#)
- [Amazon ECS Service Connect 組態概觀](#)
- [具有共用 AWS Cloud Map 命名空間的 Amazon ECS Service Connect](#)
- [Amazon ECS Service Connect 存取日誌](#)
- [加密 Amazon ECS Service Connect 流量](#)
- [使用 設定 Amazon ECS Service Connect AWS CLI](#)

定價

- Amazon ECS Service Connect 定價取決於您使用 AWS Fargate 或 Amazon EC2 基礎設施來託管容器化工作負載。在 AWS Outposts 上使用 Amazon ECS 時，定價直接採用與您使用 Amazon EC2 時所用的相同模型。如需詳細資訊，請參閱 [Amazon ECS 定價](#)。
- 使用 Amazon ECS Service Connect 無需額外付費。
- Service Connect 使用 AWS Cloud Map 時，用量不收取額外費用。
- 客戶需支付 Amazon ECS Service Connect 使用的運算資源費用，包括 vCPU 與記憶體。由於 Amazon ECS Service Connect 代理程式在客戶任務內執行，因此執行無需額外費用。任務資源會在客戶工作負載與 Amazon ECS Service Connect 代理程式之間共用。
- 搭配使用 Amazon ECS Service Connect 流量加密功能時 AWS 私有 CA，客戶會為他們建立的私有憑證授權機構和每個發行的 TLS 憑證付費。如需更多詳細資訊，請參閱 [AWS 私有憑證授權單位 定價](#)。若要預估 TLS 憑證的每月成本，客戶需要知道已啟用 TLS 的 Amazon ECS 服務數量，將其乘以憑證成本，然後再乘以六。由於 Amazon ECS Service Connect 每五天自動輪換 TLS 憑證，因此每項 Amazon ECS 服務平均每月會發行六個憑證。

Amazon ECS Service Connect 元件

使用 Amazon ECS Service Connect 時，可以將每項 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 Service 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-伺服器服務透過橫向擴充來新增任務，則將在所有任務之間對來自用戶端的新連線進行平衡。如果已更新用戶端-伺服器服務，則將在所有新版本的任務之間對來自用戶端的新連線進行平衡。

現有任務無法解析並連線到新端點。只有在同一命名空間中具有 Service Connect 組態且在此部署後開始執行的新任務，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後，命名空間中的端點清單都可能變更。現有任務與替代任務的行為會繼續保持與最近一次部署後相同的行為。

請考慮下列範例。

首先，假設您正在單一 AWS CloudFormation 範本和單一 CloudFormation 堆疊中建立公有網際網路可用的應用程式。公有探索和連線能力應由 建立 CloudFormation，包括前端用戶端服務。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，您必須使用 `dependsOn` 來指示多個 CloudFormation Amazon ECS 服務無法平行或同時進行。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，因此所有部署轉返偵測 AWS 管理主控台或 AWS SDKs AWS CLI 和其他方法來轉返，並將後端服務還原至先前的部署和組態。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。如果此組態發生問題，您可以使用部署復原偵測或 AWS 管理主控台、AWS CLI、AWS SDKs 和其他方法來復原和還原至先前的組態，並將後端服務還原至先前的部署和組態。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，Service Connect Proxy 會監聽任務定義連接埠映射中的 `containerPort`。安全群組規則必須允許來自用戶端將執行之子網路的傳入流量到達此連接埠。

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，請變用戶端-伺服器服務 Service Connect 組態中的 `ingressPortOverride`。如果變更此連接埠號碼，則必須允許此連接埠的傳入流量，此服務的流量會使用該連接埠。

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 Service Connect 在 VPC 之間傳送流量。路由表規則、網路 ACL 與安全群組的相同需求適用於這兩種 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。不過，除非 VPC 對等互連、VPC 或子網路路由表以及 VPC 網路 ACL 允許 `containerPort` 與 `ingressPortOverride` 連接埠號碼上的流量，否則 Proxy 無法連線。

對於使用 bridge 聯網模式的任務，您必須建立安全群組，其傳入規則需允許上層動態連接埠範圍的流量。然後，將該安全群組指派給 Service Connect 叢集中的所有 EC2 執行個體。

Service Connect Proxy

如果使用 Service Connect 組態建立或更新服務，Amazon ECS 會在每個新任務啟動時新增新的容器。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 會在服務中管理容器組態。因此，您可以在沒有 Service Connect 的情況下，於多項服務、命名空間與任務之間重複使用相同的任務定義。

Proxy 資源

- 對於任務定義，必須設定 CPU 與記憶體參數。

建議在 Service Connect Proxy 容器的任務 CPU 與記憶體中，額外新增 256 個 CPU 單元與至少 64 MiB 記憶體。在 AWS Fargate 上，您可以設定的最低記憶體容量為 512 MiB。在 Amazon EC2 上，任務定義記憶體是必要的。

- 對於服務，您需要在 Service Connect 組態中進行日誌設定。
- 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 會設定任務與容器，以便應用程式僅在應用程式連線至同一命名空間中的端點名稱時才連線至 Proxy。所有其他流量都不會使用 Proxy。其他流量包含相同 VPC 中的 IP 地址、AWS 服務端點和外部流量。

Load balancing

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，假設在服務中執行 WordPress 的任務，該服務在名為本機的命名空間中設定為用戶端服務。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 任務中，WordPress 應用程式使用端點名稱連線至資料庫。此名稱的連線會移至在相同任務的邊車容器中執行的 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。

使用之前的範例，該 Proxy 可以連線至任一 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

Timeout (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。

選用的參數：

`idleTimeout` – 連線在閒置時保持作用中狀態的時間 (以秒為單位)。值為 0 會停用 `idleTimeout`。

HTTP/HTTP2/GRPC 的 `idleTimeout` 預設值為 5 分鐘。

TCP 的 `idleTimeout` 預設值為 1 小時。

`perRequestTimeout` – 等待上游對每個請求發出完整回應的時間。值為 0 會關閉 `perRequestTimeout`。這僅能在應用程式容器的 `appProtocol` 為 HTTP/HTTP2/GRPC 時設定。預設值為 15 秒。

 Note

如果 `idleTimeout` 設定為小於 `perRequestTimeout` 的時間，則連線將在達到 `idleTimeout` 而不是 `perRequestTimeout` 的時候關閉。

考量事項

使用 Service Connect，請考量下列事項：

- 在 Fargate 中執行的任務必須使用 Fargate Linux 平台版本 1.4.0 或更高版本，才能使用 Service Connect。
- 容器執行個體上的 Amazon ECS 代理程式版本必須是 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 Service Connect 代理程式的更多資訊，請參閱 GitHub 上的 [Amazon ECS Service 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](#)。
- 使用 bridge 網路模式且使用 Service Connect 的任務不支援 `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任務定義。

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](#)。

- 您可以設定 Service Connect 在相同 區域中使用任何 AWS Cloud Map 命名空間，這些命名空間位於相同 或 AWS 帳戶 與您的 AWS 帳戶 共用 AWS Resource Access Manager。如需有關使用共用命名空間的詳細資訊，請參閱 [具有共用 AWS Cloud Map 命名空間的 Amazon ECS Service Connect](#)。
- 每一項服務只能屬於一個命名空間。
- 僅支援服務建立的任務。
- 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。
- 必須重新部署現有服務，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 “Service Connect 元件”](#)。
- 刪除叢集時，Service 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 流量預設為在 awsvpc 網路模式下透過 Service Connect 代理程式進行路由。如果您希望非服務流量繞過 Service Connect 代理程式，請在 Service Connect 服務組態中使用 [ingressPortOverride](#) 參數。
- 使用 TLS 的 Service Connect 不支援 bridge 聯網模式。僅支援 awsvpc 聯網模式。
- 在 awsvpc 模式下，對於純 IPv4 與雙堆疊組態的服務，Service Connect Proxy 會將流量轉送至 IPv4 localhost 127.0.0.1。對於純 IPv6 組態的服務，Proxy 會將流量轉送至 IPv6 localhost ::1。建議將應用程式設定為監聽兩個 localhost，以便在服務從純 IPv4 或雙堆疊組態更新為純 IPv6 組態時實現無縫體驗。如需詳細資訊，請參閱[EC2 的 Amazon ECS 任務聯網選項](#)及[Fargate 的 Amazon ECS 任務聯網選項](#)。

Service Connect 不支援下列項目：

- Windows 容器
- HTTP 1.0
- 獨立任務
- 使用由 CodeDeploy 提供支援的藍/綠部署與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
- FIPS 模式

Amazon ECS Service Connect 組態概觀

當您使用 Service Connect 時，需要在資源中設定參數。

下表說明了 Amazon ECS 資源的組態參數。

參數位置	應用程式類型	Description	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。如需詳細資訊，請參閱 portMappings	是
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

下列步驟概述了如何設定 Service Connect。

Important

- Service 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。
- Service Connect 不支援任務定義中的連結。

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、使用共用命名空間，或分別建立命名空間。為簡化組織管理，請建立具有您所需命名空間名稱的叢集，並為該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。Service Connect 不會使用或建立 Amazon Route 53 中的 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. 透過 Amazon CloudWatch 中的 Service Connect Proxy 監控流量。

叢集組態

您可以在建立或更新叢集時為 Service Connect 設定預設命名空間。您指定為預設的命名空間名稱可以位於相同的 AWS 區域和帳戶中，也可以位於相同的中 AWS 區域，並由另一個 AWS 帳戶使用共用 AWS Resource Access Manager。

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。根據預設，附件不會顯示在 `awscli` 中，您必須新增 `--include ATTACHMENTS` 才能查看。

如果您想要使用與 AWS 帳戶共用的命名空間 AWS RAM，請在叢集組態中指定命名空間的 Amazon Resource Name (ARN)。如需共用 AWS Cloud Map 命名空間的詳細資訊，請參閱 [具有共用 AWS Cloud Map 命名空間的 Amazon ECS Service Connect](#)。

服務組態

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，然後選取中的命名空間 AWS 帳戶或共用命名空間，以建立用戶端服務。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`。

初始運作狀態檢查組態

Service Connect 可確保將流量路由至任務之前，任務的運作狀態良好。當任務啟動時（在部署、擴展或取代期間），Service Connect 會監控任務的運作狀態，以確保已準備好接受流量。您必須定義任務定義中必要容器的運作狀態檢查，才能啟用此行為。

當任務準備好接受流量時，初始運作狀態檢查的行為會考量達到狀態的潛在延遲：

- 如果任務是 HEALTHY，則立即可供流量使用。
- 如果任務的運作狀態為 UNKNOWN，則 Service Connect 會遵循任務基本容器的容器運作狀態檢查組態（請參閱 [the section called “運作狀態檢查”](#)）來計算逾時，最多可達 8 minutes，再將其提供給流量使用，即使它仍為 UNKNOWN。
- 如果任務是 UNHEALTHY，Amazon ECS 可能會啟動替代任務。如果沒有運作狀態良好的任務，您的部署可能會根據服務的組態復原。

對於所有持續的流量，Service Connect 會根據極端值偵測使用被動運作狀態檢查，以有效率地路由流量。

具有共用 AWS Cloud Map 命名空間的 Amazon ECS Service Connect

Amazon ECS Service Connect 支援在同一 AWS 帳戶內跨多個使用共用 AWS Cloud Map 命名空間 AWS 區域。此功能可讓您建立分散式應用程式，其中在不同的中執行的服務 AWS 帳戶可以透過

Service Connect 彼此探索和通訊。共用命名空間使用 AWS Resource Access Manager (AWS RAM) 進行管理，允許安全的跨帳戶資源共用。如需共用命名空間的詳細資訊，請參閱《AWS Cloud Map 開發人員指南》中的[跨帳戶 AWS Cloud Map 命名空間共用](#)。

 Important

必須使用 `AWSRAMPermissionCloudMapECSFullPermission` 受管許可來共用命名空間，Service Connect 才能在命名空間正常運作。

當您搭配 Service Connect 使用共用 AWS Cloud Map 命名空間時，來自多個的服務 AWS 帳戶可以參與相同的服務命名空間。這對於具有多個的組織特別有用 AWS 帳戶，這些組織需要維持跨帳戶邊界 service-to-service 通訊，同時保持安全和隔離。

 Note

若要與位於不同 VPC 中的服務通訊，您需要設定 VPC 間連線功能。這可以使用 VPC 對等互連來實現。如需詳細資訊，請參閱 Amazon Virtual Private Cloud VPC Peering Guide 中的[Create or delete a VPC Peering connection](#)。

搭配 Amazon ECS Service Connect 使用共用 AWS Cloud Map 命名空間

設定 Service Connect 的共用 AWS Cloud Map 命名空間包含下列步驟：建立命名空間的命名空間擁有者、透過 AWS Resource Access Manager (AWS RAM) 共用的擁有者、接受資源共用的取用者，以及設定 Service Connect 使用共用命名空間的取用者。

步驟 1：建立 AWS Cloud Map 命名空間

命名空間擁有者會建立將與其他帳戶共用的 AWS Cloud Map 命名空間。

使用 建立共用的命名空間 AWS 管理主控台

1. 在 <https://console.aws.amazon.com/cloudmap/> 開啟 AWS Cloud Map 主控台。
2. 選擇 Create namespace (建立命名空間)。
3. 輸入命名空間名稱。所有參與帳戶中的服務都會使用此名稱。
4. 在命名空間類型欄位中，根據使用案例選擇適當的類型：
 - API 呼叫 – HTTP 命名空間，用於不使用 DNS 功能進行服務探索。

- VPC 中的 API 呼叫與 DNS 查詢 – 私有 DNS 命名空間，用於在 VPC 中使用私有 DNS 查詢進行服務探索。
- API 呼叫與公有 DNS 查詢 – 公有 DNS 命名空間，用於使用公有 DNS 查詢進行服務探索。

5. 選擇 Create namespace (建立命名空間)。

步驟 2：使用 共用命名空間 AWS RAM

命名空間擁有者使用 與其他 AWS RAM 共用命名空間 AWS 帳戶。

使用 AWS RAM 主控台共用命名空間

1. 在 <https://console.aws.amazon.com/ram/> 開啟 AWS RAM 主控台。
2. 選擇 Create resource share (建立資源共用)。
3. 在名稱欄位中，輸入資源共用的描述性名稱。
4. 在資源區段中：
 - a. 在資源類型欄位中，選擇 Cloud Map 命名空間。
 - b. 選取您在前一步驟中建立的命名空間。
5. 在受管許可區段中，指定 AWSRAMPermissionCloudMapECSFullPermission。

Important

必須使用 AWSRAMPermissionCloudMapECSFullPermission 受管許可來共用命名空間，Service Connect 才能在命名空間正常運作。

6. 在主體區段中，指定要與之共用命名空間的 AWS 帳戶。您可以輸入帳戶 ID 或組織單位 ID。
7. 選擇 Create resource share (建立資源共用)。

步驟 3：接受資源共用

命名空間取用者帳戶必須接受資源共用邀請，才能使用共用命名空間。

使用 AWS RAM 主控台接受資源共享邀請

1. 在消費者帳戶中，開啟位於 <https://console.aws.amazon.com/ram/> 的 AWS RAM 主控台。
2. 在導覽窗格中，依次選擇與我共用、資源共用。

3. 選取資源共用邀請，然後選擇接受資源共用。
4. 接受後，請記下資源詳細資訊中的共用命名空間 ARN。設定 Service Connect 服務時，您將使用此 ARN。

步驟 4：使用共用命名空間設定 Amazon ECS 服務

接受共用命名空間後，命名空間取用者可以將 Amazon ECS 服務設定為使用共用命名空間。組態與使用一般命名空間類似，但您必須指定命名空間 ARN 而不是名稱。如需詳細的服務建立程序，請參閱[建立 Amazon ECS 滾動更新部署](#)。

使用 建立具有共用命名空間的服務 AWS 管理主控台

1. 開啟主控台，網址為 <https://console.aws.amazon.com/ecs/v2>。
2. 在叢集頁面上，選擇要在其中建立服務的叢集。
3. 在服務下選擇建立。
4. 根據工作負載填入其他詳細資訊後，在 Service Connect 區段中，選擇使用 Service Connect。
5. 在命名空間欄位中，輸入共用命名空間的完整 ARN。

ARN 格式為：`arn:aws:servicediscovery:region:account-id:namespace/namespace-id`

6. 視需要針對服務類型 (用戶端或用戶端-伺服器) 進行剩餘的 Service Connect 設定。
7. 完成服務建立程序。

您也可以使用 AWS CLI AWS SDKs 設定服務，方法是在的 `namespace` 參數中指定共用命名空間 ARN `serviceConnectConfiguration`。

```
aws ecs create-service \
  --cluster my-cluster \
  --service-name my-service \
  --task-definition my-task-def \
  --service-connect-configuration '{
    "enabled": true,
    "namespace": "arn:aws:servicediscovery:us-west-2:123456789012:namespace/ns-
abcdef1234567890",
    "services": [{
      "portName": "web",
      "discoveryName": "my-service",
```

```
        "clientAliases": [{
            "port": 80,
            "dnsName": "my-service"
        }]
    }]
```

考量事項

搭配 Service Connect 使用共用 AWS Cloud Map 命名空間時，請考慮下列事項：

- AWS RAM 必須在 AWS 區域 您要使用共用命名空間的 中可用。
- 共用命名空間必須與 AWS 區域 Amazon ECS 服務和叢集位於相同的 中。
- 使用共用命名空間設定 Service Connect 時，必須使用命名空間 ARN，而不是 ID。
- 支援所有命名空間類型：HTTP、私有 DNS 與公有 DNS 命名空間。
- 如果撤銷共用命名空間的存取權，則需要與命名空間互動的 Amazon ECS 操作 (例如 CreateService、UpdateService 與 ListServicesByNamespace) 將會失敗。如需有關對共用命名空間許可問題進行疑難排解的詳細資訊，請參閱[使用共用 AWS Cloud Map 命名空間對 Amazon ECS Service Connect 進行故障診斷](#)。
- 若在共用私有 DNS 命名空間中使用 DNS 查詢進行服務探索：
 - 命名空間擁有者需要呼叫 create-vpc-association-authorization，並指定與命名空間關聯的私有託管區域的 ID 以及取用者的 VPC。

```
aws route53 create-vpc-association-authorization --hosted-zone-id Z1234567890ABC --vpc VPCRegion=us-east-1,VPCId=vpc-12345678
```

- 命名空間取用者需要呼叫 associate-vpc-with-hosted-zone，並指定私有託管區域的 ID。

```
aws route53 associate-vpc-with-hosted-zone --hosted-zone-id Z1234567890ABC --vpc VPCRegion=us-east-1,VPCId=vpc-12345678
```

- 只有命名空間擁有者可以管理資源共用。
- 命名空間取用者可以在共用命名空間內建立與管理服務，但無法修改命名空間本身。
- 無論是哪個帳戶建立服務，探索名稱在共用命名空間內都必須是唯一的。
- 共用命名空間中的服務可以探索和連線到其他可存取命名空間之 AWS 帳戶的服務。
- 為 Service Connect 啟用 TLS 並使用共用命名空間時，AWS 私有 CA 憑證認證機構 (CA) 的範圍僅限於該命名空間。撤銷共用命名空間的存取權時，CA 的存取權也會停止。

- 使用共用命名空間時，命名空間擁有者和取用者預設無法存取跨帳戶 Amazon CloudWatch 指標。目標指標只會發佈至擁有用戶端服務的帳戶。擁有用戶端服務的帳戶無法存取擁有用戶端伺服器服務的帳戶接收的指標，反之亦然。若要允許跨帳戶存取指標，請設定 CloudWatch 跨帳戶可觀測性。如需設定跨帳戶可觀測性的詳細資訊，請參閱《Amazon [CloudWatch 使用者指南](#)》中的 [CloudWatch 跨帳戶可觀測性](#)。Amazon CloudWatch 如需 Service Connect CloudWatch 指標的詳細資訊，請參閱 [Amazon ECS CloudWatch 指標](#)。

使用共用 AWS Cloud Map 命名空間對 Amazon ECS Service Connect 進行故障診斷

使用以下資訊對共用 AWS Cloud Map 命名空間和 Service Connect 的問題進行故障診斷。如需有關尋找錯誤訊息的詳細資訊，請參閱 [Amazon ECS 疑難排解](#)。

因缺少許可或撤銷命名空間存取權，會顯示與許可問題相關的錯誤訊息。

 Important

必須使用 `AWSRAMPermissionCloudMapECSFullPermission` 受管許可來共用命名空間，Service Connect 才能在命名空間正常運作。

錯誤訊息會以下列其中一種格式顯示：

An error occurred (ClientException) when calling the <OperationName> operation: User: arn:aws:iam::<account-id>:user/<user-name> is not authorized to perform: <ActionName> on resource: <ResourceArn> because no resource-based policy allows the <ActionName> action

下列案例可能會導致此格式的錯誤訊息：

叢集建立或更新失敗

當 Amazon ECS 操作如 `CreateCluster` 或因缺少 AWS Cloud Map 許可而 `UpdateCluster` 失敗時，就會發生這些問題。這些操作需要下列 AWS Cloud Map 動作的許可：

- `servicediscovery:GetNamespace`

確保取用者帳戶已接受資源共用邀請，且在 Service Connect 組態中使用了正確的命名空間 ARN。

服務建立或更新失敗

當 Amazon ECS 操作如 `CreateService` 或因缺少 AWS Cloud Map 許可而 `UpdateService` 失敗時，就會發生這些問題。這些操作需要下列 AWS Cloud Map 動作的許可：

- `servicediscovery:CreateService`
- `servicediscovery:GetNamespace`
- `servicediscovery:GetOperation` (用於建立新的 AWS Cloud Map 服務)
- `servicediscovery:GetService` (用於 AWS Cloud Map 服務已存在時)

確保取用者帳戶已接受資源共用邀請，且在 Service Connect 組態中使用了正確的命名空間 ARN。

ListServicesByNamespace 操作失敗

當 Amazon ECS ListServicesByNamespace 操作失敗時，會發生此問題。此操作需要下列 AWS Cloud Map 動作的許可：

- `servicediscovery:GetNamespace`

若要解決此問題：

- 確認取用者帳戶具有 `servicediscovery:GetNamespace` 許可。
- 呼叫 API 時使用命名空間 ARN，而不是名稱。
- 確保資源共用處於作用中狀態，且已接受邀請。

User: <iam-user> is not authorized to perform: <ActionName> on resource: <ResourceArn> with an explicit deny in an identity-based policy.

下列案例可能會導致此格式的錯誤訊息：

服務刪除失敗並停滯在 **DRAINING** 狀態

當 Amazon ECS DeleteService 操作因撤銷命名空間存取權而缺少 `servicediscovery:DeleteService` 許可時，會發生此問題。服務最初可能看起來已成功刪除，但會停滯在 DRAINING 狀態。錯誤訊息會顯示為 Amazon ECS 服務事件。

若要解決此問題，命名空間擁有者必須與取用者帳戶共用命名空間，才能完成服務刪除。

服務中的任務執行失敗

當任務因缺少許可而啟動失敗時，會發生此問題。錯誤訊息會顯示為已停止的任務錯誤。如需詳細資訊，請參閱[解決 Amazon ECS 已停止任務錯誤](#)。

執行任務需要下列 AWS Cloud Map 動作：

- `servicediscovery:GetOperation`
- `servicediscovery:RegisterInstance`

確保取用者帳戶具有所需許可，並且可以存取共用命名空間。

任務無法完全停止或停滯在 **DEACTIVATING** 或 **DEPROVISIONING** 狀態

當任務在關閉期間因為缺少許可而無法從 AWS Cloud Map 服務取消註冊時，就會發生此問題。錯誤會在任務附件中顯示為 `statusReason`，可使用 `DescribeTasks` API 擷取附件。如需詳細資訊，請參閱 Amazon Elastic Container Service API Reference 中的 [DescribeTasks](#)。

停止任務需要下列 AWS Cloud Map 動作：

- `servicediscovery:DeregisterInstance`
- `servicediscovery:GetOperation`

如果撤銷共用命名空間存取權，任務可能會保持 **DEACTIVATING** 或 **DEPROVISIONING** 狀態，直到命名空間存取權還原。需請求命名空間擁有者還原命名空間的存取權。

Amazon ECS Service Connect 存取日誌

Amazon ECS Service Connect 支援存取日誌，以提供 Service Connect Proxy 處理之個別請求的詳細遙測。存取日誌透過擷取每個請求流量中繼資料來補充現有的應用程式日誌，例如 HTTP 方法、路徑、回應碼、旗標和時間資訊。這可讓對請求層級流量模式和服務互動進行更深入的可觀測性，以有效地進行故障診斷和監控。

若要啟用存取日誌，請在 `serviceConnectConfiguration` 物件中指定 `logConfiguration` 和 `accessLogConfiguration` 物件。您可以設定日誌的格式，以及日誌是否應該在 中包含查詢參數 `accessLogConfiguration`。日誌會由 中指定的日誌驅動程式交付至目的地日誌群組 `logConfiguration`。

```
{
  "serviceConnectConfiguration": {
    "enabled": true,
    "namespace": "myapp.namespace",
    "services": [
      ...
    ],
    "logConfiguration": {
      "logDriver": "awslogs",
      "options": {
        "awslogs-group": "my-envoy-log-group",
        "awslogs-region": "us-west-2",
```

```

        "awslogs-stream-prefix": "myapp-envoy-logs"
    },
    "accessLogConfiguration": {
        "format": "TEXT",
        "includeQueryParameters": "ENABLED"
    }
}
}

```

考量事項

當您啟用存取日誌的存取權時，請考慮下列事項

- 存取日誌和應用程式日誌都會寫入 `/dev/stdout`。若要將存取日誌與應用程式日誌分開，建議使用具有自訂 Fluent Bit 或 Fluentd 組態的 `awsfirelens` 日誌驅動程式。
- 建議使用 `awslogs` 日誌驅動程式將應用程式和存取日誌傳送至相同的 CloudWatch 目的地。
- 使用平台版本 1.4.0 和更高版本的 Fargate 服務支援存取日誌。
- 根據預設，請求 ID 和字符等查詢參數會從存取日誌中排除。若要在存取日誌中包含查詢參數，請將 `includeQueryParameters` 設定為 `"ENABLED"`。

存取日誌格式

存取日誌可以使用 JSON 格式字典或文字格式字串進行格式化，不同類型存取日誌的支援命令運算子不同。

HTTP 存取日誌

HTTP 日誌預設包含下列命令運算子：

Text

```

[%START_TIME%] "%REQ(:METHOD)% %REQ(X-ENVOY-ORIGINAL-PATH?:PATH)% %PROTOCOL%"
%RESPONSE_CODE% %BYTES_RECEIVED% %BYTES_SENT% %DURATION%
%RESP(X-ENVOY-UPSTREAM-SERVICE-TIME)% "%REQ(X-FORWARDED-FOR)% " "%REQ(USER-AGENT)% "
"%REQ(X-REQUEST-ID)% " "%REQ(:AUTHORITY)% " "%UPSTREAM_HOST%" \n

```

JSON

```

{

```



```

"start_time": "%START_TIME%",
"method": "%REQ(:METHOD)%",
"path": "%REQ(X-ENVOY-ORIGINAL-PATH?:PATH)%",
"protocol": "%PROTOCOL%",
"response_code": "%RESPONSE_CODE%",
"bytes_received": "%BYTES_RECEIVED%",
"bytes_sent": "%BYTES_SENT%",
"duration_ms": "%DURATION%",
"upstream_service_time": "%RESP(X-ENVOY-UPSTREAM-SERVICE-TIME)%",
"forwarded_for": "%REQ(X-FORWARDED-FOR)%",
"user_agent": "%REQ(USER-AGENT)%",
"request_id": "%REQ(X-REQUEST-ID)%",
"authority": "%REQ(:AUTHORITY)%",
"upstream_host": "%UPSTREAM_HOST%"
}

```

HTTP2 存取日誌

除了 HTTP 日誌包含的命令運算子之外，HTTP2 日誌預設還包含 %STREAM_ID%運算子。

Text

```

[%START_TIME%] "%REQ(:METHOD)% %REQ(X-ENVOY-ORIGINAL-PATH?:PATH)% %PROTOCOL%"
%RESPONSE_CODE% %BYTES_RECEIVED% %BYTES_SENT% %DURATION%
%RESP(X-ENVOY-UPSTREAM-SERVICE-TIME)% "%REQ(X-FORWARDED-FOR)%" "%REQ(USER-AGENT)%"
"%REQ(X-REQUEST-ID)%" "%REQ(:AUTHORITY)%" "%UPSTREAM_HOST%" "%STREAM_ID%"`n

```

JSON

```

{
  "start_time": "%START_TIME%",
  "method": "%REQ(:METHOD)%",
  "path": "%REQ(X-ENVOY-ORIGINAL-PATH?:PATH)%",
  "protocol": "%PROTOCOL%",
  "response_code": "%RESPONSE_CODE%",
  "bytes_received": "%BYTES_RECEIVED%",
  "bytes_sent": "%BYTES_SENT%",
  "duration": "%DURATION%",
  "upstream_service_time": "%RESP(X-ENVOY-UPSTREAM-SERVICE-TIME)%",
  "forwarded_for": "%REQ(X-FORWARDED-FOR)%",
  "user_agent": "%REQ(USER-AGENT)%",
  "request_id": "%REQ(X-REQUEST-ID)%",

```

```

"authority": "%REQ(:AUTHORITY)%",
"upstream_host": "%UPSTREAM_HOST%",
"stream_id": "%STREAM_ID%"
}

```

gRPC 存取日誌

除了 HTTP 日誌包含的命令運算子之外，gRPC 存取日誌預設還包含 %STREAM_ID% 和 %GRPC_STATUS()% 運算子。

Text

```

[%START_TIME%] "%REQ(:METHOD)% %REQ(X-ENVOY-ORIGINAL-PATH?:PATH)% %PROTOCOL%"
%RESPONSE_CODE% %GRPC_STATUS()% %BYTES_RECEIVED% %BYTES_SENT% %DURATION%
%RESP(X-ENVOY-UPSTREAM-SERVICE-TIME)% "%REQ(X-FORWARDED-FOR)%" "%REQ(USER-AGENT)%"
"%REQ(X-REQUEST-ID)%" "%REQ(:AUTHORITY)%" "%UPSTREAM_HOST%" "%STREAM_ID%" \n

```

JSON

```

{
  "start_time": "%START_TIME%",
  "method": "%REQ(:METHOD)%",
  "path": "%REQ(X-ENVOY-ORIGINAL-PATH?:PATH)%",
  "protocol": "%PROTOCOL%",
  "response_code": "%RESPONSE_CODE%",
  "grpc_status": "%GRPC_STATUS()",
  "bytes_received": "%BYTES_RECEIVED%",
  "bytes_sent": "%BYTES_SENT%",
  "duration": "%DURATION%",
  "upstream_service_time": "%RESP(X-ENVOY-UPSTREAM-SERVICE-TIME)%",
  "forwarded_for": "%REQ(X-FORWARDED-FOR)%",
  "user_agent": "%REQ(USER-AGENT)%",
  "request_id": "%REQ(X-REQUEST-ID)%",
  "authority": "%REQ(:AUTHORITY)%",
  "upstream_host": "%UPSTREAM_HOST%",
  "stream_id": "%STREAM_ID%"
}

```

TCP 存取日誌

根據預設，TCP 存取日誌中會包含下列命令運算子：

Text

```
[%START_TIME%] %DOWNSTREAM_REMOTE_ADDRESS% %DOWNSTREAM_REMOTE_PORT%
%BYTES_RECEIVED% %BYTES_SENT% %DURATION%
%CONNECTION_TERMINATION_DETAILS% %CONNECTION_ID%\n
```

JSON

```
{
  "start_time": "%START_TIME%",
  "downstream_remote_address": "%DOWNSTREAM_REMOTE_ADDRESS%",
  "downstream_remote_port": "%DOWNSTREAM_REMOTE_PORT%",
  "bytes_received": "%BYTES_RECEIVED%",
  "bytes_sent": "%BYTES_SENT%",
  "duration": "%DURATION%",
  "connection_termination_details": "%CONNECTION_TERMINATION_DETAILS%",
  "connection_id": %CONNECTION_ID%
}
```

如需這些命令運算子的詳細資訊，請參閱 Envoy 文件中的[命令運算子](#)。

啟用 Amazon ECS Service Connect 的存取日誌

使用 Service Connect 的 Amazon ECS 服務預設不會啟用存取日誌。您可以透過下列方式啟用存取日誌。

使用 啟用存取日誌 AWS CLI

下列命令顯示如何在建立服務accessLogConfiguration時指定 AWS CLI，以使用 啟用 Amazon ECS 服務的存取日誌：

```
aws ecs create-service \
  --cluster my-cluster \
  --service-name my-service \
  --task-definition my-task-def \
  --service-connect-configuration '{
    "enabled": true,
    "namespace": "arn:aws:servicediscovery:us-west-2:123456789012:namespace/ns-
abcdef1234567890",
    "services": [{
      "portName": "web",
```

```
    "discoveryName": "my-service",
    "clientAliases": [{
      "port": 80,
      "dnsName": "my-service"
    }]
  },
  "logConfiguration": {
    "logDriver": "awslogs",
    "options": {
      "awslogs-group": "my-envoy-log-group",
      "awslogs-region": "us-west-2",
      "awslogs-stream-prefix": "myapp-envoy-logs"
    }
  },
  "accessLogConfiguration": {
    "format": "TEXT",
    "includeQueryParameters": "ENABLED"
  }
}'
```

使用主控台啟用存取日誌

如需詳細的服務建立程序，請參閱[建立 Amazon ECS 滾動更新部署](#)。

使用 建立具有共用命名空間的服務 AWS 管理主控台

1. 開啟主控台，網址為 <https://console.aws.amazon.com/ecs/v2>。
2. 在叢集頁面上，選擇要在其中建立服務的叢集。
3. 在服務下選擇建立。
4. 根據工作負載填入其他詳細資訊後，在 Service Connect 區段中，選擇使用 Service Connect。
5. 根據您的服務類型（用戶端或用戶端伺服器）視需要設定 Service Connect 設定。
6. 展開存取日誌組態。針對格式，選擇 JSON 或 TEXT。
7. 若要在存取日誌中包含查詢參數，請選取包含查詢參數。
8. 完成服務建立程序。

加密 Amazon ECS Service Connect 流量

Amazon ECS Service Connect 支援使用 Transport Layer Security (TLS) 憑證，為 Amazon ECS 服務自動加密流量。當您將 Amazon ECS 服務指向 [AWS 私有憑證授權單位 \(AWS 私有 CA\)](#) 時，Amazon

ECS 會自動佈建 TLS 憑證，以加密 Amazon ECS Service Connect 服務之間的流量。Amazon ECS 會產生、輪換與分發用於流量加密的 TLS 憑證。

Service Connect 的自動流量加密使用業界領先的加密功能來保護服務間通訊，協助您滿足安全需求。它支援使用 AWS 私有憑證授權單位 256-bit ECDSA 和 2048-bit RSA 加密的 TLS 憑證。您還可以完全控制私有憑證與簽署金鑰，協助您滿足合規需求。依預設，支援 TLS 1.3，但不支援 TLS 1.0 - 1.2。Service Connect 支援具有下列密碼的 TLS 1.3：

- TLS_AES_128_GCM_SHA256
- TLS_AES_256_GCM_SHA384
- TLS_CHACHA20_POLY1305_SHA256

Note

若要使用 TLS 1.3，必須在目標的接聽程式上將其啟用。
只有透過 Amazon ECS 代理程式傳遞的傳入與傳出流量才會加密。

Service Connect 與 Application Load Balancer 運作狀態檢查

您可以搭配 Application Load Balancer 運作狀態檢查與 TLS 1.3 加密使用 Service Connect。

Application Load Balancer 組態

使用下列設定來設定 Application Load Balancer：

- 使用 TLS 1.3 安全政策 (例如 `ELBSecurityPolicy-TLS13-1-2-2021-06`) 設定 TLS 接聽程式。如需詳細資訊，請參閱 [Security policies for your Application Load Balancer](#)。
- 使用下列設定來建立目標群組：
 - 將通訊協定設定為 HTTPS
 - 將目標群組連接至 TLS 接聽程式
 - 將運作狀態檢查連接埠設定為符合 Service Connect 服務的容器連接埠需求

Service Connect 組態

使用下列設定來設定服務：

- 將服務設定為使用 awsvpc 網路模式，因為不支援 bridge 網路模式。

- 為服務啟用 Service Connect。
- 使用下列設定來設定負載平衡器組態：
 - 指定為 Application Load Balancer 設定的目標群組
 - 將容器連接埠設定為符合 Service Connect TLS 服務的容器連接埠需求
- 避免為服務設定 `ingressPortOverride`。如需詳細資訊，請參閱 Amazon Elastic Container Service API Reference 中的 [ServiceConnectService](#)。

考量事項

使用 Application Load Balancer、TLS 與 Service Connect 時，請考量下列事項：

- 搭配 TLS 加密使用 Service Connect 時，需使用 `awsvpc` 網路模式 (而不是 `bridge` 網路模式) 進行 HTTPS 運作狀態檢查。HTTP 運作狀態檢查將繼續使用 `bridge` 模式。
- 將目標群組運作狀態檢查連接埠設定為符合 Service Connect 服務的容器連接埠 (而不是預設的 HTTPS 連接埠 (443)) 需求。

AWS 私有憑證授權單位 憑證和 Service Connect

您需要擁有基礎結構 IAM 角色。如需有關此角色的詳細資訊，請參閱 [Amazon ECS 基礎結構 IAM 角色](#)。

AWS 私有憑證授權單位 適用於 Service Connect 的模式

AWS 私有憑證授權單位 可以兩種模式執行：一般用途和短期。

- 一般用途 – 可設定任何過期日期的憑證。
- 短期 – 有效性上限為七天的憑證。

儘管 Amazon ECS 支援這兩種模式，但建議使用短期憑證。依預設，憑證每五天輪換一次，相較於一般用途模式，在短期模式下執行可大幅節省成本。

Service Connect 不支援撤銷憑證，而是改為利用會頻繁輪換的短期憑證。您有權使用 [Secrets Manager](#) 中的 [受管輪換](#) 來修改輪換頻率、停用或刪除秘密，但這樣做可能會產生下列後果。

- 較短的輪換頻率 - 由於 AWS KMS Secrets Manager AWS 私有 CA 和 Auto Scaling 的輪換工作負載增加，較短的輪換頻率會產生更高的成本。
- 較長的輪換頻率 – 如果輪換頻率超過七天，應用程式通訊將會失敗。

- 刪除秘密 – 刪除秘密會導致輪換失敗並影響客戶應用程式通訊。

如果秘密輪換失敗，會在 [AWS CloudTrail](#) 中發布 `RotationFailed` 事件。您還可以為 `RotationFailed` 設定 [CloudWatch 警示](#)。

Important

請勿將複本區域新增至秘密。這樣做會阻止 Amazon ECS 刪除秘密，因為 Amazon ECS 沒有從複寫中移除區域的許可。若已新增複寫，請執行下列命令。

```
aws secretsmanager remove-regions-from-replication \  
--secret-id SecretId \  
--remove-replica-regions region-name
```

次級憑證授權機構

您可以將任何 AWS 私有 CA 根憑證或次級憑證帶到 Service Connect TLS，以發行服務的終端實體憑證。提供的發行者會視為各地的簽署者與信任根。您可以從不同的次級 CA 為應用程式的不同部分發行終端實體憑證。使用時 AWS CLI，請提供 CA 的 Amazon Resource Name (ARN) 來建立信任鏈。

內部部署憑證授權機構

若要使用內部部署 CA，需要在 AWS 私有憑證授權單位中建立與設定次級 CA。這可確保為 Amazon ECS 工作負載發行的所有 TLS 憑證與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

在 `AmazonECSManaged : true` 中新增必要的標籤 AWS 私有 CA。

基礎設施即程式碼

搭配基礎結構即程式碼 (IaC) 工具使用 Service Connect TLS 時，務必正確設定相依性以避免問題，例如服務停滯在耗盡狀態。如果 AWS KMS 提供金鑰，則應在 Amazon ECS 服務之後刪除 IAM 角色和 AWS 私有 CA 相依性。

如果用於 Service Connect 的命名空間是共用命名空間，您可以選擇使用共用 AWS 私有 CA 資源。如需詳細資訊，請參閱 AWS 私有憑證授權單位 User Guide 中的 [Attach a policy for cross-account access](#)。

Service Connect 與 Secrets Manager

搭配 TLS 加密使用 Amazon ECS Service Connect 時，服務會透過下列方式與 Secrets Manager 互動：

Service Connect 會利用提供的基礎結構角色，在 Secrets Manager 中建立秘密。這些秘密用於儲存 TLS 憑證的關聯私有金鑰，以加密 Service Connect 服務之間的流量。

Warning

Service Connect 會自動建立與管理這些秘密，從而簡化為服務實作 TLS 加密的程序。但是，務必注意潛在的安全隱患。具有 Secrets Manager 讀取存取權的其他 IAM 角色可能可以存取這些自動建立的秘密。如果未正確設定存取控制，這可能會向未經授權的一方公開敏感的加密資料。

若要降低此風險，請遵循下列最佳實務：

- 小心管理與限制 Secrets Manager 存取權，尤其是 Service Connect 建立的秘密。
- 定期稽核 IAM 角色及其許可，確保遵循最低權限原則。

授予 Secrets Manager 讀取存取權時，請考慮排除 Service Connect 建立的 TLS 私有金鑰。為此，您可以透過在 IAM 政策中使用條件，來排除具備符合下列模式之 ARN 的秘密：

```
"arn:aws:secretsmanager:::secret:ecs-sc!"
```

以下是一個 IAM 政策範例，該政策拒絕對所有以 GetSecretValue 為字首的秘密執行 ecs-sc! 動作：

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": "secretsmanager:GetSecretValue",
      "Resource": "arn:aws:secretsmanager:*:*:secret:ecs-sc!*"
    }
  ]
}
```



```
}
```

Note

這是一般範例，可能需要根據您的特定使用案例和 AWS 帳戶組態進行調整。務必全面測試 IAM 政策，確保它們在維護安全性的同時提供預期的存取權。

透過了解 Service Connect 如何與 Secrets Manager 互動，您可以更好地管理 Amazon ECS 服務的安全性，同時利用自動 TLS 加密的優勢。

Service Connect 和 AWS Key Management Service

您可以使用 [AWS Key Management Service](#) 來加密和解密 Service Connect 資源。AWS KMS 是一項由管理的服務，您可以在 AWS 其中建立和管理加密金鑰來保護您的資料。

AWS KMS 搭配 Service Connect 使用時，您可以選擇使用 AWS 為您 AWS 管理的擁有金鑰，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 Service Connect 中指定該金鑰的 Amazon Resource Name (ARN)。

以下是範例 AWS KMS 政策。將 *user-input* 值取代實際值。

JSON

```
{
  "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": "*"
}
]
```

如需有關金鑰政策的詳細資訊，請參閱 AWS Key Management Service Developer Guide 中的 [Creating a key policy](#)。

 Note

Service 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 Service 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 [識別非對稱 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 Service Connect 啟用 TLS

您可以在建立或更新 Service Connect 服務時啟用流量加密。

使用 啟用現有命名空間中服務的流量加密 AWS 管理主控台

1. 您需要擁有基礎結構 IAM 角色。如需有關此角色的詳細資訊，請參閱 [Amazon ECS 基礎結構 IAM 角色](#)。
2. 開啟主控台，網址為 <https://console.aws.amazon.com/ecs/v2>。
3. 在導覽窗格中，選擇 Namespaces (命名空間)。
4. 選擇要為其啟用流量加密之服務所在的命名空間。
5. 選擇要啟用流量加密的服務。
6. 選擇右上角的更新服務，然後向下捲動至「Service Connect」區段。
7. 在服務資訊下方，選擇開啟流量加密以啟用 TLS。
8. 在 Service Connect TLS 角色欄位中，選擇現有的基礎結構 IAM 角色或建立一個新角色。

- 在簽署者憑證授權機構欄位中，選擇現有的憑證授權機構或建立一個新的憑證授權機構。

如需詳細資訊，請參閱[AWS 私有憑證授權單位 憑證和 Service Connect](#)。

- 針對選擇 AWS KMS key，選擇 AWS 擁有和管理的金鑰，或者您可以選擇不同的金鑰。您也可以選擇建立一個新金鑰。

如需使用 AWS CLI 為您的服務設定 TLS 的範例，請參閱[使用 設定 Amazon ECS Service Connect AWS CLI](#)。

驗證 Amazon ECS Service Connect 是否啟用 TLS

Service Connect 會在 Service Connect 代理程式啟動 TLS，並在目的地代理程式將其終止。因此，應用程式程式碼絕不會看到 TLS 互動。使用下列步驟來驗證 TLS 是否已啟用。

- 在應用程式映像中包含 openssl CLI。
- 在服務上啟用 [ECS Exec](#)，以透過 SSM 連線至任務。或者，您也可以在与服務相同的 Amazon VPC 中啟動 Amazon EC2 執行個體。
- 從要驗證的服務中擷取任務的 IP 與連接埠。您可以在 AWS Cloud Map 主控台中擷取任務 IP 地址。該資訊位於命名空間的服務詳細資訊頁面上。
- 使用 `execute-command` 登入任何任務，如下列範例所示。或者，登入在步驟 2 中建立的 Amazon EC2 執行個體。

```
$ aws ecs execute-command --cluster cluster-name \  
    --task task-id \  
    --container container-name \  
    --interactive \  
    --command "/bin/sh"
```

Note

直接呼叫 DNS 名稱不會顯示憑證。

- 在已連線的 Shell 中，使用 openssl CLI 來驗證與檢視連接至任務的憑證。

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>

```

使用 設定 Amazon ECS Service Connect AWS CLI

您可以搭配 AWS CLI 使用 Service Connect，來建立包含 Fargate 任務的 Amazon ECS 服務。

Note

您可以使用雙堆疊服務端點，透過 IPv4 和 IPv6 AWS CLI 從、SDKs 和 Amazon ECS API 與 Amazon ECS 互動。如需詳細資訊，請參閱[使用 Amazon ECS 雙堆疊端點](#)。

先決條件

下列是使用 Service Connect 的先決條件：

- 確認 AWS CLI 已安裝並設定最新版本的。如需詳細資訊，請參閱 [Installing or updating to the latest version of the AWS CLI](#)。
- 您的 IAM 使用者擁有 [AmazonECS_FullAccess](#) IAM 政策範例中指定的所需許可。
- 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 Service Connect Proxy 日誌寫入 Amazon CloudWatch Logs。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。

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，名稱的 AWS Cloud Map 命名空間 `service-connect` 不存在於此帳戶中 AWS 區域，因此命名空間是由 Amazon ECS 建立。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. (選用) 確認已在其中建立命名空間 AWS Cloud Map。您可以在建立時，使用 AWS 管理主控台或正常 AWS CLI 組態 AWS Cloud Map。

例如，使用 AWS CLI：

```
aws servicediscovery 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，將 Service Connect 設定為針對此服務的流量提供服務探索與服務網格代理。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 Service 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 logConfiguration，將 nginx 輸出從 stdout 和 stderr 傳送到 Amazon CloudWatch Logs。此任務執行角色沒有所需的額外許可，無法建立

CloudWatch Logs 日誌群組。使用 AWS 管理主控台 或 在 CloudWatch Logs 中建立日誌群組 AWS CLI。如果不想將 nginx 日誌傳送到 CloudWatch Logs，您可以移除 logConfiguration。

將任務執行角色中的 AWS 帳戶 ID 取代為您的 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。

建立 ECS 服務時，請指定 Fargate 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 `tls` 組態，請為 `awsPcaAuthorityArn`、`kmsKey` 與 IAM 角色的 `roleArn` 新增憑證 `arn`。

此服務會使用 `logConfiguration`，將 Service Connect Proxy 輸出從 `stdout` 和 `stderr` 傳送至 Amazon CloudWatch Logs。此任務執行角色沒有所需的額外許可，無法建立 CloudWatch Logs 日誌群組。使用 AWS 管理主控台 或在 CloudWatch Logs 中建立日誌群組 AWS CLI。我們建議您建立此日誌群組，並將 Proxy 日誌儲存在 CloudWatch Logs 中。如果不想將 Proxy 日誌傳送到 CloudWatch Logs，您可以移除 `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 `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",
"enableECSTags": 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，查看 CloudWatch 中由 Service Connect Proxy 建立的其他指標。

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 取得任務 ID，方法是使用 `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。不過，公有 IP 無法從 `describe-tasks` 取得，因此這些步驟涉及前往 Amazon EC2 AWS 管理主控台 或 AWS CLI 以取得彈性網路界面的詳細資訊。

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

Service Connect Proxy 會在 CloudWatch 指標中建立應用程式 (HTTP、HTTP2、gRPC 或 TCP 連線) 指標。使用 CloudWatch 主控台時，請參閱 Amazon ECS 命名空間下的 DiscoveryName、(DiscoveryName、ServiceName、ClusterName)、TargetDiscoveryName 與 (TargetDiscoveryName、ServiceName、ClusterName) 的其他指標維度。如需有關這些指標與維度的詳細資訊，請參閱 Amazon CloudWatch Logs User Guide 中的 [View Available Metrics](#)。

使用服務探索以利用 DNS 名稱連接 Amazon ECS 服務

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 Developer Guide 中的 [What Is AWS Cloud Map](#)。

服務探索可在下列 AWS 區域使用：

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
Africa (Cape Town)	af-south-1
亞太地區 (香港)	ap-east-1
亞太區域 (台北)	ap-east-2
亞太區域 (孟買)	ap-south-1
亞太地區 (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
亞太地區 (大阪)	ap-northeast-3
亞太區域 (新加坡)	ap-southeast-1
亞太區域 (雪梨)	ap-southeast-2
亞太地區 (雅加達)	ap-southeast-3
亞太地區 (墨爾本)	ap-southeast-4

區域名稱	區域
亞太地區 (馬來西亞)	ap-southeast-5
亞太區域 (紐西蘭)	ap-southeast-6
亞太區域 (泰國)	ap-southeast-7
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
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
Europe (Stockholm)	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
墨西哥 (中部)	mx-central-1
Middle East (Bahrain)	me-south-1
南美洲 (聖保羅)	sa-east-1

區域名稱	區域
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，這是要路由流量的位置。您可以透過呼叫 `aws servicediscovery create-private-dns-namespace` 命令或 Amazon ECS 主控台建立命名空間。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。如需服務探索命令的詳細資訊，請參閱《AWS Cloud Map (服務探索) AWS CLI 參考指南[list-namespaces](#)》中的[create-private-dns-namespace](#)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：
 - **服務登錄檔**：可讓您透過 DNS 或 AWS Cloud Map API 動作查詢服務，並取得一或多個可用來連線至服務的可用端點。
- **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** – 對於 A 記錄，Route 53 傳回以回應 DNS 查詢的 IPv4 地址，並在探索執行個體詳細資訊時 AWS Cloud Map 傳回，例如 `192.0.2.44`。
 - **AWS_INSTANCE_IPV6**– 針對 AAAA 記錄，Route 53 傳回以回應 DNS 查詢的 IPv6 地址，並在探索執行個體詳細資訊時 AWS Cloud Map 傳回，例如 `2001:0db8:85a3:0000:0000:abcd:0001:2345`。對於 Amazon ECS 雙堆疊服務，會同時新增 **AWS_INSTANCE_IPv4** 與 **AWS_INSTANCE_IPv6**。對於 Amazon ECS 純 IPv6 服務，僅會新增 **AWS_INSTANCE_IPv6**。
 - **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 的任務，這是容器執行個體所在的可用區域。對於使用 Fargate 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，則不會新增此自訂屬性。
- **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 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](#)。
- Amazon ECS 不支援在共用 AWS Cloud Map 命名空間中註冊服務。
- 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)。
- 如果服務任務定義使用 awsvpc 網路模式，您可以為每項服務任務建立任意組合的 A 或 SRV 記錄。如果使用 SRV 記錄，必須具備連接埠。如果服務使用雙堆疊子網路，可以另外建立 AAAA 記錄。如果服務使用純 IPv6 子網路，則無法建立 A 記錄。
- 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 與 AAAA 記錄會傳回一組對應至任務的 IP 位址。SRV 記錄會為每項任務傳回一組 IP 位址與連接埠。

- 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 代替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 參考中的 [HealthCheckCustomConfig](#)。
- 使用服務探索時建立 AWS Cloud Map 的資源必須手動清除。
- 任務與執行個體會註冊為 UNHEALTHY，直到容器運作狀態檢查傳回值。如果運作狀態檢查通過，狀態會更新為 HEALTHY。如果容器運作狀態檢查失敗，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 會執行容器層級的運作狀態檢查，並將其公開給 AWS Cloud Map 自訂的運作狀態檢查 API 操作。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

了解如何使用 AWS CLI 建立一項服務，其中包含使用服務探索的 Fargate 任務。

如需 AWS 區域 支援服務探索的清單，請參閱 [使用服務探索以利用 DNS 名稱連接 Amazon ECS 服務](#)。

如需支援 Fargate 的區域的資訊，請參閱 [the section called “AWS Fargate 區域”](#)。

Note

您可以使用雙堆疊服務端點，透過 IPv4 和 IPv6 AWS CLI 從、SDKs 和 Amazon ECS API 與 Amazon ECS 互動。如需詳細資訊，請參閱 [使用 Amazon ECS 雙堆疊端點](#)。

先決條件

在您開始教學課程之前，請務必先達成以下先決條件：

- AWS CLI 已安裝並設定最新版本的。如需詳細資訊，請參閱 [Installing or updating to the latest version of the AWS CLI](#)。
- [設定以使用 Amazon ECS](#)。中所述的步驟已完成。
- 您的 IAM 使用者擁有 [AmazonECS_FullAccess](#) IAM 政策範例中指定的所需許可。
- 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": "h2qe3s6dxftvvt7riu6lfy2f6c3j1hf4-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 h2qe3s6dxftvvt7riu6lfy2f6c3j1hf4-je6chs2e
```

輸出如下。

```
{  
  "Operation": {  
    "Id": "h2qe3s6dxftvvt7riu6lfy2f6c3j1hf4-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": "public.ecr.aws/docker/library/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\""  
      ]  
    },  
    {  
      "name": "sample-app",  
      "image": "public.ecr.aws/docker/library/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"  
  ]  
}
```



```

        "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。

建立 ECS 服務時，請指定 Fargate 以及支援服務探索的 LATEST 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：在 中驗證服務探索 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 操作，或從 VPC dig 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
```

```
--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": "Z35JQ4ZFDYPLV"  
      }  
    },  
    "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 Z35JQ4ZFDYPLV
```

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": "CJP2A2M86XW30"  
    }  
  }  
}
```

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
```

```
--cluster tutorial \  
--service ecs-service-discovery
```

8. 刪除 Amazon ECS 叢集：

```
aws ecs delete-cluster \  
--cluster tutorial
```

使用 Amazon VPC Lattice 連接、觀察和保護您的 Amazon ECS 服務

Amazon VPC Lattice 是一種全受管應用程式聯網服務，可讓 Amazon ECS 客戶觀察、保護和監控跨 AWS 運算服務、VPCs 和帳戶建置的應用程式，而無需進行任何程式碼變更。

VPC Lattice 使用目標群組，這是運算資源的集合。這些目標會執行您的應用程式或服務，可以是 Amazon EC2 執行個體、IP 地址、Lambda 函數和 Application Load Balancer。透過將其 Amazon ECS 服務與 VPC Lattice 目標群組建立關聯，客戶現在可以在 VPC Lattice 中將 Amazon ECS 任務啟用為 IP 目標。當已註冊服務的任務啟動時，Amazon ECS 會自動將任務註冊到 VPC Lattice 目標群組。

Note

使用五個 VPC Lattice 組態時，部署時間可能會比使用較少組態時稍長。

接聽程式規則用於在符合條件時將流量轉送至指定目標群組。接聽程式會使用所設定之連接埠上的通訊協定來檢查連線請求。服務會根據您在設定接聽程式時定義的規則，將請求路由至其已註冊目標。

Amazon ECS 還會根據 VPC Lattice 運作狀態檢查，自動取代運作狀態不良的任務。一旦與 VPC Lattice 相關聯，Amazon ECS 客戶也可以利用 VPC Lattice 中的許多其他跨運算連線、安全性和可觀測性功能，例如跨叢集、VPCs 和帳戶連線至服務 AWS Resource Access Manager、授權和身分驗證的 IAM 整合，以及進階流量管理功能。

Amazon ECS 客戶可以透過下列方式受益於 VPC Lattice。

- 提高開發人員生產力 – VPC Lattice 可讓您專注於建置功能，同時由 VPC Lattice 以統一的方式處理所有運算平台上的聯網、安全性與可觀測性挑戰，從而提高開發人員生產力。
- 強化安全狀態 – VPC Lattice 可讓開發人員輕鬆驗證與保護跨應用程式與運算平台的通訊、強制執行傳輸中加密，並透過 VPC Lattice 驗證政策套用精細存取控制。這可以強化安全狀態，以滿足業界領先的法規與合規需求。

- 改善應用程式可擴展性與恢復能力 – VPC Lattice 可讓您建立已部署應用程式的網路，並提供以路徑為基礎的路由、以標頭為基礎的路由與以方法為基礎的路由、身分驗證、授權與監控等功能。這些優勢不會對工作負載造成資源負荷，並且可以支援每秒產生數百萬個請求的多叢集部署，而不會增加明顯的延遲。
- 異質基礎結構的部署彈性 – VPC Lattice 在所有運算服務中提供一致的功能，例如 Amazon ECS、Fargate、Amazon EC2、Amazon EKS 與 Lambda，便於組織彈性地為每個應用程式選擇合適的基礎結構。

VPC Lattice 如何搭配 Amazon ECS 服務運作

搭配 Amazon ECS 使用 VPC Lattice 時，某些 Amazon ECS 服務的使用方式可能會變更，而某些服務則會保持不變。

Application Load Balancer

您不再需要建立特定的 Application Load Balancer，以用於 VPC Lattice 中的 Application Load Balancer 目標群組類型，然後再連結至 Amazon ECS 服務。只需將 Amazon ECS 服務設定為使用 VPC Lattice 目標群組即可。您也可以選擇同時搭配 Amazon ECS 使用 Application Load Balancer。

Amazon ECS 滾動部署

只有 Amazon ECS 滾動部署能搭配 VPC Lattice 運作，並且 Amazon ECS 會在部署期間安全地將任務引入服務和從服務中移除。不支援 CodeDeploy 與藍/綠部署。

若要進一步了解 VPC Lattice，請參閱 [Amazon VPC Lattice User Guide](#)。

建立使用 VPC Lattice 的服務

您可以使用 AWS 管理主控台 或 AWS CLI 來建立具有 VPC Lattice 的服務。

先決條件

在您開始教學課程之前，請務必先達成以下先決條件：

- AWS CLI 已安裝並設定最新版本的。如需詳細資訊，請參閱[安裝 AWS Command Line Interface](#)。

Note

您可以使用雙堆疊服務端點，透過 IPv4 和 IPv6 AWS CLI 從、SDKs 和 Amazon ECS API 與 Amazon ECS 互動。如需詳細資訊，請參閱[使用 Amazon ECS 雙堆疊端點](#)。

- [設定以使用 Amazon ECS](#)。中所述的步驟已完成。
- 您的 IAM 使用者擁有 [AmazonECS_FullAccess](#) IAM 政策範例中指定的所需許可。

建立搭配使用 VPC Lattice 的服務 AWS 管理主控台

請遵循下列步驟，透過 AWS 管理主控台建立使用 VPC Lattice 的服務。

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，選擇要建立服務的叢集。
4. 在 Services (服務) 索引標籤上，選擇 Create (建立)。

如果您之前從未建立服務，請遵循[使用主控台建立 Amazon ECS 服務](#)中的步驟，然後在到達 VPC Lattice 區段時繼續執行這些步驟。

5. 勾選按鈕以選擇開啟 VPC Lattice。
6. 若要使用現有角色，請在 Amazon ECS 的 ECS 基礎結構角色欄位中，選擇一個已建立的角色，以便在建立 VPC Lattice 目標群組時使用。若要建立新角色，請建立 ECS 基礎結構角色。
7. 選擇 VPC。

VPC 取決於您在註冊任務定義時選取的聯網模式。如果您搭配 EC2 使用 host 或 network 模式，請選擇 VPC。

對於 awsvpc 模式，系統會根據您在聯網下選擇的 VPC 自動選取 VPC，且無法變更。

8. 在目標群組下，選擇一個或多個目標群組。您需要選擇至少一個目標群組，最多可以選擇五個。選擇新增目標群組以新增額外的目標群組。針對選擇的每個目標群組，選擇連接埠名稱、通訊協定與連接埠。若要刪除目標群組，請選擇移除。

Note

- 若要新增現有的目標群組，需要使用 AWS CLI。如需如何使用 新增目標群組的說明 AWS CLI，請參閱《AWS Command Line Interface 參考》中的 [register-targets](#)。
- 雖然 VPC Lattice 服務可以有多個目標群組，但每個目標群組只能新增至一項服務。
- 若要在純 IPv6 組態中建立服務，請選擇 IP 位址類型為 IPv6 的目標群組。

9. 此時，您會導覽至 VPC Lattice 主控台以繼續設定。在這裡，您可以將新的目標群組加入接聽程式的預設動作或現有 VPC Lattice 服務的規則中。

如需詳細資訊，請參閱 [VPC Lattice 服務的接聽程式規則](#)。

Important

您需要允許安全群組或任務使用傳入規則 vpc-lattice 字首，否則運作狀態檢查可能會失敗。

建立搭配 使用 VPC Lattice 的服務 AWS CLI

使用 AWS CLI 建立具有 VPC Lattice 的服務。將每個#####替換為自己的資訊。

1. 建立目標群組組態檔案。下列範例名為 tg-config.json

```
{
  "ipAddressType": "IPV4",
  "port": 443,
  "protocol": "HTTPS",
  "protocolVersion": "HTTP1",
  "vpcIdentifier": "vpc-f1663d9868EXAMPLE"
}
```

2. 使用下列命令來建立 VPC Lattice 目標群組。

```
aws vpc-lattice create-target-group \
  --name my-lattice-target-group-ip \
  --type IP \
  --config file://tg-config.json
```

Note

若要在純 IPv6 組態中建立服務，請建立 IP 位址類型為 IPv6 的目標群組。如需詳細資訊，請參閱 AWS CLI Command Reference 中的 [create-target-group](#)。

輸出範例：

```
{
```

```

    "arn": "arn:aws:vpc-lattice:us-east-2:123456789012:targetgroup/
tg-0eaa4b9ab4EXAMPLE",
    "config": {
        "healthCheck": {
            "enabled": true,
            "healthCheckIntervalSeconds": 30,
            "healthCheckTimeoutSeconds": 5,
            "healthyThresholdCount": 5,
            "matcher": {
                "httpCode": "200"
            },
            "path": "/",
            "protocol": "HTTPS",
            "protocolVersion": "HTTP1",
            "unhealthyThresholdCount": 2
        },
        "ipAddressType": "IPv4",
        "port": 443,
        "protocol": "HTTPS",
        "protocolVersion": "HTTP1",
        "vpcIdentifier": "vpc-f1663d9868EXAMPLE"
    },
    "id": "tg-0eaa4b9ab4EXAMPLE",
    "name": "my-lattice-target-group-ip",
    "status": "CREATE_IN_PROGRESS",
    "type": "IP"
}

```

3. 下列名為 *ecs-service-vpc-lattice.json* 的 JSON 檔案是一個範例，用來將 Amazon ECS 服務連接至 VPC Lattice 目標群組。以下範例中的 `portName` 與您在任務定義的 `portMappings` 屬性的 `name` 欄位中定義的名稱相同。

```

{
    "serviceName": "ecs-service-vpc-lattice",
    "taskDefinition": "ecs-task-def",
    "vpcLatticeConfigurations": [
        {
            "targetGroupArn": "arn:aws:vpc-lattice:us-
west-2:123456789012:targetgroup/tg-0eaa4b9ab4EXAMPLE",
            "portName": "testvpclattice",
            "roleArn": "arn:aws:iam::123456789012:role/
ecsInfrastructureRoleVpcLattice"
        }
    ]
}

```

```
],  
  "desiredCount": 5,  
  "role": "ecsServiceRole"  
}
```

使用下列命令建立 Amazon ECS 服務，並使用上述 JSON 範例將其連接至 VPC Lattice 目標群組。

```
aws ecs create-service \  
  --cluster clusterName \  
  --serviceName ecs-service-vpc-lattice \  
  --cli-input-json file://ecs-service-vpc-lattice.json
```

Note

Amazon ECS Anywhere 不支援 VPC Lattice。

保護 Amazon ECS 任務，避免其因縮減事件而終止

您可以使用 Amazon ECS 任務縮減保護來保護任務，避免任務因服務自動擴展或部署的縮減事件而終止。

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。
- 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`。在將 `ProtectionEnabled` 設定為 `true` 時，任務預設會受到 2 小時保護。您隨後可以使用 `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)。如果您使用的是 AWS CLI，您可以指定 `--protection-enabled` 選項。

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。如果您使用的是 AWS CLI，您可以指定 `--no-protection-enabled` 選項。

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。在任務完成工作之後，您可以使用相同的端點清除對 `ProtectionEnabled` 屬性的設定，使任務符合在後續縮減事件期間終止的資格。

如需有關 Amazon ECS 容器代理程式端點的詳細資訊，請參閱 [Amazon ECS 任務縮減保護端點](#)。

- Amazon ECS API

如果應用程式具有追蹤作用中任務狀態的元件，則可使用 Amazon ECS API 來設定與擷取任務縮減保護。使用 `UpdateTaskProtection` 將一或多個任務標記為受保護。使用 `GetTaskProtection` 擷取保護狀態。

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 Elastic Container Service API Reference 中的 [UpdateTaskProtection](#) 與 [GetTaskProtection](#)。

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：

- 任務縮減保護僅支援從服務部署的任務。

- 任務縮減保護支援從在 Amazon ECS 受管執行個體上執行的服務部署的任務。
- 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。流量將轉移至新啟動的任務，而且只有在 `protectionEnabled` 清除或過期時才會移除較舊的任務。視 CodeDeploy 或 CloudFormation 更新的逾時而定，部署可能會逾時，而較舊的藍色任務可能仍會存在。
 - 如果您使用 CloudFormation，則更新堆疊會有 3 小時的逾時。因此，如果您將任務保護設定為 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 堆疊會顯示 `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

任務必須具備具有如下許可的 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 任務縮減保護端點

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。

從容器內對此 URI 發送 PUT 請求，設定任務縮減保護。對此 URI 的 GET 請求將傳回任務的目前保護狀態。

任務縮減保護請求參數

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` 來標記任務以實現保護。指定 `false` 以移除保護，並讓任務符合終止資格。

類型：布林值

必要：是

ExpiresInMinutes

任務受到保護的分鐘數。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任務的範例

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}'
```

60 分鐘保護任務的範例

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}'
```

24 小時保護任務的範例

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}'
```

Windows 容器範例

對於 Windows 容器，您可以使用 PowerShell 的 `Invoke-RestMethod` cmdlet 而非 `curl`。下列範例展示了與先前 `curl` 命令等效的 PowerShell 命令。

使用預設時段保護 Windows 容器任務的範例

此範例說明如何使用 PowerShell 以預設的 2 小時時段保護任務。

```
Invoke-RestMethod -Uri $env:ECS_AGENT_URI/task-protection/v1/state -Method Put -Body '{"ProtectionEnabled":true}' -ContentType 'application/json'
```

60 分鐘保護 Windows 容器任務的範例

此範例說明如何搭配 PowerShell 使用 `expiresInMinutes` 參數保護任務 60 分鐘。

```
Invoke-RestMethod -Uri $env:ECS_AGENT_URI/task-protection/v1/state -Method Put -Body
'{"ProtectionEnabled":true,"ExpiresInMinutes":60}' -ContentType 'application/json'
```

24 小時保護 Windows 容器任務的範例

此範例說明如何搭配 PowerShell 使用 `expiresInMinutes` 參數保護任務 24 小時。

```
Invoke-RestMethod -Uri $env:ECS_AGENT_URI/task-protection/v1/state -Method Put -Body
'{"ProtectionEnabled":true,"ExpiresInMinutes":1440}' -ContentType 'application/json'
```

PUT 請求將傳回以下回應。

```
{
  "protection": {
    "ExpirationDate": "2023-12-20T21:57:44.837Z",
    "ProtectionEnabled": true,
    "TaskArn": "arn:aws:ecs:us-west-2:111122223333:task/1234567890abcdef0"
  }
}
```

任務縮減保護回應參數

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)。如果任務未受到保護，則此值將為空。

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
```

對於 Windows 容器，請使用下列 PowerShell 命令來取得保護狀態：


```
Invoke-RestMethod -Uri $env:ECS_AGENT_URI/task-protection/v1/state -Method Get
```

```
{
  "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

造成例外狀況的 Amazon ECS API 呼叫 AWS 請求 ID。

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"
  }
}
```

搭配 Amazon ECS 與 Fargate 工作負載使用故障注入

客戶可以在 Amazon EC2 與 Fargate 上搭配 Amazon ECS 使用故障注入，以測試應用程式在特定受損案例下的應對表現。這些測試提供的資訊可用於最佳化應用程式的效能與恢復能力。

啟用故障注入後，Amazon ECS 容器代理程式會允許任務存取新的故障注入端點。您需要選擇加入，才能透過將 `enableFaultInjection` 任務定義參數值設定為 `true` 來使用故障注入。預設值為 `false`。

```
{
  ...
  "enableFaultInjection": true
}
```

Note

故障注入僅適用於使用 `awsvpc` 或 `host` 網路模式的任務。
故障注入不適用於 Windows。

如需有關如何在 中啟用錯誤注入的資訊 AWS 管理主控台，請參閱[使用主控台建立 Amazon ECS 任務定義](#)。

您需要在 AWS Fault Injection Service 中啟用此功能以進行測試。如需詳細資訊，請參閱[使用 AWS FIS aws : ecs : task 動作](#)。

Note

如果您未使用新的 Amazon ECS 最佳化 AMI，或擁有自訂 AMI，請安裝下列相依項：

- `tc`

- [sch_netem](#) 核心模組

Amazon ECS 故障注入端點

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。每個端點都包含 `/start`、`/stop` 與 `/status` 端點。這些端點僅接受已啟用故障注入之任務的請求，且每個端點的速率限制為每個容器每 5 秒 1 個請求。超過此限制會導致錯誤。

Note

需要具有 Amazon ECS Agent version 1.88.0+ 才能使用故障注入端點。

搭配故障注入使用的三個端點為：

- [網路黑洞連接埠端點](#)
- [網路封包遺失端點](#)
- [網路延遲端點](#)

成功的請求會產生 200 的回應碼，呼叫 `/start` 端點時訊息為 `running`，呼叫 `/stop` 端點時為 `stopped`，呼叫 `/status` 端點時為 `running` 或 `not-running`。

```
{
  "Status": <string>
}
```

失敗的請求會傳回下列其中一個錯誤代碼：

- 400 – 錯誤的請求
- 409 – 故障注入請求與另一個執行中的故障衝突
- 429 – 請求限流
- 500 – 伺服器發生非預期錯誤

```
{
```

```
"Error": <string message>
}
```

 Note

一次只能注入一個網路延遲故障或一個網路封包遺失故障。嘗試注入多個故障會導致請求遭拒。

網路黑洞連接埠端點

{ECS_AGENT_URI}/fault/v1/network-blackhole-port 端點會捨棄任務網路命名空間中特定連接埠與通訊協定的傳入或傳出流量，並與兩種模式相容：

- awsvpc – 變更會套用至任務網路命名空間
- 主機 – 變更會套用至預設網路命名空間容器執行個體

{ECS_AGENT_URI}/fault/v1/network-blackhole-port/start

此端點會啟動網路黑洞連接埠故障注入，並具有下列參數：

站點

用於黑洞連接埠故障注入的指定連接埠。

類型：整數

必要：是

通訊協定

用於黑洞連接埠故障注入的通訊協定。

類型：字串

有效值：tcp | udp

必要：是

TrafficType

故障注入使用的流量類型。

類型：字串

有效值：ingress | egress

必要：是

SourcesToFilter

受故障保護的 IPv4 或 IPv6 位址或 CIDR 區塊的 JSON 陣列。

類型：字串陣列

必要：否

以下是使用 start 端點的請求範例 (將##值取代為實際值)：

```
Endpoint: ${ECS_AGENT_URI}/fault/v1/network-blackhole-port/start
```

```
Http method:POST
```

```
Request payload:
```

```
{
  "Port": 1234,
  "Protocol": "tcp|udp",
  "TrafficType": "ingress|egress"
  "SourcesToFilter": ["${IP1}", "${IP2}", ...],
}
```

{ECS_AGENT_URI}/fault/v1/network-blackhole-port/stop

此端點會停止請求中指定的故障。此端點具有下列參數：

站點

應停止之受故障影響的連接埠。

類型：整數

必要：是

通訊協定

用於停止故障的通訊協定。

類型：字串

有效值：tcp | udp

必要：是

TrafficType

故障注入使用的流量類型。

類型：字串

有效值：ingress | egress

必要：是

以下是使用 stop 端點的請求範例 (將##值取代為實際值)：

```
Endpoint: ${ECS_AGENT_URI}/fault/v1/network-blackhole-port/stop

Http method: POST

Request payload:
{
  "Port": 1234,
  "Protocol": "tcp|udp",
  "TrafficType": "ingress|egress",
}
```

{ECS_AGENT_URI}/fault/v1/network-blackhole-port/status

此端點用於檢查故障注入的狀態。此端點具有下列參數：

站點

要檢查故障狀態的受影響連接埠。

類型：整數

必要：是

通訊協定

檢查故障狀態時使用的通訊協定。

類型：字串

有效值：tcp | udp

必要：是

TrafficType

故障注入使用的流量類型。

類型：字串

有效值：ingress | egress

必要：是

以下是使用 status 端點的請求範例 (將##值取代為實際值)：

```
Endpoint: ${ECS_AGENT_URI}/fault/v1/network-blackhole-port/status
```

```
Http method: POST
```

```
Request payload:
```

```
{
  "Port": 1234,
  "Protocol": "tcp|udp",
  "TrafficType": "ingress|egress",
}
```

網路延遲端點

{ECS_AGENT_URI}/fault/v1/network-latency 端點會將延遲與抖動新增至任務的網路介面，以將流量傳送至特定來源。端點與兩種模式相容：

- awsvpc – 變更會套用至任務網路介面
- 主機 – 變更會套用至預設網路介面

{ECS_AGENT_URI}/fault/v1/network-latency/start

此 /start 端點會開始網路延遲故障注入，並具有下列參數：

DelayMilliseconds

要新增至網路介面以用於故障注入的延遲毫秒數。

類型：整數

必要：是

JitterMilliseconds

要新增至網路介面以用於故障注入的抖動毫秒數。

類型：整數

必要：是

來源

IPv4 或 IPv6 位址或 CIDR 區塊的 JSON 陣列，作為用於故障注入的目的地。

類型：字串陣列

必要：是

SourcesToFilter

受故障保護的 IPv4 或 IPv6 位址或 CIDR 區塊的 JSON 陣列。SourcesToFilter 的優先級高於 Sources。

類型：字串陣列

必要：否

以下是使用 /start 端點的請求範例 (將##值取代為實際值)：

```
Endpoint: ${ECS_AGENT_URI}/fault/v1/network-latency/start
```

```
Http method: POST
```

```
Request payload:
```

```
{
  "DelayMilliseconds": 123,
  "JitterMilliseconds": 123,
  "Sources": ["${IP1}", "${IP2}", ...],
  "SourcesToFilter": ["${IP1}", "${IP2}", ...],
}
```

{ECS_AGENT_URI}/fault/v1/network-latency/stop and /status

{ECS_AGENT_URI}/fault/v1/network-latency/stop 端點會停止故障，{ECS_AGENT_URI}/fault/v1/network-latency/status 則會檢查故障的狀態。

以下是使用 /stop 與 /status 端點的兩個請求範例。兩者都使用 POST HTTP 方法。

```
Endpoint: ${ECS_AGENT_URI}/fault/v1/network-latency/stop
```

```
Endpoint: ${ECS_AGENT_URI}/fault/v1/network-latency/status
```

網路封包遺失端點

{ECS_AGENT_URI}/fault/v1/network-packet-loss 端點會將封包遺失新增至指定的網路介面。此端點與兩種模式相容：

- awsvpc – 變更會套用至任務網路介面
- 主機 – 變更會套用至預設網路介面

{ECS_AGENT_URI}/fault/v1/network-packet-loss/start

此 /start 端點會開始網路封包遺失故障注入，並具有下列參數：

LossPercent

封包遺失百分比

類型：整數

必要：是

來源

IPv4 或 IPv6 位址或 CIDR 區塊的 JSON 陣列，用於故障注入測試。

類型：字串陣列

必要：是

SourcesToFilter

受故障保護的 IPv4 或 IPv6 位址或 CIDR 區塊的 JSON 陣列。SourcesToFilter 的優先級高於 Sources。

類型：字串陣列

必要：否

以下是使用 `start` 端點的請求範例 (將##值取代為實際值)：

```
Endpoint: ${ECS_AGENT_URI}/fault/v1/network-packet-loss/start

Http method: POST

{
  "LossPercent": 6,
  "Sources": ["${IP1}", "${IP2}", ...],
  "SourcesToFilter": ["${IP1}", "${IP2}", ...],
}
```

`{ECS_AGENT_URI}/fault/v1/network-packet-loss/stop` and `/status`

`{ECS_AGENT_URI}/fault/v1/network-packet-loss/stop` 端點會停止故障，`{ECS_AGENT_URI}/fault/v1/network-packet-loss/status` 則會檢查故障的狀態。一次僅支援每種類型的一個故障。

以下是使用 `/stop` 與 `/status` 端點的兩個請求範例。兩者都使用 POST HTTP 方法。

```
Endpoint: ${ECS_AGENT_URI}/fault/v1/network-packet-loss/stop
```

```
Endpoint: ${ECS_AGENT_URI}/fault/v1/network-packet-loss/status
```

更新 Amazon ECS 服務參數

建立服務之後，有時可能需要更新服務參數，例如任務數量。

服務排程器啟動新任務時，會使用下列邏輯來決定叢集中的任務置放。

- 決定叢集中的哪些容器執行個體可以支援服務的任務定義。例如，它們具有所需的 CPU、記憶體、連接埠與容器執行個體屬性。
- 依預設，服務排程器會嘗試以這種方式平衡可用區域間的任務，即使您可以選擇不同的置放策略。
 - 在執行個體所在可用區域中，依此服務的執行中任務數量下限，對有效容器執行個體進行排序。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。

- 將新的服務任務放置在最佳可用區域的有效容器執行個體 (根據先前的步驟)，且偏好此服務中的執行中任務數量最少的容器執行個體。

當服務排程器停止執行中的任務時，會嘗試使用下列邏輯維護叢集中可用區域間的平衡：

- 在執行個體所在可用區域中，依此服務的執行中任務數量上限，對容器執行個體進行排序。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，則最適合終止的即為區域 B 或 C 中的容器執行個體。
- 停止最佳可用區域之容器執行個體中的任務 (根據先前的步驟)，且偏好此服務之執行中任務數量最多的容器執行個體。

使用下列清單來判斷您是否可以變更服務參數。

可用區域重新平衡

指出服務是否使用可用區域重新平衡。

您可以針對滾動部署變更此參數。

容量提供者策略

容量提供者策略的詳細資料。您可以在建立叢集、執行任務或更新服務時設定容量提供者。

使用 Fargate 時，容量提供者為 FARGATE 或 FARGATE_SPOT。

使用 Amazon EC2 時，容量提供者為 Auto Scaling 群組。

可以針對滾動部署與藍/綠部署變更容量提供者。

下列清單可提供有效的轉換：

- 將 Fargate 更新為 Auto Scaling 群組容量提供者。
- 將 EC2 更新為 Fargate 容量提供者。
- 將 Fargate 容量提供者更新為 Auto Scaling 群組容量提供者。
- 將 Amazon EC2 容量提供者更新為 Fargate 容量提供者。
- 將 Auto Scaling 群組或 Fargate 容量提供者更新回啟動類型。使用 CLI 或 API 時，需要在 `capacityProviderStrategy` 參數中傳遞空白清單。

叢集

無法變更叢集名稱。

Deployment configuration (部署組態)

部署組態包括用於偵測失敗的 CloudWatch 警示與斷路器，以及所需的組態。

如果服務無法達到穩定狀態，部署斷路器會確定服務部署是否會失敗。如果使用部署斷路器，服務部署會轉換為失敗狀態並停止啟動新任務。如果使用復原選項，當服務部署失敗後，服務會復原至上一次成功完成的部署。

更新使用 Amazon ECS 斷路器的服務時，Amazon ECS 會建立服務部署與服務修訂版。這些資源可用於檢視服務歷史記錄的詳細資訊。如需詳細資訊，請參閱[使用 Amazon ECS 服務部署檢視服務歷史記錄](#)。

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，則運作狀態百分比下限代表部署期間，服務中必須保持在 RUNNING 狀態的任務數量下限，以所需任務數量百分比表示 (無條件進位到最接近的整數)。如果服務包含使用 EC2 的任務，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。對於未使用負載平衡器的服務，如果任務處於 RUNNING 狀態，則服務會將任務視為運作狀態良好。對於使用負載平衡器的服務，如果任務處於 RUNNING 狀態且負載平衡器也回報為運作狀態良好，則服務會將任務視為運作狀態良好。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，以所需任務數量百分比表示 (無條件捨去到最接近的整數)。如果服務包含使用 EC2 的任務，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。然後會發生下列情況。

- 如果替代任務的運作狀態為 HEALTHY，服務排程器會停止運作狀態不良的任務
- 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](#)。

部署控制器

用於服務的部署控制器。部署控制器有三種類型：

- ECS
- EXTERNAL
- CODE_DEPLOY

更新服務時，可以更新其使用的部署控制器。下列清單可提供有效的轉換：

- 將 CodeDeploy 藍/綠部署 (CODE_DEPLOY) 更新為 ECS 滾動部署或藍/綠部署 (ECS)。
- 將 CodeDeploy 藍/綠部署 (CODE_DEPLOY) 更新為外部部署 (EXTERNAL)。
- 將 ECS 滾動部署或藍/綠部署 (ECS) 更新為外部部署 (EXTERNAL)。
- 將外部部署 (EXTERNAL) 更新為 ECS 滾動部署或藍/綠部署 (ECS)。

更新服務的部署控制器時，請考量下列事項：

- 如果服務使用 VPC Lattice 或 Amazon ECS Service Connect，則無法將服務的部署控制器從 ECS 部署控制器更新為任何其他控制器。
- 在正在進行的服務部署期間，無法更新服務的部署控制器。
- 如果服務上沒有負載平衡器，則無法將服務的部署控制器更新為 CODE_DEPLOY。
- 如果 deploymentConfiguration 包含警示、部署斷路器或 BLUE_GREEN 部署策略，則無法將服務的部署控制器從 ECS 更新為任何其他控制器。如需詳細資訊，請參閱[Amazon ECS 服務部署控制器與策略](#)。
- 如果您將服務的部署控制器從 ECS 更新為任何其他控制器，Amazon ECS 將不會使用您在容器定義中為 versionConsistency 指定的值。
- 如果您將服務的部署控制器從 ECS 更新為任何其他控制器，UpdateService 與 DescribeService API 回應仍將傳回 deployments 而不是 taskSets。如需有關

UpdateService 與 CreateService 的詳細資訊，請參閱 Amazon ECS API Reference 中的 [UpdateService](#) 與 [CreateService](#)。

- 如果服務使用滾動更新部署策略，將部署控制器從 ECS 更新為任何其他控制器會變更 deploymentConfiguration 中 maximumPercent 值的使用方式。maximumPercent 不僅僅用作滾動更新部署中的總任務數量上限，還用於取代運作狀態不良的任務。如需有關排程器如何取代運作狀態不良任務的詳細資訊，請參閱 [Amazon ECS 服務](#)。
- 如果您將服務的部署控制器從 ECS 更新為任何其他部署控制器，則系統會忽略使用負載平衡器組態指定的任何 advancedConfiguration。如需詳細資訊，請參閱 Amazon ECS API reference 中的 [LoadBalancer](#) 與 [AdvancedConfiguration](#)。

使用 CloudFormation 更新服務的部署控制器時，請根據正在執行的遷移類型，考量下列事項。

- 如果 CloudFormation 範本包含 EXTERNAL 部署控制器資訊以及 TaskSet 與 PrimaryTaskSet 資源，並且在從 EXTERNAL 更新為 ECS 時從範本中移除任務集資源，則在部署控制器更新為 ECS 後，DescribeTaskSet 與 DeleteTaskSet API 呼叫會傳回 400 錯誤。這會導致 CloudFormation 刪除任務集資源失敗，即使 CloudFormation 堆疊轉換為 UPDATE_COMPLETE 狀態，也是如此。如需詳細資訊，請參閱 AWS CloudFormation User Guide 中的 [Resource removed from stack but not deleted](#)。若要修正此問題，請使用 Amazon ECS DeleteTaskSet API 直接刪除任務集。如需有關如何刪除任務集的詳細資訊，請參閱 Amazon Elastic Container Service API Reference 中的 [DeleteTaskSet](#)。
- 如果要從 CODE_DEPLOY 遷移至 ECS 並使用新的任務定義，且 CloudFormation 執行復原操作，則 Amazon ECS UpdateService 請求會失敗，並顯示下列錯誤：

```
Resource handler returned message: "Invalid request provided: Unable to update task definition on services with a CODE_DEPLOY deployment controller. Use AWS CodeDeploy to trigger a new deployment. (Service: Ecs, Status Code: 400, Request ID: 0abda1e2-f7b3-4e96-b6e9-c8bc585181ac) (SDK Attempt Count: 1)" (RequestToken: ba8767eb-c99e-efed-6ec8-25011d9473f0, HandlerErrorCode: InvalidRequest)
```

- 從 ECS 成功遷移至 EXTERNAL 部署控制器後，需要手動移除 ACTIVE 任務集，因為 Amazon ECS 不再管理該部署。如需有關如何刪除任務集的資訊，請參閱 Amazon Elastic Container Service API Reference 中的 [DeleteTaskSet](#)。

所需的任務計數

要在服務中置放並保持執行的任務執行個體數量。

若要暫時停止服務，請將此值設定為 0。然後，當您準備好啟動服務時，請使用原始值更新服務。

可以針對滾動部署與藍/綠部署變更此參數。

啟用受管標籤

決定是否為服務中的任務啟用 Amazon ECS 受管標籤。

只有更新後啟動的任務才會反映更新。若要更新所有任務上的標籤，請使用強制部署選項。

可以針對滾動部署與藍/綠部署變更此參數。

啟用 ECS Exec

決定是否使用 Amazon ECS Exec。

如果不想覆寫建立服務時設定的值，可以在執行此動作時將此值設定為 null。

您可以針對滾動部署變更此參數。

運作狀態檢查寬限期

所謂寬限期是指 Amazon ECS 服務排程器在任務首次啟動後，會在這段期間內 (以秒為單位) 忽略運作狀態不良的 Elastic Load Balancing、VPC Lattice 與容器運作狀態檢查。如果您未指定運作狀態檢查寬限期值，則會使用預設值 0。如果未使用任何運作狀態檢查，healthCheckGracePeriodSeconds 會處於未使用狀態。

若服務的任務需要一些時間來啟動及回應運作狀態檢查，您可以指定運作狀態檢查的寬限期，最長為 2,147,483,647 秒 (約 69 年)。在此期間，Amazon ECS 服務排程器會忽略運作狀態檢查情形。此寬限期可防止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。

可以針對滾動部署與藍/綠部署變更此參數。

負載平衡器

更新負載平衡器時，必須使用服務連結角色。

Elastic Load Balancing 負載平衡器物件的清單。該清單包含負載平衡器名稱、容器名稱，以及要從負載平衡器存取的容器連接埠。容器名稱是在容器定義中顯示的名稱。

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。

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 組態啟動新任務，然後在新任務執行時停止舊任務。

對於使用滾動更新的服務，可以新增、更新或移除 Elastic Load Balancing 目標群組。您可以從單一目標群組更新為多個目標群組，也可以從多個目標群組更新為單一目標群組。

對於使用藍/綠部署的服務，可以透過 CodeDeploy 使用 [CreateDeployment](#) 來更新 Elastic Load Balancing 目標群組。請注意，藍/綠部署不支援多個目標群組。如需詳細資訊，請參閱[向服務註冊多個目標群組](#)。

對於使用外部部署控制器的服務，可以使用 [CreateTaskSet](#) 來新增、更新或移除負載平衡器。請注意，外部部署不支援多個目標群組。如需詳細資訊，請參閱[向服務註冊多個目標群組](#)。

傳遞空白清單以移除負載平衡器。

您可以針對滾動部署變更此參數。

網路組態

服務網路組態。

您可以針對滾動部署變更此參數。

置放限制條件

要更新服務以使用的任務置放限制條件物件陣列。如果未指定任何值，則服務的現有置放限制條件將保持不變。如果指定此值，則會覆寫為服務定義的任何現有置放限制條件。若要移除所有現有的置放限制條件，請指定空陣列。

您最多可以為每項任務指定 10 項限制條件。此限制包含任務定義中的限制條件以及執行時間指定的限制條件。

可以針對滾動部署與藍/綠部署變更此參數。

置放策略

要更新服務以使用的任務置放策略物件。如果未指定任何值，則服務的現有置放策略將保持不變。如果指定此值，則會覆寫為服務定義的現有置放策略。若要移除現有的置放策略，請指定空白物件。

可以針對滾動部署與藍/綠部署變更此參數。

平台版本

服務執行所在的 Fargate 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。

您可以針對滾動部署變更此參數。

傳播標籤

決定是否將標籤從任務定義或服務傳播到任務。如果沒有指定值，則不會傳播標籤。

只有更新後啟動的任務才會反映更新。若要更新所有任務上的標籤，請將 `forceNewDeployment` 設定為 `true`，以便 Amazon ECS 使用更新後的標籤啟動新任務。

可以針對滾動部署與藍/綠部署變更此參數。

Service Connect 組態

Amazon ECS Service Connect 的組態。此參數可決定服務如何連線至應用程式中的其他服務。

您可以針對滾動部署變更此參數。

服務登錄檔

更新服務登錄檔時，必須使用服務連結角色。

要指派給此服務的服務探索登錄檔的詳細資訊。如需詳細資訊，請參閱[服務探索](#)。

當您新增、更新或移除服務登錄檔組態時，Amazon ECS 會使用更新後的服務登錄檔組態啟動新任務，然後在新任務執行時停止舊任務。

傳遞空白清單以移除服務登錄檔。

您可以針對滾動部署變更此參數。

任務定義

用於服務的任務定義與修訂版。

如果變更任務定義中容器使用的連接埠，則可能需要更新容器執行個體的安全群組，才能使用更新後的連接埠。

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。

容器映像提取行為會因運算選項而有所不同。如需詳細資訊，請參閱下列其中一個項目：

- [為 Amazon ECS 規劃 AWS Fargate 架構](#)
- [為 Amazon ECS 規劃 EC2 容量架構](#)

- [Amazon ECS 的外部執行個體 \(Amazon ECS Anywhere\)](#)

您可以針對滾動部署變更此參數。

磁碟區組態

磁碟區的詳細資訊，即 `configuredAtLaunch`。將任務定義中的 `configuredAtLaunch` 設定為 `true` 時，此服務參數會為服務中的每項任務設定一個 Amazon EBS 磁碟區，以便在部署期間建立與連接。您可以在 [ServiceManagedEBSVolumeConfiguration](#) 中設定大小、`volumeType`、IOPS、輸送量、快照與加密。磁碟區的 `name` 必須與任務定義中的 `name` 相符。如果設定為 `null`，則不會觸發新的部署。否則，如果此組態與現有組態不同，則會觸發新的部署。

您可以針對滾動部署變更此參數。

VPC Lattice 組態

服務的 VPC Lattice 組態。這會定義服務如何與 VPC Lattice 整合以進行服務間通訊。

您可以針對滾動部署變更此參數。

AWS CDK 考量事項

AWS CDK 不會追蹤資源狀態。它不知道您是在建立服務還是更新服務。客戶應使用逃生艙直接存取 `ecs Service L1` 建構模組。

如需逃生艙的相關資訊，請參閱《AWS Cloud Development Kit (AWS CDK) v2 開發人員指南》中的[從 AWS 建構程式庫自訂建構](#)。

若要將現有服務遷移至 `ecs.Service` 建構模組，請執行下列動作：

1. 使用逃生艙存取 `Service L1` 建構模組。
2. 在 `Service L1` 建構模組中手動設定下列屬性。

如果服務使用 Amazon EC2 容量：

- `daemon?`
- `placementConstraints?`
- `placementStrategies?`
- 如果使用 `awsvpc` 網路模式，則需要設定 `vpcSubnets?` 與 `securityGroups?` 建構模組。

如果服務使用 Fargate：

- FargatePlatformVersion
- vpcSubnets? 與 securityGroups? 建構。

3. 按如下方式設定 launchType :

```
const cfnEcsService = service.node.findChild('Service') as ecs.CfnService;  
cfnEcsService.launchType = "FARGATE";
```

若要從啟動類型遷移至容量提供者，請執行下列動作：

1. 使用逃生艙存取 Service L1 建構模組。
2. 新增 capacityProviderStrategies? 建構模組。
3. 部署服務。

更新 Amazon ECS 服務

建立服務之後，有時可能需要更新服務參數，例如任務數量。

更新使用 Amazon ECS 斷路器的服務時，Amazon ECS 會建立服務部署與服務修訂版。這些資源可用於檢視服務歷史記錄的詳細資訊。如需詳細資訊，請參閱[使用 Amazon ECS 服務部署檢視服務歷史記錄](#)。

先決條件

更新服務之前，請先確認部署類型可以變更哪些服務參數。如需可變更參數的完整清單，請參閱[更新 Amazon ECS 服務參數](#)。

程序

Console

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「叢集詳細資訊」頁面上的服務區段中，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

主控台會驗證所選任務定義系列與修訂版是否與定義的運算組態相容。如果您收到警告，請確認已選取任務定義相容性與運算組態。

6. 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。
7. 如果選擇複本，若要讓 Amazon ECS 監控可用區域間的任務分佈，並在出現不平衡時重新分佈，請在可用區域服務重新平衡下，選取可用區域服務重新平衡。
8. 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](#)。
9. 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)。
10. 若要設定服務的任務部署方式，請展開部署選項區段，然後設定選項。
 - a. 在部署控制器類型欄位中，指定服務部署控制器。Amazon ECS 主控台支援下列控制器類型：ECS。
 - b. 在部署策略欄位中，選擇 Amazon ECS 用來部署新版本服務的策略。
 - c. 根據部署策略的選擇，執行下列動作：

部署策略	步驟
滾動更新	a. 在執行中任務百分比下限欄位中，請指定必須在服務部署期間執行的任務百分比值下限。如需詳細資訊，請參閱透過取代任務來部署 Amazon ECS 服務。 b. 在執行中任務百分比上限欄位中，請指定可以在服務部署期間執行的任務百分比值上限。如需詳細資訊，請參

部署策略	步驟	
	閱 透過取代任務來部署 Amazon ECS 服務 。	
藍/綠	在封裝時間欄位中，指定藍色與綠色服務修訂版應同時執行的持續時間 (以分鐘為單位)。如需詳細資訊，請參閱 Amazon ECS 藍/綠部署 。	

- d. 若要為生命週期階段執行 Lambda 函式，請在部署 lifecycle hook 下為每個唯一的 Lambda 函式執行下列動作：

- i. 選擇新增。

針對每個要執行的唯一函式重複此步驟。

- ii. 在 Lambda 函式欄位中輸入函式名稱。
 iii. 在角色欄位中，選擇您在先決條件中建立的具有藍/綠部署許可的角色。

如需詳細資訊，請參閱[Amazon ECS 藍/綠部署中 Lambda 函式所需的許可](#)。

- iv. 在生命週期階段欄位中選取 Lambda 函式執行的階段。
 v. (選用) 在勾點詳細資訊欄位中，輸入提供勾點相關資訊的鍵值對。

11. 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)。

在部署斷路器將部署設定為失敗狀態時，若要讓軟體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. 若要根據應用程式指標停止部署，請選取使用 CloudWatch 警示。然後，從 CloudWatch 警示名稱欄位中選擇警示。要建立新的警示，請前往 CloudWatch 主控台。

在 CloudWatch 警示將部署設定為失敗狀態時，若要讓軟體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。

12. 若要變更運算選項，請展開運算組態區段，然後執行下列動作：

- a. 對於 上的服務 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，請針對容量提供者策略，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。

13. (選用) 若要設定服務自動擴展功能，請展開服務自動擴展區段，然後指定下列參數。若要使用預測自動擴展功能 (該功能可查看過去從流量流程載入的資料)，請在建立服務後進行設定。如需詳細資訊，請參閱[使用歷史模式透過預測擴展來擴展 Amazon ECS 服務](#)。
 - a. 若要使用服務自動擴展，請選取 Service auto scaling (服務自動擴展)。
 - b. 在任務數量下限欄位中，輸入供服務自動擴展功能使用的任務數量下限。所需的計數不會低於此計數。
 - c. 在任務數量上限中，輸入供服務自動擴展功能使用的任務數量上限。所需的計數不會高於此計數。
 - d. 選擇政策類型。在擴展政策類型下，選擇下列其中一個選項。

使用此政策類型	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。	

使用此政策類型	執行此作業	
	• ECSServiceAverageCPUUtilization – 服務的平均 CPU 使用率。 • ECSServiceAverageMemoryUtilization – 服務的平均記憶體使用率。 • ALBRequestCountPerTarget – Application Load Balancer 目標群組中每個目標的已完成請求數。 d. 針對 Target value (目標值)，輸入服務為選取的指標保持的值。 e. 在橫向擴充冷卻時間欄位中，輸入一個橫向擴充活動 (新增任務) 完成之後、另一個橫向擴充活動開始之前，必須經過的時間 (以秒為單位)。 f. 在縮減冷卻時間欄位中，輸入一個縮減活動 (移除任務) 完成之後、另一個縮減活動開始之前，必須經過的時間 (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)。	

使用此政策類型	執行此作業	
	h. • (選用) 如果希望擴展政策在流量增加時進行橫向擴充，但不需要在流量減少時縮減，請選取關閉縮減。	

使用此政策類型	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：	

使用此政策類型	執行此作業	
	- 在動作欄位中，選取是為服務新增、移除還是設定特定的所需計數。 - 如果選擇新增或移除任務，請在值欄位中，輸入啟動擴展動作時要新增或移除的任務數量 (或現有任務的百分比)。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。選擇新增擴展動作，然後重複擴展動作步驟。 - 在冷卻時間欄位中，輸入等待上一個擴展活動生效的時間 (以秒	

使用此政策類型	執行此作業	
	為單位)。對於新增政策，這是橫向擴充活動結束之後，擴展政策封鎖縮減活動並限制一次可以橫向擴充多少項任務的時間。對於移除政策，這是一個縮減活動結束之後、另一個縮減活動開始之前必須經過的時間。	

14. (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至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。這可以是在您的 中在相同 AWS 區域 中分別建立的命名空間，AWS 帳戶 或是使用 AWS Resource Access Manager () 與您的帳戶共用的相同區域中的命名空間AWS RAM。如需共用 AWS Cloud Map 命名空間的詳細資訊，請參閱《AWS Cloud Map 開發人員指南》中的[跨帳戶 AWS Cloud Map 命名空間共用](#)
- c. (選用) 指定日誌組態。選取使用日誌收集。預設選項會將容器日誌傳送至 CloudWatch 日誌。其他日誌驅動程式選項是使用 AWS FireLens 設定。如需詳細資訊，請參閱[將 Amazon ECS 日誌傳送至 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 – 將任務設定為將容器日誌傳送至 CloudWatch Logs。系統會提供預設日誌驅動程式選項，用來代表您建立 CloudWatch 日誌群組。若要指定不同的日誌群組名稱，請變更驅動程式選項值。

- Amazon Data Firehose – 將任務設定為將容器日誌傳送至 Firehose。系統會提供預設日誌驅動程式選項，用來將日誌傳送至 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 – 將任務設定為將容器日誌傳送至 Amazon Kinesis Data Streams。系統會提供預設日誌驅動程式選項，用來將日誌傳送至 Kinesis Data Streams 串流。若要指定不同的串流名稱，請變更驅動程式選項值。
 - Amazon OpenSearch Service – 將任務設定為將容器日誌傳送至 OpenSearch Service 網域。務必提供日誌驅動程式選項。
 - Amazon S3：將任務設定為將容器日誌傳送至 Amazon S3 儲存貯體。系統會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。
- d. 若要啟用存取日誌，請遵循下列步驟：
- i. 展開存取日誌組態。針對格式，選擇 JSON 或 TEXT。
 - ii. 若要在存取日誌中包含查詢參數，請選取包含查詢參數。

 Note

若要停用存取日誌，請在格式化中選擇無。

15. 如果任務使用的資料磁碟區與部署時的組態相容，則可透過展開磁碟區區段對磁碟區進行設定。

磁碟區名稱與磁碟區類型會在建立任務定義修訂版時設定，且無法在更新服務時變更。若要更新磁碟區名稱與類型，必須建立新的任務定義修訂版，並使用新的修訂版來更新服務。

設定此磁碟區類型	執行此作業	
Amazon EBS	a. 在 EBS 磁碟區類型欄位中，選擇要連接至任務的 EBS 磁碟區類型。 b. 在大小 (GiB) 欄位中，輸入磁碟區大小的有效值 (以 GiB 為單位)。您可以指定磁碟區大小 (大小下限 1 GiB，大小上限	

設定此磁碟區類型	執行此作業	
	16,384 GiB)。除非提供快照 ID，否則需要此值。 c. 在 IOPS 欄位中，輸入磁碟區應提供的輸入/輸出操作次數 (IOPS) 上限。此值僅可針對 io1、io2 與 gp3 磁碟區類型設定。 d. 在 輸送量 (MiB/s) 欄位中，輸入磁碟區應提供的輸送量 (以 MiBps 或 MiB/s 為單位)。此值僅可針對 gp3 磁碟區類型設定。 e. 在快照 ID 欄位中，若想從快照建立磁碟區，請選擇現有的 Amazon EBS 磁碟區快照或輸入快照的 ARN。您也可以不選擇或不輸入快照 ID，建立新的空白磁碟區。 f. 如果指定了快照 ID，則可指定磁碟區初始化速率 (MiB/s)。輸入一個介於 100 至 300 MiB/s 之間的值，該值將決定從使用快照 ID 指定的快照在建立磁碟區時載入資料的速度。 g. 在檔案系統類型欄位中，選擇用於磁碟區上資料儲存與擷取的檔案系統類型。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	

設定此磁碟區類型	執行此作業	
	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。如果有檔案系統類型不相符，則任務將無法啟動。 h. 在基礎結構角色欄位中，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 AmazonECSInfrastructureRolePolicyForVolumes 受管政策連接至該角色，也可參照該政策建立並連接自己的政策，該政策需具有符合特定需求的許可。如需有關必要許可的詳細資訊，請參閱 Amazon ECS 基礎結構 IAM 角色。 i. 若想依預設設定使用 Amazon EBS 加密，請在加密欄位中選擇預設值。如果您的帳戶已預設設定加密，則會使用設定中指定的 AWS Key Management Service (AWS KMS) 金鑰來加密磁碟區。如果選擇預設，但未開啟 Amazon EBS 預設加密，則磁碟區將取消加密。	

設定此磁碟區類型	執行此作業	
	如果您選擇自訂，您可以為磁碟區加密指定 AWS KMS key 您選擇的。 如果選擇無，除非您已設定為預設加密，或者從加密快照建立磁碟區，否則磁碟區將不會加密。 j. 如果您已選擇自訂加密，則必須指定要使用 AWS KMS key 的。在 KMS 金鑰欄位中選擇 AWS KMS key 或輸入金鑰 ARN。如果您選擇使用對稱客戶受管金鑰來加密磁碟區，請確定您已在 AWS KMS key 政策中定義正確的許可。如需詳細資訊，請參閱 Amazon EBS 磁碟區的資料加密。 k. (選用) 在標籤下，您可以透過從任務定義或服務傳播標籤或提供自己的標籤，將標籤新增至 Amazon EBS 磁碟區。 如果要從任務定義傳播標籤，請在傳播標籤來源欄位中選擇任務定義。如果要從服務傳播標籤，請在傳播標籤來源欄位中選擇服務。如果選擇不傳播，或者未選擇任何值，則不會傳播標籤。	

設定此磁碟區類型	執行此作業	
	如果要提供自己的標籤，請選擇新增標籤，然後為每個新增的標籤提供鍵與值。 如需有關標記 Amazon EBS 磁碟區的詳細資訊，請參閱標記 Amazon EBS 磁碟區。	

16. (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。

- [新增標籤] 選擇新增標籤，並執行下列動作：
 - 在金鑰欄位中，輸入金鑰名稱。
 - 在值中，進入索引鍵值。
- [移除標籤] 在標籤旁邊，選擇 移除標籤。

17. 選擇更新。

AWS CLI

- 執行 `update-service`。如需有關執行 命令的資訊，請參閱《AWS Command Line Interface 參考》中的 [update-service](#)。

下列 `update-service` 範例會將服務 `my-http-service` 所需的任務計數更新為 2。

將 *user-input* 取代為實際值。

```
aws ecs update-service \
  --cluster MyCluster \
  --service my-http-service \
  --desired-count 2
```

後續步驟

追蹤部署，並檢視 Amazon ECS 斷路器服務的服務歷史記錄。如需詳細資訊，請參閱[使用 Amazon ECS 服務部署檢視服務歷史記錄](#)。

更新 Amazon ECS 服務以使用容量提供者

如果現有服務使用 Amazon EC2 或 Fargate 啟動類型，並且想要使用 Amazon ECS 受管執行個體，則需要更新服務以使用 Amazon ECS 受管執行個體容量提供者。

先決條件

為 Amazon ECS 受管執行個體建立容量提供者。如需詳細資訊，請參閱[為 Amazon ECS 受管執行個體建立容量提供者](#)。

程序

Console

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「叢集詳細資訊」頁面上的服務區段中，選取服務旁的核取方塊，然後選擇更新。
4. 選取強制執行新部署。
5. 在運算組態下，選擇「容量提供者策略」。然後，選擇下列其中一項：
 - 如果 Amazon ECS 受管執行個體容量提供者是預設容量提供者，請選擇使用叢集預設值。
 - 如果 Amazon ECS 受管執行個體容量提供者不是預設容量提供者，請選擇使用自訂 (進階)。選擇 Amazon ECS 受管執行個體容量提供者，然後在權重欄位中選擇「1」。
6. 選擇更新。

AWS CLI

- 執行 `update-service`。如需有關執行命令的資訊，請參閱《AWS Command Line Interface 參考》中的 [update-service](#)。

將 `user-input` 取代為實際值。

```
aws ecs update-service \
```

```
--cluster my-cluster \  
--service my-service \  
--capacity-provider-strategy capacityProvider=my-managed-instance-capacity-  
provider,weight=1 \  
--force-new-deployment
```

使用主控台刪除 Amazon ECS 服務

以下是您可能會刪除服務的一些原因：

- 不再需要應用程式
- 您正在將服務遷移至新環境
- 沒有主動使用應用程式
- 應用程式使用的資源超出所需資源，而您正嘗試最佳化成本

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，Amazon ECS 會一併刪除服務的所有服務部署與服務修訂版。

刪除服務時，如果仍有需要清理的執行中任務，則服務狀態會從 ACTIVE 移至 DRAINING 狀態，且服務將不再顯示於主控台或 ListServices API 操作中。在所有任務轉換為 STOPPING 或 STOPPED 狀態後，服務狀態會從 DRAINING 變為 INACTIVE。使用 DescribeServices API 操作仍可檢視處於 DRAINING 或 INACTIVE 狀態的服務。

Important

如果嘗試建立的新服務與處於 ACTIVE 或 DRAINING 狀態的現有服務同名，則會收到錯誤。

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. 在確認提示中，輸入刪除，然後選擇刪除。

將 Amazon ECS 短服務 ARN 移轉到完整 ARN

Amazon ECS 會為每項服務指派唯一的 Amazon Resource Name (ARN)。在 2021 年之前建立的服務使用短 ARN 格式：

```
arn:aws:ecs:region:aws_account_id:service/service-name
```

Amazon ECS 變更了 ARN 格式以包含叢集名稱。這是長 ARN 格式：

```
arn:aws:ecs:region:aws_account_id:service/cluster-name/service-name
```

您的服務必須具有完整 ARN 格式，才能標記您的服務。

您可以將使用短 ARN 格式的服務遷移至長 ARN 格式，而無需重新建立服務。可以使用 API、CLI 或主控台。遷移操作無法復原。

遷移程序是無縫的，並可確保服務零停機時間。在遷移期間：

- 服務可用性：服務會繼續正常執行，而不會中斷流量或功能。
- 執行中的任務：現有任務會繼續執行而不會中斷。如果在啟用 `taskLongArnFormat` 帳戶設定的情況下進行遷移，則遷移後啟動的新任務將使用長 ARN 格式。
- 容器執行個體：容器執行個體不受服務 ARN 遷移影響，並繼續正常運作。
- 服務組態：所有服務設定，包括任務定義、聯網與負載平衡器組態，均保持不變。

如果您想要使用 CloudFormation 以短 ARN 格式標記服務，則必須使用 API、CLI 或主控台遷移服務。遷移完成後，您可以使用 CloudFormation 來標記服務。

若要使用 Terraform 以短 ARN 格式標記服務，則必須透過 API、CLI 或主控台遷移服務。遷移完成後，可以使用 Terraform 來標記服務。

遷移完成後，服務會發生下列變更：

- 使用長 ARN 格式

```
arn:aws:ecs:region:aws_account_id:service/cluster-name/service-name
```

- 使用主控台進行遷移時，Amazon ECS 會將標籤新增至服務，並將鍵設定為 "ecs : serviceArnMigratedAt"，值設定為遷移時間戳記 (UTC 格式)。

此標籤會計入標籤配額。

- 當 CloudFormation 堆疊 `PhysicalResourceId` 中的 代表服務 ARN 時，該值不會變更，且會繼續為短期服務 ARN。

先決條件

在遷移服務 ARN 之前，請執行下列操作。

1. 若要查看是否存在短服務 ARN，請在 Amazon ECS 主控台中檢視服務詳細資訊 (如果服務使用短 ARN 格式，您會看到警告)，或檢視 `describe-services` 傳回的 `serviceARN` 參數。如果 ARN 不包含叢集名稱，您將擁有短 ARN。短 ARN 的格式如下：

```
arn:aws:ecs:region:aws_account_id:service/service-name
```

2. 請記下建立日期。
3. 如果您有使用短 ARN 格式的 IAM 政策，請將其更新為長 ARN 格式。

將每個 `#####` 替換為自己的資訊。

```
arn:aws:ecs:region:aws_account_id:service/cluster-name/service-name
```

如需詳細資訊，請參閱《AWS Identity and Access Management 使用者指南》中的[編輯 IAM 政策](#)。

4. 如果您有使用短 ARN 格式的工具，請將其更新為長 ARN 格式。

將每個 `#####` 替換為自己的資訊。

```
arn:aws:ecs:region:aws_account_id:service/cluster-name/service-name
```

5. 啟用服務長 ARN 格式。執行 `put-account-setting` 並將 `serviceLongArnFormat` 選項設定為 `enabled`。如需詳細資訊，請參閱 Amazon Elastic Container Service API Reference 中的[put-account-setting](#)。

如果服務的 `createdAt` 日期未知，請以根使用者身分執行命令。

```
aws ecs put-account-setting --name serviceLongArnFormat --value enabled
```

範例輸出

```
{
  "setting": {
    "name": "serviceLongArnFormat",
    "value": "enabled",
    "principalArn": "arn:aws:iam::123456789012:role/your-role",
    "type": "user"
  }
}
```

6. 啟用任務長 ARN 格式。此帳戶設定會控制服務遷移完成後啟動之新任務的 ARN 格式。執行 `put-account-setting` 並將 `taskLongArnFormat` 選項設定為 `enabled`。如需詳細資訊，請參閱 Amazon Elastic Container Service API Reference 中的 [put-account-setting](#)。

如果服務的 `createdAt` 日期未知，請以根使用者身分執行命令。

```
aws ecs put-account-setting --name taskLongArnFormat --value enabled
```

範例輸出

```
{
  "setting": {
    "name": "taskLongArnFormat",
    "value": "enabled",
    "principalArn": "arn:aws:iam::123456789012:role/your-role",
    "type": "user"
  }
}
```

Note

`taskLongArnFormat` 設定不會直接遷移現有任務。該設定只會影響啟用設定後所建立之新任務的 ARN 格式。現有的執行中任務會保留其目前的 ARN 格式，直到透過正常服務操作 (例如部署或擴展活動) 該其取代。

程序

使用下列方法遷移服務 ARN。

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. 在服務區段中，選擇 ARN 資料欄中包含警告的服務。

服務資訊頁面隨即顯示。

4. 選擇遷移至長 ARN。

「遷移服務」對話方塊隨即顯示。

5. 選擇 Migrate (遷移)。

CLI

完成先決條件後，即可標記服務。執行以下命令：

Amazon ECS 會將「針對具有短 ARN 的服務，在 tag-resource API 請求中傳遞長 ARN 格式」的行為，視為一種要求將該服務遷移至使用長 ARN 格式的訊號。

```
aws ecs tag-resource \  
  --resource-arn arn:aws:ecs:region:aws_account_id:service/cluster-name/service-name \  
  --tags key=key1,value=value1
```

下列範例將使用標籤標記 MyService，並將鍵設定為 "TestService"，值設定為 "WebServers"：

```
aws ecs tag-resource \  
  --resource-arn arn:aws:ecs:us-east-1:123456789012:service/MyCluster/MyService \  
  --tags key=TestService1,value=WebServers
```

Terraform

完成先決條件後，即可標記服務。建立 `aws_ecs_service` 資源並設定 tags 參考。如需詳細資訊，請參閱 Terraform 文件中的 [Resource : aws_ecs_service](#)。

```
resource "aws_ecs_service" "MyService" {
```

```
name      = "example"
cluster = aws_ecs_cluster.MyService.id

tags = {
  "Name"    = "MyService"
  "Environment" = "Production"
  "Department" = "QualityAssurance"
}
```

後續步驟

您可以將標籤新增至服務。如需詳細資訊，請參閱[將標籤新增至 Amazon ECS 資源](#)。

如果希望 Amazon ECS 將標籤從任務定義或服務傳播至任務，請使用 `propagateTags` 參數執行 `update-service`。如需詳細資訊，請參閱《AWS Command Line Interface 參考》中的[update-service](#)。

疑難排解

某些使用者從短 ARN 格式遷移至長 ARN 格式時，可能會遇到下列錯誤。

There was an error while migrating the ARN of service *service-name*. The specified account does not have `serviceLongArnFormat` or `taskLongArnFormat` account settings enabled. Add account settings in order to enable tagging.

若已啟用 `serviceLongArnFormat` 帳戶設定，但仍遇到此錯誤，可能是因為尚未為最初建立服務的特定 IAM 主體啟用長 ARN 格式的帳戶設定。

1. 識別建立服務的主體。

1. 在主控台中，可在 Amazon ECS 主控台的「服務詳細資訊」頁面上的組態與聯網索引標籤中的建立者欄位中取得此資訊。
2. 針對 AWS CLI，執行下列命令：

將 *user-input* 取代為實際值。

```
aws ecs describe-services --cluster cluster-name --services service-name --query 'services[0].{createdBy: createdBy}'
```

2. 為該特定主體啟用所需的帳戶設定。您可採用下列其中一種方式來這麼做：

- a. 擔任該主體的 IAM 使用者或角色。然後執行 `put-account-setting`。
- b. 使用根使用者執行命令，同時使用 `principal-arn` 指定建立主體。

範例。

將 *principal-arn* 取代為步驟 1 中的值。

```
aws ecs put-account-setting --name serviceLongArnFormat --value enabled --  
principal-arn arn:aws:iam::123456789012:role/jdoe
```

這兩種方法都會在建立服務的主體上啟用所需的 `serviceLongArnFormat` 帳戶設定，從而允許繼續進行 ARN 遷移。

Amazon ECS 服務限流邏輯

Amazon ECS 服務排程器包含保護性邏輯，當任務重複啟動失敗時，會對任務啟動進行限流。這有助於防止不必要的資源消耗並降低成本。

如果服務中的任務未能從 `PENDING` 狀態轉換為 `RUNNING` 狀態，而是直接移至 `STOPPED` 狀態，排程器會：

- 遞增重新啟動嘗試之間的時間
- 持續增加延遲，直至兩次嘗試之間的時間達到上限 27 分鐘
- 產生服務事件訊息以通知您問題

Note

27 分鐘的延遲期間上限可能會在未來更新中發生變更。

啟用限流後，您會收到此服務事件訊息：

```
(service service-name) is unable to consistently start tasks successfully.
```

節流邏輯的重要特性：

- 服務會無限期地繼續重試嘗試

- 唯一的修改是增加了重新啟動之間的時間
- 沒有使用者可設定的參數

解決限流問題

若要解決限流問題，您可以：

- 更新服務以使用新的任務定義，這會立即將服務回復到正常、非限流的運作狀態。如需詳細資訊，請參閱[更新 Amazon ECS 服務](#)。
- 解決導致任務失敗的根本原因。

觸發限流的常見任務失敗原因包括：

- 叢集資源不足 (連接埠、記憶體或 CPU)
 - 由[資源不足的服務事件訊息](#)指示
- 容器映像提取失敗
 - 可能由無效的映像名稱、標籤或許可不足造成
 - 導致 [檢視 Amazon ECS 已停止任務錯誤](#) 中的 `CannotPullContainerError`
- 磁碟空間不足
 - 導致[已停止任務錯誤](#)中的 `CannotCreateContainerError`
 - 如需解析步驟，請參閱[在 Amazon ECS 中對 Docker API error \(500\): devmapper 進行疑難排解](#)

Important

以下情況不會觸發限流邏輯：

- 達到 RUNNING 狀態後停止的任務
- 因 Elastic Load Balancing 運作狀態檢查失敗而停止的任務
- 容器命令在達到 RUNNING 狀態後以非零代碼結束的任務

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。如果未指定啟動類型，預設會使用預設的 `capacityProviderStrategy`。

更新服務時，此參數會觸發新的服務部署。

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。

更新服務時，此參數不會觸發新的服務部署。

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 是用於將容量提供者與叢集相關聯。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。可以使用 CreateCapacityProvider API 操作來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，且只需要與要使用的叢集建立關聯。

PutClusterCapacityProviders API 操作是用來在建立叢集之後，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。

更新服務時，此參數會觸發新的服務部署。

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。

在 Amazon EC2 上託管的 Amazon 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)。

更新服務時，此參數會觸發新的服務部署。

AWS Fargate 平台版本用於參考 Fargate 任務基礎設施的特定執行期環境。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 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。最多允許 255 個字母（大寫和小寫）、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 REPLICAS 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：

- REPLICAS - 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[複本排程策略](#)。
- 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[常駐程式排程策略](#)。



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。

更新服務時，此參數不會觸發新的服務部署。

如果使用 REPLICAS 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。

當您使用服務自動擴展時，如果以少於目前執行中任務數量的 desiredCount 更新目前執行中的服務，該服務會縮減規模至指定的 desiredCount。

Deployment configuration (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。

更新服務時，此參數不會觸發新的服務部署。

`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，如果服務使用 REPLICAS 服務排程器、`desiredCount` 為 4 項任務，且 `maximumPercent` 值為 200%，則排程器可能會先啟動四項新任務，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 REPLICAS 服務排程器的服務，其預設 `maximumPercent` 值為 200%。

只要用於啟動替代任務的叢集資源可用，Amazon ECS 排程器就會使用此參數，透過先啟動替代任務然後停止運作狀態不良的任務，來取代運作狀態不良的任務。如需有關排程器如何取代運作狀態不良任務的詳細資訊，請參閱 [Amazon ECS 服務](#)。

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。

如果服務使用藍/綠 (CODE_DEPLOY) 或 EXTERNAL 部署類型且任務使用 EC2 啟動類型，則百分比上限值會設定為預設值。該值用於定義容器執行個體處於 DRAINING 狀態時，服務中保持在 RUNNING 狀態的任務數量上限。

 Note

對於使用藍/綠 (CODE_DEPLOY) 或 EXTERNAL 部署類型且任務使用 EC2 的服務，您無法指定自訂的 `maximumPercent` 值。

如果服務使用藍/綠 (CODE_DEPLOY) 或 EXTERNAL 部署類型，且服務中的任務使用 Fargate，則不會使用百分比上限值。在描述服務時仍會傳回該值。

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，如果服務的 `desiredCount` 為 4 項任務，且 `minimumHealthyPercent` 為 50%，`maximumPercent` 為 100%，則服務排程器會先停止兩項現有任務以釋出叢集容量，再啟動兩項新任務。

如果任何任務運作狀態不良，並且 `maximumPercent` 不允許 Amazon ECS 排程器啟動替代任務，則排程器會將 `minimumHealthyPercent` 用作限制條件，逐個停止運作狀態不佳的任務，清理容量以啟動替代任務。如需有關排程器如何取代運作狀態不良任務的詳細資訊，請參閱 [Amazon ECS 服務](#)。

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 DAEMON 的服務 `minimumHealthyPercent` 預設值為 0% AWS CLI、AWS SDKs 和 APIs，以及 50% AWS 管理主控台。

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。

如果服務使用藍/綠 (CODE_DEPLOY) 或 EXTERNAL 部署類型且使用 EC2 的任務在執行中，則運作狀態百分比下限值會設定為預設值。該值用於定義容器執行個體處於 DRAINING 狀態時，服務中保持在 RUNNING 狀態的任務數量下限。

 Note

對於使用藍/綠 (CODE_DEPLOY) 或 EXTERNAL 部署類型且任務使用 EC2 的服務，您無法指定自訂的 `maximumPercent` 值。

若服務使用的是藍/綠 (CODE_DEPLOY) 或 EXTERNAL 部署類型，並且使用 Fargate 的任務在執行中，則不會使用運作狀態百分比下限值，儘管描述服務時會傳回該值。

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](#)。

更新服務時，此參數不會觸發新的服務部署。

`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

Amazon ECS 部署控制器支援多種部署策略：滾動更新、藍/綠、線性和金絲雀。滾動更新部署類型涉及將容器的目前執行版本取代為最新版本。藍/綠部署會建立新的環境，並

一次轉移流量。線性部署會以相等的百分比增量逐漸轉移流量。Canary 部署會先轉移一小部分的流量，然後轉移剩餘的流量。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/綠部署類型 (CODE_DEPLOY) 使用 CodeDeploy 支援的藍/綠部署模式，讓您可以在傳送生產流量之前驗證新部署的服務。

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。如果使用 Fargate，則不支援任務置放限制條件。

更新服務時，此參數不會觸發新的服務部署。

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[建立表達式以定義 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。

更新服務時，此參數不會觸發新的服務部署。

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。

Tags (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取值。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](#)。

更新服務時，此參數不會觸發新的服務部署。

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。

enableECSTags

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[將標籤用於計費](#)。

更新服務時，此參數不會觸發新的服務部署。

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。

更新服務時，此參數不會觸發新的服務部署。

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，則需要 awsvpc 網路模式。如需有關 EC2 聯網的更多資訊，請參閱[EC2 的 Amazon ECS 任務聯網選項](#)。如需有關 Fargate 聯網的更多資訊，請參閱[Fargate 的 Amazon ECS 任務聯網選項](#)。

更新服務時，此參數會觸發新的服務部署。

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

所謂寬限期是指 Amazon ECS 服務排程器在任務首次啟動後，會在這段期間內 (以秒為單位) 忽略運作狀態不良的 Elastic Load Balancing、VPC Lattice 與容器運作狀態檢查。如果您未指定運作狀態檢查寬限期間值，則會使用預設值 0。如果您沒有使用任何運作狀態檢查，則 `healthCheckGracePeriodSeconds` 處於未使用狀態。

若服務的任務需要一些時間來啟動及回應，您可以指定運作狀態檢查的寬限期，最長為 2,147,483,647 秒 (約 69 年)。在此期間，Amazon ECS 服務排程器會忽略運作狀態檢查狀態。此寬限期可防止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。

更新服務時，此參數不會觸發新的服務部署。

`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。

更新服務時，此參數會觸發新的服務部署。

建立服務之後，負載平衡器組態無法從 AWS 管理主控台變更。您可以使用 AWS Copilot AWS CloudFormation AWS CLI 或 SDK 來修改滾動 ECS 部署控制器的負載平衡器組態，而不是 AWS CodeDeploy 藍/綠或外部。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。如需有關如何修改組態的資訊，請參閱 Amazon Elastic Container Service API 參考 中的 [UpdateService](#)。

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 的任務，容器執行個體必須允許傳入連接埠映射的 `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。

更新服務時，此參數不會觸發新的服務部署。

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。

更新服務時，此參數會觸發新的服務部署。

如需詳細資訊，請參閱[使用 Service Connect 以利用簡稱連線 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

用於 Service Connect 之 AWS Cloud Map 命名空間的簡短名稱或完整 Amazon Resource Name (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。如需詳細資訊 AWS Cloud Map，請參閱《AWS Cloud Map 開發人員指南》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。

物件會從任務定義中選取連接埠、指派 AWS Cloud Map 服務的名稱，以及一組別名（也稱為端點）和連接埠，供用戶端應用程式參考此服務。

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

discoveryName 是 Amazon ECS 為此 Amazon ECS AWS Cloud Map 服務建立的新服務的名稱。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日誌組態。這不是使用 fluent-bit 或 fluentd 容器映像的 FireLens 日誌路由器容器。

accessLogConfiguration

類型：[ServiceConnectAccessLogConfiguration](#) 物件

必要：否

Service Connect 存取記錄的組態，包括日誌格式，以及日誌是否應包含查詢字串。存取日誌會擷取對服務提出之請求的詳細資訊，包括請求模式、回應碼和時間資料。若要啟用存取日誌，您還必須在 logConfiguration 中指定 serviceConnectConfiguration。

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[使用服務探索以利用 DNS 名稱連接 Amazon ECS 服務](#)。

更新服務時，此參數會觸發新的服務部署。

`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檔為 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。

可用區域重新平衡

availabilityZoneRebalancing

類型：字串

必要：否

指出服務是否使用可用區域重新平衡。有效值為 ENABLED 和 DISABLED。

更新服務時，此參數不會觸發新的服務部署。

預設行為：

- 對於新服務：如果未指定任何值，則預設值為 DISABLED。
- 對於現有服務：如果未指定任何值，Amazon ECS 會將該值預設為現有值。如果先前未設定任何值，Amazon ECS 會將該值設定為 DISABLED。

如需有關可用區域重新平衡的詳細資訊，請參閱[在可用區域之間平衡 Amazon ECS 服務](#)。

磁碟區組態

volumeConfigurations

類型：物件

必要：否

用於為由服務管理的任務建立磁碟區的組態。只有任務定義中標記為 `configuredAtLaunch` 的磁碟區才能使用此物件進行設定。

更新服務時，此參數會觸發新的服務部署。

將 Amazon EBS 磁碟區連接至由服務管理的任務時，需要此物件。如需詳細資訊，請參閱[將 Amazon EBS 磁碟區與 Amazon ECS 搭配使用](#)。

name

類型：字串

必要：是

建立或更新服務時設定的磁碟區名稱。最多允許 255 個字元 (大小寫)、數字、底線 (_) 與連字號 (-)。此值必須與任務定義中指定的磁碟區名稱相符。

managedEBSVolume

類型：物件

必要：否

用於建立 Amazon EBS 磁碟區的磁碟區組態，這些磁碟區會連接至服務建立或更新時由服務維護的任務。每項任務連接一個磁碟區。

encrypted

類型：布林值

必要：否

有效值：true|false

指定是否加密每個建立的 Amazon EBS 磁碟區。如果您已為的特定開啟 Amazon EBS 加密，AWS 區域 AWS 帳戶 但將此參數設定為 false，則會覆寫此參數，而且磁碟區會使用預設為加密指定的 KMS 金鑰進行加密。如需有關 Amazon EBS 預設加密的詳細資訊，請參閱 Amazon EBS User Guide 中的 [Enable Amazon EBS encryption by default](#)。如需有關加密連接至 Amazon ECS 任務之 Amazon EBS 磁碟區的詳細資訊，請參閱[加密儲存於連接至 Amazon ECS 任務之 Amazon EBS 磁碟區中的資料](#)。

kmsKeyId

類型：字串

必要：否

用於 Amazon EBS 加密的 AWS Key Management Service (AWS KMS) 金鑰識別符。如果指定 kmsKeyId，加密狀態必須是 true。

使用此參數指定的金鑰會覆寫 Amazon EBS 預設金鑰，或可能已指定的任何叢集層級 KMS 金鑰 (用於 Amazon ECS 受管儲存加密)。如需詳細資訊，請參閱[加密儲存於連接至 Amazon ECS 任務之 Amazon EBS 磁碟區中的資料](#)。

您可以使用下列任何方式指定 KMS 金鑰：

- 金鑰 ID – 例如 1234abcd-12ab-34cd-56ef-1234567890ab。
- 金鑰別名 – 例如 alias/ExampleAlias。
- 金鑰 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 會以非同步方式驗證 KMS 金鑰。因此，如果指定的 ID、別名或 ARN 無效，則動作雖顯示成功，但最終會失敗。如需詳細資訊，請參閱[對 Amazon EBS 磁碟區連接問題進行疑難排解](#)。

volumeType

類型：字串

必要：否

有效值：gp2|gp3|io1|io2|sc1|st1|standard

Amazon EBS 磁碟區類型。如需有關磁碟區類型的詳細資訊，請參閱 Amazon EBS User Guide 中的 [Amazon EBS volume types](#)。預設磁碟區類型為 gp3。

 Note

Fargate 任務不支援 standard 磁碟區類型。

sizeInGiB

類型：整數

必要：否

有效範圍：介於 1 到 16,384 之間的整數

EBS 磁碟區的大小 (GiB)。如果未提供快照 ID 來設定要連接的磁碟區，則必須提供大小值。如果使用快照來設定要連接的磁碟區，則預設值為快照大小。然後，您可以指定大於或等於快照大小的大小。

對於 gp2 與 gp3 磁碟區類型，有效值範圍為 1-16,384。

對於 io1 與 io2 磁碟區類型，有效值範圍為 4-16,384。

對於 st1 與 sc1 磁碟區類型，有效值範圍為 125-16,384。

對於 standard 磁碟區類型，有效值範圍為 1-1,024。

snapshotId

類型：字串

必要：否

現有 Amazon EBS 磁碟區的快照 ID，Amazon ECS 可將其用於建立要連接的新磁碟區。您必須擇一指定 snapshotId 或 sizeInGiB。

volumeInitializationRate

類型：整數

必要：否

從現有 Amazon EBS 磁碟區的快照擷取資料以建立要連接之新磁碟區的速率 (以 MiB/s 為單位)。只有在您指定了 snapshotId 時，才能指定此屬性。如需有關此磁碟區初始化速率的詳細資訊，包括支援的初始化速率範圍，請參閱 Amazon EBS User Guide 中的 [Initialize Amazon EBS volumes](#)。

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，以及磁碟區為爆量而累積的 I/O 額度比率。這是 io1 和 io2 磁碟區的必要參數。gp2、st1、sc1 或 standard 磁碟區不支援此參數。

對於 gp3 磁碟區，有效值範圍為 3,000 到 16,000。

對於 io1 磁碟區，有效值範圍為 100 到 64,000。

對於 io2 磁碟區，有效值範圍為 100 到 64,000。

throughput

類型：整數

必要：否

為連接至由服務維護之任務的磁碟區佈建的輸送量。

 Important

僅 gp3 磁碟區支援此參數。

roleArn

類型：字串

必要：是

基礎設施 (IAM) 角色的 Amazon Resource ARN AWS Identity and Access Management (ARN)，提供 Amazon ECS 許可來管理任務的 Amazon EBS 資源。如需詳細資訊，請參閱[Amazon ECS 基礎結構 IAM 角色](#)。

tagSpecifications

類型：物件

必要：否

要套用至每個 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

套用至磁碟區以協助您對其進行分類與整理的中繼資料。每個標籤皆由您定義的鍵與選用值組成。AmazonECSCreated 與 AmazonECSManaged 都是 Amazon ECS 代表您新增的預留標籤，因此您最多可以指定 48 個自己的標籤。刪除磁碟區時，也會一併刪除標籤。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選用部分。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

指定是否將標籤從任務定義或服務複製到磁碟區。如果指定了 NONE 或未指定任何值，則不會複製標籤。

fileSystemType

類型：字串

必要：否

有效值：xfs|ext3|ext4|NTFS

磁碟區上的檔案系統類型。磁碟區的檔案系統類型決定了資料在磁碟區中的儲存與擷取方式。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。如果有檔案系統類型不相符，則任務將無法啟動。

Linux 的有效值為 xfs、ext3 與 `ext4`。連接至 Linux 任務之磁碟區的預設值為 XFS。

Windows 的有效值為 NTFS。連接至 Windows 任務之磁碟區的預設值為 NTFS。

服務定義範本

以下顯示 Amazon ECS 服務定義的 JSON 表示法。

EC2

```
{
  "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,
    "availabilityZoneRebalancing": "ENABLED",
    "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": "",
      "volumeInitializationRate": 0,
      "iops": 0,
      "throughput": 0,
      "tagSpecifications": [
        {
          "resourceType": "volume",
          "tags": [
            {
              "key": "",
              "value": ""
            }
          ],
          "propagateTags": "NONE"
        }
      ],
      "roleArn": "",
      "filesystemType": ""
    }
  }
]
}

```

Fargate

```

{
  "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": ""
  }
],
"enableECSTags": true,
"propagateTags": "TASK_DEFINITION",
"enableExecuteCommand": true,
"availabilityZoneRebalancing": "ENABLED",
"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": "",
                "volumeInitializationRate": 0,
                "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。您可以編輯標籤索引鍵和值，也可以隨時從資源中移除標籤。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。如果您刪除資源，也會刪除該資源的任何標籤。

如果您使用 AWS Identity and Access Management (IAM)，您可以控制 AWS 帳戶中哪些使用者具有管理標籤的許可。

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 管理主控台、Amazon ECS API AWS CLI、或 AWS SDK 手動標記資源。
- 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。

這些標籤會傳回 AWS CLI 和 AWS SDK 回應，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。

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: <i>arn</i>
Amazon ECS 任務集	是	否	N/A
Amazon ECS 任務定義	是	否	索引鍵：ecs:taskDefinition:createdFrom Value (值)：ecs-console-v2

資源	支援標籤	支援標籤傳播	主控台自動新增的標籤
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: <i>arn</i> 索引鍵 : aws:cloudformation:stack-name Value (值) : ECS-Console-V2-Cluster- <i>EXAMPLE</i>
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 的 Amazon EC2 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。	是，僅可透過 Amazon ECS 受管執行個體的容量提供者實現。您可以將標籤從 Amazon ECS 受管執行個體的容量提供者傳播至提供者管理的所有資源，例如 Amazon EC2 執行個體、啟動範本、彈性網路介面與磁碟區。	N/A

在建立期間標記資源

下列資源支援使用 Amazon ECS API AWS CLI 或 AWS SDK 建立時標記：

- 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。當 AWS 帳戶設定為標記授權時，使用者必須具有建立資源之動作的許可，例如 `ecsCreateCluster`。如果您在資源建立動作中指定標籤，AWS 會執行額外的授權，以驗證使用者或角色是否具有建立標籤的許可。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”](#)。如需關於如何設定此選項的資訊，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。這些僅供保留 AWS 使用。您不可編輯或刪除具此字首的標籤金鑰或值。具此字首的標籤不算在每一資源的標籤數限制內。

Amazon ECS 受管標籤

當您使用 Amazon ECS 受管標籤時，Amazon ECS 會自動標記包含叢集資訊的所有新啟動的任務以及任何連接到任務的 Amazon EBS 磁碟區，使用的標籤是使用者新增的任務定義標籤或服務標籤。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。連接至獨立任務的 Amazon EBS 磁碟區會收到一個鍵為 `aws:ecs:clusterName` 且值設定為叢集名稱的標籤。如需有關 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。

連接至屬於服務之任務的 Amazon EBS 磁碟區將會收到一個鍵為 `aws:ecs:clusterName` 且值設定為叢集名稱的標籤，以及一個鍵為 `aws:ecs:serviceName` 且值設定為服務名稱的標籤。如需有關 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 `enableECSTags` 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，例如叢集自動擴展。建議不要手動修改標籤，以便 Amazon ECS 能夠有效管理這些功能。

將標籤用於計費

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。

將標籤新增至 Amazon ECS 資源

您可以標記新的或現有的任務、服務、任務定義和叢集。如需有關標記容器執行個體的資訊，請參閱[將標籤新增至 Amazon ECS 的 Amazon EC2 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 服務都可以存取標籤，包括帳單。標籤不適用於私有或敏感資料。

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。	將應用程式作為 Amazon ECS 任務執行	run-task	RunTask
建立服務。	建立 Amazon ECS 滾動更新部署	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 工作負載建立 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
為 Amazon ECS 受管執行個體建立容量提供者。	為 Amazon ECS 受管執行個體建立容量提供者	create-capacity-provider	CreateCapacityProvider

將標籤新增至現有資源 (Amazon ECS 主控台)

您可直接從資源的頁面新增或刪除與叢集、服務、任務和任務定義相關聯的標籤。

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，選取要 AWS 區域 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。
 - 在值欄位中，輸入金鑰值。
6. 選擇儲存。

將標籤新增至現有資源 (AWS CLI)

您可以使用 或 AWS CLI API 新增或覆寫一或多個標籤。

Note

您可以使用雙堆疊服務端點，透過 IPv4 和 IPv6 AWS CLI 從、SDKs 和 Amazon ECS API 與 Amazon ECS 互動。如需詳細資訊，請參閱[使用 Amazon ECS 雙堆疊端點](#)。

- AWS CLI - [tag-resource](#)
- API 動作：[TagResource](#)

Amazon ECS 受管執行個體的標籤

Amazon ECS 受管執行個體支援由 新增的自訂標籤和標籤組合 AWS，可用於成本最佳化。如需有關將標籤用於計費的詳細資訊，請參閱[將標籤用於計費](#)。

由 新增的標籤 AWS

AWS 會將下列標籤新增至容量提供者建立的每個 Amazon ECS 受管執行個體：

- Amazon ECS 會自動將預留標籤 `AmazonECSCreated` 與 `AmazonECSManaged` 新增至 Amazon ECS 受管執行個體。
- Amazon ECS 會將如下系統標籤新增至每個執行個體：
 - 鍵為 `aws:ecs:clusterName` 且值設定為叢集名稱的標籤。
 - 鍵為 `aws:ecs:capacityProviderName` 且值設定為容量提供者名稱的標籤。
 - 鍵為 `aws:ecs:containerInstanceId` 且值為 Amazon ECS 受管執行個體容器執行個體 ID 的標籤。
- Amazon EC2 會新增值為 `ecs-managed-instances` 的系統標籤 `aws:ec2:managed-launch`。此外，Amazon EC2 會新增系統標籤，表示用來建立受管執行個體的啟動範本，以及受管執行個體所屬的 Amazon EC2 Fleet。

自訂標籤

您可以透過將標籤新增至容量提供者，同時使用 `propagateTags` 屬性啟用標籤傳播，將額外的自訂標籤新增至 Amazon ECS 受管執行個體。容量提供者上的標籤會傳播到容量提供者啟動的受管執行個體

下列容量提供者定義範例示範如何在使用 `propagateTags` 的 `CAPACITY_PROVIDER` 值建立容量提供者時，透過容量提供者指定與傳播標籤。

```
{
  "name": "my-cluster-managed-instances-cp",
  "cluster": "my-cluster",
  "tags": [
    {
      "key": "tag_key",
      "value": "tag_value"
    }
  ],
  "managedInstancesProvider": {
    "infrastructureRoleArn": "arn:aws:iam::123456789012:role/ecsInfrastructureRole",
    "propagateTags": "CAPACITY_PROVIDER",
    "instanceLaunchTemplate": {
```

```
    "ec2InstanceProfileArn": "arn:aws:iam::123456789012:instance-profile/ecsInstanceProfile",
    "networkConfiguration": {
      "subnets": [
        "subnet-abcdef01234567",
        "subnet-bcdefa98765432"
      ],
      "securityGroups": [
        "sg-0123456789abcdef"
      ]
    }
  }
}
```

Note

- 在將新標籤新增至容量提供者時，新增的標籤不會傳播至現有執行個體，但會傳播至任何新建立的執行個體。
- 客戶可以直接將標籤新增至 EC2 執行個體，但這些標籤不會傳播回容量提供者
- 您也可以使用 `或` 標記或取消標記容量提供者 [將標籤新增至現有資源 \(AWS CLI\)](#) [將標籤新增至現有資源 \(Amazon ECS 主控台\)](#)

如需有關 Amazon ECS 受管執行個體容量提供者的詳細資訊，請參閱 [Amazon ECS 受管執行個體容量提供者](#)。

將標籤新增至 Amazon ECS 的 Amazon EC2 容器執行個體

您可以使用以下其中一種方法，將標籤與 Amazon ECS Amazon EC2 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。

JSON

```
{
```

```
"Version": "2012-10-17",
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "ec2:DescribeTags"
    ],
    "Resource": "*"
  }
]
```

將標籤新增至 Amazon ECS 的外部容器執行個體

您可以使用以下其中一種方法，將標籤與 Amazon ECS 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 – 將外部執行個體註冊到您的叢集之後，您可以使用 AWS 管理主控台 來新增標籤。如需詳細資訊，請參閱[將標籤新增至現有資源 \(Amazon ECS 主控台\)](#)。

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運算選項。

針對使用 Fargate 的任務，lineItem/Operation 欄會顯示 FargateTask，而且您會看到與每個任務相關聯的費用。

針對使用 EC2 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[將標籤用於計費](#)。

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

如需建立 AWS 成本和用量報告的詳細資訊，請參閱 AWS Billing 《使用者指南》中的[AWS 成本和用量報告](#)。

任務層級成本和用量報告

AWS Cost Management 可以在 AWS Cost and Usage Report 為 Amazon ECS 上的每個任務提供 CPU 和記憶體用量資料，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。如需分割成本分配資料的詳細資訊，請參閱 AWS Cost and Usage Report 《使用者指南》中的[了解分割成本分配資料](#)。

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。

設定拆分成本分配資料後，報表中的 `splitLineItem` 標頭下將會有額外資料欄。如需詳細資訊，請參閱 AWS Cost and Usage Report 《使用者指南》中的[分割明細項目詳細資訊](#)

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。

先決條件如下：

- 將 `ECS_DISABLE_METRICS` Amazon ECS 代理程式組態參數設定為 `false`。

若此設定為 `false`，Amazon ECS 代理程式會將指標傳送至 Amazon CloudWatch。在 Linux 上，此設定預設為 `false`，並將指標傳送至 CloudWatch。在 Windows 上，此設定 `true` 預設為 `true`，因此您必須將設定變更為 `false`，才能將指標傳送至 CloudWatch，AWS Cost Management 以供使用。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 AWS Cost and Usage Report 《使用者指南》中的[建立成本和用量報告](#)。

AWS Cost Management 會使用任務 CPU 和記憶體用量計算分割成本分配資料。AWS Cost Management 如果用量不可用，可以使用任務 CPU 和記憶體保留，而不是用量。如果您看到 CUR 正在使用預留，請檢查您的容器執行個體是否符合先決條件，並在 CloudWatch 中顯示任務資源使用量指標。

監控 Amazon ECS

監控是維護 Amazon ECS 和 AWS 解決方案可靠性、可用性和效能的重要部分。您應該從 AWS 解決方案的所有部分收集監控資料，以便在發生多點失敗時更輕鬆地偵錯。開始監控 Amazon ECS 之前，請建立監控計畫來回答下列問題：

- 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運算選項而定。如果您使用 Fargate 執行服務，則會提供 CPU 與記憶體使用率指標來協助您監控服務。如果使用的是 Amazon EC2，您擁有且需要監控構成底層基礎結構的 EC2 執行個體。此外，會在叢集、服務和任務層級上提供額外的 CPU 與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

使用下列最佳實務來監控 Amazon ECS。

- 讓監控優先攔截小問題，以免問題越形嚴重
- 建立包含下列問題答案的監控計畫

- 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？
- 盡可能自動化執行監控。
- 檢查 Amazon ECS 日誌檔案。如需詳細資訊，請參閱[檢視 Amazon ECS 容器代理程式日誌](#)。
- 使用執行期監控協助保護 AWS 環境中的帳戶、容器、工作負載和資料。如需詳細資訊，請參閱[使用執行時期監控識別未經授權的行為](#)。

Amazon ECS 監控工具

AWS 提供各種工具，您可以用來監控 Amazon ECS。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：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。此動作是傳送到 Amazon Simple Notification Service (Amazon SNS) 主題或 Amazon EC2 Auto Scaling 政策的通知。CloudWatch 警示不會只因處於特定狀態就調用動作，狀態必須已變更並已維持一段指定的時間。如需詳細資訊，請參閱[使用 CloudWatch 監控 Amazon ECS](#)。

如果服務的任務使用的是 Fargate，您可以使用 CloudWatch 警示，根據 CloudWatch 指標 (例如 CPU 與記憶體使用率) 來橫向擴充或縮減服務中的任務數量。如需詳細資訊，請參閱[自動擴展您的 Amazon ECS 服務](#)。

如果叢集的任務或服務使用的是 EC2，您可以使用 CloudWatch 警示，根據 CloudWatch 指標 (例如叢集記憶體保留量) 來橫向擴充或縮減容器執行個體數量。

對於利用 Amazon ECS 最佳化 Amazon Linux AMI 啟動的容器執行個體，您可以使用 CloudWatch Logs 在一個便利位置檢視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 CloudWatch 使用者指南中的[使用命令列下載並設定](#)

[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[監控容器執行個體許可](#)。

- Amazon CloudWatch Logs：透過在任務定義中指定 awslogs 日誌驅動程式，藉以監控、存放及存取您的 Amazon ECS 任務中容器的日誌檔案。如需詳細資訊，請參閱[將 Amazon ECS 日誌傳送至 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 的容器。

- Amazon CloudWatch Events：匹配事件並將它們路由至一或多個目標函式或串流以進行變更、擷取狀態資訊，以及採取修正動作。如需詳細資訊，請參閱本指南中的[使用 EventBridge 自動化因應 Amazon ECS 錯誤](#)，以及 Amazon EventBridge User Guide 中的[EventBridge is the evolution of Amazon CloudWatch Events](#)。
- Container Insights – 收集、彙總及總結容器化應用程式與微服務中的指標及日誌。Container Insights 會使用內嵌指標格式將資料收集為效能日誌事件。這些效能日誌事件是使用結構化 JSON 結構描述的項目，支援大規模擷取與儲存高基數資料。CloudWatch 可透過這些資料，在叢集、任務和服務層級建立彙總指標，作為 CloudWatch 指標。您可在 CloudWatch 自動儀表板取得 Container Insights 收集的指標，也可在 CloudWatch 主控台的指標區段進行檢視。
- AWS CloudTrail 日誌監控 – 在帳戶之間共用日誌檔案、透過將日誌檔案傳送到 CloudWatch Logs 來即時監控 CloudTrail 日誌檔案、在 Java 中寫入日誌處理應用程式，以及驗證您的日誌檔案在 CloudTrail 交付後並未變更。如需詳細資訊，請參閱本指南中的[使用記錄 Amazon ECS API 呼叫 AWS CloudTrail](#)，以及《AWS CloudTrail 使用者指南》中的[使用 CloudTrail 記錄檔案](#)。
- 執行時期監控 – 偵測 AWS 環境中叢集與容器面臨的威脅。執行時期監控使用 GuardDuty 安全代理程式，將執行時期可見性新增至個別 Amazon ECS 工作負載，例如檔案存取、程序執行與網路連線。

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 主控台儀表板可讓您 at-a-glance 檢視 AWS 環境的狀態。我們建議您也檢查您的容器執行個體上的日誌檔，以及您的任務中的容器。

- Amazon ECS 主控台：
 - EC2 的叢集指標
 - 服務指標
 - 服務運作狀態

- 服務部署事件
- CloudWatch 首頁：
 - 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 CloudWatch 執行下列動作：

- 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趨勢的圖形指標資料。
- 搜尋和瀏覽您的所有 AWS 資源指標。
- 建立與編輯要通知發生問題的警示。
- 容器運作狀態檢查 – 這是在容器本機執行的命令，用於驗證應用程式的運作狀態與可用性。您可以在任務定義中為每個容器設定該檢查。
- AWS Trusted Advisor 可協助您監控 AWS 資源，以提高效能、可靠性、安全性和成本效益。所有使用者皆可使用四個 Trusted Advisor 檢查；具有商業或企業支援計劃的使用者可使用超過 50 個檢查。如需詳細資訊，請參閱 [AWS Trusted Advisor](#)。

Trusted Advisor 具有與 Amazon ECS 相關的這些檢查：

- 容錯能力檢查 – 警示您有服務正在單一可用區域中執行。
- 容錯能力檢查 – 警示您未針對多可用區域使用分散置放策略。
- AWS Compute Optimizer 是一種服務，可分析 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 Amazon ECS 的建議](#)。

- Amazon ECS MCP 伺服器 – 使用 AI 助理監控部署、疑難排解容器問題，並使用自然語言檢查組態。MCP 伺服器提供服務運作狀態的即時可見性、分析任務失敗，並加速問題解決。如需詳細資訊，請參閱 [Amazon ECS MCP 伺服器](#)。

使用 CloudWatch 監控 Amazon ECS

您可以使用 Amazon CloudWatch 來監控 Amazon ECS 資源，前者會收集來自 Amazon ECS 的原始資料，並處理為可讀且近乎即時的指標。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 指標資料會自動以 1 分鐘的期間傳送到 CloudWatch。如需有關 CloudWatch 的詳細資訊，請參閱 [Amazon CloudWatch 使用者指南](#)。

Amazon ECS 會免費提供叢集和服務的指標。您可支付額外費用，為叢集啟用 Amazon ECS CloudWatch Container Insights 以取得每個任務的指標，包括 CPU、記憶體與 EBS 檔案系統使用率。如需更多 Container Insights 的相關資訊，請參閱 [使用具有增強可觀測性的 Container Insights 監控 Amazon ECS](#)。

考量事項

使用 Amazon ECS CloudWatch 指標時，建議考量下列事項。

- 任何在 Fargate 上託管的 Amazon ECS 服務會自動產生 CloudWatch CPU 和記憶體使用率指標，因此不需採取任何手動步驟。
- 對於託管在 Amazon EC2 執行個體上的任何 Amazon ECS 任務或服務，Amazon EC2 執行個體需要 1.4.0 版或更新版本 (Linux) 或 1.0.0 版或更新版本 (Windows) 的容器代理程式，才能產生 CloudWatch 指標。不過，我們建議您使用最新版的容器代理。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](#)」。
- 要確保產生可靠的 CloudWatch 指標，Docker 的最低版本需為 Docker 20.10.13 版及更新版本。
- Amazon EC2 容器執行個體也需要用來啟動 Amazon EC2 執行個體之 IAM 角色的 `ecs:StartTelemetrySession` 許可。如果您已在 Amazon ECS 可使用 CloudWatch 指標之前建立 Amazon ECS 容器執行個體 IAM 角色，則可能需要新增此許可。如需有關容器執行個體 IAM 角色及連接容器執行個體之受管 IAM 政策的資訊，請參閱 [Amazon ECS 容器執行個體 IAM 角色](#)。
- 您可以在 Amazon ECS 容器代理程式組態中設定 `ECS_DISABLE_METRICS=true`，停用 Amazon EC2 執行個體的 CloudWatch 指標收集功能。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 提供您可以用來監控資源的免費 CloudWatch 指標。透過這些指標，可以量測整個叢集的 CPU 與記憶體保留量及 CPU、記憶體與 EBS 檔案系統使用率，以及叢集中服務的 CPU、記憶體與 EBS 檔案系統使用率。針對 GPU 工作負載，您可以量測叢集整體的 GPU 保留。

叢集中託管 Amazon ECS 任務的基礎結構決定了哪些指標可用。對於託管在 Fargate 基礎結構上的任務，Amazon ECS 會提供 CPU、記憶體與 EBS 檔案系統使用率指標，可協助監控服務狀態。對於託管在 EC2 執行個體上的任務，Amazon ECS 會提供叢集與服務層級的 CPU、記憶體與 GPU 保留量指標，以及 CPU 與記憶體使用率指標。您需要監控單獨構成基礎設施的 Amazon EC2 執行個體。如需有關監控 EC2 執行個體的詳細資訊，請參閱 Amazon EC2 User Guide 中的 [Monitoring Amazon EC2](#)。

如需有關適用於 Amazon ECS 的建議警示的相關資訊，請參閱 Amazon CloudWatch Logs User Guide 中的下列任一主題：

- [Amazon ECS](#)
- [Amazon ECS with Container Insights](#)

檢視 Amazon ECS 指標

當叢集中有資源開始執行後，即可透過 Amazon ECS 與 CloudWatch 主控台檢視指標。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。CloudWatch 主控台提供微調和可自訂的資源顯示，以及服務中所執行任務的數量。

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. 決定要檢視的指標。

檢視下列位置的指標	步驟	
叢集	在「叢集詳細資訊」頁面中，選擇指標索引標籤。若已啟用該功能，系統亦提供前往 CloudWatch 主控台的連結，供您檢視 CloudWatch Container Insights 的指標。	
服務	在「叢集詳細資訊」頁面的服務索引標籤上，選取服務。然後，這些指標就會顯示在運作狀態與指標索引標籤上。	

CloudWatch 主控台

如果服務使用的是 Fargate，則可以在 CloudWatch 主控台中檢視 Amazon ECS 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。您可以檢視服務使用率和服務 RUNNING 任務計數。

如果服務使用的是 EC2 容量提供者，則也可以在 CloudWatch 主控台中檢視 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。

如需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 CloudWatch 指標

您可以使用 CloudWatch 用量指標來提供您帳戶的資源用量可見度。使用這些指標，以 CloudWatch 圖表和儀表板視覺化目前的服務使用狀況。

Amazon ECS 每隔 1 分鐘就會將指標傳送至 CloudWatch。這些指標為具有 RUNNING 狀態之任務的資源而收集。如果叢集、服務或其他資源沒有執行中的任務，則該期間內不會回報該資源的指標。舉例來說，如果您有叢集包含一個服務，但該服務沒有處於 RUNNING 狀態的任務，則不會向 CloudWatch 傳送任何指標。同理，如果您有兩個服務，其中一個服務有執行中的任務，而另一個沒有，則只會傳送有執行中任務之服務的指標。

指標	Description	有效維度	實用統計量	單位
CPUReservation	叢集或服務中所保留 CPU 單位的百分比。 CPU 保留量 (依 ClusterName 篩選) 的計算方式為，叢集上 Amazon ECS 任務所保留的 CPU 單元總數，除以叢集中所有已註冊 Amazon EC2 執行個體的 CPU 單元總數。僅處	ClusterName	平均值、最小值、最大值	百分比

指標	Description	有效維度	實用統計量	單位
	於 ACTIVE 或 DRAINING 狀態的 Amazon EC2 執行個體會影響 CPU 保留量指標。該指標僅適用於託管在 Amazon EC2 執行個體上的任務。			

指標	Description	有效維度	實用統計量	單位
CPUUtilization	叢集或服務或受管協助程式使用的 CPU 單位百分比。 叢集層級 CPU 使用率 (依 ClusterName 篩選) 的計算方式為，叢集上 Amazon ECS 任務所使用 CPU 單元總數，除以叢集中所有已註冊 Amazon EC2 執行個體的 CPU 單位總數。僅處於 ACTIVE 或 DRAINING 狀態的 Amazon EC2 執行個體會影響 CPU 保留量指標。叢集層級指標僅適用於託管在 Amazon EC2 執行個體上的任務。 服務層級 CPU 使用率 (依 ClusterName、ServiceName 篩選) 的計算方式為，屬於該服務之任務所	ClusterName , ServiceName	平均值、最小值、最大值	百分比

指標	Description	有效維度	實用統計量	單位
	使用的 CPU 總數，除以為這些任務保留的 CPU 單位總數。服務層級指標適用於託管在 Amazon EC2 執行個體和 Fargate 上的任務。 常駐程式層級 CPU 使用率（由篩選ClusterName，ServiceName）的測量方式為屬於受管常駐程式的任務正在使用的 CPU 單位總數，除以屬於受管常駐程式的任務預留的 CPU 單位總數。			

指標	Description	有效維度	實用統計量	單位
MemoryReservation	由叢集內執行中任務保留的記憶體百分比。 叢集記憶體保留量的計算方式為，叢集上 Amazon ECS 任務所保留的記憶體總量，除以叢集中所有已註冊 Amazon EC2 執行個體的記憶體總量。此指標只能依 ClusterName 篩選。僅處於 ACTIVE 或 DRAINING 狀態的 Amazon EC2 執行個體會影響記憶體保留量指標。叢集層級記憶體保留量指標僅適用於託管在 Amazon EC2 執行個體上的任務。	ClusterName	平均值、最小值、最大值	百分比

 **Note**
計算記憶體使用率時，如果已指定 MemoryRes

指標	Description	有效維度	實用統計量	單位
	ervation 則將以此 數值代入 計算，而 非採用記 憶體總 量。			

指標	Description	有效維度	實用統計量	單位
MemoryUtilization	叢集或服務或受管協助程式使用的記憶體百分比。 叢集層級記憶體使用率 (依 ClusterName 篩選) 的衡量方式為，叢集上 Amazon ECS 任務所使用的記憶體總量，除以叢集中所有已註冊 Amazon EC2 執行個體的記憶體總量。僅處於 ACTIVE 或 DRAINING 狀態的 Amazon EC2 執行個體會影響記憶體使用率指標。叢集層級指標僅適用於託管在 Amazon EC2 執行個體上的任務。 服務層級記憶體使用率 (依 ClusterName 、 ServiceName 篩選) 的計算方式為，屬於該服務之任務所	ClusterName , ServiceName	平均值、最小值、最大值	百分比

指標	Description	有效維度	實用統計量	單位
	使用的記憶體總量，除以為術語該服務的這些任務保留的記憶體總量。服務層級指標適用於託管在 Amazon EC2 執行個體和 Fargate 上的任務。 常駐程式層級記憶體使用率 (由篩選ClusterName , ServiceName) 的測量方式為屬於受管常駐程式的任務正在使用的總記憶體，除以屬於受管常駐程式的任務預留的總記憶體。			

指標	Description	有效維度	實用統計量	單位
EBSFilesystemUtilization	服務中任務所使用的 Amazon EBS 檔案系統百分比。 服務層級 EBS 檔案系統使用率指標 (依 ClusterName 、 ServiceName 篩選) 的計算方式為，屬於該服務之任務所使用的 EBS 檔案系統總量，除以屬於該服務的所有任務配置的 EBS 檔案系統儲存總量。服務層級 EBS 檔案系統使用率指標僅適用於託管在 Amazon EC2 執行個體 (使用容器代理程式版本 1.79.0) 與 Fargate (使用平台版本 1.4.0) 上且已連接 EBS 磁碟區的任務。	ClusterName , ServiceName	平均值、最小值、最大值	百分比

 Note
對於託管在

指標	Description	有效維度	實用統計量	單位
	Fargate 上的任務，磁碟上有僅供 Fargate 使用的空間。Fargate 使用的空間不會產生相關費用，但您可以使用 df 等工具檢視此額外儲存空間。			

指標	Description	有效維度	實用統計量	單位
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 保留量指標的計算方式為，叢集上 Amazon ECS 任務所保留的 GPU 數量，除以叢集中所有已註冊且搭載 GPU 之 Amazon EC2 執行個體的 GPU 總數。僅處於 ACTIVE 或 DRAINING 狀態的 Amazon EC2 執行個體會影響 GPU 保留量指標。	ClusterName	平均值、最小值、最大值	百分比

指標	Description	有效維度	實用統計量	單位
ActiveConnectionCount	從用戶端到在共享所選 Discovery Name 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：Discovery Name 和 Discovery Name, ServiceName, ClusterName。	Discovery Name 與 Discovery Name、ServiceName、ClusterName	平均值、最小值、最大值、總和	計數

指標	Description	有效維度	實用統計量	單位
NewConnectionCount	從用戶端到 在共享所選 Discovery Name 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。	Discovery Name 和 Discovery Name, ServiceNa me, ClusterName	平均值、最小 值、最大值、總 和	計數
Processed Bytes	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。	Discovery Name 和 Discovery Name, ServiceNa me, ClusterName	平均值、最小 值、最大值、總 和	位元組

指標	Description	有效維度	實用統計量	單位
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。 您也需在任務定義的連接埠映射中設定 appProtocol 。	Discovery Name 和 Discovery Name, ServiceName, ClusterName	平均值、最小值、最大值、總和	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。	Discovery Name 和 Discovery Name, ServiceName, ClusterName	平均值、最小值、最大值、總和	計數

指標	Description	有效維度	實用統計量	單位
HTTPCode_ Target_2X X_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	平均值、最小值、最大值、總和	計數

指標	Description	有效維度	實用統計量	單位
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。	TargetDiscoveryName 和 TargetDiscoveryName, ServiceName, ClusterName	平均值、最小值、最大值、總和	計數

指標	Description	有效維度	實用統計量	單位
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。	TargetDiscoveryName 和 TargetDiscoveryName, ServiceName, ClusterName	平均值、最小值、最大值、總和	計數

指標	Description	有效維度	實用統計量	單位
HTTPCode_ Target_5X X_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。	TargetDiscoveryName 和 TargetDiscoveryName, ServiceName, ClusterName	平均值、最小值、最大值、總和	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。	TargetDiscoveryName 和 TargetDiscoveryName, ServiceName, ClusterName	平均數	計數

指標	Description	有效維度	實用統計量	單位
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。	TargetDiscoveryName 和 TargetDiscoveryName, ServiceName, ClusterName	平均值、最小值、最大值、總和	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。	TargetDiscoveryName 和 TargetDiscoveryName, ServiceName, ClusterName	平均值、最小值、最大值	毫秒
ClientTLSNegotiationErrorCount	TLS 連線失敗的總次數。僅當已啟用 TLS 時才使用此指標。 僅當您已設定 Amazon ECS Service Connect 時，才能使用此指標。	DiscoveryName 與 DiscoveryName、ServiceName、ClusterName	平均值、最小值、最大值、總和	計數

指標	Description	有效維度	實用統計量	單位
TargetTLSNegotiationErrorCount	TLS 連線因缺少用戶端憑證、AWS 私有 CA 驗證失敗或 SAN 驗證失敗而失敗的總次數。僅當已啟用 TLS 時才使用此指標。 僅當您已設定 Amazon ECS Service Connect 時，才能使用此指標。	ServiceName、ClusterName、TargetDiscoveryName 與 TargetDiscoveryName	平均值、最小值、最大值、總和	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。舉例來說，如果您有叢集包含一個服務，但該服務沒有處於 RUNNING 狀態的任務，則不會傳送任何指標至 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。

維度	定義
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	此維度會篩選您在指定叢集內，對指定服務或協助程式中所有資源請求的資料。對於協助程式指標，ServiceName 維度值具有字首，daemon:後面接著協助程式名稱。例如，名為 my-daemon 的協助程式 my-daemon 的 ServiceName 維度值為 daemon:my-daemon。

維度	定義
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。

維度	定義
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 用量指標

您可以使用 CloudWatch 用量指標來提供您帳戶的資源用量可見度。使用這些指標，以 CloudWatch 圖表和儀表板視覺化目前的服務使用狀況。

AWS Fargate 用量指標對應至 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。如需有關 Fargate 服務配額的詳細資訊，請參閱 Amazon Web Services 一般參考 中的 [Amazon ECS endpoints and quotas](#)。

AWS Fargate 在 AWS/Usage 命名空間中發佈下列指標。

指標	Description
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	Description
Service	包含資源 AWS 的服務名稱。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 用量指標的唯一有效值為 Resource。
Resource	正在執行的資源類型。正在執行的資源類型。目前，AWS Fargate 用量指標的唯一有效值是 vCPU，其會傳回執行中執行個體的相關資訊。
Class	正在追蹤的資源類別。正在追蹤的資源類別。對於使用 vCPU 做為資源維度值的 AWS Fargate 用量指標，有效值為 Standard/OnDemand 和 Standard/Spot。

您可以使用 Service Quotas 主控台在圖形上視覺化您的用量，並設定警示，以便在 AWS Fargate 用量接近服務配額時提醒您。如需有關如何建立用量接近配額閾值時的 CloudWatch 警示的資訊，請參閱 Service Quotas User Guide 中的 [Service Quotas and Amazon CloudWatch alarms](#)

Amazon ECS 叢集保留量指標

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。此指標僅用於任務或服務託管在 EC2 執行個體上的叢集，不適用於任務託管在 AWS Fargate 上的叢集。

$$\text{Cluster CPU reservation} = \frac{(\text{Total CPU units reserved by tasks in cluster}) \times 100}{(\text{Total CPU units registered by container instances in cluster})}$$

$$\text{Cluster memory reservation} = \frac{(\text{Total MiB of memory reserved by tasks in cluster} \times 100)}{\text{Total MiB of memory registered by container instances in cluster}}$$

$$\text{Cluster CPU reservation} = \frac{\text{(Total MiB of memory reserved by tasks in cluster)}}{\text{(Total MiB of memory registered by container instances in cluster)}}$$

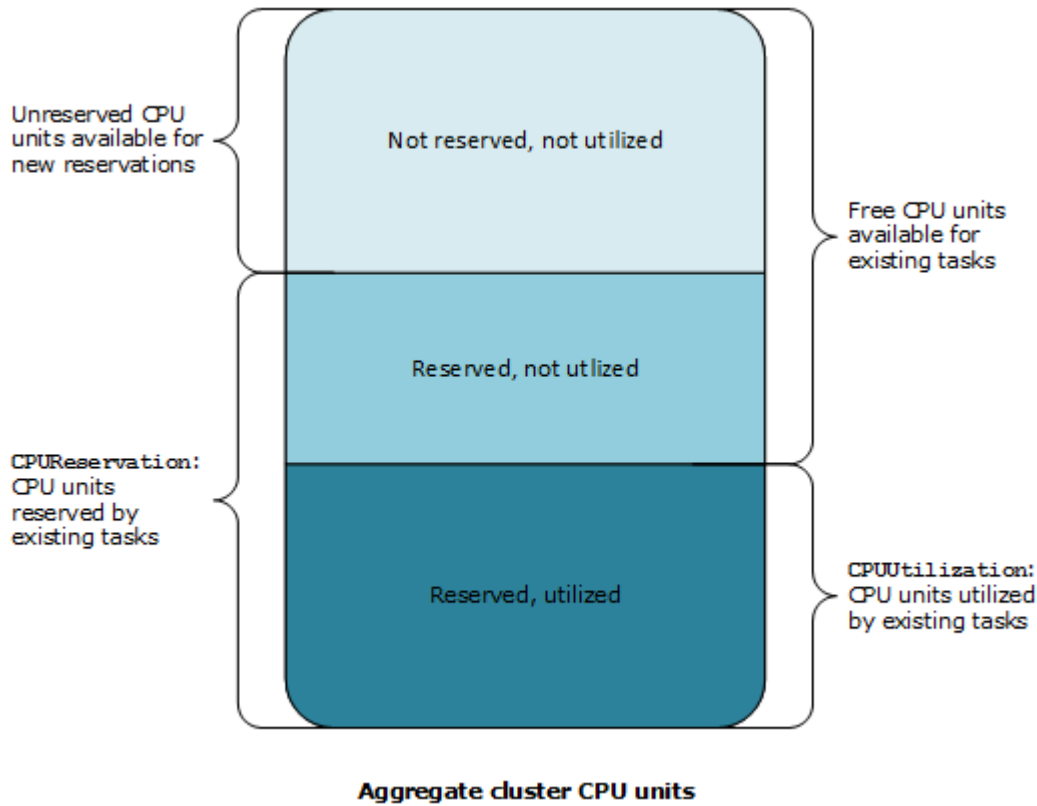
$$\text{Cluster GPU reservation} = \frac{\text{(Total GPUs reserved by tasks in cluster x 100)}}{\text{(Total GPUs registered by container instances in cluster)}}$$

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。為叢集上所有執行中之任務保留的 CPU、記憶體和 GPU 總數量皆會計算，這些數字會以叢集總登錄資源百分比的形式回報給 CloudWatch。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。此會回報到 CloudWatch，做為叢集的 55% CPU 保留和 60% 記憶體保留。

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。下方 (已保留、已使用) 和中間 (已保留、未使用) 區塊代表為叢集上現有執行中任務所保留的總 CPU 單位，或 CPUReservation CloudWatch 指標。下方區塊代表叢集上執行中任務實際使用的保留 CPU 單位或 CPUUtilization CloudWatch 指標。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

叢集使用率指標涵蓋 CPU、記憶體以及 EBS 檔案系統使用率 (當有 EBS 磁碟區連接至任務時)。這些指標僅適用於任務或服務託管在 Amazon EC2 執行個體上的叢集，不適用於任務託管在 AWS Fargate 上的叢集。

Amazon ECS 叢集層級 CPU 與記憶體使用率指標

CPU 與記憶體使用率的計算方式為，叢集中所有任務所使用的 CPU 與記憶體，相較於叢集中所有已註冊的作用中 Amazon EC2 執行個體其註冊的 CPU 與記憶體總量，所得出的百分比。僅處於 ACTIVE 或 DRAINING 狀態的 Amazon EC2 執行個體會影響叢集使用率指標。

$$\text{Cluster CPU utilization} = \frac{(\text{Total CPU units used by tasks in cluster}) \times 100}{(\text{Total CPU units registered by container instances in cluster})}$$

$$100) \quad (\text{Total MiB of memory used by tasks in cluster} \times$$


```

Cluster memory utilization =
-----
(Total MiB of memory registered by container instances in
cluster)

```

每個 Amazon EC2 執行個體上的 Amazon ECS 容器代理程式，每分鐘都會計算該 Amazon EC2 執行個體上每個執行中任務目前正在使用的 CPU 單位數與記憶體量 (MiB)，並將此資訊回報給 Amazon ECS。叢集上所有執行中之任務使用的 CPU 和記憶體總數量皆會計算，這些數字會以在叢集總登錄資源中所佔的百分比回報給 CloudWatch。

例如，叢集已註冊有兩個作用中的 Amazon EC2 執行個體：c4.4xlarge 執行個體與 c4.large 執行個體。c4.4xlarge 執行個體以 16,384 個 CPU 單位與 30,158 MiB 記憶體的配置向叢集完成註冊。c4.large 執行個體以 2,048 個 CPU 單位與 3,768 MiB 記憶體的配置完成註冊。此叢集的資源總量為 18,432 個 CPU 單位與 33,926 MiB 記憶體。

如果有十個任務在此叢集上執行且每個任務取用 1,024 個 CPU 單位與 2,048 MiB 記憶體，則此叢集上總共使用 10,240 個 CPU 單位與 20,480 MiB 記憶體。此會回報到 CloudWatch，做為叢集的 55% CPU 使用率和 60% 記憶體使用率。

Amazon ECS 叢集層級 Amazon EBS 檔案系統使用率

叢集層級 EBS 檔案系統使用率指標的計算方式為，叢集中執行中任務所使用的 EBS 檔案系統總量，除以叢集中所有任務配置的 EBS 檔案系統儲存總量。

```

(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

服務使用率指標涵蓋 CPU、記憶體以及 EBS 檔案系統使用率 (當有 EBS 磁碟區連接至任務時)。服務層級指標支援任務託管在 Amazon EC2 執行個體與 Fargate 上的服務。

服務層級 CPU 與記憶體使用率

CPU 與記憶體使用率的計算方式為，叢集上屬於某項服務的 Amazon ECS 任務使用之 CPU 與記憶體，相較於該服務的任務定義中所指定的 CPU 與記憶體，所得出的百分比。

在 CloudWatch 中檢視這些指標時，您可以選擇不同的統計量：

- 平均值：服務中所有任務的平均使用率。此值使用下面的公式計算得出。
- 最小值：服務中資源用量最低的任務使用率。此值表示資源密集型最低的任務所使用的 CPU 或記憶體，相較於任務定義中所指定的數值，所得出的百分比。
- 最大值：服務中資源用量最高的任務使用率。此值表示資源密集型最高的任務所使用的 CPU 或記憶體，相較於任務定義中所指定的數值，所得出的百分比。

下列公式說明平均值統計量的計算方式：

$$\text{Service CPU utilization} = \frac{(\text{Total CPU units used by tasks in service}) \times 100}{(\text{Total CPU units specified in the task definition}) \times (\text{Number of tasks in the service})}$$

$$\text{Service memory utilization} = \frac{(\text{Total MiB of memory used by tasks in service} \times 100)}{(\text{Total MiB of memory specified in the task definition}) \times (\text{Number of tasks in the service})}$$

 Note

上述公式僅適用於平均值統計量。最小值與最大值統計量分別代表資源使用率最低與最高的個別任務，而非所有任務的彙總計算值。

Amazon ECS 每 20 秒收集一次指標。Amazon ECS 容器代理程式每分鐘都會計算服務所擁有的每個執行中任務目前所使用的 CPU 單位數量與記憶體量 (MiB)。此資訊會回報給 Amazon ECS。叢集上執行中的服務擁有之所有任務使用的 CPU 和記憶體總量皆會計算，這些數字會以在服務的任務定義中，為服務指定的總資源中所佔的百分比，回報給 CloudWatch。最小值與最大值是在這 20 秒週期內蒐集的指標最小值與最大值。平均值為 3 個數值的彙總結果。

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)。如需有關硬性與軟性限制的詳細資訊，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%。

如果您的應用程式短時間記憶體使用率驟升，由於 Amazon ECS 每分鐘會收集多個資料點，然後將這些資料點匯集至傳送到 CloudWatch 的單一資料點，因此您會發現服務記憶體使用率並未增加。

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 檔案系統使用率的計算方式為，屬於該服務的任務所使用的 EBS 檔案系統總量，除以屬於該服務的所有任務配置的 EBS 檔案系統儲存總量。

服務 **RUNNING** 任務計數

您可以使用 CloudWatch 指標來檢視您服務中 **RUNNING** 狀態的任務數量。例如，您可以為此指標設定 CloudWatch 警示，在服務的執行中之任務數量低於指定值時提醒您。

Amazon ECS CloudWatch Container Insights 中的服務 **RUNNING** 任務計數

當您使用 Amazon ECS CloudWatch Container Insights 時，每個叢集和每個服務都有一個「正在執行的任務數」(`RunningTaskCount`) 指標。您可以透過以下方式為所有新建立的叢集使用 Container Insights：啟用 `containerInsights` 帳戶設定、在建立叢集時開啟個別叢集的設定，或使用

UpdateClusterSettings API 套用至現有叢集。CloudWatch Container Insights 收集的指標會以自訂指標的方式計費。如需了解有關 CloudWatch 定價的詳細資訊，請參閱 [CloudWatch 定價](#)。

如要查看此指標，請參閱《Amazon CloudWatch 使用者指南》中的 [Amazon ECS Container Insights 指標](#)。

Amazon ECS 服務使用率指標使用案例

下列清單提供了有關何時使用 Amazon ECS 指標的資訊。

- 資源使用率監控：使用平均值統計量來監控 CPU 與記憶體耗用模式、建立效能基準，以及偵測效能逐漸下降的趨勢。
- 成本最佳化：使用平均值統計量來監控資源用量、最適化調整容量、根據用量模式調整保留量，以及實作排程擴展。
- 效能基準測試：使用平均值統計量來比較不同服務修訂版本間的指標、建立效能 KPI 以及驗證最佳化改善效果。
- 資源底線偵測：使用平均值統計量來識別閒置期間的最低資源需求、設定適當的保留量，以及偵測異常下降情況。
- 異常偵測：使用最小值統計量找出資源使用率的異常下降情況，這些跡象可能標示潛在問題，例如初始化失敗或非預期的閒置期間。
- 擴展政策優化：使用最小值統計量，根據最低必要使用率設定最佳的縮減觸發閾值，避免過度縮減。
- 容量規劃：使用最大值統計量來設定適當的任務大小，並規劃具備足夠緩衝空間的基礎結構，以因應流量峰值。
- 效能瓶頸識別：使用最大值統計量來偵測資源飽和點、識別瓶頸，以及判斷何時增加任務大小。
- 擴展政策組態：使用最大值統計量，依據流量峰值模式設定最佳的橫向擴充觸發閾值，並設定因應突發流量的高載容量。
- SLA 合規監控：使用最大值統計量來追蹤峰值回應時間與錯誤率，確保服務效能符合既定的 SLA 要求。

相關指標資訊

如需有關 Container Insights 的詳細資訊，請參閱 CloudWatch User Guide 中的 [Container Insights use cases](#)。

使用 EventBridge 自動化因應 Amazon ECS 錯誤

使用 Amazon EventBridge，您可以自動化您的 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。AWS 服務的事件會以接近即時的方式傳送到 EventBridge。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 Logs 中的日誌群組
- 叫用 AWS Lambda 函數
- 調用 Amazon EC2 執行命令
- 將事件轉傳至 Amazon Kinesis Data Streams
- 啟用 AWS Step Functions 狀態機器
- 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》中的 [Amazon EventBridge 入門](#)。

您可以使用適用於 EventBridge 的 Amazon ECS 事件，接收有關您的 Amazon ECS 叢集目前狀態的近乎即時通知。如果任務使用的是 EC2，則可檢視容器執行個體的狀態，以及這些容器執行個體上所有執行中任務的目前狀態。如果任務使用的是 Fargate，則可檢視容器執行個體的狀態。

您可以使用 EventBridge，在 Amazon ECS 上建置自訂排程器，而自訂排程器負責協調叢集之間的任務，並以近乎即時的方式監控叢集狀態。您不需要排定和監控持續輪詢 Amazon ECS 服務以處理狀態變更的程式碼，而是改為使用任意 EventBridge 目標，以非同步的方式處理 Amazon ECS 狀態變更。目標可能包括 AWS Lambda Amazon Simple Queue Service、Amazon Simple Notification Service 或 Amazon Kinesis Data Streams。

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[因應 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](#)
- [因應 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變更，就會產生事件並將事件傳送到 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 會產生下列類型的事件並將其傳送至 EventBridge：

- 容器執行個體狀態變更
- 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、來源與詳細資訊。如果透過程式碼將事件 JSON 資料反序列化，請確保應用程式已準備好處理不明屬性，避免日後新增這些額外屬性時出現問題。

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 User Guide 中的 [AWS service event metadata](#)。
- 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。如果您使用 EventBridge 複寫 Amazon ECS 容器執行個體和任務狀態，您可以將 Amazon ECS APIs 回報的資源版本與 EventBridge 中回報的資源版本（在 `detail` 物件內）進行比較，以確認事件串流中的版本為最新版本。

服務動作事件只包含本體中的 `version` 欄位。

服務動作事件會透過 2 個不同欄位標明所屬服務：

- 對於由 `create-service` 產生的事件，服務位於 `serviceName` 欄位中。
- 對於由 `update-service` 產生的事件，服務位於 `service` 欄位中。

如果您針對服務事件使用自動化工具，則需要為這兩個欄位編寫程式碼。

如需有關如何建立服務動作事件規則的資訊，請參閱 [Amazon ECS 服務動作事件](#)。

如需有關如何整合 Amazon ECS 與 EventBridge 的其他資訊，請參閱[整合 Amazon EventBridge 與 Amazon ECS](#)。

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 管理主控台 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)。

您可以直接使用 `DeregisterContainerInstance` API 操作或使用 AWS 管理主控台 SDKs 取消註冊容器執行個體。

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` 區段類似於從 Amazon Elastic Container Service API 參考中的 [DescribeContainerInstances](#) API 操作所返回的 [ContainerInstance](#) 物件。如需有關 EventBridge 參數的詳細資訊，請參閱 Amazon EventBridge User Guide 中的 [AWS service event metadata](#)。

```
{
  "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 管理主控台、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。

若使用的是 EC2，Amazon ECS 容器代理程式會監控容器執行個體上任務的狀態。Amazon ECS 容器代理程式會回報所有狀態變更。狀態變更可能包括從 `PENDING` 到 `RUNNING` 或從 `RUNNING` 到 `STOPPED` 的變更。

您可以使用 `DeregisterContainerInstance` API 操作和 `force` 旗標，直接或使用或 AWS 管理主控台 SDKs，強制取消基礎容器執行個體的註冊。

取消登錄容器代理程式會變更容器執行個體的状态以及 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。請注意下列關於欄位的事項：

- 任務狀態變更事件中不提供事件的運作狀態。如果您需要任務的運作狀態，可以執行 [describe-tasks](#)。
- 當容器使用 Amazon ECR 託管的映像時，會傳回 `imageDigest` 欄位。
- `createdAt`、`connectivityAt`、`pullStartedAt`、`startedAt`、`pullStoppedAt` 與 `updatedAt` 欄位的值為 ISO 字串時間戳記。
- `detail-type` 值為「ECS 任務狀態變更」。
- 當為已停止的任務產生事件時，`stoppedReason` 與 `stopCode` 欄位會提供有關任務停止原因的其他資訊 (例如「使用者啟動」)。

如需有關 EventBridge 參數的詳細資訊，請參閱 Amazon EventBridge Events Reference 中的 [AWS service event metadata](#)。

如需有關如何設定 Amazon EventBridge 事件規則 (僅擷取因任務的一個必要容器已終止而導致任務停止執行的任務事件) 的資訊，請參閱[針對 Amazon ECS 任務停止事件傳送 Amazon Simple Notification Service 提醒](#)

```
{
  "version": "0",
  "id": "105f6bb1-4da6-c630-4965-35383018cbca",
  "detail-type": "ECS Task State Change",
  "source": "aws.ecs",
  "account": "123456789012",
  "time": "2025-05-06T11:02:34Z",
  "region": "us-east-1",
  "resources": [
    "arn:aws:ecs:us-east-1:123456789012:task/example-cluster/a1173316d40a45dea9"
  ],
  "detail": {
    "attachments": [
      {
        "id": "fe3a9a46-6a47-40ee-afd9-7952ae90a75a",
        "type": "eni",
        "status": "ATTACHED",
        "details": [
          {
            "name": "subnetId",
            "value": "subnet-0d0eab1bb38d5ca64"
          },
          {
            "name": "networkInterfaceId",
            "value": "eni-0103a2f01bad57d71"
          },
          {
            "name": "macAddress",
            "value": "0e:50:d1:c1:77:81"
          },
          {
            "name": "privateDnsName",
            "value": "ip-10-0-1-163.ec2.internal"
          },
          {
            "name": "privateIPv4Address",
```

```

        "value": "10.0.1.163"
      }
    ]
  },
  "attributes": [
    {
      "name": "ecs.cpu-architecture",
      "value": "x86_64"
    }
  ],
  "availabilityZone": "us-east-1b",
  "capacityProviderName": "FARGATE",
  "clusterArn": "arn:aws:ecs:us-east-1:123456789012:cluster/example-cluster",
  "connectivity": "CONNECTED",
  "connectivityAt": "2025-05-06T11:02:17.19Z",
  "containers": [
    {
      "containerArn": "arn:aws:ecs:us-east-1:123456789012:container/example-cluster/a1173316d40a45dea9/a0a99b87-baa8-4bf6-b9f1-a9a95917a635",
      "lastStatus": "RUNNING",
      "name": "web",
      "image": "nginx",
      "imageDigest":
"sha256:c15da6c91de8d2f436196f3a768483ad32c258ed4e1beb3d367a27ed67253e66",
      "runtimeId": "a1173316d40a45dea9-0265927825",
      "taskArn": "arn:aws:ecs:us-east-1:123456789012:task/example-cluster/a1173316d40a45dea9",
      "networkInterfaces": [
        {
          "attachmentId": "fe3a9a46-6a47-40ee-afd9-7952ae90a75a",
          "privateIpv4Address": "10.0.1.163"
        }
      ],
      "cpu": "99",
      "memory": "100"
    },
    {
      "containerArn": "arn:aws:ecs:us-east-1:123456789012:container/example-cluster/a1173316d40a45dea9/a2010e2d-ba7c-4135-8b79-e0290ff3cd8c",
      "lastStatus": "RUNNING",
      "name": "aws-guardduty-agent-nm401c",
      "imageDigest":
"sha256:bf9197abdf853607e5fa392b4f97ccdd6ca56dd179be3ce8849e552d96582ac8",

```

```

        "runtimeId": "a1173316d40a45dea9-2098416933",
        "taskArn": "arn:aws:ecs:us-east-1:123456789012:task/example-cluster/
a1173316d40a45dea9",
        "networkInterfaces": [
            {
                "attachmentId": "fe3a9a46-6a47-40ee-afd9-7952ae90a75a",
                "privateIpv4Address": "10.0.1.163"
            }
        ],
        "cpu": "null"
    },
    {
        "containerArn": "arn:aws:ecs:us-east-1:123456789012:container/example-
cluster/a1173316d40a45dea9/dccf0ca2-d929-471f-a5c3-98006fd4379e",
        "lastStatus": "RUNNING",
        "name": "aws-otel-collector",
        "image": "public.ecr.aws/aws-observability/aws-otel-collector:v0.32.0",
        "imageDigest":
"sha256:7a1b3560655071bcacd66902c20ebe9a69470d5691fe3bd36baace7c2f3c4640",
        "runtimeId": "a1173316d40a45dea9-4027662657",
        "taskArn": "arn:aws:ecs:us-east-1:123456789012:task/example-cluster/
a1173316d40a45dea9",
        "networkInterfaces": [
            {
                "attachmentId": "fe3a9a46-6a47-40ee-afd9-7952ae90a75a",
                "privateIpv4Address": "10.0.1.163"
            }
        ],
        "cpu": "0"
    }
],
"cpu": "256",
"createdAt": "2025-05-06T11:02:13.877Z",
"desiredStatus": "RUNNING",
"enableExecuteCommand": false,
"ephemeralStorage": {
    "sizeInGiB": 20
},
"group": "family:webserver",
"launchType": "FARGATE",
"lastStatus": "RUNNING",
"memory": "512",
"overrides": {
    "containerOverrides": [

```

```

        {
            "name": "web"
        },
        {
            "environment": [
                {
                    "name": "CLUSTER_NAME",
                    "value": "example-cluster"
                },
                {
                    "name": "REGION",
                    "value": "us-east-1"
                },
                {
                    "name": "HOST_PROC",
                    "value": "/host_proc"
                },
                {
                    "name": "AGENT_RUNTIME_ENVIRONMENT",
                    "value": "ecs-fargate"
                },
                {
                    "name": "STAGE",
                    "value": "prod"
                }
            ],
            "memory": 128,
            "name": "aws-guardduty-agent-nm401C"
        },
        {
            "name": "aws-otel-collector"
        }
    ]
},
"platformVersion": "1.4.0",
"pullStartedAt": "2025-05-06T11:02:24.162Z",
"pullStoppedAt": "2025-05-06T11:02:33.493Z",
"startedAt": "2025-05-06T11:02:34.325Z",
"taskArn": "arn:aws:ecs:us-east-1:123456789012:task/example-cluster/a1173316d40a45dea9",
"taskDefinitionArn": "arn:aws:ecs:us-east-1:123456789012:task-definition/webserver:5",
"updatedAt": "2025-05-06T11:02:34.325Z",
"version": 3

```



```
}  
}
```

Example

以下是 EC2 任務狀態變更事件的範例。

```
{  
  "version": "0",  
  "id": "a65cf262-f104-0dd5-ceda-4b09ba71a441",  
  "detail-type": "ECS Task State Change",  
  "source": "aws.ecs",  
  "account": "123456789012",  
  "time": "2025-05-12T13:12:06Z",  
  "region": "us-east-1",  
  "resources": [  
    "arn:aws:ecs:us-east-1:123456789012:task/example/c1ffa94f19a540ed8d9f7e1d2a5d"  
  ],  
  "detail": {  
    "attachments": [  
      {  
        "id": "52333e3b-b812-41a8-b057-9ed184bbe5e1",  
        "type": "eni",  
        "status": "ATTACHED",  
        "details": [  
          {  
            "name": "subnetId",  
            "value": "subnet-0d0eab1bb38d5ca64"  
          },  
          {  
            "name": "networkInterfaceId",  
            "value": "eni-0ea90f746500773a4"  
          },  
          {  
            "name": "macAddress",  
            "value": "0e:d5:9b:ce:49:fb"  
          },  
          {  
            "name": "privateDnsName",  
            "value": "ip-10-0-1-37.ec2.internal"  
          },  
          {  
            "name": "privateIPv4Address",  
            "value": "10.0.1.37"  
          }  
        ]  
      }  
    ]  
  }  
}
```

```

        }
    ]
}
],
"attributes": [
    {
        "name": "ecs.cpu-architecture",
        "value": "x86_64"
    }
],
"availabilityZone": "us-east-1b",
"capacityProviderName": "Infra-ECS-Cluster-example-fa84e0cc-
AsgCapacityProvider-0seQJU9pizmp",
"clusterArn": "arn:aws:ecs:us-east-1:123456789012:cluster/example",
"connectivity": "CONNECTED",
"connectivityAt": "2025-05-12T13:11:44.98Z",
"containerInstanceArn": "arn:aws:ecs:us-east-1:123456789012:container-instance/
example/d1d84798400f49f3b21cb61610c1e",
"containers": [
    {
        "containerArn": "arn:aws:ecs:us-east-1:123456789012:container/example/
c1ffa94f19a540ed8d9f7e1d2a5d3626/197d0994-5367-4a6d-9f9a-f075e4a6",
        "lastStatus": "RUNNING",
        "name": "aws-otel-collector",
        "image": "public.ecr.aws/aws-observability/aws-otel-collector:v0.32.0",
        "imageDigest":
"sha256:7a1b3560655071bcacd66902c20ebe9a69470d5691fe3bd36baace7c2f3c4640",
        "runtimeId":
"8e926f0ccd8fe2b459926f49584ba6d33a3d9f61398dbabe944ee6a13a8ff3a1",
        "taskArn": "arn:aws:ecs:us-east-1:123456789012:task/example/
c1ffa94f19a540ed8d9f7e1d2a5d",
        "networkInterfaces": [
            {
                "attachmentId": "52333e3b-b812-41a8-b057-9ed184bbe5e1",
                "privateIpv4Address": "10.0.1.37"
            }
        ],
        "cpu": "0"
    },
    {
        "containerArn": "arn:aws:ecs:us-east-1:123456789012:container/example/
c1ffa94f19a540ed8d9f7e1d2a5d3626/cab39ef0-9c50-459d-844b-b9d51d73d",
        "lastStatus": "RUNNING",
        "name": "web",

```

```

        "image": "nginx",
        "imageDigest":
"sha256:c15da6c91de8d2f436196f3a768483ad32c258ed4e1beb3d367a27ed67253e66",
        "runtimeId":
"9f1c73f0094f051541d9e5c2ab1e172d83c4eb5171bcc857c4504b02770ff3b8",
        "taskArn": "arn:aws:ecs:us-east-1:123456789012:task/example/
c1ffa94f19a540ed8d9f7e1d2a5d",
        "networkInterfaces": [
            {
                "attachmentId": "52333e3b-b812-41a8-b057-9ed184bbe5e1",
                "privateIpv4Address": "10.0.1.37"
            }
        ],
        "cpu": "99",
        "memory": "100"
    }
],
"cpu": "256",
"createdAt": "2025-05-12T13:11:44.98Z",
"desiredStatus": "RUNNING",
"enableExecuteCommand": false,
"group": "family:webserver",
"launchType": "EC2",
"lastStatus": "RUNNING",
"memory": "512",
"overrides": {
    "containerOverrides": [
        {
            "name": "aws-otel-collector"
        },
        {
            "name": "web"
        }
    ]
},
"pullStartedAt": "2025-05-12T13:11:59.491Z",
"pullStoppedAt": "2025-05-12T13:12:05.896Z",
"startedAt": "2025-05-12T13:12:06.053Z",
"taskArn": "arn:aws:ecs:us-east-1:123456789012:task/example/
c1ffa94f19a540ed8d9f7e1d2a5d",
"taskDefinitionArn": "arn:aws:ecs:us-east-1:123456789012:task-definition/
webserver",
"updatedAt": "2025-05-12T13:12:06.053Z",
"version": 4

```

```
}  
}
```

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。以下這個事件模式會用來建立 Amazon ECS 服務動作事件的 EventBridge 規則。如需詳細資訊，請參閱 Amazon EventBridge User Guide 中的 [Getting started with EventBridge](#)。

```
{  
  "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。

TASKS_STOPPED

服務已停止執行中的任務。

SERVICE_DEPLOYMENT_IN_PROGRESS

一項服務部署正在進行中。服務部署可為復原作業，也可為新的服務修訂版。

SERVICE_DEPLOYMENT_COMPLETED

服務部署處於穩定狀態且已完成。服務部署可為復原作業，也可為部署更新後的服務修訂版。

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。

VPC_LATTICE_TARGET_UNHEALTHY

使用 VPC Lattice 的服務偵測到 VPC Lattice 某一個目標的運作狀態不良。

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 限流限制，服務排程器已經受到限流控管。

SERVICE_DISCOVERY_OPERATION_THROTTLED

由於 AWS Cloud Map API 限流限制，服務排程器已調節。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。

SERVICE_HEALTH_UNKNOWN

服務無法描述任務的運作狀態資料。

SERVICE_DEPLOYMENT_FAILED

服務部署未達到穩定狀態。當 CloudWatch 被觸發或斷路器偵測到服務部署失敗時，就會發生這種情況。

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 User Guide 中的 [Events in EventBridge](#)。

```
{
  "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。如需詳細資訊，請參閱 Amazon EventBridge User Guide 中的 [Events in EventBridge](#)。

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 範例中，嘗試在容器執行個體 2dd1b186f39845a584488d2ef155c131 上啟動任務，但由於 CPU 不足，服務排程器無法置放任務。

```
{  
  "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。以下這個事件模式會用來建立 Amazon ECS 服務部署狀態變更事件的 EventBridge 規則。如需有關建立 EventBridge 規則的詳細資訊，請參閱 Amazon EventBridge User Guide 中的 [Getting started with Amazon EventBridge](#)。

```
{
  "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。如需詳細資訊，請參閱 [Amazon ECS 服務動作事件](#)

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 User Guide 中的 [AWS service event metadata](#)。

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-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 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[透過取代任務來部署 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[透過取代任務來部署 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."
  }
}
```

因應 Amazon ECS 事件

Amazon ECS 會傳送事件至少一次。這表示您可能會收到指定事件的多個副本。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。如果您使用 EventBridge 複寫 Amazon ECS 容器執行個體和任務狀態，您可以將 Amazon ECS APIs 回報的資源版本與 EventBridge 中 version 回報的資源版本進行比較，以驗證事件串流中的版本是否為最新版本。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 函數中的事件

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：

- **ECSCtrInstanceState** : 存放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。
- **ECSTaskState** : 存放任務的最新狀態。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
        )

```

下列範例顯示使用 Python 3.9 撰寫的 Lambda 函式，其功能為擷取任務狀態變更事件，並將這些事件儲存至下列 Amazon DynamoDB 資料表：

```

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
    )
```

使用具有增強可觀測性的 Container Insights 監控 Amazon ECS

Container Insights 會收集、彙總及總結來自容器化應用程式與微服務的指標及日志。具有增強可觀測性的 Container Insights 可提供所有 Container Insights 指標，以及其他任務和容器指標。

Container Insights 會探索叢集中所有執行中的容器，並在效能堆疊的每個層級上收集效能資料。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。CloudWatch 可透過這些資料，建立在叢集、服務和任務層級更高層級彙總的指標並做為 CloudWatch 指標。指標包含 CPU、記憶體、磁碟和網路這類資源的使用率。在 CloudWatch 自動儀表板中可取得這些指標。

指標僅反映指定時間範圍內執行中任務的資源。舉例來說，如果您有叢集包含一個服務，但該服務沒有處於 RUNNING 狀態的任務，則不會傳送任何指標至 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。

2024 年 12 月 2 日，AWS 發佈了 Container Insights 並增強了 Amazon ECS 的可觀測性。此版本針對使用 Amazon EC2 與 Fargate 的 Amazon ECS 叢集，提供增強可觀測性功能。在 Amazon ECS 上設定了具有增強可觀測性的 Container Insights 後，Container Insights 會自動收集您環境中從叢集層級到容器層級的詳細基礎結構遙測資料，並將這些關鍵效能資料顯示在精心設計的儀表板中，無需您進行繁重的可觀測性設定。如需如何設定具有增強可觀測性的 Container Insights 的詳細資訊，請參閱[具有增強可觀測性的 Container Insights](#)。

建議使用具有增強可觀測性的 Container Insights 而非普通的 Container Insight，因為前者能提供容器環境的詳細可見性，縮短問題解決的平均時間。如需詳細資訊，請參閱 Amazon CloudWatch User Guide 中的 [Amazon ECS Container Insights with enhanced observability metrics](#)。

您可以檢視的事件包括 Amazon ECS 傳送至 Amazon EventBridge 的事件。如需詳細資訊，請參閱[Amazon ECS 事件](#)。

⚠ Important

CloudWatch Container Insights 收集的指標會以自訂指標的方式計費。如需了解有關 CloudWatch 定價的詳細資訊，請參閱 [CloudWatch 定價](#)。

Amazon ECS 受管執行個體的其他指標

下表列出了使用 Container Insights 時 Amazon ECS 受管執行個體可用的其他指標。

指標	Description	維度	單位
Instance0SFilesystemUtilization	已使用的總磁碟空間百分比 (作業系統磁碟區)。	ClusterName – 啟用 ContainerInsights 時 ClusterName 、 CapacityProviderName – 啟用 ContainerInsights 時 ClusterName 、 CapacityProviderName 、 ContainerInstanceId 、 EC2InstanceId – 啟用 EnhancedContainerInsights 時	百分比
InstanceDataFilesystemUtilization	已使用的總磁碟空間百分比 (資料磁碟區)。	ClusterName – 啟用 ContainerInsights 時 ClusterName 、 CapacityProviderName – 啟用 ContainerInsights 時 ClusterName 、 CapacityProviderName 、 ContainerInstanceId 、 EC2Instant	百分比

指標	Description	維度	單位
		ceId – 啟用 Enhanced Container Insights 時	

監控 Amazon ECS 受管執行個體

監控是維護 Amazon ECS 受管執行個體工作負載可靠性、可用性和效能的重要部分。AWS 提供數種工具和服務，協助您監控容器化應用程式和基礎設施。

Container Insights 監控

CloudWatch Container Insights 能夠為容器化應用程式與微服務提供全面的監控。Container Insights 會自動收集、彙總及總結來自執行於 Amazon ECS 受管理執行個體上的容器化應用程式與微服務的指標及日志。

Container Insights 會於叢集、服務及任務層級收集指標，提供以下面向的可見性：

- CPU 與記憶體使用率
- 網路效能指標
- 儲存使用率
- 任務與服務效能

這些指標顯示在 CloudWatch 儀表板中，可用於建立警示，實現效能問題自動回應。Container Insights 還提供增強的監控功能，可協助您快速識別問題並進行疑難排解。

Note

Container Insights 需額外付費才可使用。如需有關定價的詳細資訊，請參閱 [Amazon CloudWatch 定價](#)。

執行個體監控

若要監控支援 Amazon ECS 受管執行個體工作負載的基礎基礎設施，您可以使用 CloudWatch 提供的 Amazon EC2 指標。

Amazon ECS 受管執行個體隨附兩個 Amazon EBS 磁碟區：

- 用於作業系統檔案系統的根磁碟區
- 應用程式使用的資料磁碟區

啟用 Container Insights 時，Amazon ECS 會自動發佈執行個體層級作業系統和資料檔案系統使用率指標。

Amazon ECS 受管執行個體的可用指標包括：

- Amazon EC2 指標：CPU 使用率、網路效能、磁碟操作和狀態檢查。如需詳細資訊，請參閱[使用 CloudWatch 監控執行個體](#)
- Amazon ECS 指標（啟用 Container Insights 時）：作業系統和資料磁碟區檔案系統使用率。如需詳細資訊，請參閱 [Amazon ECS Container Insights 指標](#)
- Amazon EBS 指標：IOPS、輸送量、讀取和寫入延遲。如需詳細資訊，請參閱 [Amazon EBS CloudWatch 指標](#)

Note

CloudWatch 代理程式無法做為協助程式執行，因為 Amazon ECS 受管執行個體上不支援協助程式。這表示無法使用其他需要以協助程式身分執行之 CloudWatch 代理程式的系統層級指標。

這些指標會自動提供，無需手動安裝代理程式。

Amazon ECS 受管執行個體的詳細監控

CloudWatch 提供兩種類別的監控：基本監控和詳細監控。依預設，受管執行個體會設定為基本監控。您可以選擇啟用詳細監控，以協助您更快地識別和處理操作問題。在建立或更新 Amazon ECS 受管執行個體容量提供者時，您可以啟用或停用詳細監控。

在受管執行個體上啟用詳細監控，不會影響所連接 Amazon EBS 磁碟區的監控。

下表重點說明了受管執行個體的基本監控與詳細監控之間的差異。

監控類型	描述	費用
基本監控	狀態檢查指標在 1 分鐘內可用。所有其他指標均在 5 分鐘內可用。	免費。

監控類型	描述	費用
詳細監控	所有指標 (包括狀態檢查指標) 在 1 分鐘內可用。若要取得此層級的資料，必須專門為受管執行個體啟用該類型。對於已啟用詳細監控的受管執行個體，您亦可取得類似受管執行個體的跨群組彙總資料。	我們會按 Amazon ECS 受管執行個體傳送至 CloudWatch 的指標進行計費。您不需為資料儲存付費。如需詳細資訊，請參閱 Amazon CloudWatch 定價 頁面上的付費方案和範例 1 - EC2 詳細監控。

所需的許可

若要啟用受管執行個體的詳細監控，使用者必須具有使用 MonitorInstances API 動作的許可。若要關閉受管執行個體的詳細監控，使用者必須具有使用 UnmonitorInstances API 動作的許可。

使用容器運作狀態檢查來判斷 Amazon ECS 任務運作狀態

建立任務定義時，您可以設定容器的運作狀態檢查。運作狀態檢查是在容器本機執行的命令，用於驗證應用程式運作狀態與可用性。

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。

在任務定義中定義運作狀態檢查時，容器會在容器內執行運作狀態檢查程序，然後評估結束程式碼來判斷應用程式的運作狀態。

運作狀態檢查包含下列參數：

- 命令 – 容器為了判斷自身是否良好所執行的命令。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。當您需要管道、重新導向、命令鏈接或環境變數擴展等 Shell 功能時，請使用 CMD-SHELL。例如，CMD-SHELL 可讓您使用 `curl -f http://localhost/ || exit 1` 之類的命令，其中的 `||` 運算子會由 Shell 解釋。若為無需 Shell 解釋的簡單指令，請使用 CMD。

- 時間間隔 – 每次運作狀態檢查之間的時間間隔 (以秒為單位)。
- 逾時 – 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)。
- 重試次數 – 在將容器運作狀態判定為不良之前，失敗的運作狀態檢查重試次數。
- 啟動期間 – 選用的寬限期，讓容器有時間完成啟動程序，此期間內失敗的運作狀態檢查不會計入最大重試次數。

如果運作狀態檢查在 `startPeriod` 內成功，則代表容器運作狀態良好，之後的任何故障都會計入，累積至重試次數上限。

如需有關如何在任務定義中指定運作狀態檢查的資訊，請參閱[運作狀態檢查](#)。

以下內容說明容器的可能運作狀態值：

- 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。

運作狀態檢查命令在容器上執行。因此，必須將命令包含在容器映像中。

運作狀態檢查會透過容器的回送介面，以 `localhost` 或 `127.0.0.1` 與應用程式建立連線。結束代碼 `0` 表示成功，而非零結束代碼則表示失敗。

使用容器運作狀態檢查時，建議考量下列事項：

- 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 或更新版本或 Windows 平台版本 1.1.0 或更新版本，則 Fargate 任務支援容器運作狀態檢查

Amazon ECS 如何判斷任務運作狀態

只有在任務定義中具必要性且設有運作狀態檢查命令的容器，才會被納入任務運作狀態的判斷範圍。

將依順序評估下列規則：

1. 如果一個必要容器的狀態為 UNHEALTHY，則任務狀態為 UNHEALTHY。
2. 如果一個必要容器的狀態為 UNKNOWN，則任務狀態為 UNKNOWN。

3. 如果所有必要容器的狀態為 HEALTHY，則任務狀態為 HEALTHY。

請參考下列包含 2 個必要容器的任務運作狀態範例。

容器 1 運作狀態	容器 2 運作狀態	任務運作狀態
UNHEALTHY	UNKNOWN	UNHEALTHY
UNHEALTHY	HEALTHY	UNHEALTHY
HEALTHY	UNKNOWN	UNKNOWN
HEALTHY	HEALTHY	HEALTHY

請參考下列包含 3 個必要容器的任務運作狀態範例。

容器 1 運作狀態	容器 2 運作狀態	容器 3 運作狀態	任務運作狀態
UNHEALTHY	UNKNOWN	UNKNOWN	UNHEALTHY
UNHEALTHY	UNKNOWN	HEALTHY	UNHEALTHY
UNHEALTHY	HEALTHY	HEALTHY	UNHEALTHY
HEALTHY	UNKNOWN	HEALTHY	UNKNOWN
HEALTHY	UNKNOWN	UNKNOWN	UNKNOWN
HEALTHY	HEALTHY	HEALTHY	HEALTHY

代理程式中斷連線對運作狀態檢查的影響

如果 Amazon ECS 容器代理程式與 Amazon ECS 服務中斷連線，這種情況不會導致容器轉換為 UNHEALTHY 狀態。這是設計使然，旨在確保容器於代理程式重新啟動或暫時無法使用期間仍能持續執行。運作狀態檢查的結果，反映的是 Amazon ECS 代理程式「最後回報」的狀態。因此，若容器在連線中斷前被判定為 HEALTHY，該狀態會維持不變，直至代理程式重新連線並完成下一次運作狀態檢查為止。沒有對容器運作狀態檢查的狀態做出任何假設。

檢視 Amazon ECS 容器運作狀態

您可以在主控台中檢視容器運作狀態，也可以透過 API 從 `DescribeTasks` 回應中檢視容器運作狀態。如需詳細資訊，請參閱 Amazon Elastic Container Service API Reference 中的 [DescribeTasks](#)。

如果您對容器使用記錄功能 (例如 Amazon CloudWatch Logs)，則可以設定運作狀態檢查命令，將容器運作狀態輸出轉送至日誌。請務必使用 `2&1` 同時擷取 `stdout` 與 `stderr` 資訊。下列範例使用的是 `CMD-SHELL`，因為需要管道 (`>>`) 與邏輯 OR (`||`) 運算子等 Shell 功能：

```
"command": [
  "CMD-SHELL",
  "curl -f http://localhost/ >> /proc/1/fd/1 2>&1 || exit 1"
],
```

監控 Amazon ECS 容器執行個體運作狀態

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，或是 AWS CLI 版本 2.3.6 或更新版本。如需如何更新的資訊 AWS CLI，請參閱《使用者指南第 2 版》中的 [安裝或更新最新版本的 AWS CLI](#)。AWS Command Line Interface

若要檢視容器執行個體的運作狀態，請使用 `CONTAINER_INSTANCE_HEALTH` 選項執行 `describe-container-instances`。

`overallStatus` 取決於 `details` 陣列中的個別運作狀態檢查狀態。最嚴重狀態的優先順序如下：`IMPAIRED`、`INITIALIZING`、`INSUFFICIENT_DATA` 和 `OK`。

以下是 `overallStatus` 的有效值：

- `OK` – 正在通過所有運作狀態檢查。
- `IMPAIRED` – 一或多個運作狀態檢查失敗。
- `INSUFFICIENT_DATA` – 一個或多個運作狀態檢查無法使用。
- `INITIALIZING` – 正在初始化一或多個運作狀態檢查。

以下範例示範如何執行 `describe-container-instances`。

```
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"
  }]
}
```

容器執行個體運作狀態問題

當 `overallStatus` 的狀態不是 OK 時，請嘗試下列動作：

- 等待，然後執行 `describe-container-instances`
- 透過 EC2 主控台或 CLI 檢視容器執行個體運作狀態。
- 檢閱 CloudWatch 指標。如需詳細資訊，請參閱[使用 CloudWatch 監控 Amazon ECS](#)
- 檢查 AWS Health 儀板表 以查看服務是否有任何問題。

使用應用程式追蹤資料識別 Amazon ECS 最佳化機會

Amazon ECS 與 AWS Distro for OpenTelemetry 整合，以從您的應用程式收集追蹤資料。Amazon ECS 使用 AWS Distro for OpenTelemetry 附屬容器來收集和路由追蹤資料 AWS X-Ray。如需詳細資訊，請參閱在 [AWS Amazon ECS 中設定 Distro for OpenTelemetry Collector](#)。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。

若要讓 AWS Distro for OpenTelemetry Collector 將追蹤資料傳送到其中 AWS X-Ray，您的應用程式必須設定為建立追蹤資料。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 Distro for OpenTelemetry 與 整合所需的 IAM 許可 AWS X-Ray

Amazon ECS 與 AWS Distro for OpenTelemetry 整合需要您建立任務角色，並在任務定義中指定角色。建議您設定 AWS Distro for OpenTelemetry 附屬項目，將容器日誌路由至 CloudWatch Logs。

Important

如果您也使用 AWS Distro for OpenTelemetry 整合收集應用程式指標，請確定您的任務 IAM 角色也包含該整合所需的許可。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](#)。

建立角色後，請建立包含下列許可的政策，然後將該政策連接至該角色。

- logs:PutLogEvents
- logs:CreateLogGroup
- logs:CreateLogStream
- logs:DescribeLogStreams
- logs:DescribeLogGroups
- logs:PutRetentionPolicy
- xray:PutTraceSegments
- xray:PutTelemetryRecords
- xray:GetSamplingRules
- xray:GetSamplingTargets
- xray:GetSamplingStatisticSummaries
- ssm:GetParameters

指定 AWS Distro for OpenTelemetry 附屬項目以在任務定義中 AWS X-Ray 整合

Amazon ECS 主控台使用使用追蹤收集選項，簡化建立 AWS Distro for OpenTelemetry 附屬容器的程序。如需詳細資訊，請參閱[使用主控台建立 Amazon ECS 任務定義](#)。

如果您不是使用 Amazon ECS 主控台，您可以將 AWS Distro for OpenTelemetry 附屬容器新增至任務定義。下列任務定義程式碼片段顯示容器定義，用於新增 AWS Distro for OpenTelemetry 附屬項目以進行 AWS X-Ray 整合。

```
{
  "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，並將指標匯出到 Amazon CloudWatch 或 Amazon Managed Service for Prometheus。

您可以使用收集的中繼資料，將應用程式效能資料與底層基礎結構資料建立關聯，減少解決問題耗用的平均時間。

Amazon ECS 使用 AWS Distro for OpenTelemetry 附屬容器來收集應用程式指標並將其路由至目的地。Amazon ECS 主控台體驗簡化了建立任務定義時新增此整合的流程。

主題

- [將應用程式指標匯出至 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至 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 作為自訂指標。方法是將 AWS Distro for OpenTelemetry 附屬容器新增至您的任務定義。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流程。如需詳細資訊，請參閱[使用主控台建立 Amazon ECS 任務定義](#)。

將應用程式指標匯出至有日誌群組名稱 `/aws/ecs/application/metrics` 的 CloudWatch Logs，並且您可以在 `ECS/AWSOTel/Application` 命名空間中查看指標。您的應用程式必須使用 OpenTelemetry SDK 進行檢測。如需詳細資訊，請參閱 [AWS Distro for OpenTelemetry 文件](#) 中的 AWS Distro for OpenTelemetry 簡介。

考量事項

使用 Amazon ECS on Fargate 與 AWS Distro for OpenTelemetry 整合，將應用程式指標傳送至 Amazon CloudWatch 時，應考慮下列事項。

- 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[使用具有增強可觀測性的 Container Insights 監控 Amazon ECS](#)。
- 託管在 Fargate 上的 AWS Amazon ECS 工作負載和託管在 Amazon EC2 執行個體上的 Amazon ECS 工作負載支援 Distro for OpenTelemetry 整合。目前不支援外部執行個體。
- CloudWatch 支援每個指標最多 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。如果設定的維度超過 30 個，則 CloudWatch 將無法顯示它們。發生這種情況時，應用程式指標會顯示在 ECS/AWSOTel/Application CloudWatch 指標命名空間，但沒有任何維度。您可以檢測應用程式以增加其他維度。如需詳細資訊，請參閱[AWS Distro for OpenTelemetry 文件中的使用 CloudWatch 指標搭配 AWS Distro for OpenTelemetry](#)。

AWS Distro for OpenTelemetry 與 Amazon CloudWatch 整合所需的 IAM 許可

Amazon ECS 與 AWS Distro for OpenTelemetry 整合需要您建立任務 IAM 角色，並在任務定義中指定角色。我們建議您也將 AWS Distro for OpenTelemetry 附屬項目設定為將容器日誌路由至 CloudWatch Logs，這也需要在任務定義中建立和指定任務執行 IAM 角色。Amazon ECS 主控台會代表您負責處理任務執行 IAM 角色，但任務 IAM 角色必須手動建立並新增至任務定義。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

如果您也使用 AWS Distro for OpenTelemetry 整合收集應用程式追蹤資料，請確定您的任務 IAM 角色也包含該整合所需的許可。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 Elastic Container Service 的服務角色 (IAM 主控台)

1. 登入 AWS 管理主控台 並開啟位於 <https://console.aws.amazon.com/iam/> 的 IAM 主控台。
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. 在服務或使用案例欄位中選擇 Elastic Container Service，然後選擇 Elastic Container Service 任務使用案例。
5. 選擇下一步。
6. 在新增許可區段中，搜尋並選取 `AWSDistroOpenTelemetryPolicyForXray` 政策。
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. 開啟設定許可界限區段，然後選擇使用許可界限來控制角色許可上限。

IAM 包含您帳戶中 AWS 受管和客戶受管政策的清單。
 - 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字尾，協助識別該角色的用途。

Important

當您命名角色時，請注意下列事項：

- 角色名稱在您的 中必須是唯一的 AWS 帳戶，而且無法依大小寫設為唯一。

例如，不要同時建立名為 **PRODRole** 和 **prodrole** 的角色。當角色名稱用於政策或 ARN 的一部分時，角色名稱會區分大小寫，但是當角色名稱在主控台中顯示給客戶時，例如在登入過程中，角色名稱不會區分大小寫。

- 因為其他實體可能會參考角色，所以在建立角色之後，就無法編輯其名稱。

10. (選用) 在說明中，輸入角色的說明。
11. (選用) 若要編輯使用案例和角色許可，請在步驟 1：選取受信任的實體或者步驟 2：新增許可區段中選擇編輯。
12. (選用) 若要協助識別、組織或搜尋角色，請將標籤新增為索引鍵值對。如需在 IAM 中使用標籤的詳細資訊，請參閱《IAM 使用者指南》中的 [AWS Identity and Access Management 資源的標籤](#)。
13. 檢閱角色，然後選擇 Create role (建立角色)。

在任務定義中指定 AWS Distro for OpenTelemetry 附屬項目

Amazon ECS 主控台使用使用指標收集選項，簡化了建立 AWS Distro for OpenTelemetry 附屬容器的體驗。如需詳細資訊，請參閱 [使用主控台建立 Amazon ECS 任務定義](#)。

如果您不是使用 Amazon ECS 主控台，則可以手動將 AWS Distro for OpenTelemetry 附屬容器新增至任務定義。下列任務定義範例顯示為 Amazon CloudWatch 整合新增 AWS Distro for OpenTelemetry 附屬項目的容器定義。

```
{
  "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。方法是將 AWS Distro for OpenTelemetry 附屬容器新增至您的任務定義。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流程。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。您必須使用 Prometheus 程式庫或 OpenTelemetry SDK 檢測應用程式。如需有關使用 OpenTelemetry SDK 檢測應用程式的詳細資訊，請參閱[AWS Distro for OpenTelemetry 文件](#)中的 AWS Distro for OpenTelemetry 簡介。

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

使用 Amazon ECS on Fargate 與 AWS Distro for OpenTelemetry 整合，將應用程式指標傳送至 Amazon Managed Service for Prometheus 時，應考慮下列事項。

- 託管在 Fargate 上的 AWS Amazon ECS 工作負載和託管在 Amazon EC2 執行個體上的 Amazon ECS 工作負載支援 Distro for OpenTelemetry 整合。目前不支援外部執行個體。
- 根據預設，AWS Distro for OpenTelemetry 會在匯出至 Amazon Managed Service for Prometheus 時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。如需詳細資訊，請參閱 AWS Distro for OpenTelemetry 文件中的[適用於 Amazon Managed Service for Prometheus 的 Prometheus 遠端寫入匯出器入門](#)。

AWS Distro for OpenTelemetry 與 Amazon Managed Service for Prometheus 整合所需的 IAM 許可

Amazon ECS 與使用 AWS Distro for OpenTelemetry 附屬的 Amazon Managed Service for Prometheus 整合需要您建立任務 IAM 角色，並在任務定義中指定角色。註冊任務定義之前，必須手動建立此任務的 IAM 角色。如需建立任務角色的詳細資訊，請參閱 [Amazon ECS 任務 IAM 角色](#)。

建議您也將 AWS Distro for OpenTelemetry 附屬項目設定為將容器日誌路由至 CloudWatch Logs，這也需要在任務定義中建立和指定任務執行 IAM 角色。Amazon ECS 主控台會代表您負責處理任務執行 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

如果您也使用 AWS Distro for OpenTelemetry 整合收集應用程式追蹤資料，請確定您的任務 IAM 角色也包含該整合所需的許可。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](#)。

AWS Distro for OpenTelemetry 與 Amazon Managed Service for Prometheus 整合需要下列許可：

- logs:PutLogEvents
- logs:CreateLogGroup
- logs:CreateLogStream
- logs:DescribeLogStreams
- logs:DescribeLogGroups
- cloudwatch:PutMetricData

在任務定義中指定 AWS Distro for OpenTelemetry 附屬項目

Amazon ECS 主控台使用使用指標收集選項，簡化了建立 AWS Distro for OpenTelemetry 附屬容器的體驗。如需詳細資訊，請參閱[使用主控台建立 Amazon ECS 任務定義](#)。

如果您不是使用 Amazon ECS 主控台，則可以手動將 AWS Distro for OpenTelemetry 附屬容器新增至任務定義。下列任務定義範例顯示為 Amazon Managed Service for Prometheus 整合新增 AWS Distro for OpenTelemetry 附屬項目的容器定義。

```
{
```



```

"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 Elastic Container Service 已與 [AWS CloudTrail](#) 整合，後者可記錄使用者、角色或 AWS 服務所執行的動作。CloudTrail 會將對 Amazon ECS 的所有 API 呼叫當做事件來擷取。擷取的呼叫包括來自 Amazon ECS 主控台的呼叫，以及對 Amazon ECS API 操作發出的程式碼呼叫。您可以利用 CloudTrail 所收集的資訊來判斷向 Amazon ECS 發出的請求，以及發出請求的 IP 位址、時間和其他詳細資訊。

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：

- 該請求是否使用根使用者還是使用者憑證提出。
- 請求是否代表 IAM Identity Center 使用者提出。
- 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。
- 該請求是否由另一項 AWS 服務服務提出。

當您建立帳戶 AWS 帳戶 時 CloudTrail 會在 中處於作用中狀態，而且您會自動存取 CloudTrail 事件歷史記錄。CloudTrail 事件歷史記錄為 AWS 區域中過去 90 天記錄的管理事件，提供可檢視、可搜尋、可下載且不可變的記錄。如需詳細資訊，請參閱「AWS CloudTrail 使用者指南」中的 [使用 CloudTrail 事件歷史記錄](#)。檢視事件歷史記錄不會產生 CloudTrail 費用。

如需 AWS 帳戶 過去 90 天內持續記錄的事件，請建立線索或 [CloudTrail Lake](#) 事件資料存放區。

CloudTrail 追蹤

線索能讓 CloudTrail 將日誌檔案交付至 Amazon S3 儲存貯體。使用 建立的所有線索 AWS 管理主控台 都是多區域。您可以使用 AWS CLI 建立單一或多區域追蹤。建議您建立多區域追蹤，因為您擷取 AWS 區域 帳戶中所有的活動。如果您建立單一區域追蹤，您只能檢視追蹤 AWS 區域中記錄的事件。如需追蹤的詳細資訊，請參閱《AWS CloudTrail 使用者指南》中的 [為您的 AWS 帳戶建立追蹤](#) 和 [為組織建立追蹤](#)。

您可以透過建立追蹤，免費將持續管理事件的一個複本從 CloudTrail 傳遞至您的 Amazon S3 儲存貯體，但這樣做會產生 Amazon S3 儲存費用。如需 CloudTrail 定價的詳細資訊，請參閱 [AWS CloudTrail 定價](#)。如需 Amazon S3 定價的相關資訊，請參閱 [Amazon S3 定價](#)。

CloudTrail Lake 事件資料存放區

CloudTrail Lake 讓您能夠對事件執行 SQL 型查詢。CloudTrail Lake 會將分列式 JSON 格式的現有事件轉換為 [Apache ORC](#) 格式。ORC 是一種單欄式儲存格式，針對快速擷取資料進行了最佳化。系統會將事件彙總到事件資料存放區中，事件資料存放區是事件的不可變集合，其依據為您透過套用 [進階事件選取器](#) 選取的條件。套用於事件資料存放區的選取器控制哪些事件持續存在並可供您查詢。如需 CloudTrail Lake 的詳細資訊，請參閱 AWS CloudTrail 《使用者指南》中的 [使用 AWS CloudTrail Lake](#)。

CloudTrail Lake 事件資料存放區和查詢會產生費用。建立事件資料存放區時，您可以選擇要用於事件資料存放區的 [定價選項](#)。此定價選項將決定擷取和儲存事件的成本，以及事件資料存放區的預設和最長保留期。如需 CloudTrail 定價的詳細資訊，請參閱 [AWS CloudTrail 定價](#)。

CloudTrail 中的 Amazon ECS 管理事件

[管理事件](#) 提供有關在資源上執行的管理操作的資訊 AWS 帳戶。這些也稱為控制平面操作。根據預設，CloudTrail 記錄管理事件。

Amazon Elastic Container Service 會將所有 Amazon ECS 控制平面操作記錄為管理事件。例如，對 CreateService、RunTask 和 DeleteCluster 區段的呼叫，會在 CloudTrail 日誌檔案中產生項目。如需 Amazon ECS 記錄至 CloudTrail 中的 Amazon Elastic Container Service 控制平面操作清單，請參閱 [Amazon Elastic Container Service API Reference](#)。

CloudTrail 中的 Amazon ECS 資料事件

[資料事件](#) 提供有關在資源上執行或在資源中執行的資源操作的資訊。根據預設，CloudTrail 不會記錄資料事件。CloudTrail 事件歷史記錄不會記錄資料事件。

資料事件需支付額外的費用。如需 CloudTrail 定價的詳細資訊，請參閱 [AWS CloudTrail 定價](#)。

您可以使用 CloudTrail 主控台 AWS CLI 或 CloudTrail API 操作來記錄 Amazon ECS 資源類型的資料事件。如需如何記錄資料事件的詳細資訊，請參閱 AWS CloudTrail 使用者指南中的 [使用 AWS 管理主控台 記錄資料事件](#) 和 [使用 AWS Command Line Interface 記錄資料事件](#)。

下表列出您可以記錄資料事件的 Amazon ECS 資源類型。資料事件類型 (主控台) 資料行會顯示從 CloudTrail 主控台上的資料事件類型清單中選擇的值。resources.type 值欄會顯示值，您會在使用

AWS CLI 或 CloudTrail APIs 設定進階事件選取器時指定此 `resources.type` 值。記錄到 CloudTrail 的資料 API 資料行會針對資源類型顯示記錄到 CloudTrail 的 API 呼叫。

資料事件類型 (主控台)	resources.type 值	記錄到 CloudTrail 的資料 API
AwsApiCall	AWS::ECS::ContainerInstance	<code>ecs:Poll</code> (適用於 EC2 和 Amazon ECS 受管執行個體) <code>ecs:StartTelemetrySession</code> (適用於 EC2 和 Amazon ECS 受管執行個體) <code>ecs:PutSystemLogEvents</code> (適用於 Amazon ECS 受管執行個體)

您可以設定進階事件選取器來篩選 `eventName`、`readOnly` 和 `resources.ARN` 欄位，以僅記錄對您重要的事件。如需這些欄位的詳細資訊，請參閱 AWS CloudTrail API 參考中的 [AdvancedFieldSelector](#)。

Amazon ECS 事件範例

一個事件代表任何來源提出的單一請求，並包含請求 API 操作的相關資訊、操作的日期和時間、請求參數等。CloudTrail 日誌檔案不是公有 API 呼叫的已排序堆疊追蹤，因此事件不會以任何特定順序顯示。

以下範例顯示了展示 `CreateService` 動作的 CloudTrail 事件。

Note

此範例已格式化，以提升可讀性。在 CloudTrail 日誌檔案中，所有項目和事件會合併為單一系列。此外，這個範例中受限於單一 Amazon ECS 項目。在真正的 CloudTrail 日誌檔案中，您會看到來自多個 AWS 服務的項目和事件。

```
{
  "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"  
}
```

如需有關 CloudTrail 記錄內容的詳細資訊，請參閱「AWS CloudTrail 使用者指南」中的 [CloudTrail 記錄內容](#)。

檢視 Amazon ECS 服務和任務的 CloudWatch Logs Live Tail

CloudWatch Logs Live Tail 可在擷取時檢視新日誌事件的串流清單，協助您快速進行事件疑難排解。您可以在 Amazon ECS 主控台中檢視任務與服務事件。此檢視畫面可提供任務與服務運作狀態的整合式檢視。

Amazon ECS 服務中的每個任務都有針對每個容器的專用日誌串流。如果服務向上擴展，每個任務執行個體都有專屬的一組日誌串流。日誌串流的命名慣例遵循下列模式：

```
log-group-prefix/container-name/task-id
```

單一任務會在生命週期內寫入相同的日誌串流。日誌串流包含來自該任務容器的訊息，以及來自應用程式程式碼的任何輸出。每則訊息都會加上時間戳記，包括自訂日誌。

Note

Live Tail 工作階段會按工作階段使用時間 (每分鐘) 產生費用。如需定價的詳細資訊，請參閱 [Amazon CloudWatch 定價](#)。

所需的許可

主控台 IAM 使用者啟動或停止 CloudWatch Logs Live Tail 工作階段需要以下許可：

- logs:DescribeLogGroups
- logs:StartLiveTail
- logs:StopLiveTail

程序

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

2. 決定要檢視的資源

資源	步驟	
任務	a. 在叢集頁面上，選擇叢集。 「叢集詳細資訊」頁面隨即顯示。 b. 選擇 Tasks (任務) 索引標籤。 任務詳細資訊頁面隨即顯示。 c. 選擇 Logs (日誌) 索引標籤。	
服務	a. 在叢集頁面上，選擇叢集。 「叢集詳細資訊」頁面隨即顯示。 b. 選擇所需的服務。 服務詳細資訊頁面隨即顯示。 c. 選擇 Logs (日誌) 索引標籤。	

- 選擇 CloudWatch Logs Live Tail，然後選擇開始。
- (選用) 若要篩選串流，請在篩選器下的選取日誌群組欄位中，選擇所需的日誌群組。

AWS CLI 參考

您也可以使用 AWS CLI 來啟動 CloudWatch Logs 的 Live Tail 工作階段。下列資源提供搭配 AWS CLI Live Tail 使用的詳細資訊：

- [start-live-tail 命令參考](#) – 提供 `aws logs start-live-tail` 命令的完整命令語法、參數與範例。

- [CloudWatch Logs Live Tail 使用者指南](#) - 完整指南，包括 AWS CLI 使用僅限列印和互動式模式。
- [StartLiveTail SDK 範例](#) - 搭配各種 SDK 使用 StartLiveTail API 的程式設計範例。AWS SDKs

使用 Amazon ECS 中繼資料監控工作負載

您可以使用任務與容器中繼資料對工作負載進行疑難排解，並根據執行時期環境進行組態變更。

中繼資料包括下列類別：

- 提供任務執行位置相關資訊的任務層級屬性。
- 提供 Docker ID、名稱與映像詳細資訊的容器層級屬性。

這有助於掌握容器的運作狀況。

- IP 位址、子網路與網路模式等網路設定。

這有助於進行網路組態設定與疑難排解。

- 任務狀態與運作狀態

這有助於掌握任務是否正在執行。

您可以透過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](#)。

- 容器自我檢查

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 環境變數

環境變數控制着任務的行為，例如執行任務時所使用的容量。Amazon ECS 會為任務設定下列環境變數：

- `AWS_CONTAINER_CREDENTIALS_RELATIVE_URI` – SDK 在請求憑證時使用的完整 HTTP URL 端點。其中包含通訊協定與主機。如需詳細資訊，請參閱 [AWS SDKs Reference Guide](#) 中的 [Container credential provider](#)。
- `ECS_CONTAINER_METADATA_URI_V4` – 任務中繼資料第 4 版的位址。如需詳細資訊，請參閱 [Amazon ECS 任務中繼資料端點第 4 版](#)。
- `ECS_CONTAINER_METADATA_URI` – 任務中繼資料第 3 版的位址。如需詳細資訊，請參閱 [Amazon ECS 任務中繼資料端點第 3 版](#)。
- `ECS_AGENT_URI` – Fargate 支援的不同端點的基本位址。如需詳細資訊，請參閱：
 - [Amazon ECS 任務縮減保護端點](#)
 - [Amazon ECS 故障注入端點](#)
- `AWS_EXECUTION_ENV` – 任務執行所在的運算選項相關資訊。
 - 對於 Fargate，Amazon ECS 將此設定為 `AWS_ECS_FARGATE`。
 - 對於 EC2，Amazon ECS 將此設定為 `AWS_ECS_EC2`。
- `AWS_DEFAULT_REGION` – 您 AWS 區域的預設值 AWS 帳戶。這是任務執行所在的預設區域。
- `AWS_REGION` – 任務執行所在的區域。若已完成定義，此值便會覆寫 `AWS_DEFAULT_REGION`。

您可以在任務中繼資料中檢視環境變數。如需詳細資訊，請參閱下列其中一個主題：

- Fargate – [Fargate 上任務可用的 Amazon ECS 任務中繼資料](#)
- EC2 – [EC2 上 Amazon ECS 任務的任務中繼資料](#)

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。中繼資料檔案是在主機執行個體上建立，並以 Docker 磁碟區的形式掛載在容器中，因此在 AWS Fargate 上託管任務時無法使用。

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[Amazon ECS 任務與映像自動清理](#)。

主題

- [容器中繼資料檔案位置](#)
- [啟用 Amazon ECS 容器中繼資料](#)
- [Amazon ECS 容器中繼資料檔案格式](#)

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 Windows 執行個體 (PowerShell) :

```
Get-Content -path $env:ECS_CONTAINER_METADATA_FILE
```

啟用 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 容器中繼資料檔案格式

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": "arn:aws:ecs:us-east-1:123456789012:cluster/MyCluster",
  "TaskARN": "arn:aws:ecs:us-east-1:123456789012:task/MyCluster/
b593651c4d6b44a6b2b583f45c957e15",
  "Family": "curltest-container",
```

```

    "Revision": "2",
    "DesiredStatus": "RUNNING",
    "KnownStatus": "RUNNING",
    "Limits": {
        "CPU": 0.25,
        "Memory": 512
    },
    "PullStartedAt": "2025-01-17T20:56:17.394610044Z",
    "PullStoppedAt": "2025-01-17T20:56:25.282708213Z",
    "AvailabilityZone": "us-east-1b",
    "LaunchType": "FARGATE",
    "Containers": [
        {
            "DockerId": "b593651c4d6b44a6b2b583f45c957e15-3356213583",
            "Name": "curltest", "DockerName": "curltest",
            "Image": "public.ecr.aws/amazonlinux/amazonlinux:latest",

"ImageID": "sha256:7f371357694782356b65c7fd60dd1ca124c47bd5ed1b1ffe7c0e17f562898367",
            "Labels": {
                "com.amazonaws.ecs.cluster": "arn:aws:ecs:us-
east-1:123456789012:cluster/MyCluster",
                "com.amazonaws.ecs.container-name": "curltest",
                "com.amazonaws.ecs.task-arn": "arn:aws:ecs:us-
east-1:123456789012:task/MyCluster/b593651c4d6b44a6b2b583f45c957e15",
                "com.amazonaws.ecs.task-definition-family": "curltest-
container", "com.amazonaws.ecs.task-definition-version": "2"
            },
            "DesiredStatus": "RUNNING",
            "KnownStatus": "RUNNING",
            "Limits": {
                "CPU": 2
            },
            "CreatedAt": "2025-01-17T20:56:26.180347056Z",
            "StartedAt": "2025-01-17T20:56:26.180347056Z",
            "Type": "NORMAL",
            "LogDriver": "awslogs",
            "LogOptions": {
                "awslogs-create-group": "true",
                "awslogs-group": "/ecs/curltest-container",
                "awslogs-region": "us-east-1",

```

```

        "awslogs-stream":"ecs/curltest/b593651c4d6b44a6b2b583f45c957e15"
      },
      "ContainerARN":"arn:aws:ecs:us-east-1:123456789012:container/MyCluster/
b593651c4d6b44a6b2b583f45c957e15/934575e8-5bdb-478f-b763-2341a85b690e",
      "Networks":[
        {
          "NetworkMode":"awsvpc",
          "IPv4Addresses":["10.0.1.58"]
        }
      ],
      "Snapshotter":"overlayfs"
    }
  ],
  "ClockDrift":
  {
    "ClockErrorBound":0.487801,"ReferenceTimestamp":"2025-01-17T20:56:02Z",
    "ClockSynchronizationStatus":"SYNCHRONIZED"
  },
  "FaultInjectionEnabled":false
}

```

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"
}

```

Amazon ECS 受管執行個體上任務可用的 Amazon ECS 任務中繼資料

Amazon ECS 托管執行個體上的 Amazon ECS 提供了一種方法，可擷取關於容器及其所屬任務的各種中繼資料、網路指標與 [Docker 統計資料](#)。這稱為任務中繼資料端點。任務中繼資料端點第 4 版適用於 Amazon ECS 受管執行個體上的 Amazon ECS 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地 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

- [Amazon ECS 受管執行個體上任務的 Amazon ECS 任務中繼資料端點第 4 版](#)

Amazon ECS 受管執行個體上任務的 Amazon ECS 任務中繼資料端點第 4 版

Important

如果使用託管於 Amazon EC2 執行個體上的 Amazon ECS 任務，請參閱 [Amazon ECS 任務中繼資料端點](#)。

自 Amazon ECS 受管執行個體起，我們會向任務中的每個容器中插入一個名為 `ECS_CONTAINER_METADATA_URI_V4` 的環境變數。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。

對於在 Amazon ECS 受管執行個體上執行的所有 Amazon ECS 任務，任務中繼資料端點均預設為啟用。

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。

Amazon ECS 受管執行個體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 受管執行個體上任務的 Amazon ECS 任務中繼資料第 4 版 JSON 回應](#)」。

`${ECS_CONTAINER_METADATA_URI_V4}/stats`

此路徑傳回 Docker 容器的 Docker 統計資訊。如需每個所傳回統計資訊的詳細資訊，請參閱 Docker API 文件中的 [ContainerStats](#)。

 Note

Amazon ECS 受管執行個體上的 Amazon ECS 任務，要求容器必須執行約 1 秒後，才傳回容器統計資料。

`${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。如需每個所傳回統計資訊的詳細資訊，請參閱 Docker API 文件中的 [ContainerStats](#)。

 Note

Amazon ECS 受管執行個體上的 Amazon ECS 任務，要求容器必須執行約 1 秒後，才傳回容器統計資料。

`${ECS_CONTAINER_METADATA_URI_V4}/tasks`

此路徑會傳回容器執行個體上執行之所有任務的中繼資料。回應是任務中繼資料物件的 JSON 陣列，其中每個元素的結構與來自 `/task` 端點的回應相同。如需每個任務回應的詳細資訊，請參閱 [Amazon ECS 受管執行個體上任務的 Amazon ECS 任務中繼資料第 4 版 JSON 回應](#)。

 Note

此端點僅適用於在 Amazon ECS 受管執行個體上以 Managed Daemon Service 身分執行的任務。

`${ECS_CONTAINER_METADATA_URI_V4}/tasks/stats`

此路徑會傳回容器執行個體上執行之所有任務中所有容器的 Docker 統計資料。回應是 JSON 陣列，其中每個元素都包含容器 IDs 映射至特定任務的 Docker 統計資料。如需每個所傳回統計資訊的詳細資訊，請參閱 Docker API 文件中的 [ContainerStats](#)。

 Note

此端點僅適用於在 Amazon ECS 受管執行個體上以 Managed Daemon Service 身分執行的任務。

Amazon ECS 受管執行個體上任務的 Amazon ECS 任務中繼資料第 4 版 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。

ServiceName

任務所屬的服務名稱。如果任務與服務相關聯，則會顯示 ServiceName。

VPCID

容器執行個體的 VPC ID。

TaskARN

容器所屬任務的 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。

LaunchType

任務使用的啟動類型。對於在 Amazon ECS 受管執行個體上執行的任務，啟動類型為 `MANAGED_INSTANCES`。

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 Amazon ECS 受管執行個體時，ID 為一個 32 位十六進位字串，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

映像資訊清單的 SHA-256 摘要。此摘要可用於以 `repository-url/image@sha256:digest` 格式提取映像。

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。

LogOptions

為容器定義的日誌驅動程式選項。

ContainerARN

容器的 Amazon Resource Name (ARN)。

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。

RestartCount

容器已重新啟動的次數。

Note

僅在容器已啟用重新啟動政策時，才會包含 RestartCount 中繼資料。如需詳細資訊，請參閱[使用容器重新啟動政策在 Amazon ECS 任務中重新啟動個別容器](#)。

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 Time Sync Service 來衡量時鐘準確性，並提供容器的時鐘錯誤界限。如需詳細資訊，請參閱 Amazon EC2 User Guide for Linux instances 中的 [Set the time for your Linux instance](#)。

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。

FaultInjectionEnabled

指出是否已為任務啟用了故障注入。如需詳細資訊，請參閱[搭配 Amazon ECS 與 Fargate 工作負載使用故障注入](#)。

Amazon ECS 受管執行個體上任務的 Amazon ECS 任務中繼資料第 4 版範例

以下範例顯示了在 Amazon ECS 執行個體上執行之 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": "400c0466d1e54c0691aae0f86e0f9dc6-2531612879",
  "Name": "nginx",
  "DockerName": "nginx",
  "Image": "public.ecr.aws/nginx/nginx:latest",
  "ImageID": "sha256:d5f28ef21aabddd098f3dbc21fe5b7a7d7a184720bc07da0b6c9b9820e97f25e",
  "Labels": {
    "com.amazonaws.ecs.cluster": "arn:aws:ecs:us-west-2:111122223333:cluster/managed-instances-cluster",
    "com.amazonaws.ecs.container-name": "curl",
    "com.amazonaws.ecs.task-arn": "arn:aws:ecs:us-west-2:111122223333:task/managed-instances-cluster/400c0466d1e54c0691aae0f86e0f9dc6",
    "com.amazonaws.ecs.task-definition-family": "ecs-managed-instances-task-def",
    "com.amazonaws.ecs.task-definition-version": "7"
  },
  "DesiredStatus": "RUNNING",
  "KnownStatus": "RUNNING",
  "Limits": { "CPU": 2 },
  "CreatedAt": "2025-09-24T19:39:04.88336233Z",
  "StartedAt": "2025-09-24T19:39:04.88336233Z",
  "Type": "NORMAL",
```

```

"LogDriver": "awslogs",
"LogOptions": {
  "awslogs-create-group": "true",
  "awslogs-group": "/ecs/managed-instances-task-def",
  "awslogs-region": "us-west-2",
  "awslogs-stream": "ecs/nginx/400c0466d1e54c0691aae0f86e0f9dc6"
},
"ContainerARN": "arn:aws:ecs:us-west-2:111122223333:container/managed-instances-cluster/400c0466d1e54c0691aae0f86e0f9dc6/3703f0e4-a351-4f1e-a0e7-6981ea7adc8b",
"Networks": [
  {
    "NetworkMode": "awsvpc",
    "IPv4Addresses": ["172.31.62.223"],
    "AttachmentIndex": 0,
    "MACAddress": "0e:41:0b:1e:f2:fb",
    "IPv4SubnetCIDRBlock": "172.31.48.0/20",
    "PrivateDNSName": "ip-172-31-62-223.us-west-2.compute.internal",
    "SubnetGatewayIpv4Address": "172.31.48.1/20"
  }
],
"Snapshotter": "overlayfs"
}

```

Amazon ECS 受管執行個體上任務的 Amazon ECS 任務中繼資料第 4 版範例

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-west-2:111122223333:cluster/managed-instances-cluster",
  "TaskARN": "arn:aws:ecs:us-west-2:111122223333:task/managed-instances-cluster/400c0466d1e54c0691aae0f86e0f9dc6",
  "Family": "managed-instances-task-def",
  "Revision": "7",
  "DesiredStatus": "RUNNING",
  "KnownStatus": "RUNNING",
  "Limits": { "CPU": 1, "Memory": 3072 },
  "PullStartedAt": "2025-09-24T19:38:58.682942001Z",
  "PullStoppedAt": "2025-09-24T19:39:03.091597524Z",
  "AvailabilityZone": "us-west-2d",
  "LaunchType": "MANAGED_INSTANCES",
  "Containers": [
    {

```

```

    "DockerId": "400c0466d1e54c0691aae0f86e0f9dc6-2531612879",
    "Name": "curl",
    "DockerName": "curl",
    "Image": "nginx",
    "ImageID":
"sha256:d5f28ef21aabddd098f3dbc21fe5b7a7d7a184720bc07da0b6c9b9820e97f25e",
    "Labels": {
        "com.amazonaws.ecs.cluster": "arn:aws:ecs:us-west-2:111122223333:cluster/
managed-instances-cluster",
        "com.amazonaws.ecs.container-name": "nginx",
        "com.amazonaws.ecs.task-arn": "arn:aws:ecs:us-west-2:111122223333:task/managed-
instances-cluster/400c0466d1e54c0691aae0f86e0f9dc6",
        "com.amazonaws.ecs.task-definition-family": "managed-instances-task-def",
        "com.amazonaws.ecs.task-definition-version": "7"
    },
    "DesiredStatus": "RUNNING",
    "KnownStatus": "RUNNING",
    "Limits": { "CPU": 2 },
    "CreatedAt": "2025-09-24T19:39:04.88336233Z",
    "StartedAt": "2025-09-24T19:39:04.88336233Z",
    "Type": "NORMAL",
    "LogDriver": "awslogs",
    "LogOptions": {
        "awslogs-create-group": "true",
        "awslogs-group": "/ecs/managed-instances-task-def",
        "awslogs-region": "us-west-2",
        "awslogs-stream": "ecs/nginx/400c0466d1e54c0691aae0f86e0f9dc6"
    },
    "ContainerARN": "arn:aws:ecs:us-west-2:111122223333:container/managed-instances-
cluster/400c0466d1e54c0691aae0f86e0f9dc6/3703f0e4-a351-4f1e-a0e7-6981ea7adc8b",
    "Networks": [
        {
            "NetworkMode": "awsvpc",
            "IPv4Addresses": ["172.31.62.223"],
            "AttachmentIndex": 0,
            "MACAddress": "0e:41:0b:1e:f2:fb",
            "IPv4SubnetCIDRBlock": "172.31.48.0/20",
            "PrivateDNSName": "ip-172-31-62-223.us-west-2.compute.internal",
            "SubnetGatewayIpv4Address": "172.31.48.1/20"
        }
    ],
    "Snapshotter": "overlayfs"
}
],

```



```

"ServiceName": "exec-service-2",
"ClockDrift": {
  "ClockErrorBound": 0.14120749999999999,
  "ReferenceTimestamp": "2025-09-24T19:48:37Z",
  "ClockSynchronizationStatus": "SYNCHRONIZED"
},
"FaultInjectionEnabled": false
}

```

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。

```

{
  "400c0466d1e54c0691aae0f86e0f9dc6-2531612879": {
    "read": "2025-09-24T19:51:54.899736614Z",
    "preread": "2025-09-24T19:51:44.901199024Z",
    "pids_stats": {},
    "blkio_stats": {
      "io_service_bytes_recursive": [
        { "major": 259, "minor": 1, "op": "read", "value": 0 },
        { "major": 259, "minor": 1, "op": "write", "value": 131072 },
        { "major": 259, "minor": 0, "op": "read", "value": 48173056 },
        { "major": 259, "minor": 0, "op": "write", "value": 0 },
        { "major": 252, "minor": 0, "op": "read", "value": 48173056 },
        { "major": 252, "minor": 0, "op": "write", "value": 0 }
      ],
      "io_serviced_recursive": null,
      "io_queue_recursive": null,
      "io_service_time_recursive": null,
      "io_wait_time_recursive": null,
      "io_merged_recursive": null,
      "io_time_recursive": null,
      "sectors_recursive": null
    },
    "num_procs": 0,
    "storage_stats": {},
    "cpu_stats": {
      "cpu_usage": {
        "total_usage": 670462000,
        "usage_in_kernelmode": 276072000,
        "usage_in_usermode": 394389000
      }
    }
  }
}

```

```

    },
    "system_cpu_usage": 787660000000,
    "online_cpus": 1,
    "throttling_data": {
        "periods": 0,
        "throttled_periods": 0,
        "throttled_time": 0
    }
},
"precpu_stats": {
    "cpu_usage": {
        "total_usage": 663809000,
        "usage_in_kernelmode": 273333000,
        "usage_in_usermode": 390476000
    },
    "system_cpu_usage": 777710000000,
    "online_cpus": 1,
    "throttling_data": {
        "periods": 0,
        "throttled_periods": 0,
        "throttled_time": 0
    }
},
"memory_stats": {
    "usage": 83562496,
    "stats": {
        "active_anon": 24576,
        "active_file": 258048,
        "anon": 32264192,
        "anon_thp": 0,
        "file": 48533504,
        "file_dirty": 0,
        "file_mapped": 36286464,
        "file_writeback": 0,
        "inactive_anon": 32243712,
        "inactive_file": 48271360,
        "kernel_stack": 442368,
        "pgactivate": 6,
        "pgdeactivate": 0,
        "pgfault": 18936,
        "pglazyfree": 0,
        "pglazyfreed": 0,
        "pgmajfault": 311,
        "pgrefill": 0,

```

```

    "pgscan": 0,
    "pgsteal": 0,
    "shmem": 4096,
    "slab": 1598960,
    "slab_reclaimable": 1080040,
    "slab_unreclaimable": 518920,
    "sock": 0,
    "thp_collapse_alloc": 0,
    "thp_fault_alloc": 0,
    "unevictable": 0,
    "workingset_activate": 0,
    "workingset_nodereclaim": 0,
    "workingset_refault": 0
  },
  "limit": 18446744073709551615
},
"name": "nginx",
"id": "400c0466d1e54c0691aae0f86e0f9dc6-2531612879"
}
}

```

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 awsipc 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 任務中繼資料端點第 4 版](#)。
- 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 awsipc 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 任務中繼資料端點第 3 版](#)。
- 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 awsipc 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 任務中繼資料端點第 2 版](#)。

如果您的 Amazon ECS 任務託管在 Amazon EC2 上，或者您的任務使用 host 網路模式並託管在 Amazon ECS 受管執行個體上，您也可以使用[執行個體中繼資料服務 \(IMDS\) 端點來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
```

如果 Amazon ECS 任務以純 IPv6 組態託管在 Amazon EC2 上，您可以使用 IPv6 IMDS 端點存取任務主機中繼資料。當從託管該任務的執行個體內執行時，下列命令會透過 IPv6 列出主機執行個體的 ID。

```
curl http://[fd00:ec2::254]/latest/meta-data/instance-id
```

若要存取 IPv6 IMDS 端點，請在容器執行個體上啟用 IPv6 IMDS 端點，並使用所選 SDK 的 IMDS 憑證提供者將中繼資料服務端點模式設定為 IPv6。如需有關為容器執行個體啟用 IPv6 IMDS 端點的詳細資訊，請參閱 Amazon EC2 User Guide 中的[Configure the Instance Metadata Service options](#)。如需有關 SDK 的 IMDS 憑證提供者的詳細資訊，請參閱 AWS SDKs and Tools Reference Guide 中的[IMDS credential provider](#)。

Note

當 `awsipcTrunking` 帳戶設定啟用後，將無法存取 IPv6 IMDS 端點。若要在啟用 `awsipcTrunking` 後存取容器執行個體 IAM 角色憑證，您可以改為使用任務 IAM 角色。如需關於任務 IAM 角色的詳細資訊，請參閱[Amazon ECS 任務 IAM 角色](#)。

您可以從端點取得的資訊分為幾個類別，例如 *instance-id*。如需有關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[Instance metadata categories](#)。

Amazon ECS 任務中繼資料端點第 4 版

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](#)。

任務中繼資料和網路速率統計資訊會傳送至 CloudWatch Container Insights，並可在 AWS 管理主控台中檢視。如需詳細資訊，請參閱[使用具有增強可觀測性的 Container Insights 監控 Amazon ECS](#)。

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](#)。

如需任務中繼資料範例輸出，請參閱 [Amazon ECS 任務中繼資料第 4 版範例](#)。

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 任務中繼資料第 4 版 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。

 Important

使用 `${ECS_CONTAINER_METADATA_URI_V4}/taskWithTags` 端點時，請注意每次呼叫最多會向 `ecs:ListTagsForResource` 發出兩個 API 請求（一個用於容器執行個體標籤，另一個用於任務標籤），並且任務中的任何邊車容器都可代您進行這些呼叫。頻繁的端點呼叫可能會導致 API 限流。

請考慮實作快取或批次處理策略來降低呼叫頻率，尤其是在高流量應用程式中，同時使用 AWS CloudTrail 對 API 限流問題進行偵錯。如需 `ListTagsForResource` API 限流限制的相關資訊，請參閱 Amazon Elastic Container Service API Reference 中的 [Request throttling for the Amazon ECS API](#)。如需使用偵錯 Amazon ECS API 呼叫的詳細資訊 AWS CloudTrail，請參閱 [使用記錄 Amazon ECS API 呼叫 AWS CloudTrail](#)。

`${ECS_CONTAINER_METADATA_URI_V4}/stats`

此路徑傳回特定容器的 Docker 統計資訊。如需每個所傳回統計資訊的詳細資訊，請參閱 Docker API 文件中的 [ContainerStats](#)。

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。如需每個所傳回統計資訊的詳細資訊，請參閱 Docker API 文件中的 [ContainerStats](#)。

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 任務中繼資料第 4 版 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

任務所屬的服務名稱。如果任務與服務關聯，Amazon EC2 和 Amazon ECS Anywhere 容器執行個體都會顯示 ServiceName。

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 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

映像資訊清單的 SHA-256 摘要。此摘要可用於以 `repository-url/image@sha256:digest` 格式提取映像。

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 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。

RestartCount

容器已重新啟動的次數。

Note

僅在容器已啟用重新啟動政策時，才會包含 RestartCount 中繼資料。如需詳細資訊，請參閱[使用容器重新啟動政策在 Amazon ECS 任務中重新啟動個別容器](#)。

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 任務中繼資料第 4 版範例

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。以下是作為服務 (MyService) 一部分執行之任務的範例輸出。

```
{
  "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 任務中繼資料端點第 3 版

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”](#)。如果您使用託管在上的 Amazon ECS 任務 AWS Fargate，請參閱[Amazon ECS 任務中繼資料端點第 3 版](#)，以取得 [Fargate 上的任務](#)。

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。

依預設，對於在平台版本 1.3.0 或更新版本上使用 Fargate 的任務，以及使用 EC2 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 基礎結構，或者在至少執行 Amazon ECS 容器代理程式 1.54.0 版的 Amazon EC2 Windows 基礎結構上啟動該功能，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 且平台版本低於 1.3.0 版的任務，支援任務中繼資料第 2 版端點。如需詳細資訊，請參閱[Amazon ECS 任務中繼資料端點第 2 版](#)。

任務中繼資料端點第 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 任務中繼資料第 3 版 JSON 回應](#)」。

`${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。如需每個所傳回統計資訊的詳細資訊，請參閱 Docker API 文件中的 [ContainerStats](#)。

`${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。如需每個所傳回統計資訊的詳細資訊，請參閱 Docker API 文件中的 [ContainerStats](#)。

Amazon ECS 任務中繼資料第 3 版 JSON 回應

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 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

映像資訊清單的 SHA-256 摘要。此摘要可用於以 `repository-url/image@sha256:digest` 格式提取映像。

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 Time Sync Service 來衡量時鐘準確性，並提供容器的時鐘錯誤界限。如需詳細資訊，請參閱 Amazon EC2 User Guide for Linux instances 中的 [Set the time for your Linux instance](#)。

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 任務中繼資料第 3 版範例

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 任務中繼資料端點第 2 版

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

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 且平台版本為 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 平台版本](#)。
- 使用 EC2 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 基礎結構上啟動，或在至少執行 1.54.0 版 Amazon ECS 容器代理程式的 Amazon EC2 Windows 基礎結構上啟動。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。如需每個所傳回統計資訊的詳細資訊，請參閱 Docker API 文件中的 [ContainerStats](#)。

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。如需每個所傳回統計資訊的詳細資訊，請參閱 Docker API 文件中的 [ContainerStats](#)。

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 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

映像資訊清單的 SHA-256 摘要。此摘要可用於以 `repository-url/image@sha256:digest` 格式提取映像。

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 Time Sync Service 來衡量時鐘準確性，並提供容器的時鐘錯誤界限。如需詳細資訊，請參閱 Amazon EC2 User Guide for Linux instances 中的 [Set the time for your Linux instance](#)。

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"
}
```

Fargate 上任務可用的 Amazon ECS 任務中繼資料

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

- [Fargate 上任務的 Amazon ECS 任務中繼資料端點第 4 版](#)
- [Fargate 上任務的 Amazon ECS 任務中繼資料端點第 3 版](#)

Fargate 上任務的 Amazon ECS 任務中繼資料端點第 4 版

Important

如果使用託管於 Amazon EC2 執行個體上的 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。

對於在使用平台版本 1.4.0 或更新版本上執行的所有 Amazon ECS 任務 AWS Fargate，任務中繼資料端點預設為開啟。

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「[Fargate 上任務的 Amazon ECS 中繼資料第 4 版 JSON 回應](#)」。

`${ECS_CONTAINER_METADATA_URI_V4}/stats`

此路徑傳回 Docker 容器的 Docker 統計資訊。如需每個所傳回統計資訊的詳細資訊，請參閱 Docker API 文件中的 [ContainerStats](#)。

 Note

上的 Amazon ECS 任務 AWS Fargate 需要容器執行約 1 秒，才能傳回容器統計資料。

`${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。如需每個所傳回統計資訊的詳細資訊，請參閱 Docker API 文件中的 [ContainerStats](#)。

 Note

上的 Amazon ECS 任務 AWS Fargate 需要容器執行約 1 秒，才能傳回容器統計資料。

Fargate 上任務的 Amazon ECS 中繼資料第 4 版 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。

ServiceName

任務所屬的服務名稱。對於與服務相關聯的 Fargate 任務，會顯示 ServiceName。

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 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 ephemeralStorage 物件來變更暫時性儲存量。ephemeralStorage 是以 GiB 為單位，而不是以 MiB 為單位。此 ephemeralStorage 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

映像資訊清單的 SHA-256 摘要。此摘要可用於以 `repository-url/image@sha256:digest` 格式提取映像。

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 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。

RestartCount

容器已重新啟動的次數。

Note

僅在容器已啟用重新啟動政策時，才會包含 `RestartCount` 中繼資料。如需詳細資訊，請參閱[使用容器重新啟動政策在 Amazon ECS 任務中重新啟動個別容器](#)。

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 Time Sync Service 來衡量時鐘準確性，並提供容器的時鐘錯誤界限。如需詳細資訊，請參閱 Amazon EC2 User Guide for Linux instances 中的 [Set the time for your Linux instance](#)。

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。

Fargate 上任務的 Amazon ECS 任務中繼資料第 4 版範例

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"
}

```

Fargate 上任務的 Amazon ECS 任務中繼資料第 4 版範例

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/MyEmptyCluster",
  "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":
"sha256:8059bdd0058510c03ae4c808de8c4fd2c1f3c1b6d9ea75487f1e5caa5eacea02",
  "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
    ],
    "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,
            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
  }
}
```

Fargate 上任務的 Amazon ECS 任務中繼資料端點第 3 版

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 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「[Fargate 上任務的 Amazon ECS 中繼資料第 3 版 JSON 回應](#)」。

`${ECS_CONTAINER_METADATA_URI}/stats`

此路徑傳回特定 Docker 容器的 Docker 統計資訊 JSON。如需每個所傳回統計資訊的詳細資訊，請參閱 Docker API 文件中的 [ContainerStats](#)。

`${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。如需每個所傳回統計資訊的詳細資訊，請參閱 Docker API 文件中的 [ContainerStats](#)。

Fargate 上任務的 Amazon ECS 中繼資料第 3 版 JSON 回應

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 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

映像資訊清單的 SHA-256 摘要。此摘要可用於以 `repository-url/image@sha256:digest` 格式提取映像。

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 Time Sync Service 來衡量時鐘準確性，並提供容器的時鐘錯誤界限。如需詳細資訊，請參閱 Amazon EC2 User Guide for Linux instances 中的 [Set the time for your Linux instance](#)。

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。

Fargate 上任務的 Amazon ECS 任務中繼資料第 3 版範例

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 容器自我檢查

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": "2023-10-08T20:09:11.44527186Z",
          "StartedAt": "2023-10-08T20:09:11.44527186Z",
          "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"  
}
```

使用執行時期監控識別未經授權的行為

Amazon GuardDuty 是一種威脅偵測服務，可協助保護您 AWS 環境中的帳戶、容器、工作負載和資料。GuardDuty 使用機器學習 (ML) 模型，以及異常和威脅偵測功能，持續監控不同的日誌來源和執行時期活動，以識別環境中的潛在安全風險和惡意活動和排定其優先順序。

GuardDuty 中的執行期監控透過持續監控 AWS 日誌和聯網活動來識別惡意或未經授權的行為，來保護 Fargate 和 EC2 容器執行個體上執行的工作負載。執行時期監控使用輕量且全受管的 GuardDuty 安全代理程式，可分析主機上的行為，例如檔案存取、程序執行與網路連線。這涵蓋的問題包括：提升權限、使用暴露的憑證、與惡意 IP 位址或網域通訊，以及 Amazon EC2 執行個體與容器工作負載上出現惡意軟體。如需詳細資訊，請參閱 GuardDuty User Guide 中的 [GuardDuty Runtime Monitoring](#)。

您的安全管理員可以在 AWS Organizations 中，為一或多個帳戶啟用 GuardDuty 的執行時期監控功能。他們也會選取當您使用 Fargate 時，GuardDuty 是否自動部署 GuardDuty 安全代理程式。所有叢集都會自動受到保護，GuardDuty 會代為管理安全代理程式。

在下列情況下，您也可以手動設定 GuardDuty 安全代理程式：

- 使用 EC2 容器執行個體
- 您需要精細控制權限，在叢集層級啟用執行時期監控

若要使用執行時期監控，必須設定受保護的叢集，並在 EC2 容器執行個體上安裝並管理 GuardDuty 安全代理程式。

執行時期監控如何與 Amazon ECS 搭配運作

執行時期監控使用輕量型 GuardDuty 安全代理程式，該程式會監控 Amazon ECS 工作負載的活動，藉此分析應用程式如何請求、取得存取權及消耗底層系統資源。

對於 Fargate 任務，GuardDuty 安全代理程式會作為每個任務的邊車容器執行。

對於 EC2 容器執行個體，GuardDuty 安全代理程式會在執行個體上作為程序執行。

GuardDuty 安全代理程式會收集下列資源的資料，然後將資料傳送至 GuardDuty 進行處理。您可以在 GuardDuty 主控台中檢視調查結果。您也可以將它們傳送至其他 AWS Security Hub CSPM，AWS 服務 例如第三方安全廠商，以進行彙總和修復。如需有關如何檢視與管理調查結果的資訊，請參閱 Amazon GuardDuty User Guide 中的 [Managing Amazon GuardDuty findings](#)。

- 下列 Amazon ECS API 呼叫的回應：

- [DescribeClusters](#)

當您使用 `--include TAGS` 選項時，回應參數包含執行時期監控標籤 (如果設定了標籤)。

- [DescribeTasks](#)

對於 Fargate，回應參數包含 GuardDuty 邊車容器。

- [ListAccountSettings](#)

回應參數包含執行時期監控的帳戶設定，該設定安全管理員進行。

- 容器代理程式自我檢查資料。如需詳細資訊，請參閱[Amazon ECS 容器自我檢查](#)。
- 運算選項的任務中繼資料端點：
 - [Amazon ECS 任務中繼資料端點第 4 版](#)
 - [Fargate 上任務的 Amazon ECS 任務中繼資料端點第 4 版](#)

考量事項

使用執行時期監控時，建議考量下列事項：

- 執行時期監控會產生相關成本。如需更多資訊，請參閱 [Amazon GuardDuty 定價](#)。
- Amazon ECS Anywhere 不支援執行時期監控。
- Windows 作業系統不支援執行時期監控。
- 當您在 Fargate 上使用 Amazon ECS Exec 時，必須指定容器名稱，因為 GuardDuty 安全代理程式會作為邊車容器執行。
- 您無法在 GuardDuty 安全代理程式邊車容器上使用 Amazon ECS Exec。
- 在叢集層級控制執行時期監控的 IAM 使用者，必須具備適當的 IAM 許可才能進行標籤作業。如需詳細資訊，請參閱 [《IAM 使用者指南》中的 IAM 教學課程：定義根據標籤存取 AWS 資源的許可](#)。
- Fargate 任務必須使用任務執行角色。此角色會授予任務許可，使其能代為擷取、更新與管理儲存在 Amazon ECR 私有儲存庫中的 GuardDuty 安全代理程式。
- 在 Amazon ECS 受管執行個體上執行的應用程式不支援執行時期監控。

資源使用率

您新增至叢集的標籤會計入叢集標籤配額。

GuardDuty 代理程式邊車容器不會計入每個任務定義配額的容器數目。

與大多數安全軟體一樣，GuardDuty 也有少量額外負荷。如需有關 Fargate 記憶體限制的資訊，請參閱 GuardDuty User Guide 中的 [CPU and memory limits](#)。如需有關 Amazon EC2 記憶體限制的資訊，請參閱 [CPU and memory limit for GuardDuty agent](#)。

Amazon ECS Fargate 工作負載的執行時期監控

如果您使用的是 EC2 容器執行個體，則必須手動設定執行時期監控。如需詳細資訊，請參閱 [Amazon ECS 上 EC2 工作負載的執行時期監控](#)。

您可以讓 GuardDuty 管理容器執行個體上的安全代理程式。此選項只適用於 Fargate。此選項 (GuardDuty 代理程式管理) 在 GuardDuty 中可用

當您使用 GuardDuty 代理程式管理時，GuardDuty 會執行下列操作：

- 為託管叢集的每個 VPC 建立用於 GuardDuty 的 VPC 端點。
- 在所有新的獨立 Fargate 任務與新的服務部署上，擷取並安裝最新的 GuardDuty 安全代理程式作為邊車容器。

首次啟動服務，或使用強制執行新部署選項更新現有服務時，即會進行新的服務部署。

啟用 Amazon ECS 的執行時期監控

您可以設定 GuardDuty 自動管理所有 Fargate 叢集的安全代理程式。

先決條件

以下是使用執行時期監控的先決條件：

- 對於 Linux 作業系統，Fargate 平台版本必須為 1.4.0 版或更新版本。
- Amazon ECS 的 IAM 角色與許可：
 - Fargate 任務必須使用任務執行角色。此角色會授予任務許可，使其能代為擷取、更新與管理 GuardDuty 安全代理程式。如需更多資訊，請參閱 [Amazon ECS 任務執行 IAM 角色](#)。
 - 您可以使用預先定義的標籤控制叢集的執行時期監控。如果存取政策依據標籤限制存取權，您必須明確授予 IAM 使用者標記叢集的權限。如需詳細資訊，請參閱《[IAM 使用者指南](#)》中的 [IAM 教學課程：定義根據標籤存取 AWS 資源的許可](#)。
- 連線至 Amazon ECR 儲存庫：

GuardDuty 安全代理程式儲存在 Amazon ECR 儲存庫中。每個獨立任務與服務任務都必須具備儲存庫的存取權。您可以使用下列其中一個選項：

- 對於公有子網路中的任務，您可以使用任務的公有 IP 位址，或在任務執行所在的子網路中為 Amazon ECR 建立 VPC 端點。如需詳細資訊，請參閱 Amazon Elastic Container Registry User Guide 中的 [Amazon ECR interface VPC endpoints \(AWS PrivateLink\)](#)。
- 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，或在任務執行所在的子網路中為 Amazon ECR 建立 VPC 端點。

如需詳細資訊，請參閱[私有子網路與 NAT 閘道](#)。

- 您必須具備使用 GuardDuty 所需的 `AWSServiceRoleForAmazonGuardDuty` 角色。如需詳細資訊，請參閱 Amazon GuardDuty User Guide 中的 [Service-linked role permissions for GuardDuty](#)。
- 您希望透過執行時期監控防護的所有檔案，都必須能被根使用者存取。如果您手動變更檔案的許可，則必須將其設定為 755。

以下為在 EC2 容器執行個體上使用執行時期監控的先決條件：

- 您必須使用 20230929 版或更新版本的 Amazon ECS-AMI。
- 您必須在容器執行個體上將 Amazon ECS 代理程式升級至 1.77 版或更新版本。
- 您必須使用核心版本 5.10 或更新版本。
- 如需有關支援的 Linux 作業系統與架構的資訊，請參閱 [Which operating models and workloads does GuardDuty Runtime Monitoring support](#)。
- 您可以使用 Systems Manager 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[設定 EC2 執行個體的 Systems Manager](#)。

程序

您可以在 GuardDuty 中啟用執行時期監控。如需如何啟用此功能的資訊，請參閱 Amazon GuardDuty User Guide 中的 [Enabling Runtime Monitoring](#)。

將執行時期監控新增至現有的 Amazon ECS Fargate 任務

當您啟用執行時期監控時，叢集中所有新獨立任務與新服務部署都會自動受到保護。為了維持不可變性限制，現有任務不會受到影響。

先決條件

1. 啟用執行時期監控。如需詳細資訊，請參閱[啟用 Amazon ECS 的執行時期監控](#)。
2. Fargate 任務必須使用任務執行角色。此角色會授予任務許可，使其能代為擷取、更新與管理 GuardDuty 安全代理程式。如需更多資訊，請參閱[Amazon ECS 任務執行 IAM 角色](#)。

程序

- 若要立即保護任務，您需要執行下列任一動作：
 - 對於獨立任務，請先停止任務，然後再啟動任務。如需詳細資訊，請參閱[停止 Amazon ECS 任務](#)和[將應用程式作為 Amazon ECS 任務執行](#)
 - 對於屬於服務的任務，請使用「強制執行新部署」選項來更新服務。如需詳細資訊，請參閱[更新 Amazon ECS 服務](#)。

從 Amazon ECS 叢集中移除執行時期監控

您可能想要將某些叢集排除在保護之外，例如用於測試的叢集。這會導致 GuardDuty 對叢集中的資源執行下列操作：

- 不再將 GuardDuty 安全代理程式部署至新的獨立 Fargate 任務或新的服務部署。

為了維持不可變性限制，已啟用執行時期監控的現有任務與部署不會受到影響。

- 停止計費且不再接受任務的執行時期事件。

程序

請執行下列步驟，從叢集中移除執行時期監控。

1. 使用 Amazon ECS 主控台或將叢集上的 GuardDutyManaged 標籤金鑰 AWS CLI 設定為 false。如需詳細資訊，請參閱[更新叢集](#)或[透過 CLI 或 API 使用標籤](#)。請對標籤使用以下值。

Note

鍵與值區分大小寫，且必須與字串完全相符。

索引鍵 = 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 User Guide 中的 [Delete an interface endpoint](#)。

從帳戶移除 Amazon ECS 的執行時期監控

若不想再使用執行時期監控，可在 GuardDuty 中停用此功能。如需有關如何停用此功能的資訊，請參閱 Amazon GuardDuty User Guide 中的 [Enabling Runtime Monitoring](#)。

GuardDuty 會執行以下操作：

- 針對每個託管叢集的 VPC 刪除 GuardDuty 的 VPC 端點。
- 不再將 GuardDuty 安全代理程式部署至新的獨立 Fargate 任務或新的服務部署。

為了維持不可變性限制，現有的任務與部署在停止、複寫或擴展之前不會受到影響。

- 停止計費且不再接受任務的執行時期事件。

Amazon ECS 上 EC2 工作負載的執行時期監控

當您使用 EC2 執行個體作為容量來源，或需要在 Fargate 上以叢集層級對執行時期監控進行精細控制時，請使用此選項。

您可以透過新增預先定義的標籤，為執行時期監控佈建叢集。

若為 EC2 容器執行個體，您可以下載、安裝並管理 GuardDuty 安全代理程式。

若為 Fargate，GuardDuty 會代為管理安全代理程式。

啟用 Amazon ECS 的執行時期監控

您可以為使用 EC2 執行個體的叢集啟用執行時期監控，或在 Fargate 環境中需要叢集層級的精細控制時啟用此功能。

以下是使用執行時期監控的先決條件：

- 對於 Linux 作業系統，Fargate 平台版本必須為 1.4.0 版或更新版本。
- Amazon ECS 的 IAM 角色與許可：
 - Fargate 任務必須使用任務執行角色。此角色會授予任務許可，使其能代為擷取、更新與管理 GuardDuty 安全代理程式。如需更多資訊，請參閱[Amazon ECS 任務執行 IAM 角色](#)。

- 您可以使用預先定義的標籤控制叢集的執行時期監控。如果存取政策依據標籤限制存取權，您必須明確授予 IAM 使用者標記叢集的權限。如需詳細資訊，請參閱 [《IAM 使用者指南》中的 IAM 教學課程：定義根據標籤存取 AWS 資源的許可](#)。
- 連線至 Amazon ECR 儲存庫：

GuardDuty 安全代理程式儲存在 Amazon ECR 儲存庫中。每個獨立任務與服務任務都必須具備儲存庫的存取權。您可以使用下列其中一個選項：

- 對於公有子網路中的任務，您可以使用任務的公有 IP 位址，或在任務執行所在的子網路中為 Amazon ECR 建立 VPC 端點。如需詳細資訊，請參閱 Amazon Elastic Container Registry User Guide 中的 [Amazon ECR interface VPC endpoints \(AWS PrivateLink\)](#)。
- 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，或在任務執行所在的子網路中為 Amazon ECR 建立 VPC 端點。

如需詳細資訊，請參閱[私有子網路與 NAT 閘道](#)。

- 您必須具備使用 GuardDuty 所需的 AWSServiceRoleForAmazonGuardDuty 角色。如需詳細資訊，請參閱 Amazon GuardDuty User Guide 中的 [Service-linked role permissions for GuardDuty](#)。
- 您希望透過執行時期監控防護的所有檔案，都必須能被根使用者存取。如果您手動變更檔案的許可，則必須將其設定為 755。

以下為在 EC2 容器執行個體上使用執行時期監控的先決條件：

- 您必須使用 20230929 版或更新版本的 Amazon ECS-AMI。
- 您必須在容器執行個體上將 Amazon ECS 代理程式升級至 1.77 版或更新版本。
- 您必須使用核心版本 5.10 或更新版本。
- 如需有關支援的 Linux 作業系統與架構的資訊，請參閱 [Which operating models and workloads does GuardDuty Runtime Monitoring support](#)。
- 您可以使用 Systems Manager 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[設定 EC2 執行個體的 Systems Manager](#)。

您可以在 GuardDuty 中啟用執行時期監控。如需如何啟用此功能的資訊，請參閱 Amazon GuardDuty User Guide 中的 [Enabling Runtime Monitoring](#)。

將執行時期監控新增至 Amazon ECS 叢集

設定叢集的執行時期監控，然後在 EC2 容器執行個體上安裝 GuardDuty 安全代理程式。

先決條件

1. 啟用執行時期監控。如需詳細資訊，請參閱[啟用 Amazon ECS 的執行時期監控](#)。
2. 您可以使用預先定義的標籤控制叢集的執行時期監控。如果存取政策依據標籤限制存取權，您必須明確授予 IAM 使用者標記叢集的權限。如需詳細資訊，請參閱《[IAM 使用者指南](#)》中的 [IAM 教學課程：定義根據標籤存取 AWS 資源的許可](#)。

程序

請執行下列操作，將執行時期監控新增至叢集。

1. 為每個叢集 VPC 建立用於 GuardDuty 的 VPC 端點。如需詳細資訊，請參閱 GuardDuty User Guide 中的 [Creating Amazon VPC endpoint manually](#)。
2. 設定 EC2 容器執行個體。
 - a. 將叢集中 EC2 容器執行個體上的 Amazon ECS 代理程式更新為 1.77 版或更新版本。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 GuardDuty 安全代理程式。如需詳細資訊，請參閱 GuardDuty User Guide 中的 [Managing the security agent on an Amazon EC2 instance manually](#)。

所有新的與現有的任務及部署都會立即受到保護，因為 GuardDuty 安全代理程式會在 EC2 容器執行個體上作為程序執行。

3. 使用 Amazon ECS 主控台或將叢集上的 AWS CLI GuardDutyManaged 標籤金鑰設定為 true。如需詳細資訊，請參閱[更新叢集](#)或[透過 CLI 或 API 使用標籤](#)。請對標籤使用以下值。

Note

鍵與值區分大小寫，且必須與字串完全相符。

索引鍵 = GuardDutyManaged，值 = true

將執行時期監控新增至現有的 Amazon ECS 任務

當您啟用執行時期監控時，叢集中所有新獨立任務與新服務部署都會自動受到保護。為了維持不可變性限制，現有任務不會受到影響。

先決條件

- 啟用執行時期監控。如需詳細資訊，請參閱[啟用 Amazon ECS 的執行時期監控](#)。

程序

- 若要立即保護任務，您需要執行下列任一動作：
 - 對於獨立任務，請先停止任務，然後再啟動任務。如需詳細資訊，請參閱[停止 Amazon ECS 任務及將應用程式作為 Amazon ECS 任務執行](#)。
 - 對於屬於服務的任務，請使用「強制執行新部署」選項來更新服務。如需詳細資訊，請參閱[更新 Amazon ECS 服務](#)。

從 Amazon ECS 叢集中移除執行時期監控

您可以從叢集中移除執行時期監控。此操作會導致 GuardDuty 停止監控叢集中的所有資源。

從叢集中移除執行時期監控

1. 使用 Amazon ECS 主控台或將叢集上的 GuardDutyManaged 標籤金鑰 AWS CLI 設定為 false。如需詳細資訊，請參閱[更新叢集](#)或[透過 CLI 或 API 使用標籤](#)。

Note

鍵與值區分大小寫，且必須與字串完全相符。

索引鍵 = GuardDutyManaged，值 = false

2. 解除安裝叢集中 EC2 容器執行個體上的 GuardDuty 安全代理程式。

如需詳細資訊，請參閱 GuardDuty User Guide 中的 [Uninstalling the security agent manually](#)。

3. 刪除每個叢集 VPC 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 User Guide 中的 [Delete an interface endpoint](#)。

更新 Amazon ECS 容器執行個體上的 GuardDuty 安全代理程式

如需有關如何更新 EC2 容器執行個體上的 GuardDuty 安全代理程式的資訊，請參閱 Amazon GuardDuty User Guide 中的 [Updating GuardDuty security agent](#)。

從帳戶移除 Amazon ECS 的執行時期監控

若不想再使用執行時期監控，可在 GuardDuty 中停用此功能。如需有關如何停用此功能的資訊，請參閱 Amazon GuardDuty User Guide 中的 [Enabling Runtime Monitoring](#)。

從所有叢集中移除執行時期監控。如需詳細資訊，請參閱[從 Amazon ECS 叢集中移除執行時期監控](#)。

執行時期監控疑難排解

您可能需要對執行時期監控進行疑難排解，或驗證執行時期監控已啟用且在任務與容器上執行。

主題

- [如何判斷帳戶上的執行時期監控是否處於作用中狀態？](#)
- [如何判斷叢集上的執行時期監控是否處於作用中狀態？](#)
- [如何判斷 GuardDuty 安全代理程式是否正在 Fargate 任務上執行？](#)
- [如何判斷 GuardDuty 安全代理程式是否正在 EC2 容器執行個體上執行？](#)
- [當叢集上執行的任務沒有任務執行角色時會如何？](#)
- [如何判斷我是否具備為叢集新增執行時期監控標籤的正確許可？](#)
- [無法連線至 Amazon ECR 時會如何？](#)
- [啟用執行時期監控後如何解決 Fargate 任務出現記憶體不足的錯誤？](#)

如何判斷帳戶上的執行時期監控是否處於作用中狀態？

在 Amazon ECS 主控台中，帳戶設定頁面上顯示此資訊。

您也可以使用 `list-account-settings` 選項執行 `effective-settings`。

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

Output

名稱設定為 `guardDutyActivate` 且值設定為 `on`，表示帳戶已完成設定。您必須與 GuardDuty 管理員確認管理是自動進行還是手動進行。

```
{
  "setting": {
    "name": "guardDutyActivate",
```

```
    "value": "enabled",
    "principalArn": "arn:aws:iam::123456789012:root",
    "type": "aws-managed"
  }
}
```

如何判斷叢集上的執行時期監控是否處於作用中狀態？

您可以在 GuardDuty 主控台中檢閱涵蓋範圍統計量。其中包含與您帳戶或會員帳戶相關聯的 Amazon ECS 叢集資源的資訊，即運作狀態良好的叢集在所選 AWS 區域中所有叢集中所佔的百分比。這包括使用 Fargate 與 EC2 的叢集覆蓋率。如需詳細資訊，請參閱 Amazon GuardDuty User Guide 中的 [Reviewing coverage statistics](#)。

如何判斷 GuardDuty 安全代理程式是否正在 Fargate 任務上執行？

GuardDuty 安全代理程式會作為 Fargate 任務的邊車容器執行。

在 Amazon ECS 主控台中，邊車容器顯示在任務詳細資訊頁面的容器下。

您可以執行 `describe-tasks` 並尋找名稱設定為 `aws-gd-agent` 且 `lastStatus` 設定為 `RUNNING` 的容器。

下列範例顯示了任務 `aws:ecs:us-east-1:123456789012:task/0b69d5c0-d655-4695-98cd-5d2d5EXAMPLE` 預設叢集的輸出。

```
aws ecs describe-tasks --cluster default --tasks aws:ecs:us-
east-1:123456789012:task/0b69d5c0-d655-4695-98cd-5d2d5EXAMPLE
```

Output

名為 `gd-agent` 的容器處於 `RUNNING` 狀態。

```
"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 安全代理程式是否正在 EC2 容器執行個體上執行？

執行以下命令來檢視狀態：

```
sudo systemctl status amazon-guardduty-agent
```

日誌檔案位於下列位置：

```
/var/log/amzn-guardduty-agent
```

當叢集上執行的任務沒有任務執行角色時會如何？

對於 Fargate 任務，任務會在沒有 GuardDuty 安全代理程式邊車容器的情況下啟動。GuardDuty 儀表板會在涵蓋範圍統計量儀表板中顯示任務缺少保護的狀態。

如何判斷我是否具備為叢集新增執行時期監控標籤的正確許可？

若要標記叢集，您必須同時具備對 `ecs:TagResource` 與 `CreateCluster` 執行 `UpdateCluster` 動作的許可。

以下是範例政策的程式碼片段。

```
{  
  "Statement": [  
    {  
      "Effect": "Allow",  
      "Action": [  
        "ecs:TagResource"  
      ],  
      "Resource": "*",  
      "Condition": {  
        "StringEquals": {  
          "ecs:CreateAction": "CreateCluster",  
          "ecs:CreateAction": "UpdateCluster",  
        }  
      }  
    }  
  ]  
}
```



```
]
}
```

無法連線至 Amazon ECR 時會如何？

對於 Fargate 任務，任務會在沒有 GuardDuty 安全代理程式邊車容器的情況下啟動。GuardDuty 儀表板會在涵蓋範圍統計量儀表板中顯示任務缺少保護的狀態。

啟用執行時期監控後如何解決 Fargate 任務出現記憶體不足的錯誤？

GuardDuty 安全代理程式是一種輕量型程序。不過，程序仍會根據工作負載的大小耗用資源。我們建議使用容器資源追蹤工具 (例如 Amazon CloudWatch Container Insights)，在叢集中分階段部署 GuardDuty。這些工具可協助您探索應用程式 GuardDuty 安全代理程式的耗用設定檔。然後，您可以視需要調整 Fargate 任務大小，避免可能出現的記憶體不足狀況。

使用 ECS Exec 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。在生產環境中，則可藉此取得緊急存取權，直接進入容器進行問題偵錯。

您可以從 Amazon ECS API、AWS Command Line Interface (AWS CLI)、AWS SDKs 或 AWS Copilot CLI，使用 ECS Exec 在執行中的 Linux 或 Windows 容器中執行命令。如需使用 ECS Exec 的詳細資訊，以及使用 AWS Copilot CLI 的影片逐步解說，請參閱 [Copilot GitHub 文件](#)。

您也可以使用 ECS Exec 來維護更嚴格的存取控制政策。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日誌，您可以使用 ECS Exec 檢視執行的任務，並可以使用 CloudTrail 稽核存取容器的人員。

考量事項

使用 ECS Exec 時，請注意下列事項：

- 在 Systems Manager 不支援的作業系統上執行時，ECS Exec 可能無法如預期般運作。如需有關支援的作業系統的詳細資訊，請參閱《AWS Systems Manager 使用者指南》中的 [作業系統類型](#)。
- 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 與 Windows 容器 (Amazon ECS Anywhere)
- AWS Fargate 上的 Linux 和 Windows 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
- 如果為任務設定了 HTTP Proxy，請將 NO_PROXY 環境變數設定為 "NO_PROXY=169.254.169.254,169.254.170.2"，使 EC2 執行個體中繼資料與 IAM 角色流量繞過代理伺服器。如果您未設定 NO_PROXY 環境變數，則從容器內的中繼資料端點擷取執行個體中繼資料或 IAM 角色憑證時可能會失敗。將 NO_PROXY 環境變數設定為建議值可篩選中繼資料與 IAM 流量，因此對 169.254.169.254 and 169.254.170.2 的請求不會通過 HTTP 代理。
- ECS Exec 與 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 Systems Manager VPC 端點的詳細資訊，請參閱 AWS Systems Manager 《使用者指南》中的[使用 AWS PrivateLink 為 Session Manager 設定 VPC 端點](#)。
 - 如果您將界面 Amazon VPC 端點與 Amazon ECS 搭配使用，且使用 AWS KMS key 進行加密，則必須為 建立界面 Amazon VPC 端點 AWS KMS key。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](#)。
 - 當您有任務在 Amazon EC2 執行個體上執行時，請使用 awsvpc 聯網模式。如果您沒有網際網路存取權 (例如未設定為使用 NAT 閘道)，則必須為 Systems Manager Session Manager (ssmmessages) 建立介面 Amazon VPC 端點。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 Systems Manager VPC 端點的詳細資訊，請參閱 AWS Systems Manager 《使用者指南》中的[使用 AWS PrivateLink 為 Session Manager 設定 VPC 端點](#)。
- Amazon ECS Exec 不支援以純 IPv6 組態執行的任務。如需有關以純 IPv6 組態執行任務的詳細資訊，請參閱[Fargate 的 Amazon ECS 任務聯網選項與 EC2 的 Amazon ECS 任務聯網選項](#)。
- ECS Exec 與 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 代理程式要求容器檔案系統必須具寫入權限，方能建立必要的目錄與檔案。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](#)。
- 下列功能會以邊車容器的形式執行。因此，您必須指定要在其上執行命令的容器名稱。
 - 執行時期監控
 - 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 Exec 會占用部分 CPU 與記憶體。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，或是 AWS CLI 版本 2.3.6 或更新版本。如需如何更新的資訊 AWS CLI，請參閱《使用者指南第 2 版》中的[安裝或更新最新版本的 AWS CLI](#)。AWS Command Line Interface
- 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。
- 如果您使用 `run-task` 在採用受管擴展與非同步置放 (啟動沒有執行個體的任務) 的叢集上啟動任務，則無法使用 ECS Exec。
- 您無法針對 Microsoft Nano Server 容器執行 ECS Exec。

Architecture

ECS Exec 使用 AWS Systems Manager (SSM) Session Manager 與執行中的容器建立連線，並使用 AWS Identity and Access Management (IAM) 政策控制對執行中容器中命令的存取。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 代理程式負責啟動容器內的 SSM 核心代理程式以及您的應用程式程式碼。如需詳細資訊，請參閱 [Systems Manager 工作階段管理員](#)。

您可以使用 `ExecuteCommand` 事件稽核存取容器的使用者，AWS CloudTrail 並將每個命令（及其輸出）記錄到 Amazon S3 或 Amazon CloudWatch Logs。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。

設定 ECS Exec

若要使用 ECS Exec，必須先為任務與服務啟用該功能，然後才能在容器中執行命令。

可選任務定義變更

如果您將任務定義參數 `initProcessEnabled` 設定為 `true`，則會啟動容器內的 `init` 程序。這會移除所有偵測到的殭屍 SSM 代理程式子程序。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 AWS CLI 命令時指定 `--enable-execute-command` 旗標，以開啟服務和獨立任務的 ECS Exec 功能：[create-service](#)、[start-task](#)、[update-service](#) 或 [run-task](#)。

例如，若您執行以下命令，則會為在 Fargate 上執行的新建立服務啟用 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=ENI
  \
  --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

使用 ECS Exec 記錄

您可以設定 ECS Exec 工作階段的記錄，擷取命令及其輸出，以達稽核與故障診斷之目的。

在任務與服務中啟用記錄

Important

如需了解有關 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 CloudWatch 監控 Amazon ECS](#)。

Amazon ECS 為透過 ECS Exec 執行的命令，提供了預設的記錄組態。依預設，日誌會透過任務定義中設定的 `awslogs` 日誌驅動程式傳送至 CloudWatch Logs。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。容器映像需要安裝 `script` 與 `cat`，以便將命令日誌正確上傳至 Amazon S3 或 CloudWatch Logs。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，然後將輸出記錄至名為 `cloudwatch-log-group-name` 的 CloudWatch Logs LogGroup 和名為 `s3-bucket-name` 的 Amazon S3 儲存貯體。

當您將 `CloudWatchEncryptionEnabled` 選項設定為 `true` 時，必須使用 AWS KMS 客戶受管金鑰來加密日誌群組 `true`。如需有關如何加密日誌群組的資訊，請參閱 Amazon CloudWatch Logs 《使用者指南》中的 [使用在 CloudWatch Logs 中加密日誌資料 AWS Key Management Service](#)。

```
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`：已關閉記錄。
- `DEFAULT`：日誌會傳送至設定的 `awslogs` 驅動程式。如果未設定驅動程式，則不會儲存日誌。
- `OVERRIDE`：日誌會傳送至提供的 Amazon CloudWatch Logs LogGroup 和/或 Amazon S3 儲存貯體。

Amazon CloudWatch Logs 或 Amazon S3 日誌記錄所需的 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。根據您將日誌導向 Amazon CloudWatch Logs 或 Amazon S3 的不同，所需許可也會有所差異。

Amazon CloudWatch Logs

下列政策範例會新增必要 Amazon CloudWatch Logs 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:us-east-1:111122223333: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，本機用戶端與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](#)。

您可以將以下內嵌政策新增至任務 IAM 角色，該角色需要 AWS KMS 許可。如需詳細資訊，請參閱[ECS Exec 許可](#)。

JSON

```

{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "kms:Decrypt"
            ],
            "Resource": "arn:aws:kms:us-east-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
        }
    ]
}

```

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)。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "kms:GenerateDataKey"
      ],
      "Resource": "arn:aws:kms:us-east-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
    }
  ]
}
```

使用 IAM 政策限制 ECS Exec 的存取

您可以使用下列一或多個 IAM 政策條件索引鍵，限制使用者對執行命令 API 動作的存取：

- `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。

JSON

```
{
  "Version": "2012-10-17",
```

```

    "Statement": [
      {
        "Effect": "Allow",
        "Action": [
          "ecs:ExecuteCommand",
          "ecs:DescribeTasks"
        ],
        "Resource": [
          "arn:aws:ecs:us-east-1:111122223333:task/cluster-name/*",
          "arn:aws:ecs:us-east-1:111122223333:cluster/cluster-name"
        ],
        "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。

JSON

```

{
  "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。

JSON

```
{
  "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。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": "ssm:StartSession",
      "Resource": [
        "arn:aws:ecs:us-east-1:111122223333:task/cluster-name/*"
      ]
    }
  ]
}
```

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。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": "ssm:StartSession",
      "Resource": "arn:aws:ecs:*:*:task/*",
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/Task-Tag-Key": "Exec-Task"
        }
      }
    }
  ]
}
```

```
}  
}  
}  
]  
}
```

ECS Exec 疑難排解

如需其他疑難排解協助，請參閱 [Exec 問題疑難排解](#)。

使用 ECS Exec 執行命令

您可以使用 Amazon ECS Exec 收集與容器相關的診斷資訊，並對整個容器生命週期中遇到的錯誤進行故障診斷。

先決條件

開始使用 ECS Exec 之前，請確定您已完成下列動作：

- 檢閱考量事項。如需詳細資訊，請參閱 [考量事項](#)
- 為任務與服務設定 ECS Exec。如需詳細資訊，請參閱 [設定 ECS Exec](#)
- 安裝和設定 AWS CLI。如需詳細資訊，請參閱 [Get started with the AWS CLI](#)。
- 安裝的 Session Manager 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](#)。
- 設定具備適當許可的任務角色。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](#)。
- 驗證版本需求。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，則必須使用平台版本 1.4.0 或更新版本 (Linux) 或 1.0.0 (Windows)。如需詳細資訊，請參閱 [AWS Fargate 平台版本](#)。

透過主控台操作服務任務

您可以透過主控台使用 ECS Exec 執行指令。

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「叢集詳細資訊」頁面上的服務區段中，選擇所需服務。

服務詳細資訊頁面隨即顯示。

4. 在服務詳細資訊頁面上，選擇任務。然後，選擇所需任務。
5. 在容器下選擇要在其中使用 ECS Exec 的容器。
6. 若要執行命令，請執行下列任一操作：

- 選擇連線。

CloudShell 工作階段會顯示您可以在其中執行命令的位置。

- 選擇箭頭，再選擇複製 AWS CLI 命令。

然後，您便可以在本機執行命令。

預期的結果

如果連線成功，您應該會看到來自容器的互動式 Shell 提示。您現在可以直接在容器環境中執行命令。若要退出工作階段，請選擇結束工作階段。

使用主控台操作獨立任務

您可以透過主控台使用 ECS Exec 執行指令。

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「叢集詳細資訊」頁面上的任務區段中，選擇所需任務。

任務詳細資訊頁面隨即顯示。

4. 在容器下選擇要在其中使用 ECS Exec 的容器。
5. 若要執行命令，請執行下列任一操作：

- 選擇連線。

CloudShell 工作階段會顯示您可以在其中執行命令的位置。

- 選擇箭頭，再選擇複製 AWS CLI 命令。

然後，您便可以在本機執行命令。

預期的結果

如果連線成功，您應該會看到來自容器的互動式 Shell 提示。您現在可以直接在容器環境中執行命令。若要退出工作階段，請選擇結束工作階段。

使用命令 Shell

您可透過命令 Shell 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。

若任務 ID 為 `task-id`，下面的命令會針對名為 `container-name` 的容器執行交互式 `/bin/sh` 命令。

`task-id` 是任務的 Amazon Resource Name (ARN)。

```
aws ecs execute-command --cluster cluster-name \  
  --task task-id \  
  --container container-name \  
  --interactive \  
  --command "/bin/sh"
```

預期的結果

如果命令執行成功，您應該會看到來自容器的互動式 Shell 提示。您現在可以直接在容器環境中執行命令。若要退出工作階段，請輸入 `exit` 或按 `Ctrl+D`。

AWS Compute Optimizer Amazon ECS 的建議

AWS Compute Optimizer 會針對 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 ?](#)。

上的 Amazon ECS 服務的任務和容器大小建議 AWS Fargate

AWS Compute Optimizer 會在 上產生 Amazon ECS 服務的建議 AWS Fargate。AWS Compute Optimizer 建議任務 CPU 和任務記憶體大小，以及容器 CPU、容器記憶體和容器記憶體保留大小。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 MCP 伺服器

Amazon ECS MCP 伺服器是一項全受管服務，可為開發和操作提供 AI 驅動的體驗。[模型內容通訊協定 \(MCP\)](#) 提供標準化界面，可讓您的 AI 代理器和應用程式具備 Amazon ECS 工作負載的即時情境知識，讓更準確、具備情境感知的回應和採用 AI 技術的工作流程能夠檢查、疑難排解和最佳化 Amazon ECS 工作負載。

Note

Amazon ECS MCP 伺服器目前為預覽版本，可能會有所變更。

概要

Amazon ECS MCP 伺服器可以輕鬆與任何 MCP 相容 AI 編碼助理整合，例如 [Kiro](#)，或 [Claude Code](#)、[Cursor](#) 和 [Cline](#) 等第三方工具，以增強您的開發工作流程。

在部署期間，它可提供服務部署和任務運作狀態的即時可見性。對於開發和操作，伺服器提供監控容器運作狀態、分析服務事件和調查部署問題的高階工作流程，以簡化 Amazon ECS 管理。對於偵錯和故障診斷，伺服器會透過分析任務失敗、容器日誌和網路組態的全方位診斷工具來加速問題解決。這些功能可透過自然語言互動存取，讓您能夠更直覺且有效率地執行複雜的容器操作。

Amazon ECS MCP 伺服器提供數種工具，您可以用來：

- 監控部署和服務

檢查部署狀態、追蹤服務事件，以及即時監控容器運作狀態。

- 對容器問題進行故障診斷

分析任務失敗、調查映像提取錯誤，以及使用可自訂的時段檢查容器日誌。

- 檢查網路組態

檢閱 Amazon ECS 服務的 VPC、子網路和安全群組組態。

- 解決資源相依性

識別封鎖程式，防止刪除任務定義並了解資源關係。

全受管 Amazon ECS MCP 伺服器託管在 AWS 雲端，無需本機安裝和維護。它提供企業級功能，例如自動更新和修補、透過 IAM 整合的集中式安全性、透過 CloudTrail 的全方位稽核記錄，以及證實的可擴展性、可靠性和支援 AWS。

開始使用

若要開始使用，請參閱[Amazon ECS MCP 伺服器入門](#)。

Amazon ECS MCP 伺服器入門

本指南會逐步引導您設定 AI 程式碼助理並使用 Amazon ECS MCP Server 的步驟。您將了解如何設定環境、連線至伺服器，以及透過自然語言互動開始管理 Amazon ECS 叢集。

Note

Amazon ECS MCP 伺服器目前為預覽版本，可能會有所變更。

先決條件

開始前，請確保您具備以下條件：

- [建立可存取 Amazon ECS AWS 的帳戶](#)
- [AWS CLI 使用登入資料安裝和設定](#)
- [已安裝 Python 3.10+](#)
- [已安裝的 uv](#)

設定

確認 先決條件

檢查您的 Python 版本是否為 3.10 或更新版本

```
python3 --version
```

檢查 uv 安裝

```
uv --version
```

驗證 AWS CLI 組態

```
aws configure list
```

設定 IAM 許可

您需要 [IAM](#) 許可才能對 AWS 服務提出唯讀請求，並與 MCP 伺服器互動。您可以利用[AWS 受管政策](#)或單一自訂政策進行更精細的控制。

選項 1：合併受管和自訂政策

1. 連接 AWS 受管政策 `ReadOnlyAccess` 以唯讀存取所有 AWS 服務
2. 建立並連接 MCP 許可的其他自訂政策（請參閱下面的 MCP 許可 JSON）

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "MCPServerAccess",
      "Effect": "Allow",
      "Action": [
        "ecs-mcp:InvokeReadOnlyTools",
        "ecs-mcp:UseMcp"
      ],
      "Resource": "*"
    }
  ]
}
```

選項 2：單一自訂政策（最精細的控制）

或者，您可以建立並連接包含 AWS 服務許可和 MCP 許可的單一自訂 JSON 政策：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "MCPServerAccess",
```

```

    "Effect": "Allow",
    "Action": [
        "ecs-mcp:InvokeReadOnlyTools",
        "ecs-mcp:UseMcp"
    ],
    "Resource": "*"
},
{
    "Sid": "ECSReadOnlyAccess",
    "Effect": "Allow",
    "Action": [
        "ecs:ListAccountSettings",
        "ecs:ListClusters",
        "ecs:ListContainerInstances",
        "ecs:ListTaskDefinitions",
        "ecs:ListServices",
        "ecs:ListServiceDeployments",
        "ecs:ListTasks",
        "ecs:DescribeClusters",
        "ecs:DescribeCapacityProviders",
        "ecs:DescribeContainerInstances",
        "ecs:DescribeTaskDefinition",
        "ecs:DescribeServices",
        "ecs:DescribeServiceDeployments",
        "ecs:DescribeServiceRevisions",
        "ecs:DescribeTaskSets",
        "ecs:DescribeTasks"
    ],
    "Resource": "*"
},
{
    "Sid": "CloudWatchLogsReadOnlyAccess",
    "Effect": "Allow",
    "Action": [
        "logs:DescribeLogGroups",
        "logs:DescribeLogStreams",
        "logs:GetLogEvents",
        "logs:FilterLogEvents"
    ],
    "Resource": "*"
},
{
    "Sid": "ELBReadOnlyAccess",
    "Effect": "Allow",

```

```

    "Action": [
      "elasticloadbalancing:DescribeLoadBalancers",
      "elasticloadbalancing:DescribeTargetGroups",
      "elasticloadbalancing:DescribeListeners"
    ],
    "Resource": "*"
  },
  {
    "Sid": "EC2ReadOnlyAccess",
    "Effect": "Allow",
    "Action": [
      "ec2:DescribeVpcs",
      "ec2:DescribeSubnets",
      "ec2:DescribeSecurityGroups",
      "ec2:DescribeRouteTables",
      "ec2:DescribeNetworkInterfaces"
    ],
    "Resource": "*"
  },
  {
    "Sid": "ECRReadOnlyAccess",
    "Effect": "Allow",
    "Action": [
      "ecr:DescribeRepositories",
      "ecr:DescribeImages"
    ],
    "Resource": "*"
  }
]
}

```

選擇並設定 AI 助理

安裝 MCP 相容 AI 助理或任何 MCP 相容工具。例如，您可以使用 [Kiro](#)、[Cline](#)、[游標](#) 或 [Claude Code](#)。然後，將 AI 程式碼助理設定為透過 MCP Proxy 使用 Amazon ECS MCP 伺服器 AWS，這是安全、已驗證存取 Amazon ECS MCP 伺服器的必要項目。代理充當用戶端橋接器，使用本機 AWS 登入資料處理 AWS SigV4 身分驗證。以下範例使用 Kiro CLI。請遵循[此連結](#)，進一步了解如何在 Kiro 中設定 MCP。

尋找 MCP 組態檔案

- macOS/Linux :

```
~/.kiro/settings/mcp.json
```

- Windows :

```
%USERPROFILE%\kiro\settings\mcp.json
```

如果組態檔案不存在，請建立組態檔案。

新增 MCP 伺服器組態

請務必將區域 ({region}) 預留位置取代為您想要的區域 (例如 us-west-2)。如需區域的完整清單[the section called “AWS Fargate 上的 Linux 容器”](#)，請參閱。也請務必將{profile}預留位置取代為您的[AWS CLI 設定檔名稱](#)，例如 default。

對於 Mac/Linux :

```
{
  "mcpServers": {
    "ecs-mcp": {
      "disabled": false,
      "type": "stdio",
      "command": "uvx",
      "args": [
        "mcp-proxy-for-aws@latest",
        "https://ecs-mcp.{region}.api.aws/mcp",
        "--service",
        "ecs-mcp",
        "--profile",
        "{profile}",
        "--region",
        "{region}"
      ]
    }
  }
}
```

用於 Windows :

```
{
  "mcpServers": {
```

```
"ecs-mcp": {
  "disabled": false,
  "type": "stdio",
  "command": "uvx",
  "args": [
    "--from",
    "mcp-proxy-for-aws@latest",
    "mcp-proxy-for-aws.exe",
    "https://ecs-mcp.{region}.api.aws/mcp",
    "--service",
    "ecs-mcp",
    "--profile",
    "{profile}",
    "--region",
    "{region}"
  ]
}
```

驗證組態

重新啟動 Kiro CLI、kiro-cli，確認 MCP 伺服器已載入，/mcp並檢查可用的工具 /tools。

驗證您的設定

測試連線

向您的 AI 助理詢問簡單的問題，以驗證連線：

```
List all ECS clusters in my AWS account
```

您應該會看到 Amazon ECS 叢集的清單。

與使用 Amazon ECS MCP 伺服器的 AI 助理交談

範例 1：監控部署

```
Check deployment status for my web-service in production-cluster
Show me the ALB URL for my deployed service
Get service events for the last hour
```

範例 2：調查容器運作狀態


```
Show me all tasks that failed in the last 2 hours
Why are my containers failing health checks?
Display container logs for my api-service
```

範例 3：故障診斷

```
Analyze task failures in my production cluster
Check for image pull errors in the last 30 minutes
Why is my task definition stuck in DELETE_IN_PROGRESS state?
```

範例 4：檢查組態

```
Show me the network configuration for my web-service
What security groups are attached to my service?
List all VPC and subnet details for my ECS service
```

常見組態和最佳實務

多個 AWS 設定檔

如果您使用多個 AWS 帳戶，請建立個別的 MCP 伺服器組態。

對於 Mac/Linux：

```
{
  "mcpServers": {
    "ecs-mcp-prod": {
      "disabled": false,
      "type": "stdio",
      "command": "uvx",
      "args": [
        "mcp-proxy-for-aws@latest",
        "https://ecs-mcp.{region}.api.aws/mcp",
        "--service",
        "ecs-mcp",
        "--profile",
        "production",
        "--region",
        "us-west-2"
      ]
    }
  },
}
```

```
"ecs-mcp-dev": {
  "disabled": false,
  "type": "stdio",
  "command": "uvx",
  "args": [
    "mcp-proxy-for-aws@latest",
    "https://ecs-mcp.{region}.api.aws/mcp",
    "--service",
    "ecs-mcp",
    "--profile",
    "development",
    "--region",
    "us-east-1"
  ]
}
```

安全最佳實務

請勿透過允許的輸入機制傳遞秘密或敏感資訊：

- 請勿在任何組態檔案中包含秘密或登入資料
- 不要在提示中直接將敏感資訊傳遞給模型
- 請勿在任務定義或服務組態中包含秘密
- 避免在應用程式日誌中記錄敏感資訊
- 使用 `SECRET` 或 `PARAMETER` 參數存放區來存放敏感資訊

工具組態

如需工具和組態的完整清單，請參閱 [Amazon ECS MCP 伺服器工具組態](#)。

Amazon ECS MCP 伺服器工具組態

本指南顯示全受管 Amazon ECS MCP Server 可用的所有組態。

Note

Amazon ECS MCP 伺服器目前為預覽版本，可能會有所變更。

環境變數

AWS_PROFILE (選用)

AWS 要使用的登入資料設定檔名稱；可由`--profile`命令列引數覆寫。

範例：`export AWS_PROFILE=dev`

AWS_REGION (選用)

AWS SigV4 簽署的區域；`us-west-2`如果未設定，則預設為。

範例：`export AWS_REGION=us-west-2`

引數

SigV4 MCP 端點 URL (必要)

要連線的 MCP 端點 URL。

範例：`https://ecs-mcp.us-west-2.api.aws/mcp`

--service (選用)

AWS SigV4 簽署的服務名稱；若未提供，會自動從端點主機名稱偵測。

範例：`--service ecs-mcp`

--profile (選用)

AWS 要使用的登入資料設定檔。如果未指定，則預設為`AWS_PROFILE`環境變數。

範例：`--profile dev`

--區域

AWS 要使用的區域。如果未設定，則使用`AWS_REGION`環境變數，預設為`us-east-1`。

範例：`--region us-west-2`

--唯讀 (選用)

停用可能需要寫入許可的工具（不需要寫入許可的工具會以標註`readOnlyHint=true`）。預設會啟用所有工具。

範例：--read-only

如需更多組態選項，請參閱[組態參數](#)。

工具

Amazon ECS MCP 伺服器會公開下列 [MCP 工具](#)。

唯讀工具

get_deployment_status

此工具會檢查特定 Amazon ECS 叢集和服務的 Amazon ECS 部署狀態。

- 必要的 IAM 動作：
 - `ecs:DescribeServices`
 - `elasticloadbalancing:DescribeTargetGroups`
 - `elasticloadbalancing:DescribeLoadBalancers`
- 必要參數：
 - `cluster_name` (字串) : Amazon ECS 叢集名稱
 - `service_name` (字串) : Amazon ECS 服務名稱
- 回應：工具會傳回與部署相關聯的部署狀態和詳細資訊。

fetch_service_events

此工具會擷取具有可自訂時段之診斷的 Amazon ECS 服務事件。

- 必要的 IAM 動作：
 - `ecs:DescribeServices`
- 必要參數：
 - `cluster_name` (字串) : Amazon ECS 叢集名稱
 - `service_name` (字串) : Amazon ECS 服務名稱
- 選用的參數：
 - `time_window` (整數) : 您可以用秒為單位提及時間範圍 (預設為 1 小時或 3600 秒)
 - `start_time` (字串) : ISO 格式的自訂開始時間
 - `end_time` (字串) : ISO 格式的自訂結束時間
- 回應：工具會傳回具有摘要之服務的事件清單。

fetch_task_failures

此工具可協助您使用摘要擷取和分析 Amazon ECS 任務失敗。使用此工具，您可以識別任務失敗中的任何模式。

- 必要的 IAM 動作：
 - `ecs:ListTasks`
 - `ecs:DescribeTasks`
- 必要參數：
 - `cluster_name` (字串)：任務所在的 Amazon ECS 叢集名稱
- 選用的參數：
 - `service_name` (字串)：您可以依特定服務篩選
 - `time_window` (整數)：您可以用秒為單位提及時間範圍 (預設為 1 小時或 3600 秒)
 - `start_time` (字串)：ISO 格式的自訂開始時間
 - `end_time` (字串)：ISO 格式的自訂結束時間
- 回應：工具會傳回 `TaskFailureResult`，其中包含失敗的任務和摘要

fetch_task_logs

此工具會使用彈性的時間範圍選項擷取 Amazon ECS 任務的 CloudWatch 日誌。您可以使用此工具來疑難排解執行時間問題。

- 必要的 IAM 動作：
 - `ecs:DescribeServices`
 - `ecs:DescribeTaskDefinition`
 - `logs:FilterLogEvents`
- 必要參數：
 - `cluster_name` (字串)：Amazon ECS 叢集名稱
 - `service_name` (字串)：Amazon ECS 服務名稱
- 選用的參數：
 - `time_window` (整數)：您可以用秒為單位提及時間範圍 (預設為 1 小時或 3600 秒)
 - `start_time` (字串)：ISO 格式的自訂開始時間
 - `end_time` (字串)：ISO 格式的自訂結束時間
 - `log_group_name` (字串)：指定 CloudWatch 日誌群組
- 回應：工具會使用日誌項目傳回 `TaskLogsResult`

get_task_definition_deletion_blockers

此工具可識別可防止任務定義刪除的相依性。當您執行清除操作並了解阻止 Amazon ECS 資源刪除的原因時，此工具是適當的。

- 必要的 IAM 動作：
 - `ecs:ListClusters`
 - `ecs:ListServices`
 - `ecs:DescribeServices`
 - `ecs:ListTasks`
 - `ecs:DescribeTasks`
- 必要參數：
 - `task_definition_arn` (字串)：要分析的任務定義 Amazon Resource Name (ARN)
- 回應：工具會傳回封鎖程式和刪除狀態

detect_image_pull_failures

此工具可協助您偵測和分類容器映像提取失敗。

- 必要的 IAM 動作：
 - `ecs:ListTasks`
 - `ecs:DescribeTasks`
- 必要參數：
 - `cluster_name` (字串)：Amazon ECS 叢集名稱
- 選用的參數：
 - `service_name` (字串)：您可以依特定服務篩選
 - `time_window` (整數)：您可以用秒為單位提及時間範圍 (預設為 1 小時或 3600 秒)
 - `start_time` (字串)：ISO 格式的自訂開始時間
 - `end_time` (字串)：ISO 格式的自訂結束時間
- 回應：工具會傳回失敗和摘要

fetch_network_configuration

此工具會擷取 Amazon ECS 服務網路組態詳細資訊。您可以使用此工具來了解 Amazon ECS 設定的 Amazon VPC、子網路、安全群組組態。

- 必要的 IAM 動作：
 - `ecs:DescribeServices`

- `ec2:DescribeSecurityGroups`
- `ec2:DescribeSubnets`
- 必要參數：
 - `cluster_name` (字串) : Amazon ECS 叢集名稱
 - `service_name` (字串) : Amazon ECS 服務名稱
- 回應：工具會傳回 `NetworkConfigurationResult` 的清單，其中包含網路詳細資訊

Amazon ECS 故障診斷

您可能需要對負載平衡器、任務、服務或容器執行個體進行疑難排解。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。

如需有關已停止任務的資訊，請參閱下列主題。

Action	進一步了解	
解決已停止任務錯誤。	檢視 Amazon ECS 已停止任務錯誤	
檢視已停止任務錯誤。	解決 Amazon ECS 已停止任務錯誤	
檢閱已停止任務錯誤代碼。	Amazon ECS 已停止任務錯誤訊息	
檢閱 CannotPullContainer 任務錯誤。	Amazon ECS 中的 CannotPullContainer 任務錯誤	
檢視任務 IAM 角色請求。	檢視 Amazon ECS 任務的 IAM 角色請求	
使用任務事件進行故障診斷。	主控台中的 Amazon ECS 事件擷取	

如需有關服務錯誤的資訊，請參閱下列主題。

Action	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 服務事件訊息	

Action	進一步了解	
檢閱負載平衡器問題。	對 Amazon ECS 中的服務負載平衡器進行疑難排解	
檢閱服務自動擴展問題。	Amazon ECS 中的服務自動擴展疑難排解	

如需有關任務定義錯誤的資訊，請參閱下列主題。

Action	進一步了解	
解決任務定義記憶體錯誤。	對 Amazon ECS 任務定義無效的 CPU 或記憶體錯誤進行疑難排解	

如需有關 Amazon ECS 代理程式錯誤的資訊，請參閱下列主題。

Action	進一步了解	
檢視 Amazon ECS 容器代理日誌。	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	
透過 Amazon ECS 代理程式擷取診斷詳細資訊。	透過代理程式自我檢查功能擷取 Amazon ECS 診斷詳細資訊	

如需有關 Docker 錯誤的資訊，請參閱下列主題。

Action	進一步了解	
使用 Docker 診斷。	Amazon ECS 中的 Docker 診斷	

Action	進一步了解	
啟用 Docker 偵錯模式。	在 Amazon ECS 中設定 Docker 常駐程式的詳細輸出	
對 Docker API 錯誤 500 進行疑難排解。	在 Amazon ECS 中對 Docker API error (500): devmapper 進行疑難排解	

如需有關 ECS Exec 與 Amazon ECS Anywhere 錯誤的資訊，請參閱下列主題。

Action	進一步了解	
對 ECS Exec 進行疑難排解。	對 Amazon ECS Exec 問題進行疑難排解	
對 Amazon ECS Anywhere 進行疑難排解。	對 Amazon ECS Anywhere 問題進行疑難排解	

如需有關將 Amazon EBS 磁碟區連接至 Amazon ECS 任務時所遭遇問題的資訊，請參閱下列主題：

Action	進一步了解	
Amazon EBS 磁碟區連接至 Amazon ECS 任務的疑難排解。	對 Amazon ECS 任務的 Amazon EBS 磁碟區連接問題進行疑難排解	
Amazon EBS 磁碟區連接至 Amazon ECS 任務的狀態原因。	Amazon EBS 磁碟區連接至 Amazon ECS 任務的狀態原因	

如需搭配 Amazon ECS Service Connect 使用共用 AWS Cloud Map 命名空間問題的相關資訊，請參閱下列其中一項：

Action	進一步了解	
使用共用 AWS Cloud Map 命名空間對 Amazon ECS Service Connect 進行故障診斷。	使用共用 AWS Cloud Map 命名空間對 Amazon ECS Service Connect 進行故障診斷	

如需有關限流問題的資訊，請參閱下列主題。

Action	進一步了解	
了解 Fargate 限流配額。	AWS Fargate 調節配額	
了解 Amazon ECS 限流的最佳實務。	因應 Amazon ECS 限流問題	

如需有關 API 錯誤的資訊，請參閱下列主題。

Action	進一步了解	
解決 API 錯誤。	Amazon ECS API 失敗原因	

如需採用 AI 技術的故障診斷資訊，請參閱下列內容：

Action	進一步了解	
在 主控台中使用 Amazon Q Developer 進行故障診斷。	使用 Amazon Q Developer 進行故障診斷	
使用 Amazon ECS MCP 伺服器對 AI 助理進行故障診斷。	Amazon ECS MCP 伺服器	

解決 Amazon ECS 已停止任務錯誤

當任務無法啟動時，您會在主控台與 `describe-tasks` 輸出參數 (`stoppedReason` 與 `stopCode`) 中看到錯誤訊息。

您可於主控台中檢視已停止任務，檢視時效為一小時。若要檢視已停止任務，您必須變更篩選選項。如需詳細資訊，請參閱[檢視 Amazon ECS 已停止任務錯誤](#)。

以下頁面會提供有關已停止任務的資訊。

- [了解已停止任務錯誤訊息的變更。](#)

[Amazon ECS 已停止任務錯誤訊息更新](#)

- [檢視已停止任務，取得相關錯誤成因資訊。](#)

[檢視 Amazon ECS 已停止任務錯誤](#)

- [了解已停止任務錯誤訊息和可能的錯誤原因。](#)

[Amazon ECS 已停止任務錯誤訊息](#)

- [了解如何驗證已停止任務的連線狀態並修正錯誤。](#)

[驗證 Amazon ECS 已停止任務的連線狀態](#)

Amazon ECS 已停止任務錯誤訊息更新

自 2024 年 6 月 14 日起，Amazon ECS 團隊將變更已停止任務錯誤訊息，如下表所述。`stopCode` 不會變更。如果應用程式依賴確切的錯誤訊息字串，您必須使用新字串來更新應用程式。如需相關問題的協助，請聯絡 AWS 支援。

Note

建議不要依賴自動化的錯誤訊息，因為錯誤訊息可能會有所變更。

CannotPullContainerError

舊錯誤訊息	新錯誤訊息	
CannotPullContainerError: Error response from daemon: pull access denied for <i>repository</i> , repository does not exist or may require 'docker login': denied: User: <i>roleARN</i>	• CannotPullContainerError: The task can't pull the image. Check that the role has the permissions to pull images from the registry. Error response from daemon: pull access denied for <i>repository</i> , repository does not exist or may require 'docker login': denied: User: <i>roleARN</i> is not authorized to perform: ecr:BatchGetImage on resource: <i>image</i> because no identity-based policy allows the ecr:BatchGetImage action. • CannotPullContainerError: The task can't pull the image. Check whether the image exists. Error response from daemon: pull access denied for <i>repository</i> , repository does not exist or may require 'docker login': denied: requested access to the resource is denied.	
CannotPullContainerError: Error response from daemon: Get <i>imageURI</i>: net/http:	CannotPullContainerError: The task can't pull the image. Check your network configuration. Error response from	

舊錯誤訊息	新錯誤訊息	
request canceled while waiting for connection	daemon: Get <i>image</i> : net/http: request canceled while waiting for connection (Client.Timeout exceeded while awaiting headers)	
CannotPullContainerError: ref pull has been retried 5 time(s): failed to copy: httpReadSeeker: failed open: failed to do request: Get <i>registry-uri</i> : dial tcp <ip>:<port> i/o timeout	CannotPullContainerError: The task cannot pull <i>image-uri</i> from the registry <i>registry-uri</i> . There is a connection issue between the task and the registry. Check your task network configuration. : failed to copy: httpReadSeeker: failed open: failed to do request: Get <i>registry-uri</i> : dial tcp <ip>:<port> i/o timeout	

ResourceNotFoundException

舊錯誤訊息	新錯誤訊息	
從 AWS Secrets Manager <i>##</i> 擷取秘密資料：秘密 <i>secretARN</i> : ResourceNotFoundException : Secrets Manager 找不到指定的秘密。	ResourceNotFoundException : 任務無法從中擷取具有 ARN ' <i>secretARN</i> ' 的秘密 AWS Secrets Manager。Check whether the secret exists in the specified Region. ResourceNotFoundException : 從 AWS Secrets Manager <i>##</i> 擷取秘密資料：秘密 <i>secretARN</i> : Resource	

舊錯誤訊息	新錯誤訊息	
	NotFoundException : Secrets Manager 找不到指定的秘密。	

ResourceInitializationError

舊錯誤訊息	新錯誤訊息	
ResourceInitializationError : unable to pull secrets or registry auth: execution resource retrieval failed: unable to retrieve ecr registry auth: service call has been retried 3 time(s): RequestError: send request failed caused by: Post "https://api.ecr.us-east-1.amazonaws.com/": dial tcp <ip>:<port>: i/o timeout. Please check your task network configuration.	ResourceInitializationError : unable to pull secrets or registry auth: The task cannot pull registry auth from Amazon ECR: There is a connection issue between the task and Amazon ECR. Check your task network configuration. RequestError: send request failed caused by: Post "https://api.ecr.us-east-1.amazonaws.com/": dial tcp <ip>:<port>: i/o timeout	
ResourceInitializationError : unable to pull secrets or registry auth: execution resource retrieval failed: unable to retrieve secrets from ssm: service call has been retried 5 time(s): RequestCanceled: request context canceled caused by: context deadline exceeded	ResourceInitializationError : unable to pull secrets or registry auth: unable to retrieve secrets from ssm: The task cannot pull secrets from AWS Systems Manager. There is a connection issue between the task and AWS Systems Manager Parameter Store. Check your task network configuration. RequestCanceled: request	

舊錯誤訊息	新錯誤訊息	
	context canceled caused by: context deadline exceeded	
ResourceInitializationError: failed to download env files: file download command: non empty error stream: RequestCanceled: request context canceled caused by: context deadline exceeded	ResourceInitializationError: failed to download env files: The task can't download the environment variable files from Amazon S3. There is a connection issue between the task and Amazon S3. Check your task network configuration. service call has been retried 5 time(s): RequestCanceled: request context canceled caused by: context deadline exceeded	
ResourceInitializationError : failed to validate logger args::signal:killed	ResourceInitializationError : failed to validate logger args: The task cannot find the Amazon CloudWatch log group defined in the task definition. There is a connectio n issue between the task and Amazon CloudWatch. Check your network configuration. : signal: killed	
ResourceInitializationError : unable to pull secrets or registry auth: pull command failed: : signal: killed	ResourceInitializationError : unable to pull secrets or registry auth: Check your task network configuration. : signal: killed	

檢視 Amazon ECS 已停止任務錯誤

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 管理主控台、AWS CLI、AWS SDKs、Amazon ECS API 或使用 SDKs 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。

如果任務在 Amazon EC2 或外部電腦的容器執行個體上執行，您也可以檢視容器執行時期與 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 管理主控台

您可透過主控台按下列步驟來檢查已停止任務是否有錯誤。若要檢視已停止任務，您必須變更篩選選項。

已停止任務只會在主控台中顯示 1 小時。

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。在篩選所需狀態欄位中選擇已停止。

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)。

快顯視窗隨即顯示停止原因。

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. 描述已停止任務以擷取資訊。如需詳細資訊，請參閱《AWS Command Line Interface 參考》中的 [describe-tasks](#)。

```
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 – 停止代碼會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) – 原因會提供已停止容器的額外詳細資訊。

後續步驟

檢視已停止任務，取得相關錯誤成因資訊。如需詳細資訊，請參閱[Amazon ECS 已停止任務錯誤訊息](#)。

Amazon ECS 已停止任務錯誤訊息

以下是任務意外停止時，可能會收到的錯誤訊息。

若要使用 檢查已停止的任務是否有錯誤訊息 AWS 管理主控台，請參閱 [檢視 Amazon ECS 已停止任務錯誤](#)。

Tip

您可以[Amazon ECS MCP 伺服器](#)搭配 AI 助理使用，以自然語言分析任務失敗和容器日誌。

已停止任務錯誤代碼具有與其相關聯的類別，例如 "ResourceInitializationError"。如需了解每個類別的詳細資訊，請參閱下列主題：

Category	進一步了解	
TaskFailedToStart	對 Amazon ECS TaskFailedToStart 錯誤進行疑難排解	
ResourceInitializationError	對 Amazon ECS ResourceInitializationError 錯誤進行疑難排解	
ResourceNotFoundException	對 Amazon ECS ResourceNotFoundException 錯誤進行疑難排解	
SpotInterruptionError	對 Amazon ECS SpotInterruption 錯誤進行疑難排解	
InternalError	對 Amazon ECS InternalError 錯誤進行疑難排解	
OutOfMemoryError	對 Amazon ECS OutOfMemoryError 錯誤進行疑難排解	
ContainerRuntimeError	對 Amazon ECS ContainerRuntimeError 錯誤進行疑難排解	
ContainerRuntimeTimeoutError	對 Amazon ECS ContainerRuntimeTimeoutError 錯誤進行疑難排解	

Category	進一步了解	
CannotStartContainerError	對 Amazon ECS CannotStartContainerError 錯誤進行疑難排解	
CannotStopContainerError	對 Amazon ECS CannotStopContainerError 錯誤進行疑難排解	
CannotInspectContainerError	對 Amazon ECS CannotInspectContainerError 錯誤進行疑難排解	
CannotCreateVolumeError	對 Amazon ECS CannotCreateVolumeError 錯誤進行疑難排解	
CannotPullContainer	Amazon ECS 中的 CannotPullContainer 任務錯誤	

對 Amazon ECS TaskFailedToStart 錯誤進行疑難排解

以下是部分 TaskFailedToStart 錯誤訊息以及可採取的錯誤修正動作。

若要使用 檢查已停止的任務是否有錯誤訊息 AWS 管理主控台，請參閱 [檢視 Amazon ECS 已停止任務錯誤](#)。

嘗試在子網路 **subnet-id** 中建立啟用了公有 IP 指派的網路介面時，發生意外 EC2 錯誤

當 Fargate 任務使用 awsvpc 網路模式並在具有公有 IP 位址的子網路中執行，且子網路沒有足夠的 IP 位址時，就會發生這種情況。

可在 Amazon EC2 主控台的子網路詳細資訊頁面上檢視可用 IP 位址的數量，也可透過 [describe-subnets](#) 檢視該數量。如需詳細資訊，請參閱 Amazon VPC User Guide 中的 [View your subnet](#)。

若要修正此問題，您可以建立新的子網路，在其中執行任務。

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。

選取的任務定義與選取的運算策略不相容

當您選擇啟動類型不符合叢集容量類型的任務定義時，就會發生此錯誤。您需要選取與指派給叢集的容量提供者相符的任務定義。

無法將網路介面連接至閒置的裝置索引

使用 awsvpc 聯網類型且任務沒有足夠的 CPU/記憶體時，即會發生此錯誤。首先，檢查執行個體的 CPU。如需詳細資訊，請參閱 Amazon EC2 instance types 中的 [Amazon EC2 instance type specifications](#)。取執行個體的 CPU 值，並將其乘以執行個體 ENI 數量。在任務定義中使用值 e。

代理程式

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)。

對 Amazon ECS ResourceInitializationError 錯誤進行疑難排解

以下是部分 ResourceInitialization 錯誤訊息以及可採取的錯誤修正動作。

若要使用 檢查已停止的任務是否有錯誤訊息 AWS 管理主控台，請參閱 [檢視 Amazon ECS 已停止任務錯誤](#)。

錯誤

- [The task cannot pull registry authentication from Amazon ECR. There is a connection issue between the task and Amazon ECR. Check your task network configuration.](#)
- [The task can't download the environment variable files from Amazon S3. There is a connection issue between the task and Amazon S3. Check your task network configuration.](#)
- [任務無法從 AWS Systems Manager 參數存放區提取秘密。檢查任務與 之間的網路連線 AWS Systems Manager。](#)

- [任務無法從中提取秘密 AWS Secrets Manager。There is a connection issue between the task and Secrets Manager. Check your task network configuration.](#)
- [The task can't pull the secret from Secrets Manager. The task can't retrieve the secret with ARN 'secretARN' from Secrets Manager. Check whether the secret exists in the specified Region.](#)
- [pull command failed: unable to pull secrets or registry auth Check your task network configuration.](#)
- [任務找不到任務定義中所定義的 Amazon CloudWatch 日誌群組。There is a connection issue between the task and Amazon CloudWatch. Check your network configuration.](#)
- [failed to initialize logging driver](#)
- [無法調用 EFS utils 命令來設定 EFS 磁碟區](#)

The task cannot pull registry authentication from Amazon ECR. There is a connection issue between the task and Amazon ECR. Check your task network configuration.

此錯誤表示任務無法連線至 Amazon ECR。

檢查任務與 Amazon ECR 之間的連線。如需相關資訊，請參閱[驗證 Amazon ECS 已停止任務的連線狀態](#)。

The task can't download the environment variable files from Amazon S3. There is a connection issue between the task and Amazon S3. Check your task network configuration.

當任務無法從 Amazon S3 下載環境檔案時，就會發生此錯誤。

檢查任務與 Amazon S3 VPC 端點之間的連線。如需相關資訊，請參閱[驗證 Amazon ECS 已停止任務的連線狀態](#)。

任務無法從 AWS Systems Manager 參數存放區提取秘密。檢查任務與 之間的網路連線 AWS Systems Manager。

當任務無法使用 Systems Manager 中的憑證提取任務定義中定義的映像時，就會發生此錯誤。

檢查任務與 Systems Manager VPC 端點之間的連線。如需相關資訊，請參閱[驗證 Amazon ECS 已停止任務的連線狀態](#)。

任務無法從中提取秘密 AWS Secrets Manager。There is a connection issue between the task and Secrets Manager. Check your task network configuration.

當任務無法使用 Secrets Manager 中的憑證提取任務定義中定義的映像時，就會發生此錯誤。

此錯誤表示 Systems Manager VPC 端點與任務之間存在網路連線問題。

如需有關如何驗證任務與端點之間連線的資訊，請參閱[驗證 Amazon ECS 已停止任務的連線狀態](#)。

The task can't pull the secret from Secrets Manager. The task can't retrieve the secret with ARN '**secretARN**' from Secrets Manager. Check whether the secret exists in the specified Region.

當任務無法使用 Secrets Manager 中的憑證提取任務定義中定義的映像時，就會發生此錯誤。

此問題由以下其中一個原因造成：

錯誤原因...	執行此作業...	
Secrets Manager VPC 端點與任務之間存在網路連線問題。 在錯誤訊息中看到下列任何字串時，表示存在網路問題： • dial tcp • dial udp • <ip>:<port>: i/o timeout • net/http: TLS handshake timeout • read: connection timed out • Client.Timeout exceeded while awaiting headers • net/http: request canceled while waiting for connection • signal: killed • context deadline exceeded	驗證任務與 Secrets Manager 端點之間的連線狀態。如需詳細資訊，請參閱驗證 Amazon ECS 已停止任務的連線狀態。	
任務定義中所定義的任務執行角色，沒有 Secrets Manager 所需的許可。	將 Secrets Manager 所需的許可新增至任務執行角色。如需詳細資訊，請參閱Secrets Manager 或 Systems Manager 許可。	
秘密 ARN 不存在	檢查 Secrets Manager 中是否存在 ARN。如需有關檢視	

錯誤原因...	執行此作業...
	映像的資訊，請參閱 Secrets Manager Developer Guide 中的 Find secrets in Secrets Manager 。

pull command failed: unable to pull secrets or registry auth Check your task network configuration.

當任務無法連線至 Amazon ECR、Systems Manager 或 Secrets Manager 時，就會發生此錯誤。此問題源自於網路組態錯誤。

若要修正此問題，請驗證任務與 Amazon ECR 之間的連線狀態。您也需要檢查任務與儲存秘密的服務 (Systems Manager 或 Secrets Manager) 之間的連線狀態。如需詳細資訊，請參閱[驗證 Amazon ECS 已停止任務的連線狀態](#)。

任務找不到任務定義中所定義的 Amazon CloudWatch 日誌群組。There is a connection issue between the task and Amazon CloudWatch. Check your network configuration.

當任務找不到您在任務定義中定義的 CloudWatch 日誌群組時，就會發生此錯誤。

此錯誤表示 CloudWatch VPC 端點與任務之間存在網路連線問題。

如需有關如何驗證任務與端點之間連線的資訊，請參閱[驗證 Amazon ECS 已停止任務的連線狀態](#)。

failed to initialize logging driver

當任務找不到您在任務定義中定義的 CloudWatch 日誌群組時，就會發生此錯誤。

此錯誤表示任務定義中的 CloudWatch 群組不存在。

使用下列步驟尋找缺少的 CloudWatch。

1. 執行下列命令來取得任務定義資訊。

```
aws ecs describe-task-definition \
  --task-definition task-def-name
```

查看每個容器的輸出，並記下 awslogs-group 值。

```
"logConfiguration": {
```



```

    "logDriver": "awslogs",
    "options": {
        "awslogs-group": "/ecs/example-group",
        "awslogs-create-group": "true",
        "awslogs-region": "us-east-1",
        "awslogs-stream-prefix": "ecs"
    },

```

2. 確認群組存在於 CloudWatch 中。如需詳細資訊，請參閱 Amazon CloudWatch Logs User Guide 中的 [Working with log groups and log streams](#)。

問題可能是任務定義中指定的群組不正確，也可能是日誌群組不存在。

3. 修正問題。

問題是...	執行此作業...
任務定義中指定了不正確的日誌群組。	更新任務定義，在容器定義中包含日誌群組組態。如需有關更新任務定義的資訊，請參閱 使用主控台更新 Amazon ECS 任務定義 ，或 Amazon Elastic Container Service API Reference 中的 RegisterTaskDefinition 。
CloudWatch 中不存在日誌群組	建立日誌群組。如需詳細資訊，請參閱《Amazon CloudWatch Logs 使用者指南》中的 在 CloudWatch Logs 中建立日誌群組 。

無法調用 EFS utils 命令來設定 EFS 磁碟區

下列問題可能會阻止您在任務上掛載 Amazon EFS 磁碟區：

- Amazon EFS 檔案系統設定不正確。
- 任務缺少必要許可。
- 存在與網路與 VPC 組態相關的問題。

如需有關如何偵錯和修正此問題的資訊，請參閱[為何我無法在 re : Post 上的 AWS Fargate 任務上掛載 Amazon EFS 磁碟區](#)。AWS

對 Amazon ECS ResourceNotFoundException 錯誤進行疑難排解

以下是部分 ResourceNotFoundException 錯誤訊息以及可採取的錯誤修正動作。

若要使用 檢查已停止的任務是否有錯誤訊息 AWS 管理主控台，請參閱[檢視 Amazon ECS 已停止任務錯誤](#)。

任務無法從中擷取具有 ARN '**secretARN**' 的秘密 AWS Secrets Manager。Check whether the secret exists in the specified Region.

當任務無法從 Secrets Manager 擷取秘密時，就會發生此錯誤。這表示在任務定義中指定 (並且在錯誤訊息中包含) 的秘密不存在於 Secrets Manager 中。

區域資訊包含在錯誤訊息中。

從 AWS Secrets Manager ##擷取秘密資料：秘密 **secretARN**：ResourceNotFoundException：Secrets Manager 找不到指定的秘密。

如需有關尋找秘密的資訊，請參閱 AWS Secrets Manager User Guide 中的[Find secrets in AWS Secrets Manager](#)。

使用下表來判斷並解決錯誤。

問題	動作
秘密位於與任務定義不同的區域。	- 在與任務相同的區域中建立秘密。如需詳細資訊，請參閱建立 AWS Secrets Manager 秘密。 - 使用新秘密更新任務定義。如需詳細資訊，請參閱使用主控台更新 Amazon ECS 任務定義，或 Amazon Elastic Container Service API Reference 中的RegisterTaskDefinition。

問題	動作	
任務定義的秘密 ARN 不正確。Secrets Manager 中存在正確的秘密。	使用正確的秘密更新任務定義。如需詳細資訊，請參閱 使用主控台更新 Amazon ECS 任務定義 ，或 Amazon Elastic Container Service API Reference 中的 RegisterTaskDefinition 。	
秘密不再存在。	- 在與任務相同的區域中建立秘密。如需詳細資訊，請參閱建立 AWS Secrets Manager 秘密。 - 使用新秘密更新任務定義。如需詳細資訊，請參閱使用主控台更新 Amazon ECS 任務定義，或 Amazon Elastic Container Service API Reference 中的 RegisterTaskDefinition。	

對 Amazon ECS SpotInterruption 錯誤進行疑難排解

SpotInterruption 錯誤在 Fargate 和 EC2 中發生的原因不相同。

若要使用 檢查已停止的任務是否有錯誤訊息 AWS 管理主控台，請參閱 [檢視 Amazon ECS 已停止任務錯誤](#)。

Fargate

當沒有 Fargate Spot 容量或當 Fargate 取回 Spot 容量時，就會發生 SpotInterruption 錯誤。

您可以讓任務在多個可用區域中執行，以取得更多容量。

EC2

當沒有可用的 Spot 執行個體或當 EC2 取回 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，以取得更多容量。

對 Amazon ECS InternalError 錯誤進行疑難排解

適用對象：Fargate

若要使用 檢查已停止的任務是否有錯誤訊息 AWS 管理主控台，請參閱 [檢視 Amazon ECS 已停止任務錯誤](#)。

當代理程式遇到非執行時期相關的意外內部錯誤時，就會發生 InternalError 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。

如需有關偵錯和修正此問題的資訊，請參閱 [Amazon ECS 已停止任務錯誤訊息](#)。

對 Amazon ECS OutOfMemoryError 錯誤進行疑難排解

以下是部分 OutOfMemoryError 錯誤訊息以及可採取的錯誤修正動作。

若要使用 檢查已停止的任務是否有錯誤訊息 AWS 管理主控台，請參閱 [檢視 Amazon ECS 已停止任務錯誤](#)。

container killed due to memory usage

當容器結束是由於容器中的程序消耗的記憶體比任務定義中分配的記憶體更多時，或是由於主機或作業系統限制時，就會發生此錯誤。

對 Amazon ECS ContainerRuntimeError 錯誤進行疑難排解

以下是部分 ContainerRuntimeError 錯誤訊息以及可採取的錯誤修正動作。

若要使用 檢查已停止的任務是否有錯誤訊息 AWS 管理主控台，請參閱 [檢視 Amazon ECS 已停止任務錯誤](#)。

ContainerRuntimeError

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資訊，請參閱 [為什麼我的 Amazon ECS 任務在 re : Post 上停止](#)。

AWS

對 Amazon ECS ContainerRuntimeTimeoutError 錯誤進行疑難排解

以下是部分 ContainerRuntimeTimeoutError 錯誤訊息以及可採取的錯誤修正動作。

若要使用 檢查已停止的任務是否有錯誤訊息 AWS 管理主控台，請參閱 [檢視 Amazon ECS 已停止任務錯誤](#)。

Could not transition to running; timed out after waiting 1m or Docker timeout error

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。

對 Amazon ECS CannotStartContainerError 錯誤進行疑難排解

以下是部分 CannotStartContainerError 錯誤訊息以及可採取的錯誤修正動作。

若要使用 檢查已停止的任務是否有錯誤訊息 AWS 管理主控台，請參閱 [檢視 Amazon ECS 已停止任務錯誤](#)。

failed to get container status: *<reason>*

無法啟動容器時，會發生此錯誤。

如果容器嘗試使用超過此處指定的記憶體，容器便會終止。增加分配給容器的記憶體。這是任務定義中的 memory 參數。如需詳細資訊，請參閱[the section called “記憶體”](#)。

對 Amazon ECS CannotStopContainerError 錯誤進行疑難排解

以下是部分 CannotStopContainerError 錯誤訊息以及可採取的錯誤修正動作。

若要使用 檢查已停止的任務是否有錯誤訊息 AWS 管理主控台，請參閱 [檢視 Amazon ECS 已停止任務錯誤](#)。

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資訊，請參閱[為什麼我的 Amazon ECS 任務在 re : Post 上停止](#)。
AWS

對 Amazon ECS CannotInspectContainerError 錯誤進行疑難排解

以下是部分 CannotInspectContainerError 錯誤訊息以及可採取的錯誤修正動作。

若要使用 檢查已停止的任務是否有錯誤訊息 AWS 管理主控台，請參閱 [檢視 Amazon ECS 已停止任務錯誤](#)。

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 或先前版本時，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)，或者 1.0.0 或更新版本 (Windows) 時，Fargate 代理程式會從 containerd 傳回原因。

如需有關如何偵錯和修正此問題的資訊，請參閱[為什麼我的 Amazon ECS 任務在 re : Post 上停止](#)。
AWS

對 Amazon ECS CannotCreateVolumeError 錯誤進行疑難排解

以下是部分 CannotCreateVolumeError 錯誤訊息以及可採取的錯誤修正動作。

若要使用 檢查已停止的任務是否有錯誤訊息 AWS 管理主控台，請參閱 [檢視 Amazon ECS 已停止任務錯誤](#)。

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資訊，請參閱[為什麼我的 Amazon ECS 任務在 re : Post 上停止](#)。
AWS

Amazon ECS 中的 CannotPullContainer 任務錯誤

下列錯誤表示任務無法啟動，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。

若要使用 檢查已停止的任務是否有錯誤訊息 AWS 管理主控台，請參閱 [檢視 Amazon ECS 已停止任務錯誤](#)。

 Tip

您可以[Amazon ECS MCP 伺服器](#)搭配 AI 助理使用，使用自然語言調查影像提取錯誤。

錯誤

- [The task can't pull the image. Check that the role has the permissions to pull images from the registry.](#)
- [The task cannot pull 'image-name' from the Amazon ECR repository 'repository URI'. There is a connection issue between the task and Amazon ECR. Check your task network configuration.](#)
- [The task can't pull the image. Check your network configuration](#)
- [CannotPullContainerError: pull image manifest has been retried 5 time\(s\): failed to resolve ref](#)
- [API error \(500\): Get https://111122223333.dkr.ecr.us-east-1.amazonaws.com/v2/: net/http: request canceled while waiting for connection](#)
- [API 錯誤](#)
- [write /var/lib/docker/tmp/GetImageBlob111111111: no space left on device](#)
- [ERROR: toomanyrequests: Too Many Requests or You have reached your pull rate limit.](#)
- [Error response from daemon: Get url: net/http: request canceled while waiting for connection](#)
- [ref pull has been retried 1 time\(s\): failed to copy: httpReaderSeeker: failed open: unexpected status code](#)
- [pull access denied](#)
- [pull command failed: panic: runtime error: invalid memory address or nil pointer dereference](#)
- [error pulling image conf/error pulling image configuration](#)
- [內容已取消](#)

The task can't pull the image. Check that the role has the permissions to pull images from the registry.

此錯誤表示由於許可問題，任務無法提取任務定義中指定的映像。

若要解決此問題：

1. 檢查映像是否存在於###中。如需有關檢視映像的資訊，請參閱 Amazon Elastic Container Registry User Guide 中的 [Viewing image details in Amazon ECR](#)。

2. 確認 *role-arn* 具有提取映像的正確許可。

如需有關如何更新角色的資訊，請參閱 AWS Identity and Access Management Use Guide 中的 [Update permissions for a role](#)。

任務使用以下任一角色：

- 對於使用 Fargate 的任務，此為任務執行角色。如需有關 Amazon ECR 額外許可的資訊，請參閱 [Fargate 任務透過介面端點提取 Amazon ECR 映像的許可](#)。
- 對於使用 EC2 的任務，此為容器執行個體角色。如需有關 Amazon ECR 額外許可的資訊，請參閱 [Amazon ECR 許可](#)。

The task cannot pull '*image-name*' from the Amazon ECR repository '*repository URI*'. There is a connection issue between the task and Amazon ECR. Check your task network configuration.

此錯誤表示任務無法連線至 Amazon ECR。檢查### *URI* 儲存庫的連線。

如需有關如何確認與解決此問題的資訊，請參閱[驗證 Amazon ECS 已停止任務的連線狀態](#)。

The task can't pull the image. Check your network configuration

此錯誤表示任務無法連線至 Amazon ECR。

如需有關如何確認與解決此問題的資訊，請參閱[驗證 Amazon ECS 已停止任務的連線狀態](#)。

CannotPullContainerError: pull image manifest has been retried 5 time(s): failed to resolve ref

此錯誤表示任務無法提取映像。

若要解決此問題，您可以：

- 確認任務定義中指定的映像符合儲存庫中的映像。
- Amazon ECS 會強制保障映像版本穩定性。若原始映像不再可用，您就會遇到此錯誤。映像標籤是強制實施此項機制的關鍵組成。請將任務定義中的映像標籤從 :latest 更改為特定版本。如需詳細資訊，請參閱[容器映像解析](#)。

如需有關如何確認與解決此問題的資訊，請參閱[驗證 Amazon ECS 已停止任務的連線狀態](#)。

API error (500): Get https://111122223333.dkr.ecr.us-east-1.amazonaws.com/v2/: net/http: request canceled while waiting for connection

此錯誤表示連線已逾時，原因是不存在通往網際網路的路由。

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[將應用程式作為 Amazon ECS 任務執行](#)。
- 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 端點存在連線問題。

如需如何解決此問題的資訊，請參閱 支援 網站上的[如何解決 Amazon ECS 中的 Amazon ECR 錯誤「CannotPullContainerError：API 錯誤」](#)。

```
write /var/lib/docker/tmp/GetImageBlob111111111: no space left on device
```

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。

如果使用 Amazon ECS 最佳化 AMI，您可以透過下列命令來擷取檔案系統上最大的前 20 個檔案：

```
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，執行容器可能會填滿根磁碟區。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 [JSON File logging driver](#)。

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"
  }
}
```

如果容器日誌佔用太多磁碟空間，則可使用 awslogs 日誌驅動程式。awslogs 日誌驅動程式會將日誌傳送至 CloudWatch，以釋放原本要給容器執行個體上的容器日誌使用的磁碟空間。如需詳細資訊，請參閱[將 Amazon ECS 日誌傳送至 CloudWatch](#)。

您可能需要更新 Docker 可存取的磁碟大小。

如需詳細資訊，請參閱 [CannotPullContainerError: no space left on device](#)。

ERROR: toomanyrequests: Too Many Requests or You have reached your pull rate limit.

此錯誤表示存在 Docker Hub 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](#)。

如果您已提升 Docker Hub 速率限制，並且需要身分驗證容器執行個體的 Docker 提取，請參閱[容器執行個體的私有登錄檔身分驗證](#)。

Error response from daemon: Get **url**: net/http: request canceled while waiting for connection

此錯誤表示連線已逾時，原因是不存在通往網際網路的路由。

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[將應用程式作為 Amazon ECS 任務執行](#)。
- 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 has been retried 1 time(s): failed to copy: httpReaderSeeker: failed open: unexpected status code

此錯誤表示複製映像時失敗。

若要解決此問題，請檢閱下列文章：

- 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](#)。

pull access denied

此錯誤表示無法存取映像。

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 User Guide 中的 [Private registry authentication](#)。

pull command failed: panic: runtime error: invalid memory address or nil pointer dereference

此錯誤表示由於記憶體位址無效或空指標解除參照，無法存取映像。

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。

error pulling image conf/error pulling image configuration

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[我的 Amazon ECS EC2 啟動類型任務中的如何解決「CannotPullContainerError」錯誤](#)。

內容已取消

此錯誤表示內容已取消。

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 已停止任務的連線狀態

有時候任務會因為網路連線問題而停止。這可能是間歇性問題，但最有可能是因為任務無法連線至端點。

測試任務連線狀態

您可以使用 `AWSSupport-TroubleshootECSTaskFailedToStart` 執行手冊來測試任務連線狀態。在使用執行手冊時，您需要下列資源資訊：

- 任務 ID

使用最近失敗任務的 ID。

- 任務所在的叢集

如需有關如何使用執行手冊的資訊，請參閱 AWS Systems Manager Automation runbook reference 中的 [AWSSupport-TroubleshootECSTaskFailedToStart](#)。

執行手冊會分析任務。您可以在輸出區段中檢視下列問題的結果，這些問題可能會阻止任務啟動：

- 與已設定容器登錄檔的網路連線
- VPC 端點連線
- 安全群組規則組態

修正 VPC 端點問題

當 `AWSSupport-TroubleshootECSTaskFailedToStart` 執行手冊結果顯示存在 VPC 端點問題時，請檢查下列組態：

- 建立端點的 VPC 和 VPC 端點需要使用私有 DNS。
- 請確定您有一個服務的 AWS PrivateLink 端點，而任務無法在與任務相同的 VPC 中連線到該端點。
如需詳細資訊，請參閱下列主題：

服務	服務的 VPC 端點資訊
Amazon ECR	Amazon ECR 介面 VPC 端點 (AWS PrivateLink)
Systems Manager	使用適用於 Systems Manager 的 VPC 端點來改善 EC2 執行個體的安全性
Secrets Manager	使用 AWS Secrets Manager VPC 端點
CloudWatch	CloudWatch VPC 端點
Amazon S3	AWS PrivateLink 適用於 Amazon S3 的

- 設定任務子網路的傳出規則，允許連接埠 443 的 HTTPS 與 DNS (TCP) 流量。如需詳細資訊，請參閱 Amazon Elastic Compute Cloud User Guide 中的 [Configure security group rules](#)。
- 若使用自訂網域名稱伺服器，請確認 DNS 查詢的設定。查詢必須具有連接埠 53 的傳出存取權，並使用 UDP 與 TCP 通訊協定。此外，其必須具備連接埠 443 的 HTTPS 存取權。如需詳細資訊，請參閱 Amazon Elastic Compute Cloud User Guide 中的 [Configure security group rules](#)。
- 如果子網路具有網路 ACL，則需要如下 ACL 規則：
 - 傳出規則，允許連接埠 1024-65535 的流量。
 - 傳入規則，允許連接埠 443 的 TCP 流量。

如需有關如何設定規則的資訊，請參閱 Amazon Virtual Private Cloud User Guide 中的 [Control traffic to subnets using network ACLs](#)。

修正網路問題

當 `AWSSupport-TroubleshootECSTaskFailedToStart` 執行手冊結果顯示存在網路問題時，請檢查下列組態：

在公有子網路中使用 `awsvpc` 網路模式的任務

根據執行手冊執行下列組態：

- 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[將應用程式作為 Amazon ECS 任務執行](#)。
- 您需要閘道來處理網際網路流量。任務子網路的路由表需要有通往閘道的流量路由。

如需詳細資訊，請參閱 Amazon Virtual Private Cloud User Guide 中的 [Add and remove routes from a route table](#)。

閘道類型	路由表目的地	路由表目標
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

- 如果任務子網路具有網路 ACL，則需要如下 ACL 規則：
 - 傳出規則，允許連接埠 1024-65535 的流量。
 - 傳入規則，允許連接埠 443 的 TCP 流量。

如需有關如何設定規則的資訊，請參閱 Amazon Virtual Private Cloud User Guide 中的 [Control traffic to subnets using network ACLs](#)。

在私有子網路中使用 awsvpc 網路模式的任務

根據執行手冊執行下列組態：

- 啟動任務時，在自動指派公有 IP 欄位中選擇停用。
- 在 VPC 中設定 NAT 閘道，將請求路由至網際網路。如需詳細資訊，請參閱 Amazon Virtual Private Cloud User Guide 中的 [NAT Gateways](#)。
- 任務子網路的路由表需要有通往 NAT 閘道的流量路由。

如需詳細資訊，請參閱 Amazon Virtual Private Cloud User Guide 中的 [Add and remove routes from a route table](#)。

閘道類型	路由表目的地	路由表目標
NAT	0.0.0.0/0	NAT 閘道 ID

- 如果任務子網路具有網路 ACL，則需要如下 ACL 規則：
 - 傳出規則，允許連接埠 1024-65535 的流量。

- 傳入規則，允許連接埠 443 的 TCP 流量。

如需有關如何設定規則的資訊，請參閱 Amazon Virtual Private Cloud User Guide 中的 [Control traffic to subnets using network ACLs](#)。

在公有子網路中不使用 awsvpc 網路模式的任務

根據執行手冊執行下列組態：

- 當您建立叢集時，請在 Amazon EC2 執行個體聯網區段的自動指派 IP 欄位中選擇啟用。

此選項會將公有 IP 位址指派給執行個體主要網路介面。

- 您需要閘道來處理網際網路流量。執行個體子網路的路由表需要有通往閘道的流量路由。

如需詳細資訊，請參閱 Amazon Virtual Private Cloud User Guide 中的 [Add and remove routes from a route table](#)。

閘道類型	路由表目的地	路由表目標
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，則需要如下 ACL 規則：

- 傳出規則，允許連接埠 1024-65535 的流量。
- 傳入規則，允許連接埠 443 的 TCP 流量。

如需有關如何設定規則的資訊，請參閱 Amazon Virtual Private Cloud User Guide 中的 [Control traffic to subnets using network ACLs](#)。

在私有子網路中不使用 awsvpc 網路模式的任務

根據執行手冊執行下列組態：

- 當您建立叢集時，請在 Amazon EC2 執行個體聯網區段的自動指派 IP 欄位中選擇停用。
- 在 VPC 中設定 NAT 閘道，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](#)。
- 執行個體子網路的路由表需要有通往 NAT 閘道流量的路由。

如需詳細資訊，請參閱 Amazon Virtual Private Cloud User Guide 中的 [Add and remove routes from a route table](#)。

閘道類型	路由表目的地	路由表目標
NAT	0.0.0.0/0	NAT 閘道 ID

- 如果任務子網路具有網路 ACL，則需要如下 ACL 規則：
 - 傳出規則，允許連接埠 1024-65535 的流量。
 - 傳入規則，允許連接埠 443 的 TCP 流量。

如需有關如何設定規則的資訊，請參閱 Amazon Virtual Private Cloud User Guide 中的 [Control traffic to subnets using network ACLs](#)。

檢視 Amazon ECS 任務的 IAM 角色請求

當您為 IAM 角色中的任務憑證使用提供者時，提供者請求會儲存在稽核日誌中。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

對服務問題進行疑難排解時，您首先應檢查服務事件日誌以取得診斷資訊。您可以使用 DescribeServices API、AWS CLI 或 來檢視服務事件 AWS 管理主控台。

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 服務排程器的任務置放和執行個體運作狀態事件。這些事件會擁有字首 service (**service-name**)。為了確保此事件檢視有幫助，我們只會顯示最新的前 100 個事件。系統會忽略重複的服務事件訊息，直至原因解決或經過六個小時為止。如果原因未在六小時內解決，您會收到另一則關於該原因的服務事件訊息。
- Service Auto Scaling 事件。這些事件字首為 Message，並且只有在服務以應用程式自動擴展政策設定時才會發生。

Tip

您可以[Amazon ECS MCP 伺服器](#)搭配 AI 助理使用 來分析使用自然語言的服務事件。

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. 在事件索引標籤上，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 *service-name* 服務，可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 服務事件訊息

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。

當服務處於良好狀態且任務數量達到預期，進而進入穩定狀態時，服務排程器會傳送 service (*service-name*) has reached a steady state. 服務事件。

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。

服務排程器會在找不到可用資源來新增另一個任務時傳送此事件訊息。可能的原因如下：

使用容量提供者自動擴展 EC2 執行個體。如需詳細資訊，請參閱[EC2 工作負載的 Amazon ECS 容量提供者](#)。

如果打算使用容量提供者，請確定您已傳遞容量提供者策略，或具有與叢集相關聯的預設容量提供者策略，且未傳遞啟動類型與容量提供者策略作為輸入

您的叢集中找不到任何容器執行個體

如果嘗試執行任務的叢集中沒有已註冊的容器執行個體，您會收到此錯誤。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

用於任務置放的最適配容器執行個體，其保留連接埠數量不得超過每個容器執行個體最多 100 個主機連接埠的限制。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

此任務的任務定義在其連接埠映射中使用的連接埠，與所選容器執行個體上執行的任務所使用的連接埠相同。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](#)。
- 如果您已選擇使用 `awsvpcTrunking` 帳戶設定，但您尚未在選擇使用後，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[援更多 Amazon ECS 容器網路介面的執行個體](#)。

如需選擇使用的 `awsvpcTrunking` 帳戶設定詳細資訊，請參閱 [增加 Amazon ECS Linux 容器執行個體網路介面數量](#)。

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。如果您的任務定義包含一個需要特定容器執行個體屬性的參數，而您沒有任何可用的容器執行個體可滿足此需求，即無法置放任務。

此錯誤的常見原因是，服務使用的任務採用了 `awsvpc` 網路模式與 EC2。您指定的叢集在建立服務時，並未於 `awsvpcConfiguration` 中所指定的相同子網路內註冊任何容器執行個體。

您可以使用 `AWSSupport-TroubleshootECSContainerInstance` 執行手冊進行疑難排解。執行手冊會檢閱執行個體的使用者資料是否包含正確的叢集資訊、執行個體設定檔是否包含必要的許可，以及網路組態問題。如需詳細資訊，請參閱 AWS Systems Manager Automation runbook reference User Guide 中的 [AWSSupport-TroubleshootECSContainerInstance](#)。

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「[Fargate 的 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。

用於任務置放的最適配容器執行個體，其 CPU 單位不足以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 最佳化 Amazon Linux 2 AMI 與 Amazon ECS 最佳化 Amazon Linux 2023 AMI

```
sudo systemctl restart ecs
```

- 對於 Amazon ECS 最佳化 Amazon Linux AMI

```
sudo stop ecs && sudo start ecs
```

service (***service-name***) (task ***task-id***) (instance ***instance-id***) is unhealthy in (elb ***elb-name***) due to (reason Instance has failed at least the UnhealthyThreshold number of health checks consecutively.)

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。訊息包含任務 ID，可協助識別哪些特定任務未通過運作狀態檢查。如需詳細資訊，請參閱[對 Amazon ECS 中的服務負載平衡器進行疑難排解](#)。

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 服務限流邏輯](#)。

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 支援 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。更新 `minimumHealthyPercent` 或 `maximumPercent` 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。部署組態包含在建立服務時定義的 `minimumHealthyPercent` 與 `maximumPercent` 值。這些值也可在現有的服務上進行更新。

`minimumHealthyPercent` 代表在部署期間或容器執行個體排空時，應為服務執行的任務數量下限。這表示的是服務所需任務數量的百分比。此值會向上捨入。例如，最小運作狀態良好的百分比為 50，且所需的任務數量為四，則排程器可以先停止兩個現有的任務，再開始兩個新的任務。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。

`maximumPercent` 代表在部署期間或容器執行個體排空時，應為服務執行的任務數量上限。這表示的是服務所需任務數量的百分比。此值會向下捨去。例如，最大百分比為 200，且所需的任務數量為四，則排程器可以先停止四個現有的任務，再開始四個新的任務。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。

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。

以下是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 區域”](#)。

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](#)。

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 正在從任務計數型配額轉換為 vCPU 型配額。

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：受保護的任務數量超過所需的任務數量

服務具有的受保護任務數超過所需的任務數量。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。
- 判斷哪些任務可以停止，並透過 UpdateTaskProtection API，將 protectionEnabled 選項設定為 false，取消這些任務的保護狀態。
- 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。

服務排程器會在找不到可用資源來新增另一個任務時傳送此事件訊息。可能的原因如下：

沒有與叢集關聯的容量提供者

使用 describe-services 確認您是否擁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的容量。若使用 EC2，請確定容器執行個體符合任務定義需求。

您的叢集中找不到任何容器執行個體

如果嘗試執行任務的叢集中沒有已註冊的容器執行個體，您會收到此錯誤。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

如果任務使用固定的主機連接埠映射 (例如，任務為 Web 伺服器使用主機的連接埠 80)，每個任務至少必須擁有一個容器執行個體。同一時間，一個容器只能使用一個主機連接埠。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

用於任務置放的最適配容器執行個體，其保留連接埠數量不得超過每個容器執行個體最多 100 個主機連接埠的限制。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

此任務的任務定義在其連接埠映射中使用的連接埠，與所選容器執行個體上執行的任務所使用的連接埠相同。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 數量有上限限制，且叢集中沒有具備可用 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](#)。

- 如果您已選擇使用 `awsvpcTrunking` 帳戶設定，但您尚未在選擇使用後，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[援更多 Amazon ECS 容器網路介面的執行個體](#)。

如需選擇使用的 `awsvpcTrunking` 帳戶設定詳細資訊，請參閱 [增加 Amazon ECS Linux 容器執行個體網路介面數量](#)。

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。如果您的任務定義包含一個需要特定容器執行個體屬性的參數，而您沒有任何可用的容器執行個體可滿足此需求，即無法置放任務。

此錯誤的常見原因是，如果您服務使用的是使用 `awsvpc` 網路模式和 EC2 的任務，且您指定的叢集在相同子網路中沒有註冊至其中的容器執行個體，而這個子網路指的是在建立服務時於 `awsvpcConfiguration` 指定的子網路。

您可以使用 `AWSSupport-TroubleshootECSContainerInstance` 執行手冊進行疑難排解。執行手冊會檢閱執行個體的使用者資料是否包含正確的叢集資訊、執行個體設定檔是否包含必要的許可，以及網路組態問題。如需詳細資訊，請參閱 AWS Systems Manager Automation runbook reference User Guide 中的 [AWSSupport-TroubleshootECSContainerInstance](#)。

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「[Fargate 的 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 (***service-name***) 部署失敗：任務無法啟動。

服務中的任務無法啟動。

如需有關如何偵錯已停止任務的資訊，請參閱 [Amazon ECS 已停止任務錯誤訊息](#)。

service (***service-name***) Timed out waiting for Amazon ECS Agent to start. Please check logs at /var/log/ecs/ecs-agent.log".

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。如果可透過 SSH 連線至容器執行個體，即可檢查代理程式日誌。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 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
```

service (***service-name***) task set (***taskSet-ID***) (task ***task-id***) is not healthy in target-group (***targetGroup-ARN***) due to **TARGET GROUP IS NOT FOUND**.

服務的任務集運作狀態檢查失敗，因為找不到目標群組。訊息包含任務 ID，可協助識別哪些特定任務未通過運作狀態檢查。您應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。

service (***service-name***) task set (***taskSet-ID***) (task ***task-id***) is not healthy in target-group (***targetGroup-ARN***) due to **TARGET IS NOT FOUND**.

服務的任務集運作狀態檢查失敗，因為找不到目標。訊息包含任務 ID，可協助識別哪些特定任務未通過運作狀態檢查。

IAM 許可政策設定錯誤或已變更，ECS 無法再維護您的服務

由於 IAM 許可政策設定錯誤或已變更，服務無法維護任務。與 ECS 服務或任務相關聯的 IAM 角色可能缺少必要許可。

若要解決此問題，請將必要許可新增至 IAM 角色。如需有關管理 IAM 許可政策的詳細資訊，請參閱 IAM User Guide 中的 [Adding and removing IAM identity permissions](#)。

IAM 信任關係設定錯誤或已變更，ECS 無法再維護您的服務

由於 IAM 信任關係設定錯誤或已變更，服務無法維護任務。與 ECS 服務或任務相關聯的 IAM 角色的信任政策可能不正確。

若要解決此問題，請為任務定義中使用的角色設定信任政策。如需有關為自訂角色建立信任政策的詳細資訊，請參閱 IAM User Guide 中的 [Creating a role for a custom use case](#)。

service (*service-name*) 無法啟動 deployment *deployment-id* 的##任務。

當部署工作流程成功啟動某些任務，但由於容量不足錯誤而無法啟動所有請求的任務時，服務排程器會傳送此事件訊息。這通常發生在啟用斷路器時，並提供部署失敗或復原原因的可見性。

訊息包含特定的失敗原因，例如 CPU、記憶體不足或其他資源限制。這可協助您了解需要處理哪些資源來解決部署問題。

如需詳細資訊，請參閱[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](#)。

service (*service-name*) 無法在叢集中放置任務，因為超過任務佈建容量限制。

當您的叢集達到可同時處於 PROVISIONING 狀態的 500 個任務限制時，服務排程器會傳送此事件訊息。這是叢集層級限制，而不是服務特定的問題。

當您啟動具有大量所需任務且預先佈建容量有限的服務時，或同時啟動多個服務時，通常會發生這種情況。

若要解決此問題：

- 等待現有任務完成佈建並移至 RUNNING 狀態。
- 請考慮逐步擴展您的服務，以避免達到佈建限制。
- 檢閱叢集的容量提供者組態，以確保有足夠的資源可用。

如需 Amazon ECS 服務配額的詳細資訊，請參閱《[Amazon Web Services 一般參考](#)》中的 [Amazon Elastic Container Service 端點和配額](#)。

Amazon ECS 服務運作狀態不良事件訊息

以下是服務運作狀態不良的事件訊息範例。

EC2: (service **service-name**) (task **task-id**) (instance **instance-id**) (port **port-number**) is unhealthy in (target-group **target-group-name**) due to (reason **failure-reason**)

此訊息表示在 EC2 執行個體上執行的任務運作狀態檢查失敗。如需詳細資訊，請參閱下列內容：

- [如何讓使用 Amazon EC2 啟動類型的 Amazon ECS 任務通過 Application Load Balancer 運作狀態檢查？](#)

EC2 with taskSet: (service **service-name**, taskSet **taskSet-id**) (task **task-id**) (instance **instance-id**) (port **port-number**) is unhealthy in (target-group **target-group-name**) due to (reason **failure-reason**)

此訊息表示在 EC2 執行個體上執行的任務集中的任務運作狀態檢查失敗。如需詳細資訊，請參閱下列內容：

- [如何讓使用 EC2 啟動類型的 Amazon ECS 任務通過 Application Load Balancer 運作狀態檢查？](#)

Fargate: (service **service-name**) (task **task-id**) (port **port-number**) is unhealthy in (target-group **target-group-name**) due to (reason **failure-reason**)

此訊息表示 Fargate 任務運作狀態檢查失敗。

如需有關對 Fargate 任務中的運作狀態檢查失敗進行疑難排解的詳細資訊，請參閱[如何對 Fargate 上 Amazon ECS 任務的運作狀態檢查失敗進行疑難排解？](#)文章。

Fargate with taskSet: (service **service-name**, taskSet **taskSet-id**) (task **task-id**) (port **port-number**) is unhealthy in (target-group **target-group-name**) due to (reason **failure-reason**)

此訊息表示在 Fargate 上執行的任務集中的任務運作狀態檢查失敗。

如需有關對 Fargate 任務中的運作狀態檢查失敗進行疑難排解的詳細資訊，請參閱[如何對 Fargate 上 Amazon ECS 任務的運作狀態檢查失敗進行疑難排解？](#)文章。

Amazon ECS 可用區域服務重新平衡服務事件訊息

您會看到以下服務事件訊息範例。

service (**service-name**) is not AZ balanced with **number-tasks** tasks in **Availability Zone 1**, **number-tasks** in **Availability Zone 2**, and **number-tasks** in **Availability Zone 3**. AZ Rebalancing in progress.

當任務數量未平均分散在可用區域中時，服務排程器會傳送 service (**service-name**) is not AZ balanced 服務事件。無需採取任何動作。這是資訊事件。

service (**service-name**) is AZ balanced with **number-tasks** tasks in **Availability Zone 1**, **number-tasks** tasks in **Availability Zone 2**, and **number-tasks** tasks in **Availability Zone 3**.

當可用區域服務重新平衡完成時，服務排程器會傳送 service (**service-name**) is AZ balanced 服務事件。無需採取任何動作。這是資訊事件。

service-name has started **number-tasks** tasks in **Availability Zone** to AZ Rebalance: **task-ids**.

因為服務重新平衡，服務排程器在可用區域中啟動任務時，會傳送 **service-name/task-set-name** has started **number** tasks in **Availability Zone** 服務事件。無需採取任何動作。這是資訊事件。

service-name has stopped **number-tasks** running tasks in **Availability Zone** due to AZ rebalancing: **task-id**.

因為服務重新平衡，服務排程器在可用區域中停止任務時，會傳送 **service-name/task-set-name** has stopped **number** tasks in **Availability Zone** 服務事件。無需採取任何動作。這是資訊事件。

service (**service-name**) is unable to place a task in **Availability Zone** because no container instance met all of its requirements.

因為沒有任何容器執行個體符合其所有需求，服務排程器會傳送 **service-name** is unable to place a task in **Availability Zone** 服務事件。若要解決問題，請在可用區域中啟動執行個體。

service (**service-name**) is unable to place a task in **Availability Zone**.

當您使用 Fargate 且沒有可用容量時，服務排程器會傳送 **service-name** is unable to place a task in **Availability Zone** 服務事件。

您可以在錯誤訊息的可用區域中新增其他子網路，或聯絡 支援 以取得額外的容量。

service (*service-name*) was unable to AZ Rebalance because *task-set-name* was unable to scale in due to *reason*.

當您使用任務縮減保護時，服務排程器會傳送 *service-name* was unable to AZ Rebalance because *task-set-name* was unable to scale in due to *reason* 服務事件。

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。
- 判斷哪些任務可以停止，並透過 UpdateTaskProtection API，將 protectionEnabled 選項設定為 false，取消這些任務的保護狀態。
- 將服務的所需任務計數提高到超過受保護任務的數目。

service (*service-name*) stopped AZ Rebalancing.

當可用區域重新平衡操作停止時，服務排程器會傳送 *service-name* stopped AZ Rebalancing 服務事件。這是資訊事件。Amazon ECS 會傳送提供詳細資訊的額外事件。

對 Amazon ECS 中的服務負載平衡器進行疑難排解

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](#)。

容器執行個體安全群組

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。

Elastic Load Balancing 負載平衡器未針對所有可用區域設定

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啟動位在負載平衡器未設定使用之可用區域中容器執行個體的任務，任務絕對不會通過運作狀態檢查。這會導致任務遭到終止。

Elastic Load Balancing 負載平衡器運作狀態檢查設定錯誤

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運作狀態不良，則會從負載平衡器中將其移除。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

這是負載平衡器運作狀態檢查的一部分。這是應用程式上的端點，可在應用程式運作狀態良好時傳回成功的狀態碼 (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運作不良閾值。

運作不良閾值

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。如果運作不良閾值為 2，而運作狀態檢查的間隔為 30 秒，則任務在被判定為運作狀態不良前，有 60 秒可回應運作狀態檢查 ping。您可以提高運作不良閾值或運作狀態檢查間隔，讓任務有更多的時間回應。

無法更新服務 *servicename*：任務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。如需有關如何修改組態的資訊，請參閱 Amazon Elastic Container Service API 參考 中的 [UpdateService](#)。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。

您已達到可以同時執行的任務數量限制。

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。

Amazon ECS 中的服務自動擴展疑難排解

應用程式自動擴展會在 Amazon ECS 部署正在進行時關閉縮減程序，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](#)。

主控台中的 Amazon ECS 事件擷取

Amazon ECS 主控台提供事件擷取功能，可透過 EventBridge 將服務動作和任務狀態變更等 Amazon ECS 產生的事件儲存到 Amazon CloudWatch Logs。此功能包含查詢介面，具有用於監控和故障診斷的篩選功能。

事件提供有關您的服務部署、服務、任務和執行個體如何運作的詳細資訊。您可以使用此資訊對任務或服務部署失敗進行故障診斷。

當您開啟事件擷取時，您可以存取 Amazon ECS 在您選擇的保留期間內產生的所有事件，延伸超過最後 100 個未篩選事件或僅顯示 1 小時的已停止任務的原生限制。

運作方式

事件擷取使用 EventBridge 將事件儲存在預先定義的 Amazon CloudWatch Logs 日誌群組中。Amazon ECS 主控台提供預先建置的查詢和篩選選項，並將事件相互關聯，以直覺的形式提供任務生命週期。

您可以查詢和擷取下列類型的事件：

- 服務動作事件 - 協助識別佈建或資源分配問題
- 任務生命週期事件 - 協助識別任務或容器無法啟動或停止執行的原因

Amazon ECS 主控台可讓您一鍵設定事件擷取，並提供常用的查詢和篩選，而不需要您學習查詢語言或在多個主控台之間導覽。

Event types (事件類型)

事件擷取會將所有 Amazon ECS 產生的事件儲存在下列類別中：

任務狀態變更事件

容器停止和其他終止事件，可用於疑難排解或監控任務生命週期時間表。

服務動作

像是達到穩定狀態、任務置放失敗或資源限制等事件。

服務部署狀態變更

事件，例如進行中、已完成或失敗的部署，由斷路器和復原設定觸發，以監控服務部署的狀態。

容器執行個體狀態變更

對於 EC2 和 Amazon ECS 受管執行個體上的工作負載，事件會顯示連線和中斷連線狀態。

日誌群組組態

當您開啟事件擷取時，Amazon ECS 會自動建立下列資源：

- 名為 `/aws/events/ecs/containerinsights/${clusterName}/performance` 的 Amazon CloudWatch Logs 日誌群組
- EventBridge 規則可從 `aws.ecs` 來源擷取所有事件，並將其轉送至日誌群組

您可以為日誌群組指定從 1 天到 10 年的保留期。預設保留期間為 7 天。

考量事項

使用事件擷取時，請考慮下列事項：

- 事件擷取會儲存所有事件以簡化操作。您無法在 Amazon ECS 主控台中設定規則，以僅擷取特定事件。
- Amazon ECS 主控台提供預先定義的查詢條件。對於進階查詢，請使用 Amazon CloudWatch Logs Insights 直接查詢日誌群組。
- Amazon ECS 主控台不提供即時結尾功能。直接將 Amazon CloudWatch Logs 用於即時結尾。
- 當您停用事件擷取時，會刪除 EventBridge 規則。
- 事件擷取會產生 EventBridge 資料擷取、Amazon CloudWatch Logs 儲存和查詢執行的額外費用。

如需 EventBridge 定價的資訊，請參閱 [EventBridge 定價](#)。

如需 CloudWatch 定價的相關資訊，請參閱 [CloudWatch 定價](#)。

事件型疑難排解

使用 Amazon ECS 產生的事件來回答常見的疑難排解問題。

任務失敗分析

您可以檢閱STOPPED任務狀態變更事件、停止代碼和容器結束代碼，以判斷任務在執行時無法啟動或失敗的原因。

您可以檢閱置放失敗的服務動作事件和資源限制資訊，以判斷任務因資源限制而無法置放的原因。

常見的任務失敗案例

最常見的異常任務失敗與下列問題有關：

- CI/CD 服務部署失敗
- 自動擴展失敗
- 任務重新平衡失敗
- 異常容器結束，例如out-of-memory(OOM) 錯誤

異常任務失敗會產生具有 `EssentialContainerExited` 或 `TaskFailedToStart` 停止程式碼STOPPED的任務狀態變更事件。您可以依這些停止代碼進行篩選，以檢查容器執行和停止行為。

開啟現有 Amazon ECS 叢集的事件擷取

您可以在現有的 Amazon ECS 叢集上啟用事件擷取，透過 EventBridge 將 Amazon ECS 產生的事件存放在 Amazon CloudWatch Logs 中。此功能可協助您監控任務失敗、服務部署和其他叢集活動，並進行疑難排解。

啟用事件擷取後，Amazon ECS 會建立下列資源：

- 名為 `aws-logs-aws-events-ecs-containerinsights-{clusterName}-performance` 的 Amazon CloudWatch Logs 日誌群組
- 從 `aws.ecs` 來源擷取所有事件的 EventBridge 規則

歷史記錄索引標籤會顯示在叢集檢視中，可讓您查詢任務生命週期事件和服務動作。事件擷取會立即開始，並根據您指定的保留期間存放所有 Amazon ECS 產生的事件。

先決條件

- 現有的 Amazon ECS 叢集
- 修改叢集設定和建立 Amazon CloudWatch Logs 資源的適當 IAM 許可

使用主控台開啟事件擷取

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. 選取您要啟用事件擷取的叢集。

「叢集詳細資訊」頁面隨即顯示。

4. 選擇 Configuration (組態)。
5. 在 ECS 事件區段中，選擇開啟事件擷取。

隨即顯示開啟事件擷取對話方塊。

6. 針對過期事件，選擇 Amazon CloudWatch Logs 日誌群組的保留期間。預設值為 7 天。
7. 選擇 Turn on (開啟)。

檢視 Amazon ECS 服務和任務狀態變更事件

Amazon ECS 主控台提供事件擷取功能，可透過 EventBridge 將服務動作和任務狀態變更等 Amazon ECS 產生的事件儲存到 Amazon CloudWatch Logs。此功能包含具有篩選功能的查詢界面，可增強監控和故障診斷。

事件提供有關您的服務部署、服務、任務和執行個體如何運作的詳細資訊。您可以使用此資訊對任務或服務部署失敗進行故障診斷。

您可以使用下列任一條件來篩選事件：

- 部署 ID (僅適用於服務詳細資訊頁面)
- 開始時間
- 結束時間
- 服務名稱 (僅適用於叢集詳細資訊頁面、服務詳細資訊頁面，這會預設為目前的服務)
- 任務 ID

- 任務上次狀態
- 任務定義系列
- 任務定義修訂

在叢集層級檢視事件

1. 開啟主控台，網址為 <https://console.aws.amazon.com/ecs/v2>。
2. 選擇 Clusters (叢集)。

叢集清單頁面隨即顯示。

3. 選擇 叢集。

「叢集詳細資訊」頁面隨即顯示。

4. 在歷史記錄下，決定要檢視的事件。
 - a. 若要檢視服務動作事件，請選擇服務動作事件。
 - b. 若要檢視任務狀態變更事件，請選擇任務狀態變更事件。
 - c. (選用) 在查詢條件中，輸入您要檢視之事件的篩選條件。
5. 選擇 Run query (執行查詢)。

事件會顯示在清單中。

6. 若要檢視事件的完整詳細資訊，請選擇事件。

在服務層級檢視

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「叢集詳細資訊」頁面上的服務區段中，選擇所需服務。

服務詳細資訊頁面隨即顯示。

4. 在歷史記錄下，決定要檢視的事件。
 - a. 若要檢視服務動作事件，請選擇服務動作事件。
 - b. 若要檢視任務狀態變更事件，請選擇任務狀態變更事件。
 - c. (選用) 在查詢條件中，輸入您要檢視之事件的篩選條件。

5. 選擇 Run query (執行查詢)。

事件會顯示在清單中。

6. 若要檢視事件的完整詳細資訊，請選擇事件。

對 Amazon ECS 任務定義無效的 CPU 或記憶體錯誤進行疑難排解

使用 Amazon ECS API 註冊任務定義時 AWS CLI，或者，如果您指定無效的 `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，可能會傳回如下錯誤。

```
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 vCPU 表示。註冊任務定義時，此值會轉換為整數，表示 CPU 單位。此 `memory` 值可以在任務定義中以 MiB 或 GB 表示。註冊任務定義時，此值會轉換為整數，以 MiB 表示。

對於為 `requiresCompatibilities` 參數 (即使也指定了 EC2) 指定 FARGATE 的任務定義，您必須使用下表的其中一個值。這些值決定了 CPU 與記憶體參數的支援值範圍。

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
512 (.5 vCPU)	1 GB、2 GB、3 GB、4 GB	Linux

CPU 數值	記憶體數值	AWS Fargate 支援的作業系統
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
 Note 此選項需要 Linux 平台 1.4.0 或更新版本。		
16384 (16vCPU)	介於 32 GB 與 120 GB 之間，以 8 GB 為單位遞增	Linux
 Note 此選項需要 Linux 平台 1.4.0 或更新版本。		

對於 Amazon EC2 上託管的任務，支援的任務 CPU 值介於 0.25 vCPUs 和 192 vCPUs 之間。

EC2 與 Fargate 之間的 CPU 控制機制不同：

- 對於 Amazon EC2 上託管的任務：Amazon ECS 透過 CPU 期間與 CPU 配額來控制任務大小的 CPU 硬性限制。當您在任務定義中指定 vCPU 時，Amazon ECS 會將值轉換為適用於 cgroup 的 CPU 期間與 CPU 配額設定。
- 對於在 Fargate 上託管的任務：Amazon ECS 會透過 CPU 共用機制來控制 CPU 配置。在 Fargate 任務中，CPU 配額與期間值不會用於 CPU 限制。

對於 Amazon EC2 任務，CPU 配額會控制在特定 CPU 期間授予 cgroup 的 CPU 時間量。這兩個設定都以微秒表示。當 CPU 配額等於 CPU 期間時，表示 cgroup 最多可在一個 vCPU 上達到 100% 執

行率 (若為多個 vCPU，則按相應比例總和達 100%)。CPU 配額上限為 1000000us，CPU 期間下限為 1ms。您可以使用這些值來設定 CPU 計數的限制。在不變更 CPU 配額的情況下變更 CPU 期間時，實際生效的限制會與您在任務定義中指定的值不同。

100 毫秒期間允許 0.125 至 10 範圍的 vCPU。

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`。

`ecs-init.log` 檔案包含容器代理程式生命週期管理、組態與引導的相關資訊。

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 Windows 的 Amazon ECS 日誌收集器。如需詳細資訊，請參閱 Github 上的 [Amazon ECS Logs Collector for Windows](#)。

1. 連線到您的執行個體。
2. 開啟 PowerShell，然後使用管理權限執行下列命令。命令會下載指令碼並收集日誌。

```
Invoke-WebRequest -OutFile ecs-logs-collector.ps1 https://  
raw.githubusercontent.com/aws-labs/aws-ecs-logs-collector-for-windows/master/ecs-  
logs-collector.ps1  
.\ecs-logs-collector.ps1
```

您可以啟用 Amazon ECS 代理程式與 Docker 常駐程式的偵錯記錄。此選項允許指令碼在啟用偵錯模式之前收集日誌。指令碼會重新啟動 Docker 常駐程式與 Amazon ECS 代理程式，然後終止執行個體上執行的所有容器。在執行下列命令之前，請排空容器執行個體，並將任何重要任務移至其他容器執行個體。

執行下列命令以啟用記錄。

```
.\ecs-logs-collector.ps1 -RunMode debug
```

使用 Amazon ECS 日誌收集器收集容器日誌

Note

您無法在 Amazon ECS 受管執行個體上使用 Amazon ECS 日誌收集器。

如果您不確定如何收集您的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。此工具的 [Linux](#) 和 [Windows](#) 版本都可在 GitHub 上取得。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，這有助於對 AWS 支援案例進行故障診斷。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。

Note

在 Amazon Linux Amazon ECS 最佳化 AMIs 版本 20250909 和更新版本上，Amazon ECS 日誌收集器會預先安裝在 `/opt/amazon/ecs/ecs-logs-collector.sh` 並隨時可用，無需從 GitHub 下載。如需詳細資訊，請參閱 [ECS 最佳化 AMI 文件中的 ECS Logs Collector](#)。

目前，Amazon ECS 日誌收集器支援下列作業系統：

- Amazon Linux
- Red Hat Enterprise Linux
- Ubuntu
- Windows Server

執行 Linux 的 Amazon ECS 日誌收集器 (ECS 最佳化 AMI)

1. 連線到您的容器執行個體。
2. 執行指令碼以收集日誌並建立封存。

Note

若要為 Docker 常駐程式與 Amazon ECS 容器代理程式啟用偵錯模式，請在以下命令中新增 `--mode=enable-debug` 選項。這會重新啟動 Docker 常駐程式，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 容器執行個體](#)。

```
[ec2-user ~]$ sudo /opt/amazon/ecs/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。`collect.tgz` 檔案是所有日誌的壓縮封存檔，您可以與 共用以 AWS 支援 尋求診斷協助。

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/aws-labs/ecs-logs-collector/master/ecs-logs-collector.sh
```

3. 執行指令碼以收集日誌並建立封存。

```
$ sudo bash ./ecs-logs-collector.sh
```

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 the Amazon EC2 User Guide 中的 [Connect to your Windows instance using RDP](#)。
2. 使用 PowerShell 下載 Amazon ECS 日誌收集器指令碼。

```
Invoke-WebRequest -OutFile ecs-logs-collector.ps1 https://raw.githubusercontent.com/aws-labs/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

若要為 Docker 常駐程式與 Amazon ECS 容器代理程式啟用偵錯模式，請在以下命令中新增 `-RunMode debug` 選項。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 容器執行個體](#)。

```
.\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。`collect.tgz` 檔案是所有日誌的壓縮封存檔，您可以與 AWS Support 共用，以取得診斷協助。

透過代理程式自我檢查功能擷取 Amazon ECS 診斷詳細資訊

Amazon ECS 代理程式自我檢查 API 提供 Amazon ECS 代理程式與容器執行個體整體狀態的相關資訊。

例如，您可以透過代理程式自我檢查 API 取得任務容器的 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](#)。

下例顯示兩個任務，一個目前正在執行，一個已停止。

📌 Note

以下命令透過 `python -mjson.tool` 進行管道輸送，以取得更佳的可讀性。

```
curl http://localhost:51678/v1/tasks | python -mjson.tool
```

輸出：

```
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
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"
  }
]
}

```

在上例中，停止的任務 (*090eff9b-1ce3-4db6-848a-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 容器自我檢查](#)。

Amazon ECS 中的 Docker 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。如需有關所有可用 Docker 命令列公用程式的詳細資訊，請參閱 Docker 文件中的 [Docker CLI reference](#)。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](#)」。

在 Amazon ECS 中列出 Docker 容器

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。在以下範例中，只有 Amazon ECS 容器代理程式在執行中。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"	ecs-
19 seconds ago			
console-sample-app-static-6-internalecs-emptyvolume-source-c09288a6b0cba8a53700			
f7f1f8a7a245	busybox:buildroot-2014.02	"\sh -c '/bin/sh -c	ecs-
22 hours ago	Exited (137) 9 seconds ago		
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 中檢視 Docker 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。如果您已使用 `awslogs` 日誌驅動程式設定任務，即可在 CloudWatch Logs 中找到容器日誌。如需詳細資訊，請參閱 [將 Amazon ECS 日誌傳送至 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 Docker 容器

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 中設定 Docker 常駐程式的詳細輸出

如果使用 Docker 容器或映像時發生問題，您可以在 Docker 常駐程式中啟用偵錯模式。啟用偵錯功能可提供更多常駐程式的詳細輸出，您可透過該功能擷取從容器登錄檔 (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。其他作業系統請參閱 Docker 文件中的 [Enable debugging](#) 和 [Control and configure Docker with systemd](#)。

在 Amazon ECS 最佳化 Amazon Linux AMI 中使用 Docker 常駐程式偵錯模式

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\\\"}}\"
```

在 Amazon ECS 中對 Docker **API error (500): devmapper** 進行疑難排解

以下 Docker 錯誤指出，您的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 最佳化 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 版和更新版本) 您可以縮短已停止或結束的容器保留在您容器執行個體中的時間。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，或無法在容器移除之前檢視日誌。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移除非執行中的容器與閒置的映像。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)
```

- 您可以移除容器內閒置的資料區塊。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/"
```

對 Amazon ECS Exec 問題進行疑難排解

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。

透過 Exec 檢查程式進行驗證

ECS Exec 檢查程式指令碼提供了確認與驗證 Amazon ECS 叢集及任務是否符合使用 ECS Exec 功能之先決條件的方法。ECS Exec Checker 指令碼會代表您呼叫各種 APIs，以驗證您的 AWS CLI 環境和叢集和任務都已準備好供 ECS Exec 使用。工具需要最新版本的 AWS CLI 和 jq 可供使用的。如需詳細資訊，請參閱 GitHub 上的 [ECS Exec Checker](#)。

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 Exec 許可](#)。
- 尚未安裝或未執行 SSM 代理程式。
- 有一個適用於 Amazon ECS 的 Amazon VPC 端點，但沒有一個適用於 Systems Manager Session Manager 的介面。

對 Amazon ECS Anywhere 問題進行疑難排解

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) 註冊至 Amazon ECS 叢集。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：

- 必須擷取由啟用 ID 和啟用代碼組成的 AWS Systems Manager 啟用。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。請求 Systems Manager 啟用憑證時，需指定註冊數量限制與到期日。註冊限制指定使用啟用時可註冊的執行個體數目上限。註冊限制的預設值為 1 個執行個體。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 政策用於為您的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，請依照清除程序從執行個體清除 AWS 資源，以便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，如果未偵測到或未正確設定 NVIDIA 驅動程式，則會出現錯誤。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其網路連線 AWS，則在執行個體上執行的任務仍會繼續執行，除非手動停止。AWS 還原與的連線後，Amazon ECS 容器代理程式和 Systems Manager 代理程式在外部執行個體上使用的 AWS 登入資料會自動續約。如需用於外部執行個體與之間通訊之 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，或者設定為將容器日誌傳送到 CloudWatch Logs，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](#)。

對 Fargate 上的 Java 類別載入問題進行疑難排解

在 Fargate 上執行的 Java 應用程式可能會在平台更新後遇到類別載入問題，特別是當應用程式依賴於非確定性類別載入行為時。這可能會表現為相依性注入錯誤、Spring Boot 失敗，或先前部署中不存在的其他執行時期例外狀況。

徵狀

您可能會遇到下列症狀：

- Spring Boot 相依性注入錯誤
- ClassNotFoundException 或 NoClassDefFoundError 例外狀況
- 先前在 Fargate 上正常執行的應用程式現在會間歇性失敗
- 相同的容器映像可在 Amazon EC2 上運作，但在 Fargate 上失敗
- 相同容器映像在不同部署間行為不一致

原因

這些問題通常由下列原因所致：

- 非確定性類別載入：當底層平台變更檔案的存取或快取方式時，依賴於從 JAR 檔案載入類別順序的 Java 應用程式可能會失敗。
- 平台更新：Fargate 平台版本更新可能會變更底層檔案系統行為，影響探索與載入類別的順序。
- JAR 檔案排序相依性：在未經明確相依性管理，而隱含依賴特定 JAR 載入順序的應用程式。

Resolution

若要解決 Fargate 上的 Java 類別載入問題，請實作確定性類別載入實務：

立即修正

如果您需要立即解決方法：

1. 強制指定 JAR 載入順序：明確指定在應用程式的 classpath 組態中載入 JAR 檔案的順序。
2. 使用明確相依性管理：確保在建置組態 (Maven、Gradle 等) 中明確宣告所有相依性，而不是依賴暫時性相依性。

長期最佳實務

實作這些實務，防止未來會出現的類別載入問題：

1. 確保類別載入具有決定性：
 - 在建置檔案中使用明確的相依性宣告

- 避免依賴 classpath 掃描順序
- 使用相依性管理工具來解決版本衝突
- 使用 Java 虛擬機器 (JVM) 選項，例如 `-verbose:class` 來取得 JVM 所載入類別的相關資訊。

2. Spring Boot 應用程式：

- 將 `@ComponentScan` 搭配明確的基本套件使用
- 明確設定 Bean 以避免自動設定衝突
- 使用 `@DependsOn` 註釋來控制 Bean 初始化順序

3. 建置組態：

- 使用 Maven 或 Gradle 中的相依性管理區段
- 排除造成衝突的暫時性相依性
- 使用 Maven Enforcer 外掛程式等工具來偵測相依性問題

4. 測試：

- 使用不同的 JVM 實作測試應用程式
- 執行模擬不同部署環境的整合測試
- 使用工具分析開發期間的 classpath 衝突

預防

若要防止未來部署中出現 Java 類別載入問題：

- 遵循決定性類別載入實務：設計應用程式，使其不依賴從 classpath 載入類別的順序。
- 使用明確相依性管理：一律在建置組態中明確宣告所有必要的相依性及其版本。
- 跨環境測試：定期測試不同環境與平台版本的應用程式，及早識別潛在問題。
- 監控平台更新：隨時掌握 Fargate 平台更新，並在新平台版本影響生產工作負載之前測試應用程式。

AWS Fargate 調節配額

AWS Fargate 會使用 AWS 每個區域的[權杖儲存貯體演算法](#)，將 Amazon ECS 任務和 Amazon EKS Pod 啟動率限制為配額（先前稱為限制）。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 中對 RunTask API 進行限流

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 會根據區域限制每個 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 RunTask API 的 API 限流的詳細資訊，請參閱 Amazon ECS API Reference 中的 [API request throttling](#)。

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，客戶將看到任務和 Pod 啟動速率與上述配額相比的變化。

在 Fargate 中調整速率配額

您可為 AWS 帳戶請求增加 Fargate 速率調節配額。如需詳細資訊，請參閱《Service Quotas 使用者指南》中的 [請求增加配額](#)。

對 Amazon ECS 受管執行個體進行疑難排解

使用以下程序對 Amazon ECS 受管執行個體進行故障診斷，包括常見問題、診斷技術和解決步驟。

先決條件

對 Amazon ECS 受管執行個體進行故障診斷之前，請確定您已備妥下列要求。

- AWS CLI 已安裝並設定適當的許可

如需詳細資訊，請參閱 AWS Command Line Interface User Guide 中的 [Installing or updating to the latest version of the AWS Command Line Interface](#)。

- 使用 Amazon ECS 受管執行個體容量提供者存取叢集。如需詳細資訊，請參閱[the section called “為 Amazon ECS 受管執行個體建立叢集”](#)。

常見的疑難排解案例：

檢視 Amazon ECS 受管執行個體容器代理程式日誌

您可以透過連線至在執行個體中執行的特權容器，在 Amazon ECS 受管執行個體中檢視這些 Amazon ECS 日誌檔案。

診斷步驟

部署具有權限和 Linux 功能的偵錯容器做為 Amazon ECS 任務：

設定下列環境變數。

將 *user-input* 取代為實際值。

```
export ECS_CLUSTER_NAME="your-cluster-name"
export AWS_REGION="your-region"
export ACCOUNT_ID="your-account-id"
```

使用名為 `node-debugger.json` 的 CLI JSON 檔案建立任務定義。

```
cat << EOF > node-debugger.json
{
  "family": "node-debugger",
  "taskRoleArn": "arn:aws:iam::${ACCOUNT_ID}:role/ecsTaskExecutionRole",
  "executionRoleArn": "arn:aws:iam::${ACCOUNT_ID}:role/ecsTaskExecutionRole",
```



```
"cpu": "256",
"memory": "1024",
"networkMode": "host",
"pidMode": "host",
"requiresCompatibilities": ["MANAGED_INSTANCES", "EC2"],
"containerDefinitions": [
  {
    "name": "node-debugger",
    "image": "public.ecr.aws/amazonlinux/amazonlinux:2023",
    "essential": true,
    "privileged": true,
    "command": ["sleep", "infinity"],
    "healthCheck": {
      "command": ["CMD-SHELL", "echo debugger || exit 1"],
      "interval": 30,
      "retries": 3,
      "timeout": 5
    },
    "linuxParameters": {
      "initProcessEnabled": true
    },
    "mountPoints": [
      {
        "sourceVolume": "host-root",
        "containerPath": "/host",
        "readOnly": false
      }
    ],
    "logConfiguration": {
      "logDriver": "awslogs",
      "options": {
        "awslogs-group": "/aws/ecs/node-debugger",
        "awslogs-create-group": "true",
        "awslogs-region": "${AWS_REGION}",
        "awslogs-stream-prefix": "ecs"
      }
    }
  }
],
"volumes": [
  {
    "name": "host-root",
    "host": {
      "sourcePath": "/"
```

```
    }  
  }  
]  
}  
EOF
```

註冊，然後執行任務。執行下列命令。

```
aws ecs register-task-definition --cli-input-json file://node-debugger.json  
  
TASK_ARN=$(aws ecs run-task \  
  --cluster $ECS_CLUSTER_NAME \  
  --task-definition node-debugger \  
  --enable-execute-command \  
  --capacity-provider-strategy capacityProvider=managed-instances-default,weight=1 \  
  --query 'tasks[0].taskArn' --output text)  
  
# Wait for task to be running  
aws ecs wait tasks-running --cluster $ECS_CLUSTER_NAME --tasks $TASK_ARN
```

連線至容器。執行下列命令。

```
aws ecs execute-command \  
  --cluster $ECS_CLUSTER_NAME \  
  --task $TASK_ARN \  
  --container node-debugger \  
  --interactive \  
  --command "/bin/sh"
```

檢查 Amazon ECS 代理程式日誌：

在容器的互動式工作階段中，執行下列命令：

```
# Install required tools  
yum install -y util-linux-core  
  
# View ECS agent logs  
nsenter -t 1 -m -p cat /var/log/ecs/ecs-agent.log | tail -50  
  
# Check agent registration  
nsenter -t 1 -m -p grep "Registered container instance" /var/log/ecs/ecs-agent.log
```

Example Output:

```
{"level":"info","time":"2025-10-16T12:39:37.665","msg":"Registered container instance with cluster!"}

# Verify capabilities
nsenter -t 1 -m -p grep "Response contained expected value for attribute" /var/log/ecs/ecs-agent.log
```

檢查客服人員指標：

執行下列命令以檢視日誌。

```
# View metrics logs
nsenter -t 1 -m -p cat /var/log/ecs/metrics.log | tail -20
```

任務置放問題

以下是任務置放問題的症狀：

- 任務卡在 PENDING 狀態
- 任務無法在 Amazon ECS 受管執行個體上啟動
- 資源不足錯誤

診斷步驟

執行下列命令來診斷任務置放問題，並收集叢集容量、容器執行個體和系統服務的相關資訊：

```
# Check cluster capacity
aws ecs describe-clusters --clusters cluster-name --include STATISTICS

# Check cluster capacity providers
aws ecs describe-clusters --clusters cluster-name --include STATISTICS --query 'clusters[].capacityProviders'

# List container instances
aws ecs list-container-instances --cluster cluster-name

# Check container instance details
aws ecs describe-container-instances --cluster cluster-name --container-instances container-instance-arn
```

```
# Check container instance remaining resources CPU/Mem
aws ecs describe-container-instances --cluster $ECS_CLUSTER_NAME --container-
instances container-instance-arn --query 'containerInstances[].remainingResources'

# Check container instance Security Group
aws ecs describe-container-instances --cluster $ECS_CLUSTER_NAME --container-
instances container-instance-arn --query 'containerInstances[].ec2InstanceId' --output
text
aws ec2 describe-instances --instance-ids instance-id --query
'Reservations[0].Instances[0].SecurityGroups'
aws ec2 describe-security-groups --group-ids security-group-id
```

系統服務監控：

```
# Check Containerd status
nsenter -t 1 -m -p systemctl status containerd.service

# Check Amazon ECS container agent status
nsenter -t 1 -m -p systemctl status ecs
```

Resolution

若要解決任務置放問題，請依照下列步驟確保適當的組態和容量：

- 驗證任務資源需求與可用容量
- 檢查置放限制條件和策略
- 確保已設定 Amazon ECS 受管執行個體容量提供者
- 確保任務和容器執行個體安全群組具有允許 Amazon ECS 代理程式管理端點流量的傳出規則

聯網問題

以下是聯網問題的摘要：

- 無法連線到外部服務的任務
- DNS 解析問題

診斷步驟

網路連線測試：

從偵錯容器執行下列命令：

 Note

確認連接至容量提供者或 Amazon ECS 任務的安全群組允許流量。

```
# Install DNS Utility
yum install bind-utils -y

# Test DNS resolution
nslookup amazon.com

# Test external connectivity
curl -I https://amazon.com
```

資源限制

以下是聯網問題的摘要：

- 由於記憶體限制而終止的任務
- CPU 限流
- 磁碟空間問題

診斷步驟

執行命令來監控資源和容器限制。

資源監控：

```
# Check memory usage
nsenter -t 1 -m -p free -h

# Check disk usage
nsenter -t 1 -m -p lsblk

# Check disk usage
nsenter -t 1 -m -p df -h
```

容器限制：

```
# Check OOM kills
nsenter -t 1 -m -p dmesg | grep -i "killed process"
```

容器執行個體代理程式中斷連線問題

以下是容器執行個體代理程式中斷連線問題的症狀：

- 在 Amazon ECS 主控台中顯示為中斷連線的容器執行個體
- 任務無法放置在特定執行個體上
- 日誌中的客服人員註冊失敗

診斷步驟

如果在主機上執行 ECS Exec 可存取的現有權限任務，請執行下列命令來診斷代理程式連線問題：

```
# check service status
nsenter -t 1 -m -p systemctl restart ecs
nsenter -t 1 -m -p systemctl restart containerd

# restart stopped services
nsenter -t 1 -m -p systemctl restart ecs
nsenter -t 1 -m -p systemctl restart containerd
```

否則，請強制取消註冊 Amazon ECS 受管執行個體。執行以下命令：

```
# list ECS Managed Instance container
aws ecs list-container-instances --cluster managed-instances-cluster --query
'containerInstanceArns' --output text

# deregister the specific container instance
aws ecs deregister-container-instance \
  --cluster $ECS_CLUSTER_NAME \
  --container-instance container-instance-arn \
  --force
```

Resolution

若要解決客服人員中斷連線問題，請遵循下列步驟：

- 驗證容器執行個體的 IAM 角色許可

- 檢查安全群組規則允許傳出 HTTPS 流量到 ECS 端點
- 確保與 AWS 服務的網路連線
- 視需要重新啟動 ECS 代理程式服務：`nsenter -t 1 -m -p systemctl restart ecs`
- 確認 `/etc/ecs/ecs.config` 中的 `ECS_CLUSTER` 組態與您的叢集名稱相符

Amazon ECS 受管執行個體中的日誌分析

系統日誌

使用下列命令來檢查系統日誌，並識別受管執行個體的潛在問題：

```
# Check system messages
nsenter -t 1 -m -p journalctl --no-pager -n 50

# Check kernel logs
nsenter -t 1 -m -p dmesg | tail -20

# Check for disk space errors
nsenter -t 1 -m -p journalctl --no-pager | grep -i "no space\|disk full\|enospc"
```

使用 EC2 AWS CLI 從 Amazon ECS 受管執行個體取得主控台輸出

使用 Amazon EC2 執行個體 ID 擷取主控台輸出。

將 *user-input* 取代為實際值。

```
aws ec2 get-console-output --instance-id instance-id --latest --output text
```

清除

執行下列動作來停止轉移任務並取消註冊任務定義。

```
# Stop debug task
aws ecs stop-task --cluster $ECS_CLUSTER_NAME --task $TASK_ARN

# Deregister task definition (optional)
aws ecs deregister-task-definition --task-definition node-debugger
```

其他資源

如需疑難排解 Amazon ECS 受管執行個體的詳細資訊，請參閱下列資源：

- [the section called “Amazon ECS 受管執行個體錯誤”](#)
- [疑難排解](#)
- [the section called “容器代理程式組態”](#)
- [the section called “使用 ECS Exec 監控 Amazon ECS 容器”](#)

對 Amazon ECS 受管執行個體進行疑難排解

使用 Amazon ECS 受管執行個體啟動任務時，Amazon ECS 會先嘗試將任務置放在現有容量上，並為無法置放的任務請求額外容量。如果執行個體佈建失敗，Amazon EC2 請求 ID 會包含在任務失敗訊息中。您可透過此請求 ID 在 CloudTrail 中查詢失敗請求的詳細資訊，進一步進行疑難排解。

Note

如果您選擇套用最低權限許可，並為執行個體描述檔指定自己的許可，而不是使用 AmazonECSInstanceRolePolicyForManagedInstances 受管政策，您可以新增下列許可，以協助疑難排解 Amazon ECS 受管執行個體的任務相關問題：

- `ecs:StartTelemetrySession`
- `ecs:PutSystemLogEvents`

任務定義與 Amazon ECS 受管執行個體不相容

常見原因

當任務定義包含 Amazon ECS 受管執行個體不支援的參數或組態時，就會發生此錯誤。常見的不相容問題包括不支援的網路模式、任務角色或資源需求。

Resolution

1. 確認任務定義使用設定為 `MANAGED_INSTANCES` 的 `requiresCompatibilities`。
2. 確保任務定義使用 `awsvpc` 網路模式。
3. 檢查 CPU 與記憶體值是否在 Amazon ECS 受管執行個體的支援範圍內。
4. 檢閱詳細錯誤訊息，了解具體的不相容問題詳情。

不與叢集相關聯的容量提供者

常見原因

當容量提供者策略中指定的容量提供者未與叢集建立關聯或不存在時，就會發生此錯誤。

Resolution

1. 確認容量提供者存在於帳戶與區域中。
2. 透過 Amazon ECS 主控台或 CLI 將容量提供者與叢集建立關聯。
3. 確保容量提供者在使用之前處於 ACTIVE 狀態。

基礎結構角色許可錯誤

常見原因

當 Amazon ECS 基礎結構角色缺乏代表您執行 Amazon EC2 操作的必要許可，或該角色因信任關係問題而無法擔任時，就會發生此錯誤。

Resolution

1. 確認基礎結構角色與 Amazon ECS 具有適當的信任關係。
2. 確保角色具有必要的 Amazon EC2 許可，包括 `ec2:RunInstances`、`ec2:DescribeInstances`、`iam:PassRole`。
3. 如需特定許可的詳細資訊，請檢查 CloudTrail 中的編碼授權失敗訊息。
4. 更新角色政策，納入錯誤訊息中識別到的缺少許可。

VcpuLimitExceeded 錯誤

常見原因

當您已達到目前區域中執行個體類型系列的 vCPU 服務配額時，就會發生此錯誤。在容量可用之前，Amazon ECS 受管執行個體無法啟動額外的執行個體。

Resolution

1. 透過 AWS Support Center 請求增加受影響執行個體類型系列的服務配額。
2. 建議使用屬於不同 vCPU 配額類別的不同執行個體類型。

3. 終止閒置的 Amazon EC2 執行個體，釋放 vCPU 容量。
4. 檢閱容量提供者組態，使用 vCPU 需求較低的執行個體類型。

InsufficientCapacity 與相關容量錯誤

常見原因

當 AWS 沒有足夠的容量來滿足您的執行個體請求時，就會發生這些錯誤。這可能包括請求的可用區域中執行個體容量、位址容量或磁碟區容量不足。

Resolution

1. 在容量提供者中設定多個子網路，藉此嘗試在不同可用區域中啟動執行個體。
2. 建議使用可能具有更多可用容量的不同執行個體類型。
3. 當容量可用性經常變更時，請等待並重試操作。
4. 對於持續性容量需求，建議採用預留執行個體或 Savings Plans。

UnauthorizedOperation 錯誤

常見原因

當 Amazon ECS 服務沒有執行 Amazon EC2 操作或傳遞 IAM 角色的必要許可時，就會發生此錯誤。常見案例包括缺少執行個體設定檔的 `ec2:RunInstances` 許可或 `iam:PassRole` 許可。

Resolution

1. 確認 Amazon ECS 基礎結構角色具有啟動 Amazon EC2 執行個體的必要許可。
2. 確保基礎結構角色具有 Amazon ECS 受管執行個體使用的執行個體設定檔 `iam:PassRole` 許可。
3. 如需特定許可的詳細資訊，請檢查 CloudTrail 中的編碼授權失敗訊息。
4. 更新角色政策，納入錯誤訊息中識別到的缺少許可。

任務等待容量逾時

常見原因

當執行個體啟動並註冊至叢集所需的時間超過預期時，就會發生此錯誤。這可能是由 Amazon EC2 容量限制、執行個體啟動失敗或網路連線問題所致。

Resolution

1. 檢查所在區域中 Amazon EC2 服務運作狀態是否有任何持續的問題。
2. 確認子網路有足夠的 IP 位址可用。
3. 確保安全群組允許 Amazon ECS 代理程式通訊所需的流量。
4. 建議使用多個可用區域來改善容量可用性。
5. 重試任務啟動操作，因為容量限制通常是暫時的。

網路組態錯誤

常見原因

當任務的網路需求與容量提供者的網路組態不相符，例如 VPC 不相符或缺少網路組態時，就會發生這些錯誤。

Resolution

1. 確認容量提供者已使用正確的 VPC 與子網路設定。
2. 確保安全群組與子網路屬於相同的 VPC。
3. 檢查任務定義的網路組態是否與容量提供者相容。
4. 使用正確的網路設定更新容量提供者組態。

由於執行個體卡住，無法刪除容量提供者

常見原因

當 Amazon ECS 受管執行個體卡在 ACTIVE 或 DRAINING 狀態，但執行個體上沒有執行中的任務時，就會發生這些錯誤。

Resolution

若要允許刪除容量提供者以繼續，您可以使用下列命令強制取消註冊停滯的執行個體。

```
aws ecs deregister-container-instance \
  --cluster arn:aws:ecs:us-east-1:111122223333:cluster/MyCluster \
  --container-instance arn:aws:ecs:us-east-1:111122223333:container-instance/
a1b2c3d4-5678-90ab-cdef-11111EXAMPLE \
  --force
```

對 Amazon ECS Amazon ECS 受管執行個體錯誤進行疑難排解

以下是部分 Amazon ECS 受管執行個體錯誤訊息以及可採取的錯誤修正動作。

如果沒有叢集名稱，則不支援 Amazon ECS 受管執行個體提供者

當您嘗試使用 Amazon ECS 受管執行個體提供者參數建立容量提供者，但沒有叢集欄位時，就會發生此錯誤。

若要解決此問題，請在建立容量提供者時指定有效的叢集名稱。

建立 Amazon ECS 受管執行個體容量提供者時，需要提供 Amazon ECS 受管執行個體提供者詳細資訊

當您嘗試使用 Amazon ECS 受管執行個體提供者建立容量提供者，但未提供實際物件時，就會出現此錯誤。

若要修正此問題，請指定有效的 Amazon ECS 受管執行個體提供者詳細資訊，然後再試一次。

Amazon ECS 受管執行個體容量提供者必須指定執行個體設定檔

當您嘗試使用 Amazon ECS 受管執行個體提供者建立容量提供者，但未提供 `Ec2InstanceProfile` 時，就會看到此錯誤。

若要解決此問題，請為 Amazon ECS 受管執行個體容量提供者指定有效的 EC2 執行個體設定檔。如需詳細資訊，請參閱[the section called “Amazon ECS 受管執行個體執行個體設定檔”](#)。

Amazon ECS 受管執行個體容量提供者必須指定 `InfrastructureRole`

當您嘗試使用 Amazon ECS 受管執行個體提供者建立容量提供者，但未提供 `InfrastructureRole` 時，就會顯示此訊息。

若要修正此問題，請為 Amazon ECS 受管執行個體容量提供者指定有效的基礎結構角色。如需詳細資訊，請參閱[the section called “基礎結構 IAM 角色”](#)。

Amazon ECS 受管執行個體容量提供者必須指定網路組態

當您嘗試使用 Amazon ECS 受管執行個體提供者建立容量提供者，但未提供網路組態時，就會遇到此錯誤。

若要解決此問題，請為 Amazon ECS 受管執行個體容量提供者指定具有非空白子網路的有效網路組態。如需詳細資訊，請參閱[the section called “Amazon ECS 受管執行個體的任務聯網”](#)。

沒有執行個體類型符合 Amazon ECS 受管執行個體容量提供者中指定的執行個體需求

當您嘗試使用 Amazon ECS 受管執行個體提供者建立容量提供者，但沒有 EC2 執行個體類型符合執行個體需求時，就會發生這種情況。

若要解決此問題，請檢閱並調整執行個體需求，使其與可用的 EC2 執行個體類型相符。

指定的基礎結構角色 ARN 無效

當 InfrastructureRole 未遵循特定 ARN 格式時，就會發生此錯誤。

預期格式：`arn:partition:iam::account-id:role/role-name`。指定有效的角色 ARN，然後再試一次。如需詳細資訊，請參閱[the section called “基礎結構 IAM 角色”](#)。

指定的執行個體設定檔角色 ARN 無效

當執行個體設定檔未遵循特定 ARN 格式時，就會發生此錯誤。

預期格式：`arn:partition:iam::account-id:instance-profile/profile-name`。指定有效的角色 ARN，然後再試一次。如需詳細資訊，請參閱[the section called “Amazon ECS 受管執行個體執行個體設定檔”](#)。

指定的安全群組 ID 無效

當安全群組 ID 未遵循特定格式時，就會發生此錯誤。

預期格式：`sg-xxxxxxx` 或 `sg-xxxxxxxxxxxxxxxxxxxx` [8 或 17 個字元，包括字母 (小寫) 與 "sg-" 後面的數字]。指定安全群組 ID，然後再試一次。

指定的子網路 ID 無效

當子網路 ID 未遵循特定格式時，就會發生此錯誤。

預期格式：`subnet-xxxxxxx` 或 `subnet-xxxxxxxxxxxxxxxxxxxx` [8 或 17 個字元，包括字母 (小寫) 與 "subnet-" 後面的數字]。指定子網路 ID，然後再試一次。

Amazon ECS 受管執行個體容量提供者必須指定具有非空白子網路的網路組態

當您嘗試使用 Amazon ECS 受管執行個體提供者與具有空子網路的網路組態建立容量提供者時，就會發生此錯誤。

若要解決此問題，請為 Amazon ECS 受管執行個體容量提供者指定具有非空白子網路的有效網路組態。

Amazon ECS 受管執行個體容量提供者的標籤數不能超過允許的最大值

當您嘗試建立的容量提供者超過允許的標籤數量上限 (45) 時，就會發生此錯誤。

若要解決此問題，請將標籤數量減少至 45 個或更少，然後再試一次。

propagateTags 的值無效

當您嘗試建立具有無效 propagateTags 值的容量提供者時，會發生此錯誤。

值必須是其中一個有效的 propagateTags 值。指定有效的值，然後再試一次。

具有叢集範圍限定的指定容量提供者已存在

當您嘗試建立、更新或刪除沒有叢集名稱的 Amazon ECS 受管執行個體容量提供者，但有現有的叢集範圍限定容量提供者時，就會發生此錯誤。

若要變更容量提供者組態或刪除容量提供者，請使用叢集參數更新或刪除容量提供者。

指定的容量提供者已存在於帳戶或其他叢集中

當您嘗試建立、更新或刪除具有叢集欄位的 Amazon ECS 受管執行個體容量提供者，但有現有的帳戶範圍限定容量提供者或不同叢集的現有叢集範圍限定容量提供者時，就會發生此錯誤。

若要解決此問題，請使用不同的容量提供者名稱，或使用現有的容量提供者。

容量提供者作用中叢集名稱與請求中的叢集名稱不相符

當容量提供者的作用中叢集名稱與請求中指定的叢集名稱不相符時，就會發生此錯誤。

確保請求中的叢集名稱與容量提供者的相關聯叢集相符。

容量提供者不是 Amazon ECS 受管執行個體容量提供者

當您嘗試在需要容量提供者的環境中使用非 Amazon ECS 受管執行個體容量提供者的容量提供者時，就會發生此錯誤。

確定請求使用有效的 Amazon ECS 受管執行個體容量提供者。

CapacityProvider 名稱不得為空白

當您嘗試建立容量提供者而不指定容量提供者名稱時，就會發生此錯誤。

指定有效的容量提供者名稱，然後再試一次。

更新容量提供者時需要名稱

當您嘗試更新容量提供者而不指定容量提供者名稱時，就會發生此錯誤。

指定有效的名稱，然後再試一次。

標籤不能為 null 或具有 null 元素

當您嘗試使用 null 值標記容量提供者時，就會發生此錯誤。

確保所有標籤值都已正確指定，不是 null。

Amazon ECS API 失敗原因

當您透過 Amazon ECS API、主控台或 AWS CLI 觸發的 API 動作退出時顯示 failures 錯誤訊息，以下內容可能有助於對原因進行疑難排解。失敗時會傳回原因，以及與失敗相關聯的資源的 Amazon Resource Name (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` 參數傳送至正確的區域。

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。

API 動作	失敗原因或停止原因	原因
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 單位需求大於映射至容量提供者的 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。當任務定義中的記憶體數量需求大於映射至容量提供者的 Auto Scaling 群組中定義之 Amazon EC2 執行個體上支援的記憶體數量時，通常會發生這種情況。您需要檢查容量提供者組態。

API 動作	失敗原因或停止原因	原因
	<code>TaskFailedToStart: AGENT</code>	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)。
	<code>TaskFailedToStart: MemberOf placement constraint unsatisfied</code>	沒有符合任務定義中定義之置放限制的容器執行個體。
	<code>TaskFailedToStart: ATTRIBUTE</code>	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 <code>awsvpc</code> 網路模式，但您使用 <code>ecs.capability.task-eni</code> 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「Fargate 的 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
	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。
	TASK_NOT_VALID	指定的服務不是 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。

API 動作	失敗原因或停止原因	原因
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。如需有關每個執行個體類型支援多少網路介面的詳細資訊，請參閱 Amazon EC2 User Guide 中的 IP Addresses Per Network Interface Per Instance Type。 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 <code>awsVpcConfiguration</code> 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 <code>awsvpc</code> 網路模式，但您使用 <code>ecs.capability.task-eni</code> 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「Fargate 的 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	找不到您嘗試啟動任務所在之容器執行個體。檢查是否指定了錯誤的叢集或區域，或拼錯了容器執行個體 ARN 或 ID。
	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。
StopServiceDeployment	ECS deployment failed	詐騙帳戶執行了 StopServiceDeployment API。
TagResource	InvalidParameterException	您要標記之服務的 ARN 採用短格式。您必須遷移至長格式。如需有關如何遷移的資訊，請參閱 將 Amazon ECS 短服務 ARN 移轉到完整 ARN 。
UpdateTaskProtection	DEPLOYMENT_BLOCKED	無法設定任務保護，因為一或多個受保護的任務會阻止服務部署達到穩定狀態。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	指定的服務不是 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

除了此處描述的失敗案例，API 操作也可能因例外狀況而失敗，從而導致錯誤回應。如需此類例外狀況的清單，請參閱[常見錯誤](#)。

使用 Amazon Q Developer 進行故障診斷

您可以在 [Amazon ECS 主控台中使用 Amazon Q Developer](#)，以協助診斷和解決 Amazon ECS 資源的問題。對於容器、任務、服務、部署和任務定義的特定錯誤和狀態訊息，主控台會顯示使用 Amazon Q Developer 檢查選項。當您選擇此選項時，Amazon Q Developer 會在內容中分析問題，並建議可能的原因和修復步驟。

所需的許可

- 檢視您要疑難排解之 Amazon ECS 資源的許可，例如叢集、服務、任務和任務定義。
- 在 主控台中使用 [Amazon Q Developer 的許可](#)。
- (建議) 檢視相關日誌和指標的許可，例如：
 - CloudWatch Logs
 - CloudWatch

程序

- 在 開啟 主控台<https://console.aws.amazon.com/ecs/v2>。
- 決定您要疑難排解的資源。

資源	步驟
任務	- 在叢集頁面上，選擇叢集。 - 選擇 Tasks (任務) 索引標籤。 - 選擇您要調查的任務。
容器	- 在叢集頁面上，選擇叢集。 - 選擇任務索引標籤，然後選擇任務。 - 在容器區段中，選擇您要調查的容器。

資源	步驟
服務和部署	1. 在叢集頁面上，選擇叢集。 2. 選擇 Services (服務) 索引標籤。 3. 選擇 服務，然後檢閱部署區段。
任務定義	1. 在導覽窗格中，選擇 Task Definitions (任務定義)。 2. 選擇任務定義系列，然後選擇您要調查的修訂。

3. 在資源詳細資訊頁面上，找到描述問題的狀態或運作狀態原因。
4. 按一下狀態原因以開啟快顯視窗。
5. 如果可用，請選擇使用 Amazon Q Developer 檢查。
6. 檢閱 Amazon Q Developer 提供的說明和建議的修復步驟。根據您的環境套用組態或操作變更。

考量事項

搭配使用 Amazon Q Developer 與 Amazon ECS 時，請考慮下列事項：

- 按鈕可用性 - 僅針對遇到潛在問題的資源顯示「檢查 Amazon Q Developer」按鈕。此選項不適用於運作狀態良好的資源。
- 唯讀操作 - Amazon Q Developer 整合僅執行讀取操作。它不會進行變動或寫入動作。
- 跨區域處理 - Amazon Q Developer 可能會跨 AWS 區域處理資料，以提供 AI 支援的分析。如需跨區域處理的詳細資訊，請參閱 [Amazon Q Developer 中的跨區域處理](#)。
- 僅限主控台 - 此整合只能透過主控台使用。它無法透過 AWS CLI、AWS APIs 或基礎設施做為程式碼工具使用。

進一步了解

如需使用 Amazon Q Developer 的詳細資訊，請參閱 [與 Amazon Q Developer 聊天相關 AWS](#)。

Amazon Elastic Container Service 的安全性

的雲端安全 AWS 是最高優先順序。身為 AWS 客戶，您可以受益於資料中心和網路架構，這些架構是為了滿足最安全敏感組織的需求而建置。

安全性是 AWS 與您之間共同責任。[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 Cloud AWS 中執行 AWS 服務的基礎設施。AWS 也為您提供可安全使用的服務。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 的合規計畫，請參閱[合規計畫範圍內的 AWS 服務](#)。
- 雲端的安全性 – 您的責任取決於您使用 AWS 的服務。您也必須對其他因素負責，包括資料的機密性、您的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。您也會了解如何使用其他 AWS 服務來協助您監控和保護 Amazon ECS 資源。

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 聯邦資訊處理標準 \(FIPS-140\)](#)
- [Amazon Elastic Container Service 的基礎設施安全性](#)
- [AWS Amazon ECS 的共同責任模型](#)
- [Amazon ECS 受管執行個體的共同責任模式](#)
- [Amazon ECS 的網路安全最佳實務](#)
- [Amazon ECS 任務與容器安全最佳實務](#)

Amazon Elastic Container Service 的身分和存取管理

AWS Identity and Access Management (IAM) 是一種 AWS 服務，可協助管理員安全地控制對 AWS 資源的存取。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 是您可以免費使用 AWS 服務的。

主題

- [目標對象](#)
- [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 Elastic Container Service 的 受管政策](#)
- [使用 Amazon ECS 的服務連結角色](#)
- [適用於 Amazon ECS 的 IAM 角色](#)
- [Amazon ECS 主控台必要的許可](#)
- [Amazon ECS 藍/綠部署中 Lambda 函式所需的許可](#)
- [Amazon ECS 服務自動擴展所需的 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 最佳實務](#)

目標對象

使用方式 AWS Identity and Access Management (IAM) 會根據您的角色而有所不同：

- 服務使用者 — 若無法存取某些功能，請向管理員申請所需許可 (請參閱 [對 Amazon Elastic Container Service 身分和存取進行故障診斷](#))
- 服務管理員 — 負責設定使用者存取權並提交相關許可請求 (請參閱 [Amazon Elastic Container Service 如何與 IAM 搭配使用](#))
- IAM 管理員 — 撰寫政策以管理存取控制 (請參閱 [Amazon Elastic Container Service 身分型政策範例](#))

使用身分驗證

身分驗證是您 AWS 使用身分憑證登入 的方式。您必須以 AWS 帳戶根使用者、IAM 使用者或擔任 IAM 角色身分進行身分驗證。

您可以使用身分來源的登入資料，例如 AWS IAM Identity Center (IAM Identity Center)、單一登入身分驗證或 Google/Facebook 登入資料，以聯合身分的形式登入。如需有關登入的詳細資訊，請參閱《AWS 登入 使用者指南》中的[如何登入您的 AWS 帳戶](#)。

對於程式設計存取，AWS 提供 SDK 和 CLI 以密碼編譯方式簽署請求。如需詳細資訊，請參閱《IAM 使用者指南》中的[API 請求的AWS 第 4 版簽署程序](#)。

AWS 帳戶 根使用者

當您建立時 AWS 帳戶，您會從一個名為 AWS 帳戶 theroot 使用者的登入身分開始，該身分可完整存取所有 AWS 服務和資源。強烈建議不要使用根使用者來執行日常任務。有關需要根使用者憑證的任務，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

最佳實務是要求人類使用者使用聯合身分提供者，以 AWS 服務使用臨時憑證存取。

聯合身分是來自您的企業目錄、Web 身分提供者的使用者，或使用來自身分來源的 AWS 服務憑證存取 Directory Service。聯合身分會擔任角色，而該角色會提供臨時憑證。

若需集中化管理存取權限，建議使用 AWS IAM Identity Center。如需詳細資訊，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https://docs.aws.amazon.com/IAM/latest/UserGuide/id_users.html是一種身分具備單人或應用程式的特定許可權。建議以臨時憑證取代具備長期憑證的 IAM 使用者。如需詳細資訊，請參閱《IAM 使用者指南》中的[要求人類使用者使用聯合身分提供者來 AWS 使用臨時憑證存取](#)。

[IAM 群組](#)會指定 IAM 使用者集合，使管理大量使用者的許可權更加輕鬆。如需詳細資訊，請參閱《IAM 使用者指南》中的[IAM 使用者的使用案例](#)。

IAM 角色

IAM 角色https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles.html的身分具有特定許可權，其可以提供臨時憑證。您可以透過[從使用者切換到 IAM 角色（主控台）](#)或呼叫 AWS CLI 或 AWS API 操作來擔任角色。如需詳細資訊，請參閱《IAM 使用者指南》中的[擔任角色的方法](#)。

IAM 角色適用於聯合身分使用者存取、臨時 IAM 使用者許可、跨帳戶存取權與跨服務存取，以及在 Amazon EC2 執行的應用程式。如需詳細資訊，請參閱《IAM 使用者指南》中的[IAM 中的快帳戶資源存取](#)。

使用政策管理存取權

您可以透過建立政策並將其連接到身分或資源 AWS 來控制 AWS 中的存取。政策定義與身分或資源相關聯的許可。當委託人提出請求時 AWS，會評估這些政策。大多數政策會以 JSON 文件 AWS 的形式存放在中。如需進一步了解 JSON 政策文件，請參閱《IAM 使用者指南》中的 [JSON 政策概觀](#)。

管理員會使用政策，透過定義哪些主體可在哪些條件下對哪些資源執行動作，以指定可存取的範圍。

預設情況下，使用者和角色沒有許可。IAM 管理員會建立 IAM 政策並將其新增至角色，供使用者後續擔任。IAM 政策定義動作的許可，無論採用何種方式執行。

身分型政策

身分型政策是附加至身分 (使用者、使用者群組或角色) 的 JSON 許可政策文件。這類政策控制身分可對哪些資源執行哪些動作，以及適用的條件。如需了解如何建立身分型政策，請參閱《IAM 使用者指南》中的 [透過客戶管理政策定義自訂 IAM 許可](#)。

身分型政策可分為內嵌政策 (直接內嵌於單一身分) 與受管政策 (可附加至多個身分的獨立政策)。如需了解如何在受管政策及內嵌政策之間做選擇，請參閱《IAM 使用者指南》中的 [在受管政策與內嵌政策之間選擇](#)。

資源型政策

資源型政策是附加到資源的 JSON 政策文件。範例包括 IAM 角色信任政策與 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。您必須在資源型政策中 [指定主體](#)。

資源型政策是位於該服務中的內嵌政策。您無法在資源型政策中使用來自 IAM 的 AWS 受管政策。

其他政策類型

AWS 支援其他政策類型，可設定更多常見政策類型授予的最大許可：

- 許可界限 — 設定身分型政策可授與 IAM 實體的最大許可。如需詳細資訊，請參閱《IAM 使用者指南》中的 [IAM 實體許可界限](#)。
- 服務控制政策 (SCP) — 為 AWS Organizations 中的組織或組織單位指定最大許可。如需詳細資訊，請參閱《AWS Organizations 使用者指南》中的 [服務控制政策](#)。
- 資源控制政策 (RCP) — 設定您帳戶中資源可用許可的上限。如需詳細資訊，請參閱《AWS Organizations 使用者指南》中的 [資源控制政策 \(RCP\)](#)。

- 工作階段政策 — 在以程式設計方式為角色或聯合身分使用者建立臨時工作階段時，以參數形式傳遞的進階政策。如需詳細資訊，請參閱《IAM 使用者指南》中的[工作階段政策](#)。

多種政策類型

當多種類型的政策適用於請求時，產生的許可會更複雜而無法理解。若要了解如何 AWS 在涉及多種政策類型時決定是否允許請求，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。

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
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

若要全面了解 Amazon ECS 和其他 AWS 服務如何與大多數 IAM 功能搭配使用，請參閱《[AWS IAM 使用者指南](#)》中的與 IAM 搭配使用的 [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。如需了解如何建立身分型政策，請參閱《IAM 使用者指南》中的[透過客戶管理政策定義自訂 IAM 許可](#)。

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。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

支援資源型政策：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](#)。委託人可以包含帳戶、使用者、角色、聯合身分使用者或 AWS 服務。

如需啟用跨帳戶存取權，您可以在其他帳戶內指定所有帳戶或 IAM 實體作為資源型政策的主體。如需詳細資訊，請參閱《IAM 使用者指南》中的[IAM 中的快帳戶資源存取](#)。

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。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。最佳實務是使用其 [Amazon Resource Name \(ARN\)](#) 來指定資源。若動作不支援資源層級許可，使用萬用字元 (*) 表示該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，請使用萬用字元 (*)：

```
"Resource:arn:${Partition}:ecs:${Region}:${Account}:task-definition/  
${TaskDefinitionFamilyName}:*"
```

若要指定特定任務定義修訂版，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 元素會根據定義的條件，指定陳述式的執行時機。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。若要查看所有 AWS 全域條件索引鍵，請參閱《IAM 使用者指南》中的[AWS 全域條件內容索引鍵](#)。

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	Description	評估類型
aws:RequestTag/\${TagKey}	內容金鑰的格式為 "aws:RequestTag/ <i>tag-key</i>": "<i>tag-value</i> "，其中 <i>tag-key</i> 和 <i>tag-value</i> 是標籤金鑰和值對。 檢查標籤鍵/值對是否存在於 AWS 請求中。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" 。	String
aws:ResourceTag/\${TagKey}	內容金鑰的格式為 "aws:ResourceTag/ <i>tag-key</i>": "<i>tag-value</i> "，其中 <i>tag-key</i> 和 <i>tag-value</i> 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。	String
aws:TagKeys	此內容鍵的格式為 "aws:TagKeys": " <i>tag-key</i> "，其中 <i>tag-key</i> 是沒有值的標籤鍵清單 (例如，["Dept", "Cost-Center"])。 檢查 AWS 請求中存在的標籤索引鍵。	String
ecs:ResourceTag/\${TagKey}	內容金鑰的格式為 "ecs:ResourceTag/ <i>tag-key</i>": "<i>tag-value</i> "，其中 <i>tag-key</i> 和 <i>tag-value</i> 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。	String

條件索引鍵	Description	評估類型
ecs:account-setting	內容索引鍵的格式為 "ecs:account-setting": " <i>account-setting</i> "，其中 <i>account-setting</i> 為帳戶設定的名稱。	String
ecs:auto-assign-public-ip	內文索引鍵已被格式化為 "ecs:auto-assign-public-ip": " <i>value</i> "，其中 <i>value</i> 是 "true" 或 "false"。	字串
ecs:capacity-provider	內容索引鍵的格式為 "ecs:capacity-provider": " <i>capacity-provider-arn</i> "，其中 <i>capacity-provider-arn</i> 為容量提供者的 ARN。	ARN, Null
ecs:cluster	內容金鑰的格式為 "ecs:cluster": " <i>cluster-arn</i> "，其中 <i>cluster-arn</i> 是 Amazon ECS 叢集的 ARN。	ARN, Null
ecs:compute-compatibility	內容索引鍵的格式為 "ecs:compute-compatibility": " <i>compatibility-type</i> "，其中 <i>compatibility-type</i> 為 "FARGATE"、"EC2" 或 "EXTERNAL"。	String
ecs:container-instances	內容金鑰的格式為 "ecs:container-instances": " <i>container-instance-arns</i> "，其中 <i>container-instance-arns</i> 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": " <i>container-instance-</i> "，其中 <i>container-instance-</i> 是在任務定義中定義的 Amazon ECS 容器名稱。	String
ecs:enable-execute-command	內文索引鍵已被格式化為 "ecs:enable-execute-command": " <i>value</i> "，其中 <i>value</i> 是 "true" 或 "false"。	字串

條件索引鍵	Description	評估類型
ecs:enable-service-connect	內容索引鍵的格式為 "ecs:enable-service-connect": " <i>value</i> ", 其中 <i>value</i> 是 "true" 或 "false"。	String
ecs:enable-ebs-volumes	內容索引鍵的格式為 "ecs:enable-ebs-volumes": " <i>value</i> ", 其中 <i>value</i> 是 "true" 或 "false"。	String
ecs:enable-managed-tags	內容索引鍵的格式為 "ecs:enable-managed-tags": " <i>value</i> ", 其中 <i>value</i> 是 "true" 或 "false"。	String
ecs:enable-vpc-lattice	內容索引鍵的格式為 "ecs:vpc-lattice": " <i>value</i> ", 其中 <i>value</i> 是 "true" 或 "false"。	String
ecs:fargate-ephemeral-storage-kms-key	內容金鑰已格式化, "ecs:fargate-ephemeral-storage-kms-key": " <i>key</i> " 其中 ## 是 AWS KMS 金鑰的 ID。	String
ecs:namespace	內容索引鍵已格式化, "ecs:namespace": " <i>namespace-arn</i> " 其中 <i>namespace-arn</i> 是 AWS Cloud Map 命名空間的 ARN。	ARN, Null
ecs:propagate-tags	內容索引鍵的格式為 "ecs:propagate-tags": " <i>value</i> ", 其中 # 為 "TASK_DEFINITION"、"SERVICE" 或 "NONE"。	String
ecs:service	內容金鑰的格式為 "ecs:service": " <i>service-arn</i> ", 其中 <i>service-arn</i> 是 Amazon ECS 服務的 ARN。	ARN, Null
ecs:task-definition	內容金鑰的格式為 "ecs:task-definition": " <i>task-definition-arn</i> ", 其中 <i>task-definition-arn</i> 為 Amazon ECS 任務定義的 ARN。	ARN, Null
ecs:subnet	內容索引鍵的格式為 "ecs:subnet": " <i>subnets</i> ", 其中 ### 為使用 awsvpc 模式的任務或服務的子網路。	String

條件索引鍵	Description	評估類型
ecs:task	內容索引鍵的格式為 "ecs:task":" <i>task-arn</i> "，其中 <i>task-arn</i> 為 Amazon ECS 服務的 ARN。	ARN, Null
ecs:task-cpu	內容索引鍵的格式為 "ecs:task-cpu":" <i>task-cpu</i> "，其中 <i>task-cpu</i> 為任務 CPU，以整數表示，其中 1024 相當於 1 個 vCPU。	Integer
ecs:task-memory	內容索引鍵的格式為 "ecs:task-memory":" <i>task-memory</i> "，其中 <i>task-memory</i> 為以 MiB 表示的任務記憶體。	Integer

若要查看 Amazon ECS 條件索引鍵的清單，請參閱 Service Authorization Reference 中的 [Condition keys for Amazon Elastic Container Service](#)。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，依據稱為標籤的屬性來定義許可。您可以將標籤連接至 IAM 實體 AWS 和資源，然後設計 ABAC 政策，以便在委託人的標籤符合資源上的標籤時允許操作。

如需根據標籤控制存取，請使用 `aws:ResourceTag/key-name`、`aws:RequestTag/key-name` 或 `aws:TagKeys` 條件索引鍵，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[使用 ABAC 授權定義許可](#)。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

臨時登入資料提供 AWS 資源的短期存取權，當您使用聯合或切換角色時，會自動建立。AWS 建議您動態產生臨時登入資料，而不是使用長期存取金鑰。如需詳細資訊，請參閱《IAM 使用者指南》中的[IAM 中的臨時安全憑證與可與 IAM 搭配運作的 AWS 服務](#)。

Amazon ECS 的轉送存取工作階段

支援轉寄存取工作階段 (FAS)：是

轉送存取工作階段 (FAS) 使用呼叫的委託人許可 AWS 服務，並結合 AWS 服務請求向下游服務提出請求。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。如需詳細資訊，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的一種服務角色 AWS 服務。服務可以擔任代表您執行動作的角色。服務連結角色會出現在您的中 AWS 帳戶，並由服務擁有。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。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。

如需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 \(主控台\)](#)。

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 使用者檢視自身的許可](#)
- [Amazon ECS 叢集範例](#)
- [Amazon ECS 容器執行個體範例](#)
- [Amazon ECS 任務定義範例](#)
- [執行 Amazon ECS 任務範例](#)
- [啟動 Amazon ECS 任務範例](#)
- [列出並描述 Amazon ECS 任務範例](#)
- [建立 Amazon ECS 服務範例](#)

- [根據標籤描述 Amazon ECS 服務](#)
- [拒絕 Amazon ECS Service 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 受管政策並轉向最低權限許可 – 若要開始將許可授予您的使用者和工作負載，請使用將許可授予許多常見使用案例的 AWS 受管政策。它們可在您的 中使用 AWS 帳戶。我們建議您定義特定於使用案例 AWS 的客戶受管政策，以進一步減少許可。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](#)。
- 套用最低權限許可 – 設定 IAM 政策的許可時，請僅授予執行任務所需的許可。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。如果透過特定 例如 使用服務動作 AWS 服務，您也可以使用條件來授予其存取權 CloudFormation。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](#)。
- 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，可協助您撰寫安全且實用的政策。如需詳細資訊，請參閱《IAM 使用者指南》中的 [使用 IAM Access Analyzer 驗證政策](#)。
- 需要多重要素驗證 (MFA) – 如果您的案例需要 IAM 使用者或 中的根使用者 AWS 帳戶，請開啟 MFA 以提高安全性。如需在呼叫 API 操作時請求 MFA，請將 MFA 條件新增至您的政策。如需詳細資訊，請參閱《IAM 使用者指南》中的 [透過 MFA 的安全 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 使用者檢視自身的許可

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。此政策包含在主控台或使用 或 AWS CLI AWS API 以程式設計方式完成此動作的許可。

```
{
  "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 叢集範例

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 *

JSON

```

{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [

```

```

        "ecs:CreateCluster",
        "ecs:ListClusters"
    ],
    "Resource": "*"
}
]
}

```

JSON

```

{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ecs:DescribeClusters",
        "ecs>DeleteCluster"
      ],
      "Resource": ["arn:aws:ecs:us-east-1:123456789012:cluster/cluster-name"]
    }
  ]
}

```

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。

JSON

```

{
  "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-1:123456789012: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-1:123456789012:cluster/default"}
      }
    }
  ]
}

```

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。

JSON

```

{
  "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-1:111122223333: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-1:111122223333: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：

JSON

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ecs:DeregisterContainerInstance",
        "ecs:ListContainerInstances"
      ],
      "Resource": "arn:aws:ecs:us-east-1:123456789012:cluster/cluster_name"
    }
  ]
}
```

Amazon ECS 任務定義範例

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。

JSON

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ecs:RegisterTaskDefinition",
        "ecs:ListTaskDefinitions",
        "ecs:DescribeTaskDefinition"
      ],
      "Resource": "*"
    }
  ]
}
```

執行 Amazon ECS 任務範例

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：

JSON

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": ["ecs:RunTask"],
      "Condition": {
        "ArnEquals": {"ecs:cluster": "arn:aws:ecs:us-east-1:123456789012:cluster/cluster_name"}
      },
      "Resource": ["arn:aws:ecs:us-east-1:123456789012:task-definition/task_family:*"]
    }
  ]
}
```

啟動 Amazon ECS 任務範例

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，以便 Resource 映射相符。

JSON

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": ["ecs:StartTask"],
      "Condition": {
        "ArnEquals": {
          "ecs:cluster": "arn:aws:ecs:us-east-1:123456789012:cluster/cluster_name",
          "ecs:container-instances": ["arn:aws:ecs:us-east-1:123456789012:container-instance/cluster_name/container_instance_UUID"]
        }
      },
      "Resource": ["arn:aws:ecs:us-east-1:123456789012:task-definition/task_family:*"]
    }
  ]
}
```

列出並描述 Amazon ECS 任務範例

以下 IAM 政策可讓使用者說明指定叢集中的指定任務：

JSON

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": ["ecs:DescribeTasks"],
      "Condition": {
        "ArnEquals": {"ecs:cluster": "arn:aws:ecs:us-east-1:123456789012:cluster/cluster_name"}
      },
      "Resource": ["arn:aws:ecs:us-east-1:123456789012: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 管理主控台中建立 Amazon ECS 服務：

JSON

```
{
  "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": "*"
    }
  ]
}
```

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。

JSON

```
{
  "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 Service 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。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ecs:CreateService",
        "ecs:UpdateService"
      ],
      "Condition": {
        "ArnEquals": {
          "ecs:cluster": "arn:aws:ecs:us-east-1:123456789012:cluster/cluster_name",
          "ecs:namespace": "arn:aws:servicediscovery:us-east-1:123456789012:namespace/namespace_id"
        }
      },
      "Resource": "*"
    }
  ]
}
```

AWS Amazon Elastic Container Service 的 受管政策

若要新增許可給使用者、群組和角色，使用 AWS 受管政策比自行撰寫政策更容易。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。這些政策涵蓋常見的使用案例，可在您的帳戶中使用 AWS。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 受管政策。您無法變更 AWS 受管政策中的許可。服務偶爾會將其他許可新增至 AWS 受管政策，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)。當新功能啟動或新操作可用時，服務最有可能更新 AWS 受管政策。服務不會從 AWS 受管政策移除許可，因此政策更新不會破壞您現有的許可。

此外，AWS 支援跨多個服務之任務函數的受管政策。例如，ReadOnlyAccess AWS 受管政策提供所有 AWS 服務和資源的唯讀存取權。當服務啟動新功能時，會為新操作和資源 AWS 新增唯讀許可。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](#)。

AmazonECS_FullAccess

您可將 AmazonECS_FullAccess 政策連接到 IAM 身分。此政策會授予 Amazon ECS 資源的管理存取權，並授予 IAM 身分（例如使用者、群組或角色）存取權給與 Amazon ECS 整合 AWS 的服務，以使用所有 Amazon ECS 功能。使用此政策可存取 AWS 管理主控台中可用的所有 Amazon ECS 功能。

若要檢視此政策的許可，請參閱 AWS Managed Policy Reference 中的[AmazonECS_FullAccess](#)。

AmazonECSInfrastructureRolePolicyForVolumes

您可將 AmazonECSInfrastructureRolePolicyForVolumes 受管政策連接到 IAM 實體。

此政策會授予 Amazon ECS 代表您進行 AWS API 呼叫所需的許可。您可以在啟動 Amazon ECS 任務與服務時，將此政策連接至隨磁碟區組態提供的 IAM 角色。此角色允許 Amazon ECS 管理連接至任務的磁碟區。如需詳細資訊，請參閱[Amazon ECS 基礎結構 IAM 角色](#)。

若要檢視此政策的許可，請參閱 AWS Managed Policy Reference 中的[AmazonECSInfrastructureRolePolicyForVolumes](#)。

AmazonEC2ContainerServiceforEC2Role

您可將 AmazonEC2ContainerServiceforEC2Role 政策連接到 IAM 身分。此政策會授予管理許可，允許 Amazon ECS 容器執行個體 AWS 代表您呼叫。如需詳細資訊，請參閱[Amazon ECS 容器執行個體 IAM 角色](#)。

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。

若要檢視此政策的許可，請參閱 AWS Managed Policy Reference 中的[AmazonEC2ContainerServiceforEC2Role](#)。

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。
- 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
```

AmazonEC2ContainerServiceEventsRole

您可將 AmazonEC2ContainerServiceEventsRole 政策連接到 IAM 身分。此政策授予許可，允許 Amazon EventBridge (之前為 CloudWatch Events) 代您執行任務。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](#)。

若要檢視此政策的許可，請參閱 AWS Managed Policy Reference 中的 [AmazonEC2ContainerServiceEventsRole](#)。

AmazonECSTaskExecutionRolePolicy

AmazonECSTaskExecutionRolePolicy 受管 IAM 政策會授予 Amazon ECS 容器代理程式和 AWS Fargate 容器代理程式代表您進行 AWS API 呼叫所需的許可。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](#)。

若要檢視此政策的許可，請參閱 AWS Managed Policy Reference 中的 [AmazonECSTaskExecutionRolePolicy](#)。

AmazonECSServiceRolePolicy

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 [AWSServiceRoleForECS](#) 服務連結角色。

若要檢視此政策的許可，請參閱 AWS Managed Policy Reference 中的 [AmazonECSServiceRolePolicy](#)。

AmazonECSInfrastructureRolePolicyForServiceConnectTransportLayerSecurity

您可將

AmazonECSInfrastructureRolePolicyForServiceConnectTransportLayerSecurity 政策附加至 IAM 實體。此政策會授予管理存取權 AWS 私有憑證授權單位、Secrets Manager 和其他必要 AWS 服務，以代表您管理 Amazon ECS Service Connect TLS 功能。

若要檢視此政策的許可，請參閱 AWS Managed Policy Reference 中的 [AmazonECSInfrastructureRolePolicyForServiceConnectTransportLayerSecurity](#)。

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](#)。

若要檢視此政策的許可，請參閱 AWS Managed Policy Reference 中的 [AWSApplicationAutoscalingECSServicePolicy](#)。

AWSCodeDeployRoleForECS

您無法將 AWSCodeDeployRoleForECS 連接至您的 IAM 實體。此政策會連接到服務連結角色，可讓 CodeDeploy 代表您執行動作。如需詳細資訊，請參閱《AWS CodeDeploy 使用者指南》中的 [建立 CodeDeploy 的服務角色](#)。

若要檢視此政策的許可，請參閱 AWS Managed Policy Reference 中的 [AWSCodeDeployRoleForECS](#)。

AWSCodeDeployRoleForECSLimited

您無法將 AWSCodeDeployRoleForECSLimited 連接至您的 IAM 實體。此政策會連接到服務連結角色，可讓 CodeDeploy 代表您執行動作。如需詳細資訊，請參閱《AWS CodeDeploy 使用者指南》中的 [建立 CodeDeploy 的服務角色](#)。

若要檢視此政策的許可，請參閱 AWS Managed Policy Reference 中的 [AWSCodeDeployRoleForECSLimited](#)。

AmazonECSInfrastructureRolePolicyForLoadBalancers

您可將 AmazonECSInfrastructureRolePolicyForLoadBalancers 政策附加至 IAM 實體。此政策授予許可，允許 Amazon ECS 代表您管理 Elastic Load Balancing 資源。政策包括：

- 描述接聽程式、規則、目標群組與目標運作狀態的唯讀許可
- 向目標群組註冊與取消註冊目標的許可
- 修改 Application Load Balancer 與 Network Load Balancer 接聽程式的許可
- 修改 Application Load Balancer 規則的許可

這些許可使 Amazon ECS 能夠在建立或更新服務時自動管理負載平衡器組態，確保將流量正確路由至容器。

若要檢視此政策的許可，請參閱 AWS Managed Policy Reference 中的 [AmazonECSInfrastructureRolePolicyForLoadBalancers](#)。

AmazonECSInfrastructureRolePolicyForManagedInstances

您可將 AmazonECSInfrastructureRolePolicyForManagedInstances 政策附加至 IAM 實體。此政策授予 Amazon ECS 所需的許可，代表您為 ECS 受管執行個體建立並更新 Amazon EC2 資源。政策包括：

- 為受管執行個體建立並管理 Amazon EC2 啟動範本的許可
- 使用 CreateFleet 與 RunInstances 佈建 Amazon EC2 執行個體的許可
- 在 ECS 建立的 Amazon EC2 資源上建立並管理標籤的許可
- 將 IAM 角色傳遞至受管執行個體之 Amazon EC2 執行個體的許可
- 為 Amazon EC2 Spot 執行個體建立服務連結角色的許可
- 描述 Amazon EC2 資源的唯讀許可，包括執行個體、執行個體類型、啟動範本、網路介面、可用區域、安全群組、子網路、VPCs、EC2 映像和容量保留
- 列出 Amazon ResourceGroups 資源的唯讀許可，需要基礎許可才能取得標記資源並列出 Amazon CloudFormation 堆疊資源

這些許使 Amazon ECS 能夠為 ECS 受管執行個體自動佈建並管理 Amazon EC2 執行個體，確保底層運算資源的適當設定與生命週期管理。

若要檢視此政策的許可，請參閱 AWS Managed Policy Reference 中的 [AmazonECSInfrastructureRolePolicyForManagedInstances](#)。

AmazonECSInfrastructureRolePolicyForVpcLattice

您可將 AmazonECSInfrastructureRolePolicyForVpcLattice 政策附加至 IAM 實體。此政策提供代表您管理 Amazon ECS 工作負載中 VPC Lattice 功能所需的其他 AWS 服務資源的存取權。

若要檢視此政策的許可，請參閱 AWS Managed Policy Reference 中的 [AmazonECSInfrastructureRolePolicyForVpcLattice](#)。

代表您存取 Amazon ECS 工作負載中管理 VPC Lattice 功能所需的其他 AWS 服務資源。

AmazonECSInfrastructureRoleforExpressGatewayServices

您可將 AmazonECSInfrastructureRoleforExpressGatewayServices 政策附加至 IAM 實體。此政策授予 Amazon ECS 所需的許可，以代表您使用 Express Services 建立和更新 Web 應用程式。政策包括：

- 為 Amazon ECS Application Auto Scaling 建立服務連結角色的許可
- 建立、修改和刪除 Application Load Balancer、接聽程式、規則和目標群組的許可
- 建立、修改和刪除 ECS 受管資源 VPC 安全群組的許可
- 透過 ACM 請求、管理和刪除 SSL/TLS 憑證的許可
- 設定 Amazon ECS 服務 Application Auto Scaling 政策和目標的許可
- 建立和管理自動擴展觸發的 CloudWatch 警示的許可
- 描述負載平衡器、VPC 資源、憑證、自動擴展組態和 CloudWatch 警示的唯讀許可

這些許可可讓 Amazon ECS 自動佈建和管理 Express Services Web 應用程式所需的基礎設施元件，包括負載平衡、安全群組、SSL 憑證和自動擴展組態。

若要檢視此政策的許可，請參閱《AWS 受管政策參考》中的 [AmazonECSInfrastructureRoleforExpressGatewayServices](#)。

AmazonECSComputeServiceRolePolicy

AmazonECSComputeServiceRolePolicy 政策會連接至 AmazonECSComputeServiceRole 服務連結角色。如需詳細資訊，請參閱[使用角色管理 Amazon ECS 受管執行個體](#)。

此政策包含下列許可，這些許可可能讓 Amazon ECS 完成下列任務：

- Amazon ECS 可以描述與刪除啟動範本。
- Amazon ECS 可以描述與刪除啟動範本版本。
- Amazon ECS 可以終止執行個體。
- Amazon ECS 可以描述下列執行個體資料參數：
 - 執行個體
 - 執行個體網路介面：Amazon ECS 可以描述網路介面資訊來管理 EC2 執行個體生命週期。
 - 執行個體事件視窗：Amazon ECS 可以描述事件視窗資訊來判斷可否中斷工作流程來修補執行個體。
 - 執行個體狀態：Amazon ECS 可以描述執行個體狀態來監控執行個體運作狀態。

若要檢視此政策的許可，請參閱 AWS Managed Policy Reference 中的 [AmazonECSComputeServiceRolePolicy](#)。

AmazonECSInstanceRolePolicyForManagedInstances

此 AmazonECSInstanceRolePolicyForManagedInstances 政策提供許可，讓 Amazon ECS 受管執行個體向 Amazon ECS 叢集註冊，並與 Amazon ECS 服務通訊。

此政策包含下列許可，這些許可可能讓 Amazon ECS 受管執行個體完成下列任務：

- 向 Amazon ECS 叢集註冊及取消註冊。
- 提交容器執行個體狀態變更。
- 提交任務狀態變更。
- 探索 Amazon ECS 代理程式的輪詢端點。

若要檢視此政策的許可，請參閱 AWS Managed Policy Reference 中的 [AmazonECSInstanceRolePolicyForManagedInstances](#)。

AWS 受管政策的 Amazon ECS 更新

檢視自此服務開始追蹤 Amazon ECS AWS 受管政策更新以來的詳細資訊。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	Date
更新AmazonECSInfrastructureRolePolicyForManagedInstances 政策	此AmazonECSInfrastructureRolePolicyForManagedInstances 政策已更新為下列許可，以支援 Amazon ECS 受管執行個體上的容量保留： • resource-groups:ListGroupResources 已新增許可，以允許 Amazon ECS 擷取容量保留的群組。若要執行此命令，也需要下列許可：cloudformation:DescribeStacks 、 cloudform	2026 年 2 月 24 日

變更	描述	Date
	ation:ListStackResources 、 tag:GetResources • ec2:DescribeCapacityReservations 已新增許可，以允許 Amazon ECS 收集與 Amazon ECS 受管執行個體整合所需的容量保留詳細資訊。 • ec2:RunInstances 許可已修改為包含 Amazon ECS 的 Amazon Resource Groups 群組資源，以佈建目標 Amazon EC2 容量保留。	
向 AmazonECSServiceRolePolicy 新增許可。	AmazonECSServiceRolePolicy 受管 IAM 政策已更新，包含 ssmmessages:OpenDataChannel 許可。此許可允許 Amazon ECS 開啟 ECS Exec 工作階段的資料通道。	2026 年 1 月 20 日

變更	描述	Date
更新AmazonECSInfrastructureRolePolicyForManagedInstances 政策	AmazonECSInfrastructureRolePolicyForManagedInstances 政策已更新，以修改CreateFleet 許可。子網路、安全群組和 EC2 映像的資源型條件已移除，因為： • Amazon ECS 不會建立或管理子網路和安全群組，因此不包含必要的AmazonECSManaged 標籤。 • 雖然 Amazon ECS 擁有和管理 EC2 映像，但客戶看不到這些標籤，導致無法套用以資源為基礎的政策。 此變更可確保政策與缺少預期 ECS 管理標籤的資源正確運作。	2025 年 12 月 15 日
向 AmazonECSServiceRolePolicy 新增許可。	AmazonECSServiceRolePolicy 受管 IAM 政策已更新為新的 Amazon EC2 許可，允許 Amazon ECS 為與事件 Windows 相關聯的服務和叢集擷取 Amazon EC2 事件 Windows。	2025 年 11 月 20 日
向 AmazonECSServiceRolePolicy 新增許可。	AmazonECSServiceRolePolicy 受管 IAM 政策已更新為新的 Amazon EC2 許可，允許 Amazon ECS 佈建和取消佈建任務 ENI。	2025 年 11 月 14 日

變更	描述	Date
更新 AmazonECSInfrastructureRolePolicyForServiceConnectTransportLayerSecurity 政策	已更新 AmazonECSInfrastructureRolePolicyForServiceConnectTransportLayerSecurity 受管 IAM 政策，將 secretsmanager : DescribeSecret 許可分隔為其自己的政策陳述式。許可會繼續僅針對 Amazon ECS 建立的秘密範圍限定 Amazon ECS 存取範圍，並使用 ARN 模式比對，而不是用於範圍界定的資源標籤。這可讓 Amazon ECS 在整個生命週期中監控秘密狀態，包括刪除秘密的時間。	2025 年 11 月 13 日
新增 the section called "AmazonECSInfrastructureRoleforExpressGatewayServices"	新增了新的 AmazonECSInfrastructureRoleforExpressGatewayServices 政策，提供 Amazon ECS 使用 Express Services 建立和管理 Web 應用程式的權限。	2025 年 11 月 21 日
新增 the section called "AmazonECSInstanceRolePolicyForManagedInstances"	新增了 AmazonECSInstanceRolePolicyForManagedInstances 政策，該政策授予 Amazon ECS 受管執行個體向 Amazon ECS 叢集進行註冊所需的許可。	2025 年 9 月 30 日
新增 the section called "AmazonECSInfrastructureRolePolicyForManagedInstances"	新增了 AmazonECSInfrastructureRolePolicyForManagedInstances 政策，該政策授予 Amazon ECS 建立與管理 Amazon EC2 受管資源的存取權。	2025 年 9 月 30 日

變更	描述	Date
新增 AmazonECSComputeServiceRolePolicy	允許 Amazon ECS 管理 Amazon ECS 受管執行個體與相關資源。	2025 年 8 月 31 日
新增許可至 the section called “AmazonEC2ContainerServiceforEC2Role”	AmazonEC2ContainerServiceforEC2Role 受管 IAM 政策已更新，包含 <code>ecs:ListTagsForResource</code> 許可。此許可允許 Amazon ECS 代理程式透過任務中繼資料端點 (<code>\${ECS_CONTAINER_METADATA_URI_V4}/taskWithTags</code>) 擷取任務與容器執行個體標籤。	2025 年 8 月 4 日
向 AmazonECSInfrastructureRolePolicyForLoadBalancers 新增許可。	AmazonECSInfrastructureRolePolicyForLoadBalancers 受管 IAM 政策已更新，新增了描述、取消註冊與註冊目標群組的許可。	2025 年 7 月 25 日
新增 the section called “AmazonECSInfrastructureRolePolicyForLoadBalancers” 政策	新增了新的 AmazonECSInfrastructureRolePolicyForLoadBalancers 政策，可讓您存取管理與 Amazon ECS 工作負載相關聯之負載平衡器所需的其他 AWS 服務資源。	2025 年 7 月 15 日

變更	描述	Date
向 AmazonECSServiceRolePolicy 新增許可。	AmazonECSServiceRolePolicy 受管 IAM 政策已更新，新增了 AWS Cloud Map 許可，使 Amazon ECS 能夠更新自身所管理服務的 AWS Cloud Map 服務屬性。	2025 年 7 月 15 日
新增許可至 AmazonECSServiceRolePolicy	AmazonECSServiceRolePolicy 受管 IAM 政策已更新，新增了 AWS Cloud Map 許可，使 Amazon ECS 能夠更新自身所管理服務的 AWS Cloud Map 服務屬性。	2025 年 6 月 24 日
向 AmazonECSInfrastructureRolePolicyForVolumes 新增許可	AmazonECSInfrastructureRolePolicyForVolumes 政策已更新，新增了 ec2:DescribeInstances 許可。此許可有助於防止連接至相同容器執行個體上執行之 Amazon ECS 任務的 Amazon EBS 磁碟區發生裝置名稱衝突。	2025 年 6 月 2 日
新增 AmazonECSInfrastructureRolePolicyForVpcLattice	代表您存取 Amazon ECS 工作負載中管理 VPC Lattice 功能所需的其他 AWS 服務資源。	2024 年 11 月 18 日
向 AmazonECSInfrastructureRolePolicyForVolumes 新增許可	AmazonECSInfrastructureRolePolicyForVolumes 政策已更新，允許客戶透過快照建立 Amazon EBS 磁碟區。	2024 年 10 月 10 日

變更	描述	Date
新增了許可至 the section called “AmazonECS_FullAccess”	AmazonECS_FullAccess 政策已更新，向名為 ecsInfrastructureRole 的角色新增 IAM 角色的 iam:PassRole 許可。這是建立的預設 IAM 角色 AWS 管理主控台，旨在用作 ECS 基礎設施角色，允許 Amazon ECS 管理連接到 ECS 任務的 Amazon EBS 磁碟區。	2024 年 8 月 13 日
新增 AmazonECSInfrastructureRolePolicyForServiceConnectTransportLayerSecurity 政策	新增了新的 AmazonECS InfrastructureRolePolicyForServiceConnectTransportLayerSecurity 政策 AWS KMS，提供對 Secrets Manager 的管理存取權 AWS 私有憑證授權單位，並可讓 Amazon ECS Service Connect TLS 功能正常運作。	2024 年 1 月 22 日
新增政策 AmazonECSInfrastructureRolePolicyForVolumes	新增了 AmazonECS InfrastructureRolePolicyForVolumes 政策。此政策會授予 Amazon ECS 進行 AWS API 呼叫所需的許可，以管理與 Amazon ECS 工作負載相關聯的 Amazon EBS 磁碟區。	2024 年 1 月 11 日

變更	描述	Date
新增許可至 AmazonECS ServiceRolePolicy	AmazonECSServiceRolePolicy 受管 IAM 政策已更新，新增了 events 許可以及額外的 autoscaling 與 autoscaling-plans 許可。	2023 年 12 月 4 日
向 AmazonEC2ContainerServiceEventsRole 新增許可	受AmazonECSServiceRolePolicy 管 IAM 政策已更新，以允許存取 DiscoverInstancesRevision API AWS Cloud Map 操作。	2023 年 10 月 4 日
新增許可至 AmazonEC2ContainerServiceforEC2Role	AmazonEC2ContainerServiceforEC2Role 政策已修改為新增 ecs:TagResource 許可，其中包括一個條件，該條件將許可僅限於新建立的叢集和已註冊的容器執行個體。	2023 年 3 月 6 日
新增許可至 the section called "AmazonECS_FullAccess"	AmazonECS_FullAccess 政策已修改為新增 elasticloadbalancing:AddTags 許可，其中包括一個條件，該條件將許可僅限於新建立的負載平衡器、目標群組、規則和已建立的接聽程式。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

Amazon Elastic Container Service 的逐步淘汰 AWS 受管 IAM 政策

下列 AWS 受管 IAM 政策會逐步淘汰。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。

AmazonEC2ContainerServiceFullAccess

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](#)。

AmazonEC2ContainerServiceRole

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](#)。

AmazonEC2ContainerServiceAutoscaleRole

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。如果目前使用由 AmazonEC2ContainerServiceFullAccess 政策授予的許可，但這些許可不在 AmazonECS_FullAccess 政策中，您可以將這些許可新增至內嵌政策聲明。如需詳細資訊，請參閱 [AWS Amazon Elastic Container Service 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。

更新群組、使用者或角色以使用 AmazonECS_FullAccess 政策 (AWS 管理主控台)

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. 對於目前正在使用 AmazonEC2ContainerServiceFullAccess 政策的每個 IAM 角色，選取角色並使用下列步驟分開淘汰政策，並連接 AmazonECS_FullAccess 政策。
 - a. 在 Permissions (許可) 標籤中，選擇 AmazonEC2ContainerServiceFullAccess 政策旁邊的 X。
 - b. 選擇新增許可。
 - c. 選擇 Attach existing policies directly (直接連接現有政策)，搜尋並選取 AmazonECS_FullAccess 政策，然後選擇 Next: Review (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 [generate-service-last-accessed-details](#) 命令來產生報告，其中包含上次使用淘汰政策時的詳細資訊。

```
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. 搭配使用先前輸出的任務 ID 與 [get-service-last-accessed-details](#) 命令來擷取服務的上次存取報告。此報告會顯示上次使用淘汰政策的 IAM 實體的 Amazon Resource Name (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 Elastic Container Service 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。服務連結角色是由 Amazon ECS 預先定義，並包含該服務代表您呼叫其他 AWS 服務所需的所有許可。

主題

- [使用角色允許 Amazon ECS 管理叢集](#)
- [使用角色管理 Amazon ECS 受管執行個體](#)

使用角色允許 Amazon ECS 管理叢集

Amazon Elastic Container Service 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。服務連結角色是由 Amazon ECS 預先定義，並包含該服務代表您呼叫其他 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。

您必須先刪除服務連結角色的相關資源，才能將其刪除。如此即可保護 Amazon ECS 資源，避免您不小心移除資源的存取許可。

如需有關支援服務連結角色的其他服務的資訊，請參閱[AWS 使用 IAM 的服務](#)，並在服務連結角色欄中尋找具有是的服務。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 `AWSServiceRoleForECS` 的服務連結角色：這是允許 Amazon ECS 管理叢集的角色。

`AWSServiceRoleForECS` 服務連結角色信任下列服務可擔任該角色：

- `ecs.amazonaws.com`

名稱為 `AmazonECSServiceRolePolicy` 的角色許可政策允許 Amazon ECS 針對指定資源完成下列動作：

- 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 會管理必要的 Route 53 和 AWS Cloud Map 資源，供服務探索使用。
- 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 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。
- 動作：Amazon ECS 可以更新 Amazon ECS 管理之服務的服務 AWS Cloud Map 屬性。
- 動作：Amazon ECS 可以在啟動和停止任務時調用 Amazon EC2 佈建和取消佈建 ENI。
- 動作：Amazon ECS 可以為與事件 Windows 相關聯的服務和叢集擷取 Amazon EC2 事件 Windows。

您必須設定許可，以允許您的使用者、群組或角色建立、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

您不需要手動建立服務連結角色，當您在、或 AWS API 中建立叢集 AWS 管理主控台，或建立 AWS CLI 或更新服務時，Amazon ECS 會為您建立服務連結角色。

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。Amazon ECS 自 2017 年 1 月 1 日開始支援服務連結角色，若您在這之前就有使用此服務，則 Amazon ECS 會在帳戶中建立 AWSServiceRoleForECS 角色。若要進一步了解，請參閱[我的 中出現的新角色 AWS 帳戶](#)。

您也可以使用 IAM 主控台透過 AWSServiceRoleForECS 使用案例建立服務連結角色。在 AWS CLI 或 AWS API 中，使用 IAM 建立具有服務名稱 `ecs.amazonaws.com` 的服務連結角色。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](#)」。如果您刪除此服務連結角色，您可以使用此相同的程序以再次建立該角色。

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

您不需要手動刪除 `AWSServiceRoleForECS` 角色。當您刪除 AWS CLI、AWS 管理主控台或 AWS API 中所有區域中的叢集時，Amazon ECS 會為您清除資源並刪除服務連結角色。

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，務必清除您的服務連結角色，之後才能以手動方式將其刪除。

清除服務連結角色

在您使用 IAM 刪除服務連結角色之前，您必須先刪除該角色所使用的任何資源。

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。

刪除 `AWSServiceRoleForECS` (主控台) 所使用的 Amazon ECS 資源

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。如需詳細資訊，請參閱[更新 Amazon ECS 服務](#)及[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](#)。

刪除 `AWSServiceRoleForECS` (CLI) 使用的 Amazon ECS 資源

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。如需詳細資訊，請參閱 AWS Command Line Interface 參考中的 [update-service](#) 和 [delete-service](#)。
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 [deregister-container-instance](#)。

3. 刪除所有區域中的所有 Amazon ECS 叢集。如需更多詳細資訊，請參閱 [delete-cluster](#)。

刪除 AWSServiceRoleForECS (API) 所使用的 Amazon ECS 資源

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。如需詳細資訊，請參閱 Amazon ECS API Reference 中的 [UpdateService](#) 與 [DeleteService](#)。
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 [DeregisterContainerInstance](#)。
3. 刪除所有區域中的所有 Amazon ECS 叢集。如需詳細資訊，請參閱 [DeleteCluster](#)。

手動刪除服務連結角色

使用 IAM 主控台 AWS CLI、或 AWS API 來刪除 AWSServiceRoleForECS 服務連結角色。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 Regions and endpoints](#)。

使用角色管理 Amazon ECS 受管執行個體

Amazon Elastic Container Service 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。服務連結角色是由 Amazon ECS 預先定義，並包含該服務代表您呼叫其他 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。

您必須先刪除服務連結角色的相關資源，才能將其刪除。如此即可保護 Amazon ECS 資源，避免您不小心移除資源的存取許可。

如需有關支援服務連結角色的其他服務的資訊，請參閱[AWS 使用 IAM 的服務](#)，並在服務連結角色欄中尋找具有是的服務。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 AWSServiceRoleForECSCompute 的服務連結角色：此角色可允許 Amazon ECS 管理由 Amazon ECS 受管執行個體容量提供者佈建的 Amazon EC2 受管執行個體。

AWSServiceRoleForECSCCompute 服務連結角色信任下列服務擔任該角色：

- `ecs-compute.amazonaws.com`

名為 [AmazonECSCComputeServiceRolePolicy](#) 的角色許可政策允許 Amazon ECS 完成以下動作：

- Amazon ECS 可以描述與刪除啟動範本。
- Amazon ECS 可以描述與刪除啟動範本版本。
- Amazon ECS 可以終止執行個體。
- Amazon ECS 可以描述下列執行個體資料參數：
 - 執行個體
 - 執行個體網路介面：Amazon ECS 可以描述網路介面資訊來管理 EC2 執行個體生命週期。
 - 執行個體事件視窗：Amazon ECS 可以描述事件視窗資訊來判斷可否中斷工作流程來修補執行個體。
 - 執行個體狀態：Amazon ECS 可以描述執行個體狀態來監控執行個體運作狀態。

您必須設定許可，以允許您的使用者、群組或角色建立、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

您不需要手動建立服務連結角色，當您在 AWS 管理主控台、AWS CLI 或 AWS API 中為 Amazon ECS 受管執行個體建立容量提供者時，Amazon ECS 會為您建立服務連結角色。

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。Amazon ECS 自 2017 年 1 月 1 日開始支援服務連結角色，若您在這之前就有使用此服務，則 Amazon ECS 會在帳戶中建立 AmazonECSCComputeServiceRolePolicy 角色。若要進一步了解，請參閱 [我的 中出現的新角色 AWS 帳戶](#)。

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。當您為 Amazon ECS 受管執行個體建立容量提供者時，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 AmazonECSComputeServiceRolePolicy 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

您不需要手動刪除 AmazonECSComputeServiceRolePolicy 角色。當您刪除 AWS 管理主控台、AWS CLI 或 AWS API 中所有區域中的所有 Amazon ECS 受管執行個體容量提供者時，Amazon ECS 會為您清除資源並刪除服務連結角色。

手動刪除服務連結角色

使用 IAM 主控台 AWS CLI、或 AWS API 來刪除 AmazonECSComputeServiceRolePolicy 服務連結角色。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 Regions and endpoints](#)。

適用於 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，將相關許可授予 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的啟動類型以及您使用的功能。請使用下列表來判斷 Amazon ECS 需要哪些 IAM 角色。

Role	定義	適用時機	其他資訊
任務執行 角色	此角色允許 Amazon ECS 代表您使用其他 AWS 服務。	您的任務託管於 AWS Fargate 或外部執行個體上，且符合下列條件： 從 Amazon ECR 私有的儲存庫提取容器映像。	Amazon ECS 任務執行 IAM 角色

Role	定義	適用時機	其他資訊
		- 從與執行任務不同帳戶的 Amazon ECR 私有儲存庫提取容器映像。 - 使用 <code>awslogs</code> 日誌驅動程式將容器日誌傳送到 CloudWatch Logs。 您的任務託管在 AWS Fargate 或 Amazon EC2 執行個體上，並且： - 使用私有登錄檔身分驗證。 - 使用執行時期監控。 - 任務定義會使用 Secrets Manager 秘密或 AWS Systems Manager 參數存放區參數來參考敏感資料。	
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	此角色允許您的 EC2 執行個體或外部執行個體向叢集完成註冊程序。	您的任務託管於 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

Role	定義	適用時機	其他資訊
Amazon ECS Anywhere 角色	此角色可讓您的外部執行個體存取 AWS APIs。	您的任務託管於外部執行個體上。	Amazon ECS Anywhere IAM 角色
負載平衡器角色的 Amazon ECS 基礎結構	此角色允許 Amazon ECS 代表您管理叢集中的負載平衡器資源，以實現藍/綠部署。	您想使用 Amazon ECS 藍/綠部署。	負載平衡器的 Amazon ECS 基礎結構 IAM 角色
Amazon ECS CodeDeploy 角色	此角色允許 CodeDeploy 更新您的服務。	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 EventBridge 角色	此角色允許 EventBridge 更新您的服務。	您可以透過 EventBridge 規則與目標來排程任務。	Amazon ECS EventBridge IAM 角色

Role	定義	適用時機	其他資訊
Amazon ECS 基礎結構角色	此角色允許 Amazon ECS 管理叢集中的基礎結構資源。	- 您想使用 Amazon ECS 受管執行個體作為容量。 - 您想將 Amazon EBS 磁碟區連接至 Fargate 或 EC2 啟動類型 Amazon ECS 任務。基礎結構角色允許 Amazon ECS 管理您任務的 Amazon EBS 磁碟區。 - 您想使用 Transport Layer Security (TLS) 來加密 Amazon ECS Service Connect 服務之間的流量。 - 您想建立 VPC Lattice 目標群組。	Amazon ECS 基礎結構 IAM 角色
執行個體設定檔	此角色允許 Amazon ECS 受管執行個體安全地擔任基礎結構角色。	您可以在叢集中使用 Amazon ECS 受管執行個體。	Amazon ECS 受管執行個體執行個體設定檔

Amazon ECS 中 IAM 角色的最佳實務

Amazon ECS 所需的角色取決於任務定義的啟動類型以及您使用的功能。建議在資料表中建立個別角色，而非共用角色。

Role	定義	適用時機	其他資訊
任務執行 角色	此角色允許 Amazon ECS 代表您使用其他 AWS 服務。	您的任務託管於 AWS Fargate 或外部執行個體上，且符合下列條件： - 從 Amazon ECR 私有儲存庫提取容器映像。 - 從與執行任務不同帳戶的 Amazon ECR 私有儲存庫提取容器映像。 - 使用 awslogs 日誌驅動程式將容器日誌傳送到 CloudWatch Logs。 您的任務託管在 AWS Fargate 或 Amazon EC2 執行個體上，並且： - 使用私有登錄檔身分驗證。 - 使用執行時期監控。 - 任務定義會使用 Secrets Manager 秘密或 AWS Systems Manager 參數存放區參數來參考敏感資料。	Amazon ECS 任務執行 IAM 角色

Role	定義	適用時機	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	此角色允許您的 EC2 執行個體或外部執行個體向叢集完成註冊程序。	您的任務託管於 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 APIs。	您的任務託管於外部執行個體上。	Amazon ECS Anywhere IAM 角色
Amazon ECS CodeDeploy 角色	此角色允許 CodeDeploy 更新您的服務。	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 EventBridge 角色	此角色允許 EventBridge 更新您的服務。	您可以透過 EventBridge 規則與目標來排程任務。	Amazon ECS EventBridge IAM 角色

Role	定義	適用時機	其他資訊
Amazon ECS 基礎結構角色	此角色允許 Amazon ECS 管理叢集中的基礎結構資源。	- 您想將 Amazon EBS 磁碟區連接至 Fargate 或 EC2 啟動類型 Amazon ECS 任務。基礎結構角色允許 Amazon ECS 管理您任務的 Amazon EBS 磁碟區。 - 您想使用 Transport Layer Security (TLS) 來加密 Amazon ECS Service Connect 服務之間的流量。 - 您想建立 VPC Lattice 目標群組。	Amazon ECS 基礎結構 IAM 角色

任務 角色

建議您為任務指派角色。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。

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 SDKs 會在進行 AWS API 呼叫時，自動從 `AWS_CONTAINER_CREDENTIALS_RELATIVE_URI` 環境變數擷取這些登入資料。如需如何續約登入資料的資訊，請參閱在 rePost [上續約 AWS 登入](#) 資料。

輸出包含存取金鑰對，其中包含私密存取金鑰 ID 和您的應用程式用來存取 AWS 資源的私密金鑰。它還包含一個字符，AWS 用於驗證登入資料是否有效。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予 Amazon ECS 容器代理程式代表您呼叫特定 AWS API 動作的許可。例如，當您使用 AWS Fargate 時，Fargate 需要 IAM 角色，以允許其從 Amazon ECR 提取映像並將日誌寫入 CloudWatch Logs。當任務參考存放在 中的秘密時 AWS Secrets Manager，例如映像提取秘密時，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

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` – (選用) 允許主體描述與 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` – (選用) 允許主體建立 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` – (選用) 允許主體從叢集中取消註冊 Amazon ECS 容器執行個體。Amazon ECS 容器代理程式不會呼叫此 API 操作，但此許可仍會保留，有助確保向後相容性。
- `ecs:DiscoverPollEndpoint` – (必要) 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` – (必要) 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` – (必要) 允許主體向叢集註冊容器執行個體。Amazon ECS 容器代理程式會使用此許可，向叢集註冊 Amazon EC2 執行個體，以及支援資源標籤傳播。
- `ecs:StartTelemetrySession` – (選用) 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及指標。

雖然此許可並非必要，但仍建議新增此許可，以允許容器執行個體指標啟動擴展動作，同時接收與運作狀態檢查命令相關的報告。

- `ecs:TagResource` – (選用) 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 容器執行個體的状态。Amazon ECS 容器代理程式使用此許可，用於 Spot 執行個體耗盡。
- `ecs:Submit*` – (必要) 這包括 `SubmitAttachmentStateChanges`、`SubmitContainerStateChange`、`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 `SubmitContainerStateChange` 許可，但仍會保留以確保向後相容性。

- `ecr:GetAuthorizationToken` – (選用) 允許主體擷取授權權杖。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限為 12 小時。
- `ecr:BatchCheckLayerAvailability` – (選用) 將容器映像推送至 Amazon ECR 私有儲存庫時，會逐層檢查每個映像層是否已推送。如果已推送，則會略過映像層。
- `ecr:GetDownloadUrlForLayer` – (選用) 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` – (選用) 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` – (選用) 允許主體為指定的日誌群組建立 CloudWatch Logs 日誌串流。
- `logs:PutLogEvents` – (選用) 允許主體將批次日誌事件上傳至指定的日誌串流。

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，請改為將 IAM 角色用於任務。

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 169.254.169.254/32 --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 **awsvpc** 網路模式

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。它是預設模式 AWS Fargate。它是您可以用來將安全群組指派給任務的唯一網路模式。

使用上次存取的資訊來精簡角色

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 提供的最新存取權限資訊，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[AWS 使用上次存取資訊在 中精簡許可](#)。

監控 AWS CloudTrail 可疑活動

您可以監控 AWS CloudTrail 是否有任何可疑活動。大多數 AWS API 呼叫都會記錄 AWS CloudTrail 為事件。它們由 AWS CloudTrail Insights 進行分析，並且您會收到與 write API 呼叫相關的任何可疑行為的提醒。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": "AR0A36C6WWEJ2YEXAMPLE: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。

Note

Amazon ECS 會定期傳送角色的臨時憑證給在執行個體上執行的代理程式，從而使代理程式能夠使用這些許可，但任務中的容器無法直接存取這些許可。有關容器內應用程式程式碼需要執行的 IAM 許可，請參閱 [Amazon ECS 任務 IAM 角色](#)。

以下是任務執行 IAM 角色的常見使用案例：

- 您的任務託管於 AWS Fargate、Amazon ECS 受管執行個體或外部執行個體，以及：
 - 從 Amazon ECR 私有儲存庫提取容器映像。
 - 從與執行任務不同帳戶的 Amazon ECR 私有儲存庫提取容器映像。
 - 使用 `awslogs` 日誌驅動程式將容器日誌傳送到 CloudWatch Logs。如需詳細資訊，請參閱 [將 Amazon ECS 日誌傳送至 CloudWatch](#)。
- 您的任務託管在 AWS Fargate 或 Amazon EC2 執行個體上，並且：
 - 使用私有登錄檔身分驗證。如需詳細資訊，請參閱 [私有登錄檔身分驗證許可](#)。
 - 使用執行時期監控。
 - 任務定義透過 Secrets Manager 秘密或 AWS Systems Manager Parameter Store 參數來引用敏感資料。如需詳細資訊，請參閱 [Secrets Manager 或 Systems Manager 許可](#)。

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 `AmazonECSTaskExecutionRolePolicy` 的受管政策，包含上述常用案例要求的許可。如需詳細資訊，請參閱 AWS Managed Policy Reference Guide 中的 [AmazonECSTaskExecutionRolePolicy](#)。針對特殊使用案例，可能需要將內嵌政策新增至任務執行角色

Amazon ECS 主控台會建立任務執行角色。您可手動連接任務的受管 IAM 政策，讓 Amazon ECS 為日後推出的功能和增強功能新增許可。您可以使用 IAM 主控台搜尋功能來搜尋 `ecsTaskExecutionRole`，確認帳戶是否已具備任務執行角色。如需詳細資訊，請參閱 IAM user guide 中的 [IAM console search](#)。

如果您以經身分驗證的使用者提取映像，受到 [Docker Hub usage and limits](#) 變更的影響會較小。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

如果帳戶還沒有任務執行角色，請使用下列步驟來建立角色。

AWS 管理主控台

建立 Elastic Container Service 的服務角色 (IAM 主控台)

1. 登入 AWS 管理主控台 並開啟位於 <https://console.aws.amazon.com/iam/> 的 IAM 主控台。
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. 在服務或使用案例欄位中選擇 Elastic Container Service，然後選擇 Elastic Container Service 任務使用案例。
5. 選擇下一步。
6. 在新增許可區段中，搜尋 `AmazonECSTaskExecutionRolePolicy` 並選取政策。
7. 選擇下一步。
8. 在角色名稱欄位中，輸入 `ecsTaskExecutionRole`。
9. 檢閱角色，然後選擇 Create role (建立角色)。

AWS CLI

將所有 *user input* 取代為自己的資訊。

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：

JSON

```
{
  "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 `AmazonECSTaskExecutionRolePolicy` 政策連接至 `ecsTaskExecutionRole` 角色。

```
aws iam attach-role-policy \
  --role-name ecsTaskExecutionRole \
  --policy-arn arn:aws:iam::aws:policy/service-role/AmazonECSTaskExecutionRolePolicy
```

建立角色之後，請為角色新增下列功能的其他許可。

功能	額外許可
使用 Secrets Manager 憑證從外部的私有登錄檔提取容器映像 AWS（例如 Docker Hub、Quay.io：// 或您自己的私有登錄檔）	私有登錄檔身分驗證許可

功能	額外許可
透過 Systems Manager 或 Secrets Manager 傳遞敏感資料	Secrets Manager 或 Systems Manager 許可
讓 Fargate 任務透過介面端點提取 Amazon ECR 映像	Fargate 任務透過介面端點提取 Amazon ECR 映像的許可
Amazon S3 儲存貯體中的主機組態檔案	Amazon S3 檔案儲存許可
設定 Container Insights 以檢視 Amazon ECS 生命週期事件	在 Container Insights 中啟用 Amazon ECS 生命週期事件所需的許可
在 Container Insights 中檢視 Amazon ECS 生命週期事件	在 Container Insights 中檢視 Amazon ECS 生命週期事件所需的許可

私有登錄檔身分驗證許可

私有登錄檔身分驗證可讓您的 Amazon ECS 任務從需要身分驗證憑證的 AWS（例如 Docker Hub、Quay.io 或您自己的私有登錄檔）外部的私有登錄檔提取容器映像。此功能透過 Secrets Manager 安全地儲存登錄檔憑證，這些憑證隨後會在任務定義中透過 `repositoryCredentials` 參數進行引用。

如需設定私有登錄檔身分驗證的詳細資訊，請參閱[在 Amazon ECS 中使用非AWS 容器映像](#)。

若要存取內含私有登錄檔憑證的秘密，請將以下許可以內嵌政策形式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](#)。

- `secretsmanager:GetSecretValue` – 從 Secrets Manager 擷取私有登錄檔憑證時需要。
- `kms:Decrypt` - 只有在您的秘密使用自訂 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。

JSON

```
{
  "Version": "2012-10-17",
```

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "kms:Decrypt",
      "secretsmanager:GetSecretValue"
    ],
    "Resource": [
      "arn:aws:secretsmanager:us-east-1:111122223333:secret:secret_name",
      "arn:aws:kms:us-east-1:111122223333:key/key_id"
    ]
  }
]
```

Secrets Manager 或 Systems Manager 許可

允許容器代理程式提取必要 AWS Systems Manager 或 Secrets Manager 資源的許可。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。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "secretsmanager:GetSecretValue"
      ],
```

```

    "Resource": [
      "arn:aws:secretsmanager:us-east-1:111122223333: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：

JSON

```

{
  "Version": "2012-10-17",
  "Statement": [

```



```
{
  "Effect": "Allow",
  "Action": [
    "ssm:GetParameters",
    "secretsmanager:GetSecretValue",
    "kms:Decrypt"
  ],
  "Resource": [
    "arn:aws:ssm:us-east-1:111122223333:parameter/parameter_name",
    "arn:aws:secretsmanager:us-east-1:111122223333:secret:secret_name",
    "arn:aws:kms:us-east-1:111122223333:key/key_id"
  ]
}
```

Fargate 任務透過介面端點提取 Amazon ECR 映像的許可

當 Amazon ECR 設定為使用介面 VPC 端點時，只要您啟動使用 Fargate 的任務 (可從 Amazon ECR 中提取映像) 時，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。您可以將 VPC 限制為託管任務與端點的 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：

JSON

```
{
  "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": "arn:aws:ecr:*:*:repository/*",
      "Condition": {
        "StringEquals": {
          "aws:sourceVpce": "vpce-0123456789abcdef0"
        }
      }
    }
  ]
}

```

Amazon ECR 許可

當您需要從 Amazon ECR 私有儲存庫提取容器映像時，需要如下許可。任務執行角色應具有這些許可，以允許 Amazon ECS 容器與 Fargate 代理程式代表您提取容器映像。對於基本 ECS 實作，這些許可應新增至任務執行角色，而非任務 IAM 角色。

Amazon ECS 任務執行角色受管政策 (AmazonECSTaskExecutionRolePolicy) 包含從 Amazon ECR 提取映像的必要許可。如果使用受管政策，則不需要另外新增這些許可。

如果要建立自訂政策，請包含下列許可，以允許從 Amazon ECR 提取映像：

JSON

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ecr:BatchGetImage",

```

```

        "ecr:GetDownloadUrlForLayer",
        "ecr:GetAuthorizationToken"
    ],
    "Resource": "*"
}
]
}

```

請注意，如果應用程式程式碼需要直接與 Amazon ECR API 互動，這些許可與任務 IAM 角色中可能需要的許可不同。如需有關 Amazon ECR 的任務 IAM 角色許可的資訊，請參閱 [Amazon ECR 許可](#)。

Amazon S3 檔案儲存許可

如果您指定的組態檔案託管於 Amazon S3，則任務執行角色必須包含組態檔案的 `s3:GetObject` 許可，以及檔案所在 Amazon S3 儲存貯體的 `s3:GetBucketLocation` 許可。如需詳細資訊，請參閱 Amazon Simple Storage Service User Guide 中的 [Policy actions for Amazon S3](#)。

下列範例政策會新增需要的許可，以便從 Amazon S3 中擷取檔案。指定 Amazon S3 儲存貯體的名稱和組態檔案名稱。

JSON

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject"
      ],
      "Resource": [
        "arn:aws:s3:::amzn-s3-demo-bucket/folder_name/config_file_name"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetBucketLocation"
      ],
      "Resource": [
        "arn:aws:s3:::amzn-s3-demo-bucket"
      ]
    }
  ]
}

```

```
    ]  
  }  
]  
}
```

重要安全考量事項

使用與 Amazon S3 儲存貯體整合的 Amazon ECS 功能時，請實作儲存貯體擁有權驗證，以防止儲存貯體接管攻擊。如果沒有適當的驗證，如果惡意執行者刪除並重新建立具有相同名稱的 Amazon S3 儲存貯體，您的任務可能會在無意中載入惡意組態或將敏感資料傳送至攻擊者控制的儲存貯體。

建議的 IAM 政策條件：

```
"Condition": {  
  "StringEquals": {  
    "aws:ResourceAccount": "TRUSTED-ACCOUNT-ID"  
  }  
}
```

將 *TRUSTED-ACCOUNT-ID* 取代為擁有 S3 儲存貯體 AWS 的帳戶 ID。

此條件可確保您的任務執行角色只能存取指定受信任帳戶擁有的 Amazon S3 儲存貯體。

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。IAM 角色中授予的許可將指派給任務中執行的容器。此角色可讓您的應用程式程式碼（在容器中執行）使用其他 AWS 服務。當您的應用程式存取 Amazon S3 等其他服務時 AWS，需要任務角色。

Note

Amazon ECS 容器與 Fargate 代理程式無法存取這些許可。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](#)。

以下為使用任務角色的優勢：

- 問題分離：如果使用的是 EC2，任務 IAM 角色可讓您指定容器的 IAM 許可，無需使用這些 EC2 執行個體設定檔來指定這些許可（如需詳細資訊，請參閱 [AWS Identity and Access Management User](#)

Guide 中的 [Using instance profiles](#))。因此，您可以在 ECS 容器執行個體上獨立且統一地部署應用程式，無需修改與 EC2 執行個體相關聯的 IAM 許可。

- 可稽核性：可透過 CloudTrail 取得存取與事件記錄，確保可回顧性稽核。任務憑證有連接到工作階段的 "taskArn" 內容，所以 CloudTrail 日誌會顯示角色憑證是為哪個任務所授予。
- 統一憑證交付：ECS 會將 IAM 角色憑證交付至您的容器，使其能透過妥善定義的介面進行存取，不受與任務相關聯的運算選項影響。在 ECS Fargate 上，EC2 執行個體設定檔不適用於任務中的容器。當您在容器中使用 AWS SDK 或 CLI 時，任務 IAM AWS 角色可讓您將 IAM 許可與容器建立關聯，而不論運算選項為何。如需 AWS 開發套件如何存取這些登入資料的詳細資訊，請參閱[容器登入資料提供者](#)。

Important

容器本身並非安全界限，使用任務 IAM 角色也不會改變此本質。每個在 Fargate 上執行的任務都有專用的隔離界限，並且不會與另一個任務共用底層核心、CPU 資源、記憶體資源或彈性網路介面。在 ECS 上的 EC2 與外部容器執行個體中，沒有任務隔離 (這與 Fargate 不同)，因此容器可能會存取同一容器執行個體上其他任務的憑證。這些容器也可以存取指派給 [ECS 容器執行個體角色](#) 的許可。遵循[角色建議](#)中的建議，封鎖對容器 Amazon EC2 執行個體中繼資料服務的存取 (如需詳細資訊，請參閱 Amazon EC2 User Guide 中的 [Use the Instance Metadata Service to access instance metadata](#))。

請注意，當您為任務指定 IAM 角色時，該任務容器中的 AWS CLI 或其他 SDKs 只會使用任務角色提供的 AWS 登入資料，而不會從其執行的 Amazon EC2 或外部執行個體繼承任何 IAM 許可。

建立任務 IAM 角色

為任務建立要使用的 IAM 政策時，政策必須包含您希望任務中的容器擔任的許可。您可以使用現有的 AWS 受管政策，也可以從頭開始建立符合您特定需求的自訂政策。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。這些登入資料可讓您的任務在不呼叫的情況下提出 AWS API 請求 `sts:AssumeRole`，以擔任已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 User Guide 中的 [Updating a role trust policy](#)。

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 政策連接至該角色，藉此授予任務中的容器您所需的許可。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。

如需有關所在區域的服務端點的資訊，請參閱 Amazon Web Services 一般參考 Guide 中的 [Service endpoints](#)。

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。取代區域識別符，並指定您在啟動任務時使用 AWS 的帳號。

Important

建立任務 IAM 角色時，建議將 `aws:SourceAccount` 或 `aws:SourceArn` 條件索引鍵用於與角色關聯的信任關係政策，進一步設定許可的範圍，防止出現混淆代理人安全問題。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代理人問題以及如何保護 AWS 您的帳戶，請參閱《IAM 使用者指南》中的 [混淆代理人問題](#)。

JSON

```
{
  "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"
      }
    }
  }
}
]
}

```

按照如下程序建立政策，透過範例政策從 Amazon S3 擷取物件。將所有 *user input* 值取代為實際值。

AWS 管理主控台

若要使用 JSON 政策編輯器來建立政策

1. 登入 AWS 管理主控台 並開啟位於 <https://console.aws.amazon.com/iam/> 的 IAM 主控台。
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/*"
      ]
    }
  ]
}

```

```
}
```

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

將所有 *user input* 值取代為實際值。

1. 建立稱為 s3-policy.json 的檔案，其中具有以下內容。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject"
      ],
      "Resource": [
        "arn:aws:s3:::my-task-secrets-bucket/*"
      ]
    }
  ]
}
```

2. 使用以下命令，透過 JSON 政策文件檔案建立 IAM 政策。將所有 *user input* 值取代為實際值。


```
aws iam create-policy \  
  --policy-name taskRolePolicy \  
  --policy-document file://s3-policy.json
```

透過如下程序建立服務角色。

AWS 管理主控台

建立 Elastic Container Service 的服務角色 (IAM 主控台)

1. 登入 AWS 管理主控台 並開啟位於 <https://console.aws.amazon.com/iam/> 的 IAM 主控台。
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. 在服務或使用案例欄位中選擇 Elastic Container Service，然後選擇 Elastic Container Service 任務使用案例。
5. 選擇下一步。
6. 在新增許可欄位中，搜尋並選擇您建立的政策。
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

1. 建立名為 ecs-tasks-trust-policy.json 的檔案，其中包含用於任務 IAM 角色的信任政策。檔案應包含以下內容。取代區域識別符，並指定您在啟動任務時使用 AWS 的帳號。

JSON

```
{  
  "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. 透過以下命令擷取您所建立 IAM 政策的 ARN。將 *taskRolePolicy* 取代為您建立的政策名稱。

```

aws iam list-policies --scope Local --query 'Policies[?
PolicyName==`taskRolePolicy`].Arn'

```

4. 將您建立的 IAM 政策連接至 `ecsTaskRole` 角色。將 `policy-arn` 取代為您建立的政策 ARN。

```

aws iam attach-role-policy \
  --role-name ecsTaskRole \
  --policy-arn arn:aws:iam:111122223333:aws:policy/taskRolePolicy

```

建立角色之後，請為角色新增下列功能的其他許可。

功能	額外許可
使用 ECS Exec	ECS Exec 許可
使用私有 Amazon ECR 儲存庫中的映像	Amazon ECR 許可
使用 EC2 執行個體 (Windows 與 Linux)	Amazon EC2 執行個體的額外組態
使用外部執行個體	外部執行個體的額外組態
使用 Windows EC2 執行個體	Amazon EC2 Windows 執行個體的額外組態

Amazon ECR 許可

當應用程式程式碼需直接與 Amazon ECR 儲存庫互動時，要具備如下許可。請注意，對於只需從 Amazon ECR 提取映像的基本實作，任務 IAM 角色層級不需要這些許可。但 Amazon ECS 任務執行角色應該具有這些許可。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](#)」。

如果在容器中執行的應用程式程式碼需直接與 Amazon ECR API 互動，您應該將如下許可新增至任務 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 User Guide 中的 [Adding and Removing IAM Policies](#)。

將以下政策套用至任務 IAM 角色，為需直接與 Amazon ECR 互動的容器應用程式新增必要許可：

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ecr:BatchGetImage",
        "ecr:GetDownloadUrlForLayer",
        "ecr:GetAuthorizationToken"
      ],
      "Resource": "*"
    }
  ]
}
```

```
}
```

ECS Exec 許可

[ECS Exec](#) 功能需要任務 IAM 角色，以授予容器在受管 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 User Guide 中的 [Adding and Removing IAM Policies](#)。

針對您的任務 IAM 角色使用下列政策來新增所需的 SSM 許可。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ssmmessages:CreateControlChannel",
        "ssmmessages:CreateDataChannel",
        "ssmmessages:OpenControlChannel",
        "ssmmessages:OpenDataChannel"
      ],
      "Resource": "*"
    }
  ]
}
```

Amazon EC2 執行個體的額外組態

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 執行個體至少需要 1.11.0 版的容器代理，才能使用任務角色；不過，建議使用最新版本的容器代理程式。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](#)」。如果使用的是 Amazon ECS 最佳化 AMI，您的執行個體至少需要 1.11.0-1 版的 ecs-init 套件。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 最佳化 Linux AMI](#)。

如果容器執行個體未使用 Amazon ECS 最佳化 AMI，請將 `--net=host` 選項新增至 `docker run` 命令，為所需組態啟動代理程式及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 \(適用於非 Amazon ECS 最佳化 AMI\)](#)」。您也需要在容器執行個體上設定下列聯網命令，以便任務中的容器可以擷取其 AWS 登入資料：

```
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
```

## 外部執行個體的額外組態

您的外部執行個體至少需要 1.11.0 版的容器代理，才能使用任務 IAM 角色；不過，建議使用最新版本的容器代理程式。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 最佳化 Linux AMI](#)。

如果容器執行個體未使用 Amazon ECS 最佳化 AMI，請將 --net=host 選項新增至 docker run 命令，為所需組態啟動代理程式及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 \(適用於非 Amazon ECS 最佳化 AMI\)](#)」。您也需要在容器執行個體上設定下列聯網命令，以便任務中的容器可以擷取其 AWS 登入資料：

```
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 Windows 執行個體的額外組態

### Important

這僅適用於使用任務角色的 EC2 Windows 容器。

具有 Windows 功能的任務角色需要在 EC2 上進行額外組態設定。

- 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 容器引導指令碼](#)」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[使用負載平衡分佈 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 容器引導指令碼

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。角色會在您用來登入主控台或執行 AWS CLI 命令的帳戶中建立。

### 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 協助程式代表您呼叫 AWS APIs 所需的許可。如需受管政策的詳細資訊，請參閱 [AmazonEC2ContainerServiceforEC2Role](#)。

### 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，請透過如下程序連接受管 IAM 政策。

## AWS 管理主控台

建立 Elastic Container Service 的服務角色 (IAM 主控台)

1. 登入 AWS 管理主控台 並開啟位於 <https://console.aws.amazon.com/iam/> 的 IAM 主控台。
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. 在服務或使用案例欄位中，選擇 Elastic Container Service，然後選擇 Elastic Container Service 的 EC2 角色使用案例。
5. 選擇下一步。
6. 在許可政策區段中，確認已選取 AmazonEC2ContainerServiceforEC2Role 政策。

### Important

AmazonEC2ContainerServiceforEC2Role 受管政策應連接至容器執行個體 IAM 角色，否則您會收到使用 AWS 管理主控台 建立叢集的錯誤。

7. 選擇下一步。
8. 在角色名稱欄位中，輸入 ecsInstanceRole
9. 檢閱角色，然後選擇 Create role (建立角色)。

## AWS CLI

將所有 *user input* 值取代為實際值。

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. 使用以下命令，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 `AmazonEC2ContainerServiceForEC2Role` 受管政策連接至角色。

```
aws iam attach-role-policy \
 --policy-arn arn:aws:iam::aws:policy/service-role/
AmazonEC2ContainerServiceforEC2Role \
 --role-name ecsInstanceRole
```

建立角色之後，請為角色新增下列功能的其他許可。

| 功能                         | 額外許可                            |
|----------------------------|---------------------------------|
| Amazon ECR 具有容器映像          | <a href="#">Amazon ECR 許可</a>   |
| 讓 CloudWatch Logs 監控容器執行個體 | <a href="#">監控容器執行個體許可</a>      |
| Amazon S3 儲存貯體中的主機組態檔案     | <a href="#">Amazon S3 唯讀存取權</a> |

## Amazon ECR 許可

容器執行個體使用的 Amazon ECS 容器執行個體角色，必須擁有以下 Amazon ECR IAM 政策許可。

JSON

```
{
 "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。

## 設定 awsvpcTrunking 帳戶設定所需的許可

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。當您使用此功能時，建議建立 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 主幹功能的任務。如需有關 awsvpcTrunking 帳戶設定的資訊，請參閱[透過帳戶設定使用 Amazon ECS 功能](#)。

容器執行個體使用的容器執行個體角色，必須擁有以下設定帳戶設定的 IAM 政策許可

## JSON

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Action": [
 "ecs:ListAccountSettings",
 "ecs:ListAttributes",
 "ecs:PutAccountSetting"
],
 "Resource": "*"
 }
]
}
```

若要使用容器執行個體角色，請將如下內容新增至執行個體使用者資料：

```
#!/bin/bash
aws ecs put-account-setting --name awsvpcTrunking --value enabled --region region
ECS_CLUSTER=MyCluster>> /etc/ecs/ecs.config
EOF
```

如需有關新增使用者資料至 EC2 執行個體的詳細資訊，請參閱 Amazon EC2 User Guide 中的 [Run commands on your Linux instance at launch](#)。

## Amazon S3 唯讀存取權

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` 檔案的複本存放在私有儲存貯體中，使用 Amazon EC2 使用者資料安裝，AWS CLI 然後在執行個體啟動 `/etc/ecs/ecs.config` 時將您的組態資訊複製到。

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 [將 Amazon ECS 容器執行個體組態儲存於 Amazon S3 中](#)。

您可以使用下列 AWS CLI 命令來允許容器執行個體角色的 Amazon S3 唯讀存取。將 *ecsInstanceRole* 取代為您建立的角色名稱。

```
aws iam attach-role-policy \
 --role-name ecsInstanceRole \
 --policy-arn arn:aws:iam::aws:policy/AmazonS3ReadOnlyAccess
```

您也可以透過 IAM 主控台為角色新增 Amazon S3 唯讀存取權 (AmazonS3ReadOnlyAccess)。如需詳細資訊，請參閱 AWS Identity and Access Management User Guide 中的 [Updating permissions for a role](#)。

## 監控容器執行個體許可

在容器執行個體將日誌資料傳送至 CloudWatch Logs 之前，您必須建立 IAM 政策，以允許 Amazon ECS 代理程式將客戶的應用程式日誌寫入 CloudWatch (通常透過 `awslogs` 驅動程式處理)。建立政策後，請將政策連接至 `ecsInstanceRole`。

## AWS 管理主控台

若要使用 JSON 政策編輯器來建立政策

1. 登入 AWS 管理主控台，並在 <https://console.aws.amazon.com/iam/> 開啟 IAM 主控台。
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。

建立政策後，請將政策連接至容器執行個體角色。如需有關如何將政策連接至角色的資訊，請參閱 AWS Identity and Access Management User Guide 中的 [Updating permissions for a role](#)。

## 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. 使用以下命令，透過 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. 透過以下命令擷取您所建立 IAM 政策的 ARN。將 *cwlogspolicy* 取代為您建立的政策名稱。

```
aws iam list-policies --scope Local --query 'Policies[?
PolicyName==`cwlogspolicy`].Arn'
```

4. 使用下列命令，透過政策 ARN 將政策連接至容器執行個體 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

當您向叢集註冊現場部署伺服器或虛擬機器 (VM) 時，伺服器或 VM 需要 IAM 角色才能與 AWS APIs 通訊。您只需要為每個 AWS 帳戶建立此 IAM 角色一次。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 管理主控台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 主控台搜尋功能來搜尋 *ecsAnywhereRole*，確認帳戶是否已具備角色。如需詳細資訊，請參閱 IAM user guide 中的 [IAM console search](#)。

AWS 提供兩個 受管 IAM 政策，可用於建立 ECS Anywhere IAM 角色、AmazonSSMManagedInstanceCore 和 AmazonEC2ContainerServiceforEC2Role 政策。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。如需更多詳細資訊，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：



- 使用 `awslogs` 日誌驅動程式將容器日誌傳送到 CloudWatch Logs。
- 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` 角色還包含從 Amazon ECR 中提取映像所需的許可，任務執行角色無需再包含這些許可。

## 建立 Amazon ECS Anywhere 角色

將所有 *user input* 取代為自己的資訊。

1. 建立以 `ssm-trust-policy.json` 為名且具有下列信任政策的本機檔案。

JSON

```
{
 "Version": "2012-10-17",
 "Statement": {
 "Effect": "Allow",
 "Principal": { "Service": [
 "ssm.amazonaws.com"
] },
 "Action": "sts:AssumeRole"
 }
}
```

2. 使用下列 AWS CLI 命令來建立角色，並連接信任政策。

```
aws iam create-role --role-name ecsAnywhereRole --assume-role-policy-document
file://ssm-trust-policy.json
```

3. 使用下列命令連接 AWS 受管政策。

```
aws iam attach-role-policy --role-name ecsAnywhereRole --policy-arn
arn:aws:iam::aws:policy/AmazonSSManagedInstanceCore
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 User Guide 中的 [Creating a role using custom trust policies \(console\)](#)。

## Amazon ECS 基礎結構 IAM 角色

Amazon ECS 基礎結構 IAM 角色允許 Amazon ECS 代表您管理叢集中的基礎結構資源，並在以下情況使用：

- 您想將 Amazon EBS 磁碟區連接至 Fargate 或 EC2 啟動類型 Amazon ECS 任務。基礎結構角色允許 Amazon ECS 管理您任務的 Amazon EBS 磁碟區。

您可以使用 `AmazonECSInfrastructureRolePolicyForVolumes` 受管政策。

- 您想使用 Transport Layer Security (TLS) 來加密 Amazon ECS Service Connect 服務之間的流量。

您可以使用

`AmazonECSInfrastructureRolePolicyForServiceConnectTransportLayerSecurity` 受管政策。

- 您想要建立 Amazon VPC Lattice 目標群組。

您可以使用 `AmazonECSInfrastructureRolePolicyForVpcLattice` 受管政策。

- 您想要在 Amazon ECS 叢集中使用 Amazon ECS 受管執行個體。基礎結構角色允許 Amazon ECS 管理受管執行個體的生命週期。

您可以使用 `AmazonECSInfrastructureRolePolicyForManagedInstances` 受管政策。

- 您想要使用快速模式。基礎設施角色允許 Amazon ECS 佈建和管理快速模式服務所需的基礎設施元件，包括負載平衡、安全群組、SSL 憑證和自動擴展組態。

您可以使用 `AmazonECSInfrastructureRoleforExpressGatewayServices` 受管政策。

當 Amazon ECS 擔任此角色代表您採取動作時，事件會顯示在 AWS CloudTrail 中。如果 Amazon ECS 使用角色來管理連接至任務的 Amazon EBS 磁碟區，CloudTrail 日誌 `roleSessionName` 將為 `ECSTaskVolumesForEBS`。如果角色用於加密 Service Connect 服務之間的流量，CloudTrail 日誌 `roleSessionName` 將為 `ECSServiceConnectForTLS`。如果角色用於建立 VPC Lattice 的目標群組，CloudTrail 日誌 `roleSessionName` 將為 `ECSNetworkingWithVPC_Lattice`。如果角色用於管理 Amazon ECS 受管執行個體，CloudTrail 日誌 `roleSessionName` 將為 `ECSManagedInstancesForCompute`。您可以使用此名稱，透過篩選使用者名稱，在 CloudTrail 主控台中搜尋事件。

Amazon ECS 提供受管政策，其中包含磁碟區連接、TLS、VPC Lattice 與受管執行個體所需的許可。如需詳細資訊，請參閱《AWS 受管政策參考指南》中的

[AmazonECSInfrastructureRolePolicyForVolumes](#)、[AmazonECSInfrastructureRolePolicyForServiceConnectTransportLayerSecurity](#) 和 [AmazonECSInfrastructureRolePolicyForExpressGatewayServices](#)。

## 建立 Amazon ECS 基礎結構角色

將所有 *user input* 取代為自己的資訊。

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：

JSON

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Sid": "AllowAccessToECSForInfrastructureManagement",
 "Effect": "Allow",
 "Principal": {
 "Service": "ecs.amazonaws.com"
 },
 "Action": "sts:AssumeRole"
 }
]
}
```

2. 使用以下 AWS CLI 命令，`ecsInfrastructureRole` 使用您在上一個步驟中建立的信任政策來建立名為 `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，將受管政策連接至 `ecsInfrastructureRole` 角色。
  - 若要將 Amazon EBS 磁碟區連接至 Fargate 或 EC2 啟動類型 Amazon ECS 任務，請連接 `AmazonECSInfrastructureRolePolicyForVolumes` 受管政策。
  - 若要使用 Transport Layer Security (TLS) 加密 Amazon ECS Service Connect 服務之間的流量，請連接 `AmazonECSInfrastructureRolePolicyForServiceConnectTransportLayerSecurity` 受管政策。

- 若要建立 Amazon VPC Lattice 目標群組，請連接 `AmazonECSInfrastructureRolePolicyForVpcLattice` 受管政策。
- 您想要在 Amazon ECS 叢集中使用 Amazon ECS 受管執行個體，請連接 `AmazonECSInfrastructureRolePolicyForManagedInstances` 受管政策。

```
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
```

```
aws iam attach-role-policy \
 --role-name ecsInfrastructureRole \
 --policy-arn arn:aws:iam::aws:policy/
AmazonECSInfrastructureRolePolicyForManagedInstances
```

您也可以透過 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 User Guide 中的 [Creating a role using custom trust policies \(console\)](#)。

#### Important

如果 Amazon ECS 正在使用基礎結構角色來管理連接至任務的 Amazon EBS 磁碟區，請在停止使用 Amazon EBS 磁碟區的任務之前確保下列事項。

- 角色未被刪除。
- 角色的信任政策未被修改，不會移除 Amazon ECS 存取權 (`ecs.amazonaws.com`)。
- 未移除受管政策 `AmazonECSInfrastructureRolePolicyForVolumes`。如果必須修改角色的許可，至少保留 `ec2:DetachVolume`、`ec2>DeleteVolume`、`ec2:DescribeVolumes` 用於刪除磁碟區。

在停止具有已連接 Amazon EBS 磁碟區的任務之前刪除或修改角色，將導致任務卡在 DEPROVISIONING 中，且相關聯的 Amazon EBS 磁碟區無法刪除。Amazon ECS 將定期自動重試停止任務與刪除磁碟區的操作，直至還原必要的許可為止。您可透過 [DescribeTasks](#) API，檢視任務的磁碟區連接狀態及對應的狀態原因。

建立檔案之後，您必須授予使用者將角色傳遞至 Amazon ECS 的許可。

將基礎結構角色傳遞至 Amazon ECS 的許可

若要使用 ECS 基礎結構 IAM 角色，您必須授予使用者將角色傳遞至 Amazon ECS 的許可。將下列 `iam:PassRole` 許可連接至使用者。將 *ecsInfrastructureRole* 取代為您建立的基礎結構角色名稱。

JSON

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Action": "iam:PassRole",
 "Effect": "Allow",
 "Resource": ["arn:aws:iam::*:role/ecsInfrastructureRole"],
 "Condition": {
 "StringEquals": {"iam:PassedToService": "ecs.amazonaws.com"}
 }
 }
]
}
```

如需有關 `iam:Passrole` 和更新使用者許可的詳細資訊，請參閱AWS Identity and Access Management 《使用者指南》中的[授予使用者將角色傳遞至 AWS 服務的許可](#)和[變更 IAM 使用者的許可](#)。

## Amazon ECS 受管執行個體執行個體設定檔

執行個體設定檔是 IAM 容器，只擁有一個 IAM 角色，並允許 Amazon ECS 受管執行個體安全地擔任該角色。執行個體設定檔包含執行個體角色，ECS 代理程式會擔任此角色來向叢集註冊執行個體，並與 ECS 服務通訊。

### Important

如果您使用 Amazon ECS AWS 受管執行個體搭配受管基礎設施政策，執行個體描述檔必須命名為 `ecsInstanceRole`。如果您使用基礎結構角色的自訂政策，執行個體設定檔可以採用不同名稱。

### 使用信任政策建立角色

將所有 *user input* 取代為自己的資訊。

1. 建立名為 `ecsInstanceRole-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：

JSON

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Principal": { "Service": "ec2.amazonaws.com" },
 "Action": "sts:AssumeRole"
 }
]
}
```

2. 使用以下 AWS CLI 命令，`ecsInstanceRole` 使用您在上一個步驟中建立的信任政策來建立名為的角色。

```
aws iam create-role \
 --role-name ecsInstanceRole \
 --assume-role-policy-document file://ecsInstanceRole-trust-policy.json
```

3. 將 AWS 受管AmazonECSInstanceRolePolicyForManagedInstances政策連接至ecsInstanceRole角色。

```
aws iam attach-role-policy \
 --role-name ecsInstanceRole \
 --policy-arn arn:aws:iam::aws:policy/
AmazonECSInstanceRolePolicyForManagedInstances
```

 Note

如果您選擇套用最低權限許可，並改為指定自己的許可，您可以新增下列許可，以協助疑難排解 Amazon ECS 受管執行個體的任務相關問題：

- `ecs:StartTelemetrySession`
- `ecs:PutSystemLogEvents`

您也可以透過 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 User Guide 中的 [Creating a role using custom trust policies \(console\)](#)。

建立檔案之後，您必須授予使用者將角色傳遞至 Amazon ECS 的許可。

使用 建立執行個體描述檔 AWS CLI

建立角色之後，請使用 AWS CLI 建立執行個體設定檔：

```
aws iam create-instance-profile --instance-profile-name ecsInstanceRole
```

將角色新增至執行個體設定檔：

```
aws iam add-role-to-instance-profile \
 --instance-profile-name ecsInstanceRole \
 --role-name ecsInstanceRole
```

驗證設定檔是否建立成功：

```
aws iam get-instance-profile --instance-profile-name ecsInstanceRole
```

## 負載平衡器的 Amazon ECS 基礎結構 IAM 角色

負載平衡器的 Amazon ECS 基礎結構 IAM 角色允許 Amazon ECS 代表您管理叢集中的基礎結構資源，並在以下情況使用：

- 您想要搭配 Amazon ECS 使用藍/綠部署。基礎結構角色允許 Amazon ECS 管理部署的負載平衡器資源。
- 您需要 Amazon ECS 在部署操作期間建立、修改或刪除負載平衡器資源，例如目標群組與接聽程式。

當 Amazon ECS 擔任此角色代表您採取動作時，事件會顯示在 AWS CloudTrail 中。如果 Amazon ECS 使用此角色來管理藍/綠部署的負載平衡器資源，CloudTrail 日誌 `roleSessionName` 將為 `ECSNetworkingWithELB` 或 `ecs-service-scheduler`。您可以使用此名稱，透過篩選使用者名稱，在 CloudTrail 主控台中搜尋事件。

Amazon ECS 提供受管政策，其中包含負載平衡器管理所需的許可。如需詳細資訊，請參閱 AWS Managed Policy Reference Guide 中的 [AmazonECSInfrastructureRolePolicyForLoadBalancers](#)。

### 負載平衡器的 Amazon ECS 基礎結構角色

將所有 *user input* 取代為自己的資訊。

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：

JSON

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Sid": "AllowAccessToECSForInfrastructureManagement",
 "Effect": "Allow",
 "Principal": {
 "Service": "ecs.amazonaws.com"
 },
 "Action": "sts:AssumeRole"
 }
]
}
```



```
}
```

2. 使用以下 AWS CLI 命令，`ecsInfrastructureRoleForLoadBalancers` 使用您在上一個步驟中建立的信任政策來建立名為 `ecsInfrastructureRoleForLoadBalancers` 的角色。

```
aws iam create-role \
 --role-name ecsInfrastructureRoleForLoadBalancers \
 --assume-role-policy-document file://ecs-infrastructure-trust-policy.json
```

3. 將 AWS 受管 `AmazonECSInfrastructureRolePolicyForLoadBalancers` 政策連接至 `ecsInfrastructureRoleForLoadBalancers` 角色。

```
aws iam attach-role-policy \
 --role-name ecsInfrastructureRoleForLoadBalancers \
 --policy-arn arn:aws:iam::aws:policy/
AmazonECSInfrastructureRolePolicyForLoadBalancers
```

您也可以透過 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 User Guide 中的 [Creating a role using custom trust policies \(console\)](#)。

#### Important

如果 Amazon ECS 正在使用基礎結構角色來管理藍/綠部署的負載平衡器資源，請在刪除或修改角色之前確保下列事項：

- 若進行中的部署作業尚未完成，不得刪除該角色。
- 角色的信任政策未被修改，不會移除 Amazon ECS 存取權 (`ecs.amazonaws.com`)。
- 若進行中的部署作業尚未完成，不得移除 `AmazonECSInfrastructureRolePolicyForLoadBalancers` 受管政策。

在藍/綠部署進行期間刪除或修改角色，可能導致部署失敗，使服務處於不一致狀態。

建立檔案之後，您必須授予使用者將角色傳遞至 Amazon ECS 的許可。

## 將基礎結構角色傳遞至 Amazon ECS 的許可

若要將 ECS 基礎結構 IAM 角色用於負載平衡器，您必須授予使用者將角色傳遞至 Amazon ECS 的許可。將下列 `iam:PassRole` 許可連接至使用者。將 *`ecsInfrastructureRoleForLoadBalancers`* 取代為您建立的基礎結構角色名稱。

### JSON

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Action": "iam:PassRole",
 "Effect": "Allow",
 "Resource": [
 "arn:aws:iam::*:role/ecsInfrastructureRoleForLoadBalancers",
 "Condition": {
 "StringEquals": {
 "iam:PassedToService": "ecs.amazonaws.com"
 }
 }
]
 }
]
}
```

如需有關 `iam:PassRole` 和更新使用者許可的詳細資訊，請參閱《AWS Identity and Access Management 使用者指南》中的[授予使用者將角色傳遞至 AWS 服務的許可](#)和[變更 IAM 使用者的許可](#)。

## Amazon ECS CodeDeploy IAM 角色

在您搭配使用 CodeDeploy 藍/綠部署類型與 Amazon ECS 之前，CodeDeploy 服務需要許可來代您更新 Amazon ECS 服務。這些許可由 CodeDeploy IAM 角色 (`ecsCodeDeployRole`) 提供。

### Note

使用者也需要許可來使用 CodeDeploy；[所需的 IAM 許可](#) 中描述了這些許可。

提供兩個受管政策。如需詳細資訊，請參閱 AWS Managed Policy Reference Guide 中的如下項目：

- [AWSCodeDeployRoleForECS](#) – 授予 CodeDeploy 使用相關聯動作更新任何資源的許可。

- [AWSCodeDeployRoleForECSLimited](#) – 授予 CodeDeploy 更多的有限許可。

## 建立 CodeDeploy 角色

您可以透過如下程序建立 Amazon ECS 的 CodeDeploy 角色

### AWS 管理主控台

#### 建立 CodeDeploy 的服務角色 (IAM 主控台)

1. 登入 AWS 管理主控台 並開啟位於 <https://console.aws.amazon.com/iam/> 的 IAM 主控台。
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. 在服務或使用案例欄位中選擇 CodeDeploy，然後選擇 CodeDeploy – ECS 使用案例。
5. 選擇下一步。
6. 在連接許可政策區段中，確保已選取 AWSCodeDeployRoleForECS 政策。
7. 選擇下一步。
8. 在角色名稱欄位中輸入 ecsCodeDeployRole。
9. 檢閱角色，然後選擇 Create role (建立角色)。

### AWS CLI

將所有 *user input* 取代為自己的資訊。

1. 建立名為 codedeploy-trust-policy.json 的檔案，其中包含用於 CodeDeploy IAM 角色的信任政策。

#### JSON

```
{
 "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 `AWSCodeDeployRoleForECSLimited` 受管政策連接至 `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執行角色時，您必須將每個任務執行角色或任務角色覆寫的 `iam:PassRole` 許可新增至 `CodeDeploy` 角色作為政策。

### 任務執行角色許可

當服務中的任務需要任務執行角色時，您必須將每個任務執行角色或任務角色覆寫的 `iam:PassRole` 許可新增至 `CodeDeploy` 角色作為政策。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，將該政策連接至 `CodeDeploy` 角色。

### 建立政策

#### AWS 管理主控台

若要使用 JSON 政策編輯器來建立政策

1. 登入 AWS 管理主控台 並開啟位於 <https://console.aws.amazon.com/iam/> 的 IAM 主控台。

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。

建立政策後，請將政策連接至 CodeDeploy 角色。如需有關如何將政策連接至角色的資訊，請參閱 AWS Identity and Access Management User Guide 中的 [Update permissions for a role](#)。

## AWS CLI

將所有 *user input* 取代為自己的資訊。

1. 建立稱為 `blue-green-iam-passrole.json` 的檔案，其中具有以下內容。

JSON

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Action": "iam:PassRole",
 "Resource": ["arn:aws:iam::*:role/code-deploy-role"],
 "Condition": {
 "StringEquals": {"iam:PassedToService":
 "ecs.amazonaws.com"}
 }
 }
]
}
```

2. 使用以下命令，透過 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. 透過以下命令擷取您所建立 IAM 政策的 ARN。

```
aws iam list-policies --scope Local --query 'Policies[?
PolicyName==`cdTaskExecutionPolicy`].Arn'
```

4. 使用以下命令，將政策連接至 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

在搭配 EventBridge 規則與目標使用 Amazon ECS 排程任務之前，需先授予 EventBridge 服務代表您執行 Amazon ECS 任務的許可。這些許可是由 EventBridge IAM 角色 (`ecsEventsRole`) 所提供。

AmazonEC2ContainerServiceEventsRole 政策顯示如下。

JSON

```
{
 "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"]
 }
 }
 }
]
}
```

若排程任務需要使用任務執行角色、任務角色或任務角色覆寫，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` 許可新增至 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。

## JSON

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Action": "iam:PassRole",
 "Resource": [
 "arn:aws:iam::111122223333:role/ecsTaskExecutionRole_or_TaskRole_name"
]
 }
]
}
```

設定排程任務時，您可以選擇讓 為您 AWS 管理主控台 建立 EventBridge 角色。如需詳細資訊，請參閱[使用 Amazon EventBridge 排程器排程 Amazon ECS 任務](#)。

### 建立 EventBridge 角色

將所有 *user input* 取代為自己的資訊。

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：

## JSON

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Sid": "",
 "Effect": "Allow",
 "Principal": {
 "Service": "events.amazonaws.com"
 },
 "Action": "sts:AssumeRole"
 }
]
}
```



```
}
```

2. 使用以下命令，透過在上一個步驟中建立的信任政策來建立名為 `ecsEventsRole` 的 IAM 角色。

```
aws iam create-role \
 --role-name ecsEventsRole \
 --assume-role-policy-document file://eventbridge-trust-policy.json
```

3. 使用下列命令將 AWS 受管 `AmazonEC2ContainerServiceEventsRole` 連接至 `ecsEventsRole` 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/>) 來建立角色。如需詳細資訊，請參閱 IAM User Guide 中的 [Creating a role using custom trust policies \(console\)](#)。

將政策連接至 **ecsEventsRole** 角色

您可以透過如下程序，將任務執行角色的許可新增至 EventBridge IAM 角色。

## AWS 管理主控台

若要使用 JSON 政策編輯器來建立政策

1. 登入 AWS 管理主控台 並開啟位於 <https://console.aws.amazon.com/iam/> 的 IAM 主控台。
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::111122223333: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。

建立政策後，請將政策連接至 EventBridge 角色。如需有關如何將政策連接至角色的資訊，請參閱 AWS Identity and Access Management User Guide 中的 [Update permissions for a role](#)。

## AWS CLI

將所有 *user input* 取代為自己的資訊。

1. 建立稱為 ev-iam-passrole.json 的檔案，其中具有以下內容。

### JSON

```

{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Action": "iam:PassRole",
 "Resource": [

```

```

 "arn:aws:iam::111122223333:role/ecsTaskExecutionRole_or_TaskRole_name"
]
}
]
}

```

2. 使用以下 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. 透過以下命令擷取您所建立 IAM 政策的 ARN。

```

aws iam list-policies --scope Local --query 'Policies[?
PolicyName==`eventsTaskExecutionPolicy`].Arn'

```

4. 使用以下命令，透過政策 ARN 將政策連接至 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 AWS Managed Policy Reference 中的 [AmazonECS\\_FullAccess](#)。

### 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](#)
- 建立要用於排程任務的 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至叢集所需的許可

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 (ecsExternalInstanceRole) 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。
  - iam:AttachRolePolicy
  - iam:CreateRole
  - iam:CreateInstanceProfile
  - iam:AddRoleToInstanceProfile
  - iam:ListInstanceProfilesForRole
  - iam:GetRole
- ssm – 允許主體使用 Systems Manager 註冊外部執行個體。

### Note

若要選擇現有 ecsExternalInstanceRole，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 ecsExternalInstanceRole 角色。

## JSON

```
{
 "Statement": [
 {
 "Effect": "Allow",
 "Action": [
 "iam:AttachRolePolicy",
 "iam:CreateRole",
 "iam:CreateInstanceProfile",
 "iam:AddRoleToInstanceProfile",
```

```
 "iam:ListInstanceProfilesForRole",
 "iam:GetRole"
],
 "Resource": "arn:aws:iam::*:role/ecsExternalInstanceRole"
},
{
 "Effect": "Allow",
 "Action": ["iam:PassRole", "ssm:CreateActivation"],
 "Resource": "arn:aws:iam::*:role/ecsExternalInstanceRole"
}
]
```

## 註冊任務定義所需的許可

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。
  - iam:AttachRolePolicy
  - iam:CreateRole
  - iam:GetRole

### 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。

## JSON

```
{
 "Statement": [
 {
 "Effect": "Allow",
 "Action": [
 "iam:AttachRolePolicy",
```

```
 "iam:CreateRole",
 "iam:GetRole"
],
 "Resource": "arn:aws:iam::*:role/ecsTaskExecutionRole"
}
]
```

## 透過 Amazon Q Developer 在主控台中提供建議所需的許可

若要讓 Amazon Q Developer 在 Amazon ECS 主控台中提供建議，您必須為 IAM 使用者或角色啟用正確的 IAM 許可。您必須新增 `codewhisperer:GenerateRecommendations` 許可。

### JSON

```
{
 "Statement": [
 {
 "Sid": "AmazonQDeveloperPermissions",
 "Effect": "Allow",
 "Action": ["codewhisperer:GenerateRecommendations"],
 "Resource": "*"
 }
]
}
```

若要在 Amazon ECS 主控台中使用內嵌聊天，您必須為 IAM 使用者或角色啟用正確的 IAM 許可。您必須新增 `q:SendMessage` 許可。

### JSON

```
{
 "Statement": [
 {
 "Sid": "AmazonQDeveloperInlineChatPermissions",
 "Effect": "Allow",
 "Action": ["q:SendMessage"],
 "Resource": "*"
 }
]
}
```

```
]
}
```

## 為排程任務建立 EventBridge 規則所需的許可

當您排程任務且想要建立新的 CloudWatch Events (ecsEventsRole) 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。

JSON

```
{
 "Statement": [
 {
 "Effect": "Allow",
 "Action": [
 "iam:AttachRolePolicy",
 "iam:CreateRole",
 "iam:GetRole",
 "iam:PassRole"
],
 "Resource": "arn:aws:iam::*:role/ecsEventsRole"
 }
]
}
```

## 檢視服務部署所需的許可

當您遵循授予最低權限的最佳實務時，必須新增額外許可，才能在主控台中檢視服務部署。

您需要存取下列動作：

- ListServiceDeployments
- DescribeServiceDeployments
- DescribeServiceRevisions

您需要存取下列資源：

- 服務
- 服務部署
- 服務修訂版

下列範例政策包含必要的許可，並將動作限制於指定的服務。

將 `account`、`cluster-name` 與 `service-name` 取代為實際值。

JSON

```
{
 "Statement": [
 {
 "Effect": "Allow",
 "Action": [
 "ecs:ListServiceDeployments",
 "ecs:DescribeServiceDeployments",
 "ecs:DescribeServiceRevisions"
],
 "Resource": [
 "arn:aws:ecs:us-east-1:123456789012:service/cluster-name/service-name",
 "arn:aws:ecs:us-east-1:123456789012:service-deployment/cluster-name/service-name/*",
 "arn:aws:ecs:us-east-1:123456789012:service-revision/cluster-name/service-name/*"
]
 }
]
}
```



```
 }
]
}
```

## 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 [Adding and Removing IAM Policies](#)

- events:DescribeRule
- events:ListTargetsByRule
- logs:DescribeLogGroups

### JSON

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Action": [
 "events:DescribeRule",
 "events:ListTargetsByRule",
 "logs:DescribeLogGroups"
],
 "Resource": "*"
 }
]
}
```

## 在 Container Insights 中啟用 Amazon ECS 生命週期事件所需的許可

設定生命週期事件需要以下許可：

- events:PutRule
- events:PutTargets
- logs:CreateLogGroup

## JSON

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Action": [
 "events:PutRule",
 "events:PutTargets",
 "logs:CreateLogGroup"
],
 "Resource": "*"
 }
]
}
```

### 使用的 Amazon ECS 主控台所需的許可 CloudFormation

使用 AWS 管理主控台 建立資源之前，您需要確保擁有正確的 IAM 許可。如需有關如何設定 Amazon ECS 主控台許可的資訊，請參閱 [Amazon ECS 主控台所需的許可](#)。

Amazon ECS 主控台採用 技術 AWS CloudFormation，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](#)。

#### 建立叢集所需的許可

當您在主控台中建立叢集時，需要額外的許可，以授予您管理 CloudFormation 堆疊的許可。

以下是所需的其他許可：

- `cloudformation` - 允許委託人建立和管理 CloudFormation 堆疊。在使用 AWS 管理主控台 建立 Amazon ECS 叢集以及後續管理這些叢集時，這為必要項。

- `ssm` – 允許 CloudFormation 參考最新的 Amazon ECS 最佳化 AMI。使用 建立 Amazon ECS 叢集 時，這是必要的 AWS 管理主控台。

下列政策包含必要的 CloudFormation 許可，並將動作限制在 Amazon ECS 主控台中建立的資源。

JSON

```
{
 "Statement": [
 {
 "Effect": "Allow",
 "Action": [
 "cloudformation:CreateStack",
 "cloudformation>DeleteStack",
 "cloudformation:DescribeStack*",
 "cloudformation:UpdateStack"
],
 "Resource": [
 "arn:*:cloudformation:*:*:stack/Infra-ECS-Cluster-*"
]
 },
 {
 "Effect": "Allow",
 "Action": "ssm:GetParameters",
 "Resource": [
 "arn:aws:ssm:*:*:parameter/aws/service/ecs/optimized-ami/amazon-linux-2*/*",
 "arn:aws:ssm:*:*:parameter/aws/service/ecs/optimized-ami/amazon-linux-2023*/*"
]
 }
]
}
```

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，則需要額外許可，以便主控台可以在使用叢集自動擴展功能時，將標籤新增至自動擴展群組。

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。

JSON

```
{
 "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": "*"
 }
]
}
```

建立服務所需的許可

當您在主控台中建立服務時，需要額外的許可，以授予您管理 CloudFormation 堆疊的許可。以下是所需的其他許可：

- `cloudformation` - 允許委託人建立和管理 CloudFormation 堆疊。在使用 AWS 管理主控台 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。

JSON

```
{
 "Statement": [
 {
 "Effect": "Allow",
 "Action": [
 "cloudformation:CreateStack",
 "cloudformation>DeleteStack",
 "cloudformation:DescribeStack*",
 "cloudformation:UpdateStack"
],
 "Resource": [
 "arn:*:cloudformation:*:*:stack/ECS-Console-V2-Service-*"
]
 }
]
}
```

## Amazon ECS 藍/綠部署中 Lambda 函式所需的許可

當您在 Amazon ECS 藍/綠部署中使用 Lambda 函式作為部署 lifecycle hook 時，您需要建立具有特定許可的 IAM 角色。這個角色可讓 Amazon ECS 在部署生命週期的各個階段調用 Lambda 函式。

以下是所需的其他許可：

- `lambda:InvokeFunction` – 允許 Amazon ECS 在部署程序期間調用設定為 lifecycle hook 的 Lambda 函式。

對於信任政策，您需要允許服務擔任該角色：

## JSON

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Principal": {
 "Service": [
 "ecs.amazonaws.com"
]
 },
 "Action": "sts:AssumeRole"
 }
]
}
```

## Amazon ECS 服務自動擴展所需的 IAM 許可

透過結合 Amazon ECS、CloudWatch 和 Application Auto Scaling API 可實現服務自動擴展。服務是使用 Amazon ECS 建立與更新的，警示是使用 CloudWatch 建立的，而擴展政策是使用 Application Auto Scaling 建立的。

除建立及更新服務所需的標準 IAM 許可外，也需要與服務自動擴展設定互動的下列許可。

- 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*`

Application Auto Scaling 服務也需要描述 Amazon ECS 服務和 CloudWatch 警示的許可，以及代您修改服務所需計數的許可。此 `sns`：權限許可用於 CloudWatch 在超出閾值時向 Amazon SNS 主題傳送的通知。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](#)。

如果您已在 Amazon ECS 可使用 CloudWatch 指標之前建立您的 Amazon ECS 容器執行個體角色，則您可能需要新增 `ecs:StartTelemetrySession` 許可。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標籤，AWS 會執行額外的授權，以確認指派正確的許可來建立標籤。

- `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。

下列範例示範一個政策，該政策允許使用者在建立叢集期間同時新增標籤。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 控制對特定標籤的存取

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。

如需有關多值條件的詳細資訊，請參閱 IAM User Guide 中的 [Conditions with multiple context keys or values](#)。

## 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。這稱為以屬性型存取控制 (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 範例政策](#)。

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 範例政策

您可以使用 IAM 政策來授予使用者許可，以檢視和使用 Amazon ECS 主控台內的特定資源。您可以使用上一節中的範例政策；不過，這些政策是專為使用 AWS CLI 或 AWS SDK 提出的請求而設計。

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。

JSON

```
{
 "Version": "2012-10-17",
 "Statement": [
```

```
{
 "Action": [
 "ecs:DeleteCluster"
],
 "Effect": "Allow",
 "Resource": "arn:aws:ecs:us-east-1:111122223333: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](#)
- [我未獲得執行 iam:PassRole 的授權](#)
- [我想要允許以外的人員 AWS 帳戶 存取我的 Amazon ECS 資源](#)
- [我的 Amazon ECS 受管執行個體描述檔發生問題](#)
- [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。

## 我未獲得執行 iam:PassRole 的授權

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 服務可讓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 沒有將角色傳遞給服務的許可。

```
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。

## 我想要允許 以外的人員 AWS 帳戶 存取我的 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](#)。
- 若要了解如何在您擁有 AWS 帳戶的資源之間提供存取權，請參閱《[IAM 使用者指南](#)》中的[在您的 AWS 帳戶的另一個 IAM 使用者中提供存取權](#)。
- 若要了解如何將資源的存取權提供給第三方 AWS 帳戶，請參閱《[IAM 使用者指南](#)》中的[將存取權提供給第三方 AWS 帳戶擁有](#)。
- 如需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 \(聯合身分\)](#)。
- 如需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 中的跨帳戶資源存取](#)。

## 我的 Amazon ECS 受管執行個體描述檔發生問題

如果您使用 Amazon ECS 提供的 `AmazonECSInfrastructureRolePolicyForManagedInstances` 受管政策，則執行個體描述檔必須具有 `ecsInstanceRole` 做為字首。原因是受管政策僅獲授權傳遞具有此字首的角色。

## 其他疑難排解資源

下列頁面提供錯誤代碼的資訊：

- [Amazon ECS 已停止任務錯誤訊息](#)
- [檢視 Amazon ECS 服務事件訊息](#)

## Amazon ECS 的 IAM 最佳實務

您可以使用 AWS Identity and Access Management (IAM) 透過以規則為基礎的政策來管理和控制對 AWS 服務和資源的存取，以用於身分驗證和授權目的。更具體地說，透過此服務，您可以使用套用到使用者、群組或角色的政策來控制對 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 overview of access management: Permissions and policies](#)。

### 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。如需詳細資訊，請參閱 [Amazon ECS 的政策資源](#)。

### 讓叢集資源作為管理界限

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。

### 透過自動化管道，隔離最終使用者與 API

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》中的 [教學課程：具 CodePipeline 的 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。

JSON

```
{
 "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 主體會啟動與停止所有標籤鍵為 Department、標籤值為 Accounting 的任務。

JSON

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Action": [
 "ecs:StartTask",
 "ecs:StopTask",
 "ecs:RunTask"
]
 }
]
}
```

```
],
 "Resource": "arn:aws:ecs:us-east-1:123456789012:cluster/my-cluster",
 "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 AWS Identity and Access Management User Guide 中的 [AWS security audit guidelines](#)。

## 在 Amazon Elastic Container Service 中記錄和監控

監控是維護 Amazon Elastic Container Service 和 AWS 解決方案可靠性、可用性和效能的重要部分。您應該從解決方案的所有部分 AWS 收集監控資料，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 警示

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。此動作是傳送到 Amazon Simple Notification Service (Amazon SNS) 主題或 Amazon EC2 Auto Scaling 政策的通知。CloudWatch 警示不會只因處於特定狀態就調用動作，狀態必須已變更並已維持一段指定的時間。如需詳細資訊，請參閱[使用 CloudWatch 監控 Amazon ECS](#)。

如果服務的任務使用的是 Fargate，您可以使用 CloudWatch 警示，根據 CloudWatch 指標 (例如 CPU 與記憶體使用率) 來橫向擴充或縮減服務中的任務數量。如需詳細資訊，請參閱[自動擴展您的 Amazon ECS 服務](#)。

如果叢集的任務或服務使用的是 EC2，您可以使用 CloudWatch 警示，根據 CloudWatch 指標 (例如叢集記憶體保留量) 來橫向擴充或縮減容器執行個體數量。

### Amazon CloudWatch Logs

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 的容器。

## Amazon CloudWatch Events

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。如需詳細資訊，請參閱本指南中的[使用 EventBridge 自動化因應 Amazon ECS 錯誤](#)，以及 Amazon EventBridge User Guide 中的[EventBridge is the evolution of Amazon CloudWatch Events](#)。

## AWS CloudTrail 日誌

CloudTrail 提供由 Amazon ECS AWS 中的使用者、角色或服務所採取之動作的記錄。您可以利用 CloudTrail 所收集的資訊來判斷向 Amazon ECS 發出的請求，以及發出請求的 IP 地址、人員、時間和其他詳細資訊。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 利用從為數十萬 AWS 客戶提供服務的最佳實務。會 Trusted Advisor 檢查您的 AWS 環境，然後在有機會節省成本、改善系統可用性和效能，或協助填補安全漏洞時提出建議。所有 AWS 客戶都可以存取五個 Trusted Advisor 檢查。擁有商業或企業支援計劃的客戶可以檢視所有 Trusted Advisor 檢查。

如需詳細資訊，請參閱《支援 使用者指南》中的[AWS Trusted Advisor](#)。

## AWS Compute Optimizer

AWS Compute Optimizer 是一種 服務，可分析 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 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 主控台儀表板可讓您at-a-glance檢視 AWS 環境的狀態。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

若要了解 是否 AWS 服務 在特定合規計劃範圍內，請參閱[AWS 服務 合規計劃範圍內](#)然後選擇您感興趣的合規計劃。如需一般資訊，請參閱[AWS 合規計劃](#)。

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[下載報告下載 AWS Artifact](#)。



您使用時的合規責任 AWS 服務 取決於資料的機密性、您公司的合規目標，以及適用的法律和法規。如需使用時合規責任的詳細資訊 AWS 服務，請參閱 [AWS 安全文件](#)。

## Amazon ECS 的合規與安全最佳實務

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。

### 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](#)

### 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 儲存選項都可以使用各種靜態加密機制，例如 Amazon S3、Amazon EBS 和 AWS KMS。您可以部署覆蓋網路 (例如 VNS3 或 Weave Net)，確保在容器之間傳輸的 PHI 完全加密，或提供備援加密層。您還應使用完整的記錄功能，並將所有容器日誌導向至 Amazon CloudWatch。如需有關使用基礎結構安全性的最佳實務資訊，請參閱《安全支柱 AWS Well Architected Framework》中的 [基礎結構保護](#)。

### AWS Security Hub CSPM

使用 AWS Security Hub CSPM。這 AWS 服務 可讓您全面檢視其中的安全狀態 AWS。Security Hub CSPM 使用安全控制來評估您的 AWS 資源，並檢查是否符合安全產業標準和最佳實務。如需支援的服務和控制項清單，請參閱 [Security Hub CSPM 控制項參考](#)。

## 搭配 Amazon ECS 執行時期監控使用 Amazon GuardDuty

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。GuardDuty 使用機器學習 (ML) 模型，以及異常和威脅偵測功能，持續監控不同的日誌來源和執行時期活動，以識別環境中的潛在安全風險和惡意活動和排定其優先順序。

在 GuardDuty 中使用執行時期監控來識別惡意或未經授權的行為。執行時期監控透過持續監控 AWS 日誌和聯網活動來識別惡意或未經授權的行為，來保護在 Fargate 和 EC2 上執行的工作負載。執行時期監控使用輕量且全受管的 GuardDuty 安全代理程式，可分析主機上的行為，例如檔案存取、程序執行與網路連線。這涵蓋的問題包括：提升權限、使用暴露的憑證、與惡意 IP 位址或網域通訊，以及 Amazon EC2 執行個體與容器工作負載上出現惡意軟體。如需詳細資訊，請參閱 GuardDuty User Guide 中的 [GuardDuty Runtime Monitoring](#)。

### 合規建議

我們建議您儘早在業務中與合規計劃擁有者互動，並使用 AWS 共同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。如需詳細資訊，請參閱 [AWS Amazon ECS 的共同責任模型](#)。

## AWS Fargate 聯邦資訊處理標準 (FIPS-140)

美國聯邦資訊處理標準 (FIPS-140) 是一項美國和加拿大政府的安全標準，具體說明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。如需有關 FIPS-140 合規的詳細資訊，請參閱 [Federal Information Processing Standard \(FIPS\) 140-3](#)。

### 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。
- Fargate 支援 FIPS-140 版本 140.3
- FIPS-140 合規預設為關閉。您必須將其開啟。
- Amazon CloudWatch 不支援雙堆疊 FIPS 端點，因此無法監控採用純 IPv6 組態且需符合 FIPS-140 規範的 Amazon ECS 任務。
- 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 聯邦資訊處理標準 \(FIPS-140\) 合規”](#)。
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。

將 `cluster-name` 取代為叢集名稱，將 `task-id` 取代為任務的 ID 或 ARN，並將 `container-name` 取代為任務中要執行命令的容器名稱。

傳回值 "1" 表示您正在使用 FIPS。

```
aws ecs execute-command \
 --cluster cluster-name \
 --task task-id \
 --container container-name \
 --interactive \
 --command "cat /proc/sys/crypto/fips_enabled"
```

## 使用 CloudTrail 進行 Fargate FIPS-140 稽核

當您建立 AWS 帳戶時，您的帳戶中會開啟 CloudTrail。當 API 和主控台活動在 Amazon ECS 中發生時，該活動會與事件歷史記錄中的其他 AWS 服務事件一起記錄在 CloudTrail 事件中。您可以在 AWS 帳戶中檢視、搜尋和下載最近的事件。如需詳細資訊，請參閱《使用 CloudTrail 事件歷史記錄檢視事件》<https://docs.aws.amazon.com/awscloudtrail/latest/userguide/view-cloudtrail-events.html>。

若持續記錄您 AWS 帳戶中的事件，包括 Amazon ECS 的事件，請建立 CloudTrail 用來將日誌檔案交付至 Amazon S3 儲存貯體的線索。根據預設，當您在主控台建立線索時，線索會套用到所有區域。線索會記錄 AWS 分割區中所有區域的事件，並將日誌檔案傳送到您指定的 Amazon S3 儲存貯體。此外，您可以設定其他 AWS 服務，以進一步分析和處理 CloudTrail 日誌中所收集的事件資料。如需詳細資訊，請參閱[the section called “使用 記錄 Amazon ECS API 呼叫 AWS CloudTrail”](#)。

以下範例顯示的是示範 `PutAccountSettingDefault` API 動作的 CloudTrail 日誌項目：

```
{
 "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

Amazon Elastic Container Service 是受管服務，受到 AWS 全球網路安全的保護。如需 AWS 安全服務以及如何 AWS 保護基礎設施的資訊，請參閱[AWS 雲端安全](#)。若要使用基礎設施安全最佳實務來設計您的 AWS 環境，請參閱安全支柱 AWS Well-Architected Framework 中的[基礎設施保護](#)。

您可以使用 AWS 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。

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，這只會隔離網路內特定 VPC 對指定 Amazon ECS 資源 AWS 的網路存取。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。介面端點採用一種技術 AWS PrivateLink，可讓您使用私有 IP 地址私下存取 Amazon ECS APIs。AWS PrivateLink 會限制 VPC 和 Amazon ECS 與 Amazon 網路之間的所有網路流量。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 日新增區域的端點考量事項

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：

### Note

如果您未設定所有端點，流量會經過公有端點，而不是 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`
- 當您使用範本在新區域中建立 AWS 資源，且範本已從 2023 年 12 月 23 日之前引進的 區域複製時，取決於複製區域，請執行下列其中一項操作。

例如，複製來源區域是美國東部 (維吉尼亞北部) (us-east-1)。複製目標區域是加拿大西部 (卡加利) (ca-west-1)。

| Configuration         | Action                                                                                                                                                                                                 |  |
|-----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--|
| 複製來源區域沒有任何 VPC 端點。    | 為新區域建立所有三個 VPC 端點 (例如 <code>com.amazonaws.ca-west-1.ecs-agent</code> )。                                                                                                                                |  |
| 複製來源區域包含區域特定的 VPC 端點。 | <ol style="list-style-type: none"><li>為新區域建立所有三個 VPC 端點 (例如 <code>com.amazonaws.ca-west-1.ecs-agent</code>)。</li><li>刪除複製區域的所有三個 VPC 端點 (例如 <code>com.amazonaws.us-east-1.ecs-agent</code>)。</li></ol> |  |

## Fargate 的 Amazon ECS VPC 端點的考量事項

當 Fargate 任務所部署的 VPC 內已存在 `ecr.dkr` 與 `ecr.api` 的 VPC 端點時，Fargate 任務會使用該端點。如果沒有 VPC 端點，則會使用 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 的任務不需要 Amazon ECS 的介面 VPC 端點，但您可能需要以下要點中所述的 Amazon ECR、Secrets Manager 或 Amazon CloudWatch Logs 的介面 VPC 端點。
- 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 \(AWS PrivateLink\)](#)。

**⚠ 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[Fargate 任務透過介面端點提取 Amazon ECR 映像的許可](#)。
  - 如果任務以純 IPv6 組態設定且使用 Amazon ECR 雙堆疊映像 URI，注意 Amazon ECR 不支援雙堆疊介面 VPC 端點。如需詳細資訊，請參閱 Amazon Elastic Container Registry User Guide 中的[Getting started with making requests over IPv6](#)。
- 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 VPC 沒有網際網路閘道且任務使用的是 awslogs 日誌驅動程式來將日誌資訊傳送至 CloudWatch Logs，您必須為 CloudWatch Logs 建立介面 VPC 端點。如需詳細資訊，請參閱《Amazon CloudWatch Logs 使用者指南》中的[使用 CloudWatch Events 搭配介面 VPC 端點](#)。
  - 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 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 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 VPC 沒有網際網路閘道且任務使用的是 awslogs 日誌驅動程式來將日誌資訊傳送至 CloudWatch Logs，您必須為 CloudWatch Logs 建立介面 VPC 端點。如需詳細資訊，請參閱《Amazon CloudWatch Logs 使用者指南》中的[使用 CloudWatch Events 搭配介面 VPC 端點](#)。
- 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。

## 了解 Amazon ECS 端點命名模式

必須了解的是，Amazon ECS 代理程式可能會向帶有編號字尾的端點發出請求，例如：

- 代理程式端點的 `ecs-a-1.region.amazonaws.com`、`ecs-a-2.region.amazonaws.com` 等
- 遙測端點的 `ecs-t-1.region.amazonaws.com`、`ecs-t-2.region.amazonaws.com` 等

此行為的成因在於 Amazon ECS 代理程式會透過 [DiscoverPollEndpoint](#) API 動態判定應連線的特定端點。如果 VPC 端點未正確處理這些帶編號的端點變體，即使您已針對基本名稱設定 VPC 端點，代理程式仍會回復使用公開端點。

### DiscoverPollEndpoint API 的角色

Amazon ECS 代理程式會透過 [DiscoverPollEndpoint](#) API 探索要用於輪詢任務的適當端點。當代理程式呼叫此 API 時，會收到可能包含編號字尾的特定端點 URL。為確保 VPC 端點正常運作，網路組態必須允許代理程式：



1. 存取 DiscoverPollEndpoint API
2. 連線至傳回的端點 URL，包括帶有編號字尾的端點 URL

如果正在疑難排解 VPC 端點連線問題，請確認代理程式可以同時連線至基本端點，以及 DiscoverPollEndpoint API 可能傳回的任何編號變化。

## 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[AWS 使用界面 VPC 端點程序存取](#)服務來建立下列端點。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。

### Note

如果您未設定所有端點，流量會經過公有端點，而不是 VPC 端點。

當您建立端點時，Amazon ECS 也會為端點建立私有 DNS 名稱。例如，ecs-agent 為 `ecs-a.region.amazonaws.com`，ecs-telemetry 為 `ecs-t.region.amazonaws.com`。

- `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 端點使用 `ecs:poll` API，而 ecs-telemetry 端點使用 `ecs:poll` 與 `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": [
 {
 "Principal": "*",
 "Effect": "Allow",
 "Action": [
 "ecs:CreateCluster",
 "ecs:ListClusters"
]
 }
]
}
```

```
],
 "Resource": [
 "*"
]
 }
]
}
```

## AWS Amazon ECS 的共同責任模型

安全與合規是 AWS 和 客戶之間共同責任。此共用模型有助於減輕客戶的操作負擔，因為 AWS 操作、管理和控制從主機作業系統和虛擬化層到服務操作所在設施的實體安全性的元件。客戶應承擔相關責任並負責管理訪客作業系統 (包括更新與安全性修補程式)，以及其他相關應用程式軟體，還有設定 AWS 提供的安全群組防火牆。客戶應審慎思考所選的服務，因為使用的服務、服務與客戶 IT 環境的整合情形，以及適用的法律與法規不同，客戶應承擔的責任也會不同。此共同責任的性質也提供允許部署的彈性和客戶控制。

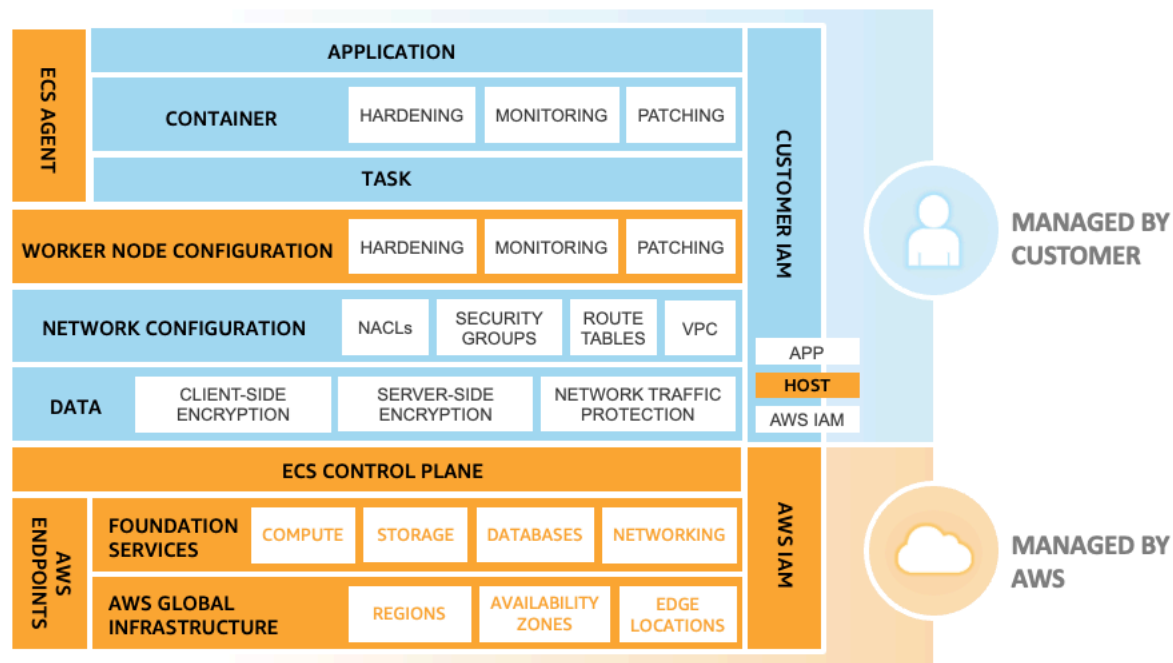
### Fargate

下圖顯示 Fargate 啟動類型的共同責任模式。Fargate 會在隔離的硬體虛擬化環境中執行所有工作負載。因此，每個任務都會取得專用基礎結構容量。在 Fargate 上執行的容器化工作負載不會與其他任務共用作業系統、Linux 核心、網路介面、暫時性儲存、CPU 或記憶體。使用 Fargate 時，客戶不需負責保護執行其容器的運算基礎結構。Fargate 將佈建與修補客戶工作負載執行所在的基礎結構。如需詳細資訊，請參閱[Amazon ECS 上 AWS Fargate 的任務淘汰和維護](#)。

您負責管理下列資源：

- 網路組態，包括 VPC、NACL、安全群組與路由表
- 用戶端與服務儲存體加密。如需詳細資訊，請參閱[Amazon ECS 任務的儲存選項](#)。
- 容器映像。如需詳細資訊，請參閱[Amazon ECS 任務與容器安全最佳實務](#)。
- 使用任務角色的應用程式 IAM 許可。如需詳細資訊，請參閱[Amazon ECS 任務 IAM 角色](#)。

## AWS Shared Responsibility Model for Amazon ECS with Fargate

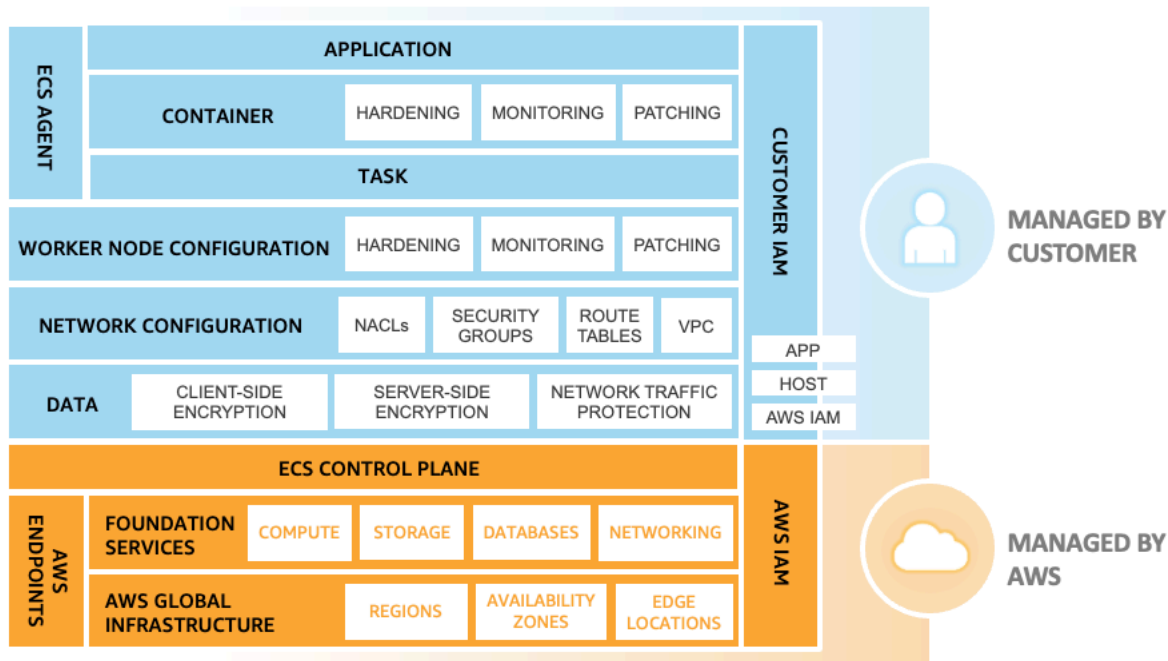


## EC2

下圖顯示 EC2 的共同責任。當您在 EC2 執行個體上執行任務時，除下列資源外，您還要負責維護 EC2 執行個體：

- Amazon ECS 代理程式。
- EC2 執行個體 AMI，包括修補與強化。
- 網路組態，包括 VPC、NACL、安全群組與路由表。
- 用戶端與服務儲存體加密。如需詳細資訊，請參閱[Amazon ECS 任務的儲存選項](#)。
- 容器映像。如需詳細資訊，請參閱[Amazon ECS 任務與容器安全最佳實務](#)。
- 使用任務角色的應用程式 IAM 許可。如需詳細資訊，請參閱[Amazon ECS 任務 IAM 角色](#)。

## AWS Shared Responsibility Model for Amazon ECS



## Amazon ECS 受管執行個體的共同責任模式

Amazon ECS 受管執行個體為容器化工作負載提供受管解決方案，結合了 Fargate 的操作簡單性，以及完整範圍的 Amazon EC2 執行個體類型和功能的存取權。AWS 處理基礎設施佈建、修補、擴展和維護，同時客戶仍可控制其應用程式和特定組態。

與 Fargate 不同，在 Amazon ECS 受管執行個體上執行的容器化工作負載，會與相同執行個體上的其他任務共用作業系統、Linux 核心、網路介面、暫時性儲存、CPU、記憶體與 GPU 資源。Amazon ECS 透過在較大型執行個體上置放多個任務來最佳化基礎結構使用率，將閒置容量降至最低。

## AWS 責任

使用 Amazon ECS 受管執行個體時，AWS 負責：

- 執行個體佈建與生命週期管理
- 作業系統修補與安全性更新
- 基礎結構擴展與最佳化
- 執行個體替代與維護 (執行個體生命週期上限為 21 天)
- 存取控制限制 (無 SSH 存取、無 SSM Session Manager 存取)

- Amazon EC2 執行個體儲存體加密，這是直接連接至執行個體的儲存體。如需詳細資訊，請參閱 [Amazon EC2 中的資料保護](#)。
- Amazon ECS 會在建立時管理連接至 Amazon EC2 執行個體的磁碟區，包括根磁碟區與資料磁碟區。
- Amazon ECS 在底層使用 Amazon EC2 受管執行個體。如需有關 Amazon EC2 受管執行個體的詳細資訊，請參閱 [Security in Amazon EC2](#)。

## 客戶責任

您負責管理下列資源：

- 網路組態，包括 VPC、NACL、安全群組與路由表
- 用戶端與服務儲存體加密。如需詳細資訊，請參閱 [Amazon ECS 任務的儲存選項](#)。
- 容器映像。如需詳細資訊，請參閱 [Amazon ECS 任務與容器安全最佳實務](#)。
- 使用任務角色的應用程式 IAM 許可。如需詳細資訊，請參閱 [Amazon ECS 任務 IAM 角色](#)。
- 應用程式層級組態與監控
- 任務與服務定義
- 共用底層執行個體資源之工作負載的安全考量
- 啟用時的特殊權限容器組態與增強版 Linux 功能 (CAP\_NET\_ADMIN、CAP\_BPF 等)
- 透過 Amazon ECS API 的管理操作 (無法透過 SSH 或 SSM 直接存取執行個體)

## 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。其中包括傳輸中加密、網路分隔和隔離、防火牆、流量路由和可觀測性。

### 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 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 負載平衡器搭配 SNI 使用智慧憑證選取演算法。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](#)
- 使用 TLS 憑證的端對端加密

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的容器](#)
- [將 Transport Layer Security \(TLS\) 維護到容器第 2 部分：使用 AWS 私有憑證授權單位](#)

## 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 聯網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。它也是 Amazon ECS 上可用於 AWS Fargate 任務的唯一模式。

## 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路介面 (ENI) 並將其附加至任務。佈建 ENI 時，任務會註冊在 AWS 安全群組中。安全群組做為虛擬防火牆，可用於控制傳入及傳出的流量。



如果使用自訂防火牆搭配任務或服務，請新增傳出規則，允許 Amazon ECS 代理程式管理端點 (ecs-a-\*.*region*.amazonaws.com)、遙測端點 (ecs-t-\*.*region*.amazonaws.com) 與 Service Connect Envoy 管理端點 (ecs-sc.*region*.api.aws) 的流量。

## AWS PrivateLink 和 Amazon ECS

AWS PrivateLink 是一種聯網技術，可讓您為不同的 AWS 服務建立私有端點，包括 Amazon ECS。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 中，並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 端點的說明，請參閱 [Amazon ECS 介面 Amazon VPC 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的所有 AWS 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。這可防止不屬於的 IAM 主體推送和提取映像 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。

## 容器代理程式設定

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。

### 在適用的情況下搭配 Amazon ECS 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。如果服務位於 ALB 之後，請使用安全群組，僅允許來自其他使用與 ALB 相同安全群組之網路資源的傳入流量。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，在獨立的 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 [VPC 端點](#)。

## 在 Amazon ECS 需要時設定 AWS PrivateLink 端點

您應該在必要時設定 AWS PrivateLink 端點。如果您的安全政策阻止您將網際網路閘道 (IGW) 連接至 Amazon VPCs，請設定 Amazon ECS 和其他服務的 AWS PrivateLink 端點，例如 Amazon ECR、AWS Secrets Manager 和 Amazon CloudWatch。

## 使用 Amazon VPC Flow Logs 分析 Amazon ECS 中長時間執行任務的流入及流出流量

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。

## Amazon ECS 任務與容器安全最佳實務

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」。如需詳細資訊，請參閱 distroless 上的 [GitHub](#) 文件。

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 Docker 文件中的[Multi-stage builds](#)。

## 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。

Amazon ECR 提供兩種使用通用漏洞披露 (CVE) 資料庫的基本掃描版本：

- AWS 原生基本掃描 – AWS 使用原生技術，現在為 GA 且建議使用。這項改進的基本掃描機制旨在為客戶提供更精準的掃描結果，跨多種主流作業系統強化漏洞偵測能力。這有助於客戶進一步提升容器映像的安全性。依預設，所有新客戶登錄檔都會選擇加入此改進版本。
- Clair 基本掃描 – 舊版基本掃描採用開放原始碼 Clair 專案技術，現已棄用。如需 Clair 的詳細資訊，請參閱 GitHub 上的 [Clair](#)。

[AWS 依區域](#)列出的所有區域都支援 AWS 原生和 Clair 基本掃描，但 2024 年 9 月之後新增的區域除外。由於 Clair 支援已被棄用，新增區域將不再支援 Clair，且自 2025 年 10 月 1 日起，所有區域都將停止支援 Clair。

若有，Amazon ECR 會使用上游分發來源提供的 CVE 嚴重性程度。若無，則會使用通用漏洞評分系統 (CVSS) 分數。CVSS 分數可用於取得 NVD 漏洞嚴重性等級。如需詳細資訊，請參閱 [NVD 漏洞嚴重性等級](#)。

您也可以從 Amazon ECR 主控台內或呼叫 [DescribeImageScanFindings](#) API 來查看掃描的結果。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 CSPM](#)說明如何從 AWS Security Hub CSPM 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。您不一定知道影像中的內容，而且影像可能會有漏洞。

## 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。您可以使用 Amazon Inspector。如需詳細資訊，請參閱 Amazon Inspector User Guide 中的 [Scanning Amazon ECR container images with Amazon Inspector](#)。

## 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 Dockerfile 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：

- [Dockerfile-lint](#) 是來自 RedHat 的開放原始碼工具，可用於檢查檔案是否符合最佳實務。

- [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 需要您指定 CPU 和記憶體限制，因為它會將這些值用於計費目的。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，您可以使用 AWS Lambda 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。如需有關每個 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 的影像會使用 a AWS Key Management Service (AWS KMS) 受管金鑰自動進行靜態加密。如果您寧願使用自己的金鑰，Amazon ECR 現在支援使用客戶受管金鑰 (CMK) 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。

您可以透過下列任一教學課程，進一步了解如何開始使用 Amazon ECS。

| 教學課程概觀                      | 進一步了解                                                  |  |
|-----------------------------|--------------------------------------------------------|--|
| 開始在 Fargate 上使用 Amazon ECS。 | <a href="#">了解如何為 Fargate 建立 Amazon ECS Linux 任務</a>   |  |
| 開始在 Fargate 上使用 Windows 容器。 | <a href="#">了解如何為 Fargate 建立 Amazon ECS Windows 任務</a> |  |
| 開始使用 EC2 Windows 容器。        | <a href="#">了解如何為 EC2 建立 Amazon ECS Windows 任務</a>     |  |

您可以使用下列任何教學課程，使用在 Amazon ECS 上部署任務 AWS CLI

| 教學課程概觀                   | 進一步了解                                                         |  |
|--------------------------|---------------------------------------------------------------|--|
| 為 Fargate 建立 Linux 任務。   | <a href="#">使用 為 Fargate 建立 Amazon ECS Linux 任務 AWS CLI</a>   |  |
| 為 Fargate 建立 Windows 任務。 | <a href="#">使用 為 Fargate 建立 Amazon ECS Windows 任務 AWS CLI</a> |  |
| 為 EC2 建立 Linux 任務。       | <a href="#">使用 為 EC2 建立 Amazon ECS 任務 AWS CLI</a>             |  |

您可以透過下列任一教學課程，進一步了解監控與記錄相關知識。

| 教學課程概觀                        | 進一步了解                                                 |  |
|-------------------------------|-------------------------------------------------------|--|
| 設定一個簡單的 Lambda 函式，用於監聽任務事件並將之 | <a href="#">設定 Amazon ECS 監聽 CloudWatch Events 事件</a> |  |



| 教學課程概觀                                                                                           | 進一步了解                                                                        |  |
|--------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------|--|
| 寫入 CloudWatch Logs 日誌串流。                                                                         |                                                                              |  |
| 設定一項 Amazon EventBridge 事件規則，使其僅擷取因其中一個必要容器終止而導致任務停止執行的任務事件。                                     | <a href="#">針對 Amazon ECS 任務停止事件傳送 Amazon Simple Notification Service 提醒</a> |  |
| 串連最初屬於一筆內容但分割至多筆記錄或日誌明細行的日誌訊息。                                                                   | <a href="#">串連多行或堆疊追蹤的 Amazon ECS 日誌訊息</a>                                   |  |
| 在於 Amazon ECS 中執行的 Windows 執行個體上部署 Fluent Bit 容器，將 Windows 任務產生的日誌串流至 Amazon CloudWatch 進行集中式記錄。 | <a href="#">在 Amazon ECS Windows 容器上部署 Fluent Bit</a>                        |  |

您可以透過下列任一教學課程，進一步了解如何在 Amazon ECS 上搭配 Active Directory 身分驗證使用群組受管服務帳戶。

| 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。     | <a href="#">在 Amazon ECS 上為 EC2 Linux 容器使用 gMSA</a>           |  |
| 在 EC2 上搭配 Windows 容器使用群組受管服務帳戶。   | <a href="#">了解如何在 Amazon ECS 上為 EC2 Windows 容器使用 gMSA</a>     |  |
| 在 Fargate 上搭配 Linux 容器使用群組受管服務帳戶。 | <a href="#">在 Fargate 上為 Linux 容器使用 gMSA</a>                  |  |
| 建立一個執行 Windows 容器的任務，該容器具備透過無網域   | <a href="#">gMSA 使用 搭配無網域使用 Amazon ECS Windows 容器 AWS CLI</a> |  |

| 教學課程概觀                           | 進一步了解 |  |
|----------------------------------|-------|--|
| 群組受管服務帳戶存取 Active Directory 的憑證。 |       |  |

## 使用 為 Fargate 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。使用 AWS CLI 的最新版本。如需有關如何升級至最新版本的詳細資訊，請參閱 [Installing or updating to the latest version of the AWS CLI](#)。

### Note

您可以使用雙堆疊服務端點，透過 IPv4 和 IPv6 AWS CLI 從、SDKs 和 Amazon ECS API 與 Amazon ECS 互動。如需詳細資訊，請參閱 [使用 Amazon ECS 雙堆疊端點](#)。

### 主題

- [先決條件](#)
- [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](#)

## 先決條件

本教學課程假設已完成下列先決條件。

- AWS CLI 已安裝並設定最新版本的。如需安裝或升級 AWS CLI、[安裝或更新至最新版本 AWS CLI](#) 的詳細資訊。
- 已完成「[設定以使用 Amazon ECS。](#)」中的步驟。

- 您的 IAM 使用者擁有 [AmazonECS\\_FullAccess](#) IAM 政策範例中指定的所需許可。
- 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 角色及其他先決條件的詳細資訊，請參閱[使用 Amazon ECS Exec 監控 Amazon ECS 容器](#)。
- （選用）AWS CloudShell 是一種工具，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AWS CloudShell 《使用者指南》中的[什麼是 AWS CloudShell ?](#)。

## 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 [先決條件](#) 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
```

## 使用 為 Fargate 建立 Amazon ECS Windows 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。如需有關如何升級至最新版本的詳細資訊，請參閱 [Installing or updating to the latest version of the AWS CLI](#)。

### Note

您可以使用雙堆疊服務端點，透過 IPv4 和 IPv6 AWS CLI 從、SDKs 和 Amazon ECS API 與 Amazon ECS 互動。如需詳細資訊，請參閱[使用 Amazon ECS 雙堆疊端點](#)。

### 主題

- [先決條件](#)
- [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](#)

## 先決條件

本教學課程假設已完成下列先決條件。

- AWS CLI 已安裝並設定最新版本的。如需安裝或升級的詳細資訊 AWS CLI，請參閱[安裝或更新至最新版本的 AWS CLI](#)。
- 已完成「[設定以使用 Amazon ECS。](#)」中的步驟。
- 您的 IAM 使用者擁有 [AmazonECS\\_FullAccess](#) IAM 政策範例中指定的所需許可。
- 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 是一種工具，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AWS CloudShell 《使用者指南》中的[什麼是 AWS CloudShell？](#)。

## 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 AWS CLI 兩種方式傳遞至 `aws ecs register-task-definition`：您可以將任務定義 JSON 儲存為檔案，並使用 `--cli-input-json file://path_to_file.json` 選項將其傳遞。

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-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
```

## 使用 為 EC2 建立 Amazon ECS 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、執行任務，以及執行其他常見案例。使用 AWS CLI 的最新版本。如需有關如何升級至最新版本的詳細資訊，請參閱 [Installing or updating to the latest version of the AWS CLI](#)。

### Note

您可以使用雙堆疊服務端點，透過 IPv4 和 IPv6 AWS CLI 從、SDKs 和 Amazon ECS API 與 Amazon ECS 互動。如需詳細資訊，請參閱[使用 Amazon ECS 雙堆疊端點](#)。

### 主題

- [先決條件](#)
- [建立 叢集](#)

- [使用 Amazon ECS AMI 啟動容器執行個體](#)
- [列出容器執行個體](#)
- [描述容器執行個體](#)
- [註冊任務定義](#)
- [列出任務定義](#)
- [建立服務](#)
- [列出服務](#)
- [描述服務](#)
- [描述執行中的任務](#)
- [測試 Web 伺服器](#)
- [清除資源](#)

## 先決條件

本教學課程假設已完成下列先決條件：

- AWS CLI 已安裝並設定最新版本的。如需安裝或升級的詳細資訊 AWS CLI，請參閱[安裝或更新至最新版本的 AWS CLI](#)。
- 已完成「[設定以使用 Amazon ECS。](#)」中的步驟。
- 您的 IAM 使用者擁有 [AmazonECS\\_FullAccess](#) IAM 政策範例中指定的所需許可。
- 您已建立可供使用的容器執行個體 IAM 角色。如需詳細資訊，請參閱 [Amazon ECS 容器執行個體 IAM 角色](#)。
- 您已建立可供使用的 VPC。如需詳細資訊，請參閱[the section called “建立 Virtual Private Cloud”](#)。
- （選用）AWS CloudShell 是一種工具，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AWS CloudShell 《使用者指南》中的[什麼是 AWS CloudShell？](#)。

## 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"
 }
}
```

## 使用 Amazon ECS AMI 啟動容器執行個體

容器執行個體是執行 Amazon ECS 容器代理程式並已在叢集中註冊的 EC2 執行個體。在本節中，您將使用 ECS 最佳化 AMI 來啟動 EC2 執行個體。

使用 啟動容器執行個體 AWS CLI

1. AWS 區域 使用下列命令擷取 的最新 ECS 最佳化 Amazon Linux 2 AMI ID。此命令使用 AWS Systems Manager 參數存放區來取得最新的 ECS 最佳化 AMI ID。AMI 包含預先安裝的 Amazon ECS 容器代理程式與 Docker 執行時期。

```
aws ssm get-parameters --names /aws/service/ecs/optimized-ami/amazon-linux-2/
recommended --query 'Parameters[0].Value' --output text | jq -r '.image_id'
```

輸出：

```
ami-abcd1234
```

2. 建立一個安全群組，該群組允許透過 SSH 存取來管理容器執行個體，也允許 Web 伺服器的 HTTP 存取。

```
aws ec2 create-security-group --group-name ecs-tutorial-sg --description "ECS tutorial security group"
```

輸出：

```
{
 "GroupId": "sg-abcd1234"
}
```

3. 執行下列命令，將傳入規則新增至安全群組。

```
aws ec2 authorize-security-group-ingress --group-id sg-abcd1234 --protocol tcp --port 80 --cidr 0.0.0.0/0
```

輸出：

```
{
 "Return": true,
 "SecurityGroupRules": [
 {
 "SecurityGroupRuleId": "sgr-efgh5678",
 "GroupId": "sg-abcd1234",
 "GroupOwnerId": "123456789012",
 "IsEgress": false,
 "IpProtocol": "tcp",
 "FromPort": 80,
 "ToPort": 80,
 "CidrIpv4": "0.0.0.0/0"
 }
]
}
```

安全群組現在允許來自指定 IP 範圍的 SSH 存取，也允許來自任何位置的 HTTP 存取。在生產環境中，您應將 SSH 存取限制為特定 IP 位址，並視需要考量限制 HTTP 存取。

4. 建立一個用於對容器執行個體進行 SSH 存取的 EC2 金鑰對。

```
aws ec2 create-key-pair --key-name ecs-tutorial-key --query 'KeyMaterial' --output
text > ecs-tutorial-key.pem
chmod 400 ecs-tutorial-key.pem
```

私有金鑰會儲存至本機電腦，且具有適用於 SSH 存取的適當許可。

5. 使用 ECS 最佳化 AMI 啟動 EC2 執行個體，並將執行個體設定為加入叢集。

```
aws ec2 run-instances --image-id ami-abcd1234 --instance-type t3.micro --key-
name ecs-tutorial-key --security-group-ids sg-abcd1234 --iam-instance-profile
Name=ecsInstanceRole --user-data '#!/bin/bash
echo ECS_CLUSTER=MyCluster >> /etc/ecs/ecs.config'
{
 "Instances": [
 {
 "InstanceId": "i-abcd1234",
 "ImageId": "ami-abcd1234",
 "State": {
 "Code": 0,
 "Name": "pending"
 },
 "PrivateDnsName": "",
 "PublicDnsName": "",
 "StateReason": {
 "Code": "pending",
 "Message": "pending"
 },
 "InstanceType": "t3.micro",
 "KeyName": "ecs-tutorial-key",
 "LaunchTime": "2025-01-13T10:30:00.000Z"
 }
]
}
```

使用者資料指令碼會設定 Amazon ECS 代理程式，將執行個體註冊至 MyCluster。執行個體使用 ecsInstanceRole IAM 角色，該角色為代理程式提供必要許可。



## 列出容器執行個體

在您啟動容器執行個體後的幾分鐘內，Amazon ECS 代理程式會將該執行個體註冊至您的 MyCluster 叢集。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 MyCluster
```

輸出：

```
{
 "containerInstanceArns": [
 "arn:aws:ecs:us-east-1:aws_account_id:container-instance/container_instance_ID"
]
}
```

## 描述容器執行個體

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 MyCluster --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。

## 註冊任務定義

您必須先註冊任務定義，才能在 Amazon ECS 叢集中執行任務。任務定義是分在一組的容器清單。以下範例是使用 nginx 映像的簡單任務定義。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](#)」。

```
{
 "family": "nginx-task",
 "containerDefinitions": [
 {
 "name": "nginx",
 "image": "public.ecr.aws/ecs-sample-image/amazon-ecs-sample:latest",
 "cpu": 256,
 "memory": 512,
 "essential": true,
 "portMappings": [
 {
 "containerPort": 80,
 "hostPort": 80,
 "protocol": "tcp"
 }
]
 }
],
 "requiresCompatibilities": ["EC2"],
 "networkMode": "bridge"
}
```

上述範例 JSON 可以透過 AWS CLI 兩種方式傳遞至：您可以將任務定義 JSON 儲存為檔案，並使用 `--cli-input-json file://path_to_file.json` 選項將其傳遞。或者，您可以逸出 JSON 中的引號，在命令列上傳遞 JSON 容器定義。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/nginx.json
```

`register-task-definition` 會在完成註冊後，傳回任務定義的描述。

```
{
 "taskDefinition": {
 "taskDefinitionArn": "arn:aws:ecs:us-east-1:123456789012:task-definition/nginx-task:1",
 "family": "nginx-task",
 "revision": 1,
 "status": "ACTIVE",
 "containerDefinitions": [
 {
 "name": "nginx",
 "image": "public.ecr.aws/docker/library/nginx:latest",
 "cpu": 256,
 "memory": 512,
 "essential": true,
 "portMappings": [
 {
 "containerPort": 80,
 "hostPort": 80,
 "protocol": "tcp"
 }
],
 "environment": [],
 "mountPoints": [],
 "volumesFrom": []
 }
],
 "volumes": [],
 "networkMode": "bridge",
 "compatibilities": [
 "EC2"
],
 "requiresCompatibilities": [
```

```
 "EC2"
]
}
}
```

## 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` 與 `revision` 值，可在呼叫 `create-service` 時一起使用。

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

輸出：

```
{
 "taskDefinitionArns": [
 "arn:aws:ec2:us-east-1:aws_account_id:task-definition/sleep360:1",
 "arn:aws:ec2:us-east-1:aws_account_id:task-definition/sleep360:2",
 "arn:aws:ec2:us-east-1:aws_account_id:task-definition/nginx-task: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"
]
}
```

## 建立服務

在為帳戶註冊任務並啟動已註冊至叢集的容器執行個體後，即可建立 Amazon ECS 服務。該服務會使用您註冊的任務定義，同時執行並維持所需數量的任務。在此範例中，您會在 `MyCluster` 叢集中置放 `nginx:1` 任務定義的單一執行個體。

```
aws ecs create-service --cluster MyCluster --service-name nginx-service --task-
definition nginx-task:1 --desired-count 1
```

輸出：

```
{
 "service": {
```

```
 "serviceArn": "arn:aws:ecs:us-east-1:aws_account_id:service/MyCluster/nginx-
service",
 "serviceName": "nginx-service",
 "clusterArn": "arn:aws:ecs:us-east-1:aws_account_id:cluster/MyCluster",
 "taskDefinition": "arn:aws:ecs:us-east-1:aws_account_id:task-definition/nginx-
task:1",
 "desiredCount": 1,
 "runningCount": 0,
 "pendingCount": 0,
 "launchType": "EC2",
 "status": "ACTIVE",
 "createdAt": "2025-01-13T10:45:00.000Z"
 }
}
```

## 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 ID 或完整的 ARN，稍後將用以描述服務。

```
aws ecs list-services --cluster MyCluster
```

輸出：

```
{
 "taskArns": [
 "arn:aws:ecs:us-east-1:aws_account_id:task/task_ID"
]
}
```

## 描述服務

使用下列命令描述服務，取得有關服務的詳細資訊。

```
aws ecs describe-services --cluster MyCluster --services nginx-service
```

輸出：

```
{
```

```

 "services": [
 {
 "serviceArn": "arn:aws:ecs:us-east-1:aws_account_id:service/MyCluster/
nginx-service",
 "serviceName": "nginx-service",
 "clusterArn": "arn:aws:ecs:us-east-1:aws_account_id:cluster/MyCluster",
 "taskDefinition": "arn:aws:ecs:us-east-1:aws_account_id:task-definition/
nginx-task:1",
 "desiredCount": 1,
 "runningCount": 1,
 "pendingCount": 0,
 "launchType": "EC2",
 "status": "ACTIVE",
 "createdAt": "2025-01-13T10:45:00.000Z",
 "events": [
 {
 "id": "abcd1234-5678-90ab-cdef-1234567890ab",
 "createdAt": "2025-01-13T10:45:30.000Z",
 "message": "(service nginx-service) has started 1 tasks: (task
abcd1234-5678-90ab-cdef-1234567890ab)."
 }
]
 }
]
 }
}

```

## 描述執行中的任務

描述服務之後，請執行下列命令，取得作為服務一部分的執行中任務的詳細資訊。

```
aws ecs list-tasks --cluster MyCluster --service-name nginx-service
```

輸出：

```

{
 "tasks": [
 {
 "taskArn": "arn:aws:ecs:us-east-1:aws_account_id:task/MyCluster/
abcd1234-5678-90ab-cdef-1234567890ab",
 "clusterArn": "arn:aws:ecs:us-east-1:aws_account_id:cluster/MyCluster",
 "taskDefinitionArn": "arn:aws:ecs:us-east-1:aws_account_id:task-definition/
nginx-task:1",

```



```

 "containerInstanceArn": "arn:aws:ecs:us-east-1:aws_account_id:container-
instance/MyCluster/abcd1234-5678-90ab-cdef-1234567890ab",
 "lastStatus": "RUNNING",
 "desiredStatus": "RUNNING",
 "containers": [
 {
 "containerArn": "arn:aws:ecs:us-east-1:aws_account_id:container/
MyCluster/abcd1234-5678-90ab-cdef-1234567890ab/abcd1234-5678-90ab-cdef-1234567890ab",
 "taskArn": "arn:aws:ecs:us-east-1:aws_account_id:task/MyCluster/
abcd1234-5678-90ab-cdef-1234567890ab",
 "name": "nginx",
 "lastStatus": "RUNNING",
 "networkBindings": [
 {
 "bindIP": "0.0.0.0",
 "containerPort": 80,
 "hostPort": 80,
 "protocol": "tcp"
 }
]
 }
],
 "createdAt": "2025-01-13T10:45:00.000Z",
 "startedAt": "2025-01-13T10:45:30.000Z"
 }
]
}

```

## 測試 Web 伺服器

### 測試 Web 伺服器

1. 透過執行下列命令，擷取容器執行個體的公有 IP 位址。

```
aws ec2 describe-instances --instance-ids i-abcd1234 --query
'Reservations[0].Instances[0].PublicIpAddress' --output text
```

輸出：

```
203.0.113.25
```

2. 擷取 IP 位址之後，請使用 IP 位址執行下列 curl 命令。

```
curl http://203.0.113.25
```

輸出：

```
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
...
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you can see this page, the nginx web server is successfully installed and
 working.</p>
...
</body>
</html>
```

出現的 nginx 歡迎頁面即確認服務已成功執行，可從網際網路存取。

## 清除資源

若要避免產生費用，需清理在此教學課程中建立的資源。

### 清理資源

1. 將服務的所需任務數量更新為零，然後刪除服務。

```
aws ecs update-service --cluster MyCluster --service nginx-service --desired-count
0
{
 "service": {
 "serviceArn": "arn:aws:ecs:us-east-1:123456789012:service/MyCluster/nginx-
service",
 "serviceName": "nginx-service",
 "desiredCount": 0,
 "runningCount": 1,
 "pendingCount": 0,
 "status": "ACTIVE"
 }
}
```

```
}
```

2. 等待執行中任務停止，然後刪除服務。

```
aws ecs delete-service --cluster MyCluster --service nginx-service
{
 "service": {
 "serviceArn": "arn:aws:ecs:us-east-1:123456789012:service/MyCluster/nginx-
service",
 "serviceName": "nginx-service",
 "status": "DRAINING"
 }
}
```

3. 終止已建立的容器執行個體。

```
aws ec2 terminate-instances --instance-ids i-abcd1234
{
 "TerminatingInstances": [
 {
 "InstanceId": "i-abcd1234",
 "CurrentState": {
 "Code": 32,
 "Name": "shutting-down"
 },
 "PreviousState": {
 "Code": 16,
 "Name": "running"
 }
 }
]
}
```

4. 清理已建立的安全群組與金鑰對。

```
aws ec2 delete-security-group --group-id sg-abcd1234
aws ec2 delete-key-pair --key-name ecs-tutorial-key
rm ecs-tutorial-key.pem
```

5. 刪除 Amazon ECS 叢集。

```
aws ecs delete-cluster --cluster MyCluster
{
```

```
"cluster": {
 "clusterArn": "arn:aws:ecs:us-east-1:123456789012:cluster/MyCluster",
 "clusterName": "MyCluster",
 "status": "INACTIVE"
}
```

## 設定 Amazon ECS 監聽 CloudWatch Events 事件

了解如何設定一個簡單的 Lambda 函式，用於監聽 Amazon ECS 任務事件並將之寫入 CloudWatch Logs 日誌串流。

### 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 “為 Fargate 工作負載建立叢集”](#)」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. 在 <https://console.aws.amazon.com/lambda/> 開啟 AWS Lambda 主控台。
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。如果一切設定正確，則在此教學課程的最後，您會看到事件詳細資訊出現在與此 Lambda 函數建立關聯的 CloudWatch Logs 日誌串流中。

## 6. 選擇儲存。

## 步驟 2：註冊事件規則

接著，您會建立一個 CloudWatch Events 事件規則，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 AWS 管理主控台 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 Events 呼叫 Lambda 函數的許可。如果您使用 建立事件規則 AWS CLI，則需要明確授予此許可。如需詳細資訊，請參閱 Amazon EventBridge User Guide 中的 [Events in Amazon EventBridge](#) 與 [Amazon EventBridge event patterns](#)。

### 將事件路由至 Lambda 函數

1. 透過 <https://console.aws.amazon.com/cloudwatch/> 開啟 CloudWatch 主控台。
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": [
 {
 "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": "public.ecr.aws/docker/library/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"
]
}
```

## 5. 選擇建立。

## 步驟 4：測試您的規則

最後，您會建立一個 CloudWatch Events 事件規則，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. 透過 <https://console.aws.amazon.com/cloudwatch/> 開啟 CloudWatch 主控台。
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。

# 針對 Amazon ECS 任務停止事件傳送 Amazon Simple Notification Service 提醒

設定一項 Amazon EventBridge 事件規則，使其僅擷取因其中一個必要容器終止而導致任務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。在此教學課程的最後，您會在此叢集上執行任務，以測試您的 Amazon SNS 主題和 EventBridge 規則設定是否正確。

## 先決條件：設定 Amazon SNS 的許可

若要允許 EventBridge 發佈到 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       | TaskStoppedAlert |  |
| 通訊協定     | Email            |  |
| Endpoint | 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。

如需有關如何建立和訂閱 Amazon SNS 主題的資訊，請參閱《Amazon EventBridge 使用者指南》中的 [在 Amazon EventBridge 中建立規則](#)，並使用下表來決定要選取的選項。

| 選項   | Value                                                                                                                                                                             |  |
|------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--|
| 規則類型 | 具有事件模式的規則                                                                                                                                                                         |  |
| 事件來源 | AWS 事件或 EventBridge 合作夥伴事件                                                                                                                                                        |  |
| 事件模式 | 自訂模式 (JSON 編輯器)                                                                                                                                                                   |  |
| 事件模式 | <pre>{   "source": [     "aws.ecs"   ],   "detail-type": [     "ECS Task State Change"   ],   "detail": {     "lastStatus": [       "STOPPED"     ],     "stoppedReason": [</pre> |  |

| 選項                 | Value                                                                                                                                                                                                             |  |
|--------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--|
|                    | <pre>         "EssentialContainer": true,         "ContainerDefinition": {           "Image": "amazonlinux:latest",           "Command": [             "sleep 5"           ]         }       }     ]   } } </pre> |  |
| Target type (目標類型) | AWS 服務                                                                                                                                                                                                            |  |
| Target             | 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 definition (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": "public.ecr.aws/amazonlinux/amazonlinux:latest",
 "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的 Amazon ECS 日誌訊息

從 AWS for Fluent Bit 2.22.0 版開始，包含多行篩選條件。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。您可以按照 GitHub 上的範例來串連由容器執行時間分割的日誌訊息：[FireLens 範例：串聯部分/分割容器日誌](#)。

## 所需的 IAM 許可

請務必先確定您擁有必要的 IAM 許可，讓容器代理程式可從 Amazon ECR 中提取容器映像，也讓容器可將日誌路由到 CloudWatch Logs。

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。

您需要具備下列許可：

- logs:CreateLogStream
- logs:CreateLogGroup
- logs:PutLogEvents

## 確定何時使用多行日誌設定

以下是您在 CloudWatch Logs 主控台中看到的範例日誌片段，其中包含預設日誌設定。您可以查看以 log 開頭的行，以確定是否需要多行篩選條件。當內容相同時，您可以使用多行日誌設定。在此範例中，內容為 "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.module.(MyProject.oneMoreMethod.java:18)",
{
 "container_id": "82ba37cada1d44d389b03e78caf74faa-EXAMPLE",
 "container_name": ": "example-app",
 "source": "stdout",
 "log": ": " at
com.myproject.module.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.module.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。這些步驟向您展示如何串聯多行並將日誌傳送到 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

您可以將剖析器檔案和組態檔案放置在 Docker 影像中除了 `/fluent-bit/etc/fluent-bit.conf` 以外的任何位置，因為該檔案路徑會由 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。

## 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，儲存庫名稱從 `fluent-bit-multiline-repo` 變更為 `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 ECR 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 管理主控台。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。

確認 Amazon CloudWatch 中的多行日誌訊息是否顯示為串連

1. 透過 <https://console.aws.amazon.com/cloudwatch/> 開啟 CloudWatch 主控台。
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/application` 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

您可以將組態檔案放置在 Docker 影像中除了 `/fluent-bit/etc/fluent-bit.conf` 以外的任何位置，因為該檔案路徑會由 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，儲存庫名稱從 `fluent-bit-multiline-repo` 變更為 `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：`aws 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 管理主控台。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。

確認 Amazon CloudWatch 中的多行日誌訊息是否顯示為串連

1. 透過 <https://console.aws.amazon.com/cloudwatch/> 開啟 CloudWatch 主控台。
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/application` 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\nngoroutine 4 [running]:\npanic(0x45cb40,\n0x47ad70)\n /usr/local/go/src/runtime/panic.go:542 +0x46c fp=0xc42003f7b8\nsp=0xc42003f710 pc=0x422f7c\nmain.main.func1(0xc420024120)\n foo.go:6\n+0x39 fp=0xc42003f7d8 sp=0xc42003f7b8 pc=0x451339\nruntime.goexit()\n /usr/\nlocal/go/src/runtime/asm_amd64.s:2337 +0x1 fp=0xc42003f7e0 sp=0xc42003f7d8\npc=0x44b4d1\ncreated by main.main\n foo.go:5 +0x58\n\nngoroutine 1 [chan receive]:\n\nruntime.gopark(0x4739b8, 0xc420024178, 0x46fcd7, 0xc, 0xc420028e17, 0x3)\n /usr/\nlocal/go/src/runtime/proc.go:280 +0x12c fp=0xc420053e30 sp=0xc420053e00 pc=0x42503c\n\nruntime.goparkunlock(0xc420024178, 0x46fcd7, 0xc, 0x100f010040c217, 0x3)\n\n /usr/local/go/src/runtime/proc.go:286 +0x5e fp=0xc420053e70 sp=0xc420053e30\npc=0x42512e\nruntime.chanrecv(0xc420024120, 0x0, 0xc420053f01, 0x4512d8)\n\n /usr/local/go/src/runtime/chan.go:506 +0x304 fp=0xc420053f20 sp=0xc420053e70\npc=0x4046b4\nruntime.chanrecv1(0xc420024120, 0x0)\n /usr/local/go/src/runtime/\nchan.go:388 +0x2b fp=0xc420053f50 sp=0xc420053f20 pc=0x40439b\nmain.main()\n\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\nngoroutine 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\nngoroutine 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 Windows 容器上部署 Fluent Bit

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。它可用來將日誌路由到各種 AWS 目的地，例如 Amazon CloudWatch Logs、Firehose Amazon S3 和 Amazon OpenSearch Service。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。如需詳細資訊，請參閱 GitHub 網站上的 [aws-for-fluent-bit](#)。

本教學課程會逐步引導您如何在 Amazon ECS 中執行的 Windows 執行個體上部署 Fluent Bit 容器，從而將 Windows 任務產生的日誌串流至 Amazon CloudWatch 進行集中式記錄。

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，如果您的 task\_1 帶有 container\_1 和 container\_2，而 task\_2 帶有 container\_3，則下列項目為 CloudWatch 日誌串流：

- /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

#### Note

您可以使用雙堆疊服務端點，透過 IPv4 和 IPv6 AWS CLI 從、SDKs 和 Amazon ECS API 與 Amazon ECS 互動。如需詳細資訊，請參閱[使用 Amazon ECS 雙堆疊端點](#)。

## 步驟

- [先決條件](#)
- [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：註冊會將日誌路由到 CloudWatch 的 Windows Fluent Bit 任務定義](#)
- [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 CloudWatch 上的日誌](#)
- [步驟 9：清除](#)

## 先決條件

本教學課程假設已完成下列先決條件：

- AWS CLI 已安裝並設定最新版本的。如需詳細資訊，請參閱 [Installing or updating to the latest version of the AWS CLI](#)。
- 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。在本教學課程中，叢集名稱是 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 端點連上子網路的私有子網路。如需有關 Amazon CloudWatch 端點的詳細資訊，請參閱《AWS 一般參考》中的 [Amazon CloudWatch 端點和配額](#)。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「ecsInstanceRole」的 Amazon ECS 容器執行個體角色。如需詳細資訊，請參閱 [Amazon ECS 容器執行個體 IAM 角色](#)。
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 傳送日誌到 CloudWatch，您需要將下列許可附加到任務 IAM 角色。

JSON

```
{
 "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 Advanced details (進階詳細資料) 下的 IAM instance profile (IAM 執行個體設定檔) 欄位中，選擇 `ecsInstanceRole`。
  - 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 CloudWatch](#)，即基於《Fluent Bit 官方手冊》中 [Amazon CloudWatch](#) 的 Fluent Bit 外掛程式。

或者，您可以使用 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 Fluent Bit 組態如下所示。

取代下列變數：

- 將 *region* 取代為您要將 Amazon CloudWatch Logs 傳送到的區域。

```
[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 $ecs_task_arn ^([a-z-:0-9]+)/([a-zA-Z0-9-_]+)/([a-z0-9-9])$
 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`。
- 新標籤將路由至輸出 `cloudwatch_logs` 外掛程式，該外掛程式會使用 CloudWatch 輸出外掛程式的 `log_group_template` 和 `log_stream_prefix` 選項來建立前文所述的日誌群組和串流。如需其他資訊，請參閱《Fluent Bit 官方手冊》中的[組態參數](#)。

## 步驟 4：註冊會將日誌路由到 CloudWatch 的 Windows Fluent Bit 任務定義

註冊會將日誌路由到 CloudWatch 的 Windows Fluent Bit 任務定義。

**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` 替換為任務 IAM 角色的 Amazon Resource Name (ARN)
- 將 `region` 替換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` 替換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
```

- 執行下列命令以列出您的任務。

取代下列變數：

- 將 `region` 替換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，您也可以在此任務中使用 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* 替換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

- 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* 替換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
```

- 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 CloudWatch 上的日誌

為了驗證您的 Fluent Bit 設定，請在 CloudWatch 主控台中檢查下列日誌群組：

- `/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. 透過 <https://console.aws.amazon.com/cloudwatch/> 開啟 CloudWatch 主控台。
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 中的日誌群組清單中 AWS 區域，您應該會看到以下內容：

- `/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 “停止任務”](#)。
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
```

## 在 Amazon ECS 上為 EC2 Linux 容器使用 gMSA

Amazon ECS 透過名為群組受管服務帳戶 (gMSA) 的特殊類型服務帳戶，支援為 EC2 上的 Linux 容器提供 Active Directory 身分驗證。

.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。如需有關在 Fargate 上為 Linux 容器使用 gMSA 的詳細資訊，請參閱[在 Fargate 上為 Linux 容器使用 gMSA](#)。
- 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。RSATActive Director 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)。

| 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          |

## 先決條件

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：
  - Directory Service Active Directory. Directory Service 是在 Amazon EC2 上託管的 AWS 受管 Active Directory。如需詳細資訊，請參閱《AWS Directory Service 管理指南》中的 [AWS Managed Microsoft AD 入門](#)。
  - 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[在 Amazon ECS 上為 EC2 Linux 容器使用 gMSA](#)。
- 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 GitHub 上的 [aws/credentials-fetcher](#)。

- 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 gMSA 做為初始登入資料，AWS Secrets Manager 則任務執行角色需要的 IAM 許可。
  - 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](#)。

- 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。
- 在 Active Directory 中建立使用者之後 AWS Secrets Manager，在 中建立秘密。如需詳細資訊，請參閱[建立 AWS Secrets Manager 秘密](#)。
- 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"}
```

- 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-fetcher && 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 -euo 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-fetcher && 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。如需有關 ECS 代理程式組態的詳細資訊，請參閱 GitHub 上的 [Amazon ECS 容器代理程式](#)。

- 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。

## JSON

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Action": [
 "secretsmanager:GetSecretValue"
],
 "Resource": "arn:aws:secretsmanager:us-east-1:123456789012:secret:my-secret-AbCdEf"
 }
]
}
```

**Note**

如果您使用自己的 KMS 金鑰來加密秘密，則必須將必要的許可新增至此角色，並將此角色新增至 AWS KMS 金鑰政策。

## 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。

## JSON

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Sid": "VisualEditor",
 "Effect": "Allow",
 "Action": [
 "s3:Get*",
 "s3:List*"
],
 "Resource": [
 "arn:aws:s3:::amzn-s3-demo-bucket",
 "arn:aws:s3:::amzn-s3-demo-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。

## JSON

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Action": [
 "ssm:GetParameters"
],
 "Resource": "arn:aws:ssm:us-east-1:123456789012:parameter/my-parameter"
 }
]
}
```

## 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。

## 在 Fargate 上為 Linux 容器使用 gMSA

Amazon ECS 透過名為群組受管服務帳戶 (gMSA) 的特殊類型服務帳戶，支援為 Fargate 上的 Linux 容器提供 Active Directory 身分驗證。

.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 上為 Linux 容器使用 gMSA 之前，請考量以下事項：

- 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。RSATActive Director 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，然後將其儲存於 Amazon S3 儲存貯體中。

- 一個任務只能支援一個 Active Directory。

## 先決條件

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：
  - Directory Service Active Directory. Directory Service 是在 Amazon EC2 上託管的 AWS 受管 Active Directory。如需詳細資訊，請參閱《AWS Directory Service 管理指南》中的 [AWS Managed Microsoft AD 入門](#)。
  - 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 中有一個現存的 gMSA 帳戶，也有一個具備存取 gMSA 服務帳戶許可的使用者。如需詳細資訊，請參閱[建立無網域 gMSA 的 Active Directory 使用者](#)。
- 您已有 Amazon S3 儲存貯體。如需詳細資訊，請參閱 Amazon S3 User Guide 中的 [Creating a bucket](#)。

## 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

使用無網域 gMSA 時，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該憑證存取網域。使用不同網域的任務可在同一容器上執行。您可以在 CredSpec 檔案中提供 AWS Secrets Manager 中的秘密名稱。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 Active Directory 網域中設定使用者。Active Directory 中的使用者必須具有存取您在任務中使用之 gMSA 服務帳戶的許可。



- b. 您擁有可以解析 Active Directory 網域的 VPC 和子網路。設定 VPC 時，需在 DHCP 選項中設定指向 Active Directory 服務名稱的網域名稱。如需有關為 VPC 設定 DHCP 選項的資訊，請參閱 Amazon Virtual Private Cloud User Guide 中的 [Work with DHCP option sets](#)。
- c. 在 中建立秘密 AWS Secrets Manager。
- d. 建立憑證規格檔案。

## 設定許可和祕密

針對每個應用程式與每項任務定義，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. 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. 成為 Active Directory 使用者之後，請在其中建立秘密 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。

## JSON

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Action": [
 "secretsmanager:GetSecretValue"
],
 "Resource": [
 "arn:aws:secretsmanager:us-east-1:111122223333:secret:MySecret"
]
 }
]
}
```

```
]
 }
}
```

### Note

如果您使用自己的 KMS 金鑰來加密秘密，則必須將必要的許可新增至此角色，並將此角色新增至 AWS KMS 金鑰政策。

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，請新增下列許可作為 Amazon ECS 任務執行 IAM 角色的內嵌政策。

JSON

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Sid": "VisualEditor",
 "Effect": "Allow",
 "Action": [
```

```

 "s3:GetObjectVersion",
 "s3:ListBucket"
],
 "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"
}
}
}
}
```

## 建立 CredSpec 並將其上傳至 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。

在建立憑證規格檔案之後，請將其上傳至 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，將 CredSpec 上傳至 Amazon S3。將 *amzn-s3-demo-bucket* 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，使用下列命令：

```
$ Write-S3Object -BucketName "amzn-s3-demo-bucket" -Key "ecs-domainless-gmsa-credspec"
-File "gmsa-cred-spec.json"
```

下列 AWS CLI 命令使用 sh 和相容 Shell 所使用的反斜線接續字元。

```
$ aws s3 cp gmsa-cred-spec.json \
s3://amzn-s3-demo-bucket/ecs-domainless-gmsa-credspec
```

## gMSA 使用 搭配無網域使用 Amazon ECS Windows 容器 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

- [先決條件](#)
- [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](#)

## 先決條件

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 IAM 使用者擁有 [AmazonECS\\_FullAccess](#) IAM 政策範例中指定的所需許可。
- AWS CLI 已安裝並設定最新版本的。如需安裝或升級的詳細資訊 AWS CLI，請參閱[安裝 AWS Command Line Interface](#)。

### Note

您可以使用雙堆疊服務端點，透過 IPv4 和 IPv6 AWS CLI 從、SDKs 和 Amazon ECS API 與 Amazon ECS 互動。如需詳細資訊，請參閱[使用 Amazon ECS 雙堆疊端點](#)。

- 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：
  - Directory Service Active Directory. Directory Service 是在 Amazon EC2 上託管的 AWS 受管 Active Directory。如需詳細資訊，請參閱《AWS Directory Service 管理指南》中的 [AWS Managed Microsoft AD 入門](#)。
  - 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)。

| 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 是一種工具，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 AWS CloudShell 《使用者指南》中的 [什麼是 AWS CloudShell？](#)。

## 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。

**Note**

此步驟會建立兩個不同的帳戶：為容器提供身分的群組受管服務帳戶 (gMSA)，以及用於網域身分驗證的一般使用者帳戶。這些帳戶的用途各不相同，因此應使用不同的名稱。

## 1. 產生金鑰分佈服務根金鑰

**Note**

如果您使用的是 Directory Service，則可以略過此步驟。

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 `ActiveDirectory PowerShell` 模組，在網域控制器上以網域管理員權限執行下列 PowerShell cmdlet。如需有關此模組的詳細資訊，請參閱 Microsoft Learn 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 `ccg.exe` 擷取 gMSA 密碼，請從有權存取網域的 Windows 伺服器或用戶端執行下列 PowerShell 命令。將 `ExampleAccount` 取代為要用於 gMSA 帳戶的名稱，並將 `example-domain` 取代為 Active Directory 網域名稱 (例如：網域為 `contoso.com`，則使用 `contoso`)。

- 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 "example-domain" -ServicePrincipalNames "host/ExampleAccount", "host/example-domain" -PrincipalsAllowedToRetrieveManagedPassword "ExampleAccountHosts"
```
- d. 建立一個具有不會過期的永久密碼的使用者。這些登入資料會存放在 中，AWS Secrets Manager 並由每個任務用來加入網域。這是與前文建立的 gMSA 帳戶不同的使用者帳戶。將 ExampleServiceUser 取代為要用於該服務使用者帳戶的名稱。

```
PS C:\> New-ADUser -Name "ExampleServiceUser" -AccountPassword (ConvertTo-SecureString -AsPlainText "Test123" -Force) -Enabled 1 -PasswordNeverExpires 1
```

- e. 

```
PS C:\> Add-ADGroupMember -Identity "ExampleAccountHosts" -Members "ExampleServiceUser"
```
- f. 安裝用於在 Active Directory 中建立 CredSpec 物件的 PowerShell 模組並輸出 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。使用服務使用者帳戶名稱 (而非 gMSA 帳戶名稱) 作為使用者名稱。預留密碼的 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。此命令與 PowerShell 不相容。您必須修改命令才能將其與 PowerShell 搭配使用。

```
$ aws secretsmanager create-secret \
--name gmsa-plugin-input \
--description "Amazon ECS - gMSA Portable Identity." \
--secret-string "{\"username\":\"ExampleServiceUser\",\"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[未加入網域的容器主機使用案例的其他憑證規格組態](#)。

- 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**\"。

- 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": "example-domain",
 "DnsName": "example-domain",
 "NetBiosName": "example-domain"
},
"ActiveDirectoryConfig": {
 "GroupManagedServiceAccounts": [
 {
 "Name": "ExampleAccount",
 "Scope": "example-domain"
 },
 {
 "Name": "ExampleAccount",
 "Scope": "example-domain"
 }
],
 "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，將 CredSpec 上傳至 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。此命令與 PowerShell 不相容。您必須修改命令才能將其與 PowerShell 搭配使用。

```
$ 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。根據預設 AWS CLI，此叢集用於、SDKs 和 CloudFormation。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 管理主控台、AWS CLI SDKs 或 建立叢集 CloudFormation。叢集中的設定和組態不會影響 gMSA。

此步驟使用 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 Elastic Container Registry 中提取容器映像，並從應用程式中提取 stdout 和 stderr 並記錄到 Amazon CloudWatch Logs 中。

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`。

JSON

```
{
 "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:us-east-1:111122223333:secret:gmsa-plugin-AbCdEf"
 }
]
}
```

## 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 Exec 許可](#)。

此步驟使用 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：

JSON

```
{
 "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`，但預留以下步驟的名稱。

以下命令使用由 `sh` 和相容 shell 所使用的反斜線接續字元。此命令與 PowerShell 不相容。您必須修改命令才能將其與 PowerShell 搭配使用。

```
$ 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：

## JSON

```
{
 "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。此命令與 PowerShell 不相容。您必須修改命令才能將其與 PowerShell 搭配使用。

```
$ 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。此命令與 PowerShell 不相容。您必須修改命令才能將其與 PowerShell 搭配使用。

```
$ 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. 若要啟動在 中為 Amazon ECS 設定的 Amazon EC2 Windows 執行個體 AWS 管理主控台，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 管理主控台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。這些屬性不會顯示在 中 AWS 管理主控台，因此本教學課程會使用 AWS CLI。

此步驟使用 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。此命令與 PowerShell 不相容。您必須修改命令才能將其與 PowerShell 搭配使用。

```
$ 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。此命令與 PowerShell 不相容。您必須修改命令才能將其與 PowerShell 搭配使用。

```
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。此命令與 PowerShell 不相容。您必須修改命令才能將其與 PowerShell 搭配使用。

```
$ aws ecs execute-command \
 --task MyTaskID \
 --container windows_sample_app \
 --interactive \
 --command powershell.exe
```

輸出將是 PowerShell 提示字元。

2. 在容器內的 PowerShell 終端中執行以下命令。

```
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。此命令與 PowerShell 不相容。您必須修改命令才能將其與 PowerShell 搭配使用。

```
$ 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 AWS 管理主控台，查看任務的停止原因欄位，了解任務停止的原因。在 AWS CLI 中，描述任務並查看 stoppedReason。如需 AWS 管理主控台 和 中的步驟 AWS CLI，請參閱 [檢視 Amazon ECS 已停止任務錯誤](#)。

### 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。

## 了解如何在 Amazon ECS 上為 EC2 Windows 容器使用 gMSA

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](#)
- [先決條件](#)
- [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[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)。

| 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 |

| 儲存位置                             | Linux    | Windows           |
|----------------------------------|----------|-------------------|
| 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          |

## 先決條件

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：
  - Directory Service Active Directory. Directory Service 是在 Amazon EC2 上託管的 AWS 受管 Active Directory。如需詳細資訊，請參閱《AWS Directory Service 管理指南》中的 [AWS Managed Microsoft AD 入門](#)。
  - 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gMSA做為初始登入資料，Amazon EC2 執行個體角色 AWS Secrets Manager 需要的 IAM 許可。
  - 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 [gMSA 使用 搭配 無網域使用 Amazon ECS Windows 容器 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[未加入網域的容器主機使用案例的其他憑證規格組態](#)。

### 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 替換為 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。

## JSON

```
{
 "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。

## JSON

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Action": [
 "ssm:GetParameters"
],
 "Resource": "arn:aws:ssm:*:111122223333:parameter/example*"
 }
]
}
```

```
}
```

## 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"
],
 ...
 }
],
 ...
}
```

## 透過 EC2 Image Builder 建置自訂的 Amazon ECS 最佳化 AMI

AWS 建議您使用 Amazon ECS 最佳化 AMIs 因為它們已預先設定執行容器工作負載的需求和建議。在某些情況下，您可能需要自訂 AMI 來新增額外軟體。您可以透過 EC2 Image Builder 來建立、管理與部署伺服器映像。您將保留帳戶中所建立自訂映像的所有權。您可透過 EC2 Image Builder 管道，自動化執行映像的更新與系統修補作業；也可以使用獨立命令，建立包含所定義組態資源的映像。

您需要為映像建立一份配方。該配方包含父映像以及任何額外元件。您也需要建立一個用於分發自訂 AMI 的管道。

您需要為映像建立一份配方。Image Builder 映像配方是一個文件，用於定義基礎映像與要套用至基礎映像的元件，以便產生輸出 AMI 映像所需的組態。您也需要建立一個用於分發自訂 AMI 的管道。如需詳細資訊，請參閱 EC2 Image Builder User Guide 中的 [How EC2 Image Builder works](#)。

建議使用下列任一 Amazon ECS 最佳化 AMI 作為 EC2 Image Builder 中的「父映像」：

- Linux
  - Amazon ECS 最佳化 AL2023 x86
  - Amazon ECS 最佳化 Amazon Linux 2023 (arm64) AMI
  - Amazon ECS 最佳化 Amazon Linux 2 核心 5 AMI
  - Amazon ECS 最佳化 Amazon Linux 2 x86 AMI
- Windows
  - Amazon ECS 最佳化 Windows 2022 Full x86
  - Amazon ECS 最佳化 Windows 2022 Core x86
  - Amazon ECS 最佳化 Windows 2019 Full x86
  - Amazon ECS 最佳化 Windows 2019 Core x86
  - Amazon ECS 最佳化 Windows 2016 Full x86

我們也建議選取「使用最新可用的作業系統版本」。該管道將對父映像採用語意版本控制，此舉有助於在自動排程任務中偵測相依性更新。如需詳細資訊，請參閱 EC2 Image Builder User Guide 中的 [Semantic versioning](#)。

AWS 會使用安全性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，需定期檢查是否有父映像的更新。若有更新，必須使用更新後的 AMI 建立新的配方版本。此舉可確保自訂映像包含最新版本的父映像。如需有關如何建立工作流程來使用新建的 AMI 自動更新 Amazon ECS 叢集中 EC2 執行個體的資訊，請參閱 [How to create an AMI hardening pipeline and automate updates to your ECS instance fleet](#)。

您也可以指定透過受管 EC2 Image Builder 管道發布之父映像的 Amazon Resource Name (ARN)。Amazon 會透過受管管道，定期發布 Amazon ECS 最佳化 AMI 映像。這些映像可公開存取。您必須具備正確的許可，才能存取映像。在 Image Builder 配方中使用映像 ARN 而非 AMI 時，管道會在每次執行時自動使用最新版本的父映像。此方法可免除每次更新時手動建立新配方版本的需求。

## 清理 Amazon ECS 最佳化 AMI

使用 Amazon ECS 最佳化 AMI 作為父映像時，必須清理該映像，以防止暫時性問題。Amazon ECS 最佳化 AMI 已預先設定，可讓 Amazon ECS 代理程式自動啟動，並作為容器執行個體向 Amazon ECS 註冊。若未適當清理就將其用作基礎映像，可能會導致自訂 AMI 出現問題。

若要清理該映像以供日後使用，請建立執行下列命令的元件，停止 ECS-init 套件和 Docker 程序：

```
sudo systemctl stop ecs
```

```
sudo systemctl stop docker
```

從目前的執行個體中移除所有日誌檔案，避免在儲存映像時預留這些檔案。使用 [Security best practices for EC2 Image Builder](#) 中的範例指令碼，從執行個體清理各種檔案。

若要清理 Amazon ECS 特定資料，請執行下列命令：

```
sudo rm -rf /var/log/ecs/*
sudo rm /var/lib/ecs/data/agent.db
```

如需建立自訂 Amazon ECS 最佳化 AMIs 的詳細資訊，請參閱 AWS 知識中心的[如何從 Amazon ECS 最佳化 AMI 建立自訂 AMI？](#)。

## 在基礎結構即程式碼 (IaC) 中使用映像 ARN

您可以使用 EC2 Image Builder 主控台或基礎設施做為程式碼（例如 CloudFormation）或 AWS SDK 來設定配方。當您在配方中指定父系映像時，您可以指定 EC2 AMI ID、映像建置器映像 ARN、AWS Marketplace 產品 ID 或容器映像。AWS 會公開發佈 Amazon ECS 最佳化 AMI 的 AMI IDs 和映像建置器映像 AMIs ARNs。映像 ARN 的格式如下：

```
arn:${Partition}:imagebuilder:${Region}:${Account}:image/${ImageName}/${ImageVersion}
```

ImageVersion 採用下列格式。將 *major*、*minor* 與 *patch* 分別取代為最新值。

```
<major>.<minor>.<patch>
```

您可以將 *major*、*minor* 與 *patch* 取代為特定值，也可以使用映像的無版本 ARN，如此一來，管道就能總是與父映像的最新版本保持同步。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 服務自動解析至映像的最新版本。使用無版本 ARN 可確保參考一律指向可用的最新映像，簡化在部署中維護最新映像的程序。透過主控台建立配方時，EC2 Image Builder 會自動識別父映像的 ARN。透過基礎結構即程式碼 (IaC) 建立配方時，必須自行識別 ARN，並選取所需的映像版本，或使用無版本 ARN 指定可用的最新映像。建議您建立自動化指令碼來篩選並僅顯示符合條件的映像。下列 Python 指令碼示範如何擷取 Amazon ECS 最佳化 AMI 的清單。

指令碼接受兩個選用引數：owner 與 platform，預設值分別為 "Amazon" 與 "Windows"。owner 引數的有效值包括：Self、Amazon、Shared 與 ThirdParty。platform 引數的有效值為 Windows 與 Linux。例如，若執行指令碼時，將 owner 引數設定為 Amazon、platform 引數設定為 Linux，該指令碼會產生一份由 Amazon 發布的映像清單，內含 Amazon ECS 最佳化映像。

```
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 使用映像 ARN CloudFormation

Image Builder 映像配方是一份藍圖，其中指定了實現輸出映像的目標組態所需的父映像與元件。您需要使用 `AWS::ImageBuilder::ImageRecipe` 資源。將 `ParentImage` 值設定為映像 ARN。請使用目標映像的無版本 ARN，確保管道一律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 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"
```



如需有關 [AWS::ImageBuilder::ImageRecipe](#) 資源的詳細資訊，請參閱 AWS CloudFormation User Guide。

您可以透過設定 `AWS::ImageBuilder::ImagePipeline` 資源的 `Schedule` 屬性，在管道中自動建立新映像。排程包含開始條件與 Cron 表達式。如需詳細資訊，請參閱《AWS CloudFormation 使用者指南》中的 [AWS::ImageBuilder::ImagePipeline](#)。

下列 `AWS::ImageBuilder::ImagePipeline` 範例中，管道會於每日協調世界時 (UTC) 上午 10:00 執行一次建置作業。設定下列 `Schedule` 值：

- 將 `PipelineExecutionStartCondition` 設定為 `EXPRESSION_MATCH_AND_DEPENDENCY_UPDATES_AVAILABLE`。僅當父映像、元件等依賴資源 (其語意版本中使用萬用字元 'x') 發生更新時，才會啟動建置作業。此設定可確保建置作業包含這些資源的最新更新。
- 將 `ScheduleExpression` 設定為 Cron 表達式 (`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 * * ? *)'
```

## 在 Terraform 中使用映像 ARN

在 Terraform 中指定管道父映像及排程的方法，與 AWS CloudFormation 中的方法一致。您需要使用 `aws_imagebuilder_image_recipe` 資源。將 `parent_image` 值設定為映像 ARN。請使用目標映像的無版本 ARN，確保管道一律使用最新版本的映像。如需詳細資訊，請參閱 Terraform 文件中的 [aws\\_imagebuilder\\_image\\_recipe](#)。

在 `aws_imagebuilder_image_pipeline` resource 的排程組態區塊中，將 `schedule_expression` 引數值設定為您選擇的 Cron 表達式 (用於指定管

道的執行頻率)，並將 `pipeline_execution_start_condition` 設定為 `EXPRESSION_MATCH_AND_DEPENDENCY_UPDATES_AVAILABLE`。如需詳細資訊，請參閱 Terraform 文件中的 [aws\\_imagebuilder\\_image\\_pipeline](#)。

## 在 Amazon ECS 上使用 AWS 深度學習容器

AWS 深度學習容器提供一組 Docker 影像，用於在 Amazon ECS 的 TensorFlow 和 Apache MXNet（培養）中訓練和提供模型。Deep Learning Containers 實現最佳化環境與 TensorFlow、Nvidia CUDA（適用於 GPU 執行個體）和 Intel MKL（適用於 CPU 執行個體）程式庫。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。

若要開始在 Amazon ECS 上使用 Deep Learning Containers（不搭配 Elastic Inference），請參閱 AWS 深度學習 AMLs Developer Guide 中的 [Amazon ECS setup](#)。

## 教學課程：使用藍/綠部署建立服務

Amazon ECS 已將藍/綠部署整合至 Amazon ECS 主控台內的建立服務精靈。如需詳細資訊，請參閱 [建立 Amazon ECS 滾動更新部署](#)。

以下教學課程說明如何在 AWS CLI 透過使用藍/綠部署類型的 Fargate 任務建立 Amazon ECS 服務。

### Note

已為 CloudFormation 新增對於執行藍/綠部署的支援。如需詳細資訊，請參閱 AWS CloudFormation 使用者指南中的 [使用 CloudFormation 透過 CodeDeploy 執行 Amazon ECS 藍/綠部署](#)。

## 先決條件

本教學課程假設已完成下列先決條件：

- AWS CLI 已安裝並設定最新版本的。如需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 IAM 使用者擁有 [AmazonECS\\_FullAccess](#) IAM 政策範例中指定的所需許可。
- 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”](#)。

- 已建立 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": "sample-fargate",
 "networkMode": "awsvpc",
```

```

"containerDefinitions": [
 {
 "name": "sample-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"
 }
]

```

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
```

## 步驟 5：建立 AWS CodeDeploy 資源

請依照下列步驟建立 CodeDeploy 應用程式、CodeDeploy 部署群組的 Application Load Balancer 目標群組，以及 CodeDeploy 部署群組。

### 建立 CodeDeploy 資源

1. 使用 [create-application](#) 命令建立 CodeDeploy 應用程式。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 [create-target-group](#) 命令建立第二個 Application Load Balancer 目標群組，該群組會在您建立 CodeDeploy 部署群組時用到。

```
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 [create-deployment-group](#) 命令建立 CodeDeploy 部署群組。

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 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"
}
```

接著建立 CodeDeploy 部署群組。

```
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：建立與監控 CodeDeploy 部署

使用以下步驟來建立和上傳應用程式規格檔案 (AppSpec 檔案) 和 CodeDeploy 部署。

### 建立和監控 CodeDeploy 部署

1. 使用以下步驟來建立和上傳 AppSpec 檔案。
  - a. 建立名為 `appspec.yaml` 並具有 CodeDeploy 部署群組內容的檔案。此範例會使用您先前在教學課程中建立的資源。

```
version: 0.0
Resources:
 - TargetService:
 Type: AWS::ECS::Service
 Properties:
 TaskDefinition: "arn:aws:ecs:region:aws_account_id:task-
definition/first-run-task-definition:7"
 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](#) 命令來將 AppSpec 檔案上傳至 Amazon S3 儲存貯體。

```
aws s3 cp ./appspec.yaml s3://tutorial-bluegreen-bucket/appspec.yaml
```

2. 使用下列步驟來建立 CodeDeploy 部署。

- a. 建立名為 `create-deployment.json` 並具有 CodeDeploy 部署內容的檔案。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"
}
```

- c. 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
```

繼續擷取狀態變為 Succeeded 前的部署詳細資訊，如下列輸出所示。

```
{
 "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": 1543431490.226,
 "lifecycleEvents": [
 {
 "lifecycleEventName": "BeforeInstall",
 "startTime": 1543431361.022,
 "endTime": 1543431361.433,
 "status": "Succeeded"
 },
 {
 "lifecycleEventName": "Install",
 "startTime": 1543431361.678,
 "endTime": 1543431485.275,
 "status": "Succeeded"
 },
 {
 "lifecycleEventName": "AfterInstall",
 "startTime": 1543431485.52,
 "endTime": 1543431486.033,
 "status": "Succeeded"
 },
 {
 "lifecycleEventName": "BeforeAllowTraffic",
 "startTime": 1543431486.838,
 "endTime": 1543431487.483,
 "status": "Succeeded"
 },
 {
 "lifecycleEventName": "AfterAllowTraffic",
 "startTime": 1543431487.483,
 "endTime": 1543431487.483,
 "status": "Succeeded"
 }
]
 }
 }
}
```

```

 "lifecycleEventName": "AllowTraffic",
 "startTime": 1543431487.748,
 "endTime": 1543431488.488,
 "status": "Succeeded"
 },
 {
 "lifecycleEventName": "AfterAllowTraffic",
 "startTime": 1543431489.152,
 "endTime": 1543431489.885,
 "status": "Succeeded"
 }
],
"status": "Succeeded",
"taskSetsInfo": [
 {
 "identifer": "ecs-svc/9223370493425779968",
 "desiredCount": 1,
 "pendingCount": 0,
 "runningCount": 1,
 "status": "ACTIVE",
 "trafficWeight": 0.0,
 "targetGroup": {
 "name": "bluegreentarget1"
 }
 },
 {
 "identifer": "ecs-svc/9223370493423413672",
 "desiredCount": 1,
 "pendingCount": 0,
 "runningCount": 1,
 "status": "PRIMARY",
 "trafficWeight": 100.0,
 "targetGroup": {
 "name": "bluegreentarget2"
 }
 }
]
}
}
}
}

```

## 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 [delete-deployment-group](#) 命令來刪除 CodeDeploy 部署群組。

```
aws deploy delete-deployment-group \
 --application-name tutorial-bluegreen-app \
 --deployment-group-name tutorial-bluegreen-dg \
 --region us-east-1
```

2. 使用 [delete-application](#) 命令來刪除 CodeDeploy 應用程式。

```
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](#) 命令來將 AppSpec 檔案從 Amazon S3 儲存貯體中刪除。

```
aws s3 rm s3://tutorial-bluegreen-bucket/appspect.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
```

# Amazon ECS 服務配額

當您建立時 AWS 帳戶，我們會根據您的區域為您的 AWS 資源設定預設配額（也稱為限制）。如果您嘗試超過資源的配額，請求會失敗。

Service Quotas 主控台是一個集中位置，您可以在其中檢視和管理 AWS 服務的配額，並請求增加您使用的許多資源的配額。使用我們提供的配額及來管理資源。計劃在您需要的時候，先行請求提高配額。

如需有關 Amazon ECS 與 Fargate 配額的詳細資訊，請參閱 Amazon Web Services 一般參考 中的 [Amazon ECS endpoints and quotas](#)。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 受管執行個體遵循 Amazon EC2 配額。如需詳細資訊，請參閱 Amazon EC2 User Guide 中的 [Amazon EC2 service quotas](#)。

## 在 中管理您的 Amazon ECS AWS Fargate 和服務配額 AWS 管理主控台

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 管理主控台

使用 檢視 Amazon ECS 和 Fargate 服務配額 AWS 管理主控台

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)。

若要使用 進一步使用服務配額，AWS 管理主控台 請參閱 [Service Quotas 使用者指南](#)。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。

## AWS CLI

使用 檢視 Amazon ECS 和 Fargate 服務配額 AWS CLI

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 使用服務配額的詳細資訊 AWS CLI，請參閱 [Service Quotas 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。

## 因應 Amazon ECS 服務配額與 API 限流限制

Amazon ECS 與數個 整合 AWS 服務，包括 Elastic Load Balancing AWS Cloud Map 和 Amazon EC2。透過這種緊密整合，Amazon ECS 包含數個功能，例如服務負載平衡、Service Connect、任務



聯網與叢集自動擴展。Amazon ECS 及其所整合的其他 AWS 服務 服務均設定有服務配額與 API 速率限制，可確保一致的效能與使用率。這些服務配額也可防止意外佈建超過所需的資源，並防範可能導致帳單金額增加的惡意行為。

透過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 [Request throttling for the Amazon ECS API](#)。

在 Amazon ECS 上擴展工作負載時，建議考量下列服務配額。

- AWS Fargate 具有配額，可限制每個 中並行執行中的任務數量 AWS 區域。Amazon ECS 上的隨需任務與 Fargate Spot 任務都有配額。每個服務配額的計算範圍，亦涵蓋您在 Fargate 上執行的所有 Amazon EKS Pod。
- 對於在 Amazon EC2 執行個體上執行的任務，每個叢集可註冊的 Amazon EC2 執行個體數量上限為 5,000 個。如果搭配 Auto Scaling 群組容量提供者使用 Amazon ECS 叢集自動擴展，或者自行管理叢集的 Amazon EC2 執行個體，則此配額可能會成為部署瓶頸。如果需要更多容量，您可以建立更多叢集或請求提高服務配額。
- 如果搭配 Auto Scaling 群組容量提供者使用 Amazon ECS Tasks in the PROVISIONING state per cluster 叢集自動擴展，請在擴展服務時將配額納入考量。此配額指每個叢集中處於 PROVISIONING 狀態的任務數上限，容量提供者可針對這些任務增加容量。當同時啟動大量任務時，就很容易達到此配額上限。若同時部署數十個服務，每個服務包含數百個任務，就可能出現這種情況。此時，若叢集容量不足，容量提供者需啟動新的容器執行個體來置放這些任務。在容量提供者啟動額外 Amazon EC2 執行個體的同時，Amazon ECS 服務排程器通常會持續平行啟動任務。但由於叢集容量不足，此操作可能會受到限流。隨著新容器執行個體的啟動，Amazon ECS 服務排程器會採用退避與指數限流策略，重試任務置放操作。因此，您可能會遇到部署或擴展時間延長的情況。為避免此情況，您可透過以下任一方式規劃服務部署。一種方式是部署大量無需增加叢集容量的任務，一種方式是保留備用叢集容量以應對新任務啟動。

除了在擴展工作負載時考慮 Amazon ECS 服務配額之外，也請考慮與 Amazon ECS AWS 服務 整合之其他的服務配額。

## Elastic Load Balancing

您可以設定 Amazon ECS 服務使用 Elastic Load Balancing，將流量平均分發至所有任務。如需如何選擇負載平衡器的詳細資訊與建議的最佳實務，請參閱[使用負載平衡分佈 Amazon ECS 服務流量](#)。

## Elastic Load Balancing 服務配額

在擴展工作負載時，建議考量下列 Elastic Load Balancing 服務配額。大多數 Elastic Load Balancing 服務配額都可調整，您可透過 Service Quotas 主控台申請提高上限。

### Application Load Balancer

在使用 Application Load Balancer 時，根據實際使用案例，可能需要申請提高以下項目的配額：

- Targets per Application Load Balancer 配額，即 Application Load Balancer 後端的目標數量。
- Targets per Target Group per Region 配額，即目標群組後端的目標數量。

如需詳細資訊，請參閱 User Guide for Application Load Balancers 中的 [Quotas for your Application Load Balancer](#)。

### Network Load Balancer

Network Load Balancer 的目標註冊數量存在更嚴格的限制。使用 Network Load Balancer 時，通常建議啟用跨可用區域支援功能，但此功能此功能會對 Targets per Availability Zone Per Network Load Balancer 帶來額外的擴展限制，亦即每個 Network Load Balancer 於各可用區域中可容納的目標數量上限將受到規範。如需詳細資訊，請參閱 User Guide for Network Load Balancers 中的 [Quotas for your Network Load Balancers](#)。

## Elastic Load Balancing API 限流

在設定 Amazon ECS 服務使用負載平衡器時，必須先通過目標群組運作狀態檢查，才能將服務視為正常運作。為了執行這些運作狀態檢查，Amazon ECS 會代為調用 Elastic Load Balancing API 操作。如果帳戶中有大量使用負載平衡器設定的服務，可能會因為 RegisterTarget、DeregisterTarget 與 DescribeTargetHealth 這幾項 Elastic Load Balancing API 操作遭遇限流，導致服務部署速度減緩。發生限流時，Amazon ECS 服務事件訊息中會發生限流錯誤。

如果您遇到 AWS Cloud Map API 限流，您可以聯絡 支援 以取得有關如何提高 AWS Cloud Map API 限流限制的指引。如需有關監控及疑難排解此類限流錯誤的詳細資訊，請參閱[因應 Amazon ECS 限流問題](#)。

## 彈性網路介面

隨著任務使用 `awsvpc` 網路模式，Amazon ECS 會為每個任務佈建唯一的彈性網路介面 (ENI)。當 Amazon ECS 服務使用 Elastic Load Balancing 負載平衡器時，這些網路介面也會註冊為該服務中定義的對應目標群組的目標。

### 彈性網路介面服務配額

當您執行使用 `awsvpc` 網路模式的任務時，每個任務都會連接唯一的彈性網路介面。如果必須透過國際網路到達這些任務，請將公有 IP 位址指派給這些任務的彈性網路介面。在擴展 Amazon ECS 工作負載時，請考量這兩個重要的配額：

- Network interfaces per Region 配額，這是中 AWS 區域 您帳戶的最大網路介面數量。
- Elastic IP addresses per Region 配額，即 AWS 區域中彈性 IP 位址的最大數量。

這兩個服務配額皆可調整，您可以直接透過 Service Quotas 主控台提出調增申請。如需詳細資訊，請參閱 Amazon Virtual Private Cloud User Guide 中的 [Amazon VPC service quotas](#)。

對於託管在 Amazon EC2 執行個體上的 Amazon ECS 工作負載，當執行使用 `awsvpc` 網路模式的任務時，建議考量 Maximum network interfaces 服務配額，即每個 Amazon EC2 執行個體的網路執行個體數量上限。此配額會限制可在執行個體上置放的任務數量。此配額無法調整，就未於 Service Quotas 主控台中提供。如需詳細資訊，請參閱 Amazon EC2 User Guide 中的 [IP addresses per network interface per instance type](#)。

雖然您無法變更可連接至 Amazon EC2 執行個體的網路介面數量，但可透過啟用彈性網路介面主幹功能來增加可用網路介面的總數。例如，依預設，`c5.large` 執行個體最多可有 3 個網路介面。執行個體的主要網路介面視為一個配額。因此，您可以將額外的兩個網路介面連接至執行個體。因為每個使用 `awsvpc` 網路模式的任務都需要網路介面，所以通常只能對此執行個體類型執行兩個這類任務。這可能會導致叢集容量的使用率偏低。如果啟用彈性網路介面主幹功能，即可提高網路介面密度，從而在每個執行個體上部署更多任務。啟用主幹功能後，`c5.large` 執行個體最多可有 12 個網路介面。該執行個體本身已具備主要網路介面，而 Amazon ECS 會建立並連接「主幹」網路介面至該執行個體。因此，透過此組態，您可在該執行個體上執行 10 個任務，而非預設的 2 個任務。如需詳細資訊，請參閱 [增加 Amazon ECS Linux 容器執行個體網路介面數量](#)。

### 彈性網路介面 API 限流

在執行使用 `awsvpc` 網路模式的任務時，Amazon ECS 會調用下列 Amazon EC2 API。每個 API 皆設有不同的 API 限流。如需詳細資訊，請參閱 Amazon EC2 API Reference 中的 [Request throttling for the Amazon EC2 API](#)。

- CreateNetworkInterface
- AttachNetworkInterface
- DetachNetworkInterface
- DeleteNetworkInterface
- DescribeNetworkInterfaces
- DescribeVpcs
- DescribeSubnets
- DescribeSecurityGroups
- DescribeInstances

若在彈性網路介面佈建工作流程中遭遇 Amazon EC2 API 呼叫限流，Amazon ECS 服務排程器會自動採用指數退避機制進行重試。這些淘汰有時可能導致任務啟動延遲，進而造成部署速度減緩。當發生 API 限流時，服務事件訊息中會顯示 `Operations are being throttled. Will try again later.`。如果您持續符合 Amazon EC2 API 限流，您可以聯絡 [支援](#) 以取得有關如何提高 API 限流限制的指引。如需有關監控及疑難排解限流錯誤的詳細資訊，請參閱[因應限流問題](#)。

## AWS Cloud Map

Amazon ECS 服務探索和 Service Connect 使用 AWS Cloud Map APIs 來管理 Amazon ECS 服務的命名空間。如果服務有大量任務，建議考量下列事項。

### AWS Cloud Map 服務配額

當 Amazon ECS `Tasks per service` 服務設定為使用服務探索或 Service Connect 時，服務任務數量上限的配額會受到 `AWS Cloud Map Instances per service` 該服務執行個體數量上限的服務配額影響。具體來說，AWS Cloud Map 服務配額會將每個服務可執行的執行個體數減少到最多 1,000 個。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 限流

Amazon ECS 會代表您呼叫 `ListInstances`、`RegisterInstance`、`GetInstancesHealthStatus` 和 `DeregisterInstance` AWS Cloud Map APIs。這些 API 可用於服務探索，會在您啟動任務時執行運作狀態檢查。當同時使用服務探索與大量任務的多個服務同時部署時，這可能會導致超過 AWS Cloud Map API 限流限制。發生這種情況時，您可能會看到以下訊息：在 Amazon ECS 服務事件訊息 `Operations are being throttled. Will try again later` 中，以及較慢的部署和任務啟動速度。AWS Cloud Map 不會記錄這些 APIs 的限流限制。如果

您遇到這些調節，您可以聯絡 [支援](#) 以取得提高 API 調節限制的指引。如需有關監控及疑難排解此類限流錯誤的更多建議，請參閱[因應 Amazon ECS 限流問題](#)。

# Amazon ECS API 參考

除了 AWS 管理主控台 和 AWS Command Line Interface (AWS CLI) 之外，Amazon ECS 還提供一個 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 AWS CLI 的詳細資訊，請參閱 [《Amazon ECS AWS Command Line Interface 參考》](#)。

## 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                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       | Date            |
|---------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------|
| 將許可更新為 AmazonECSInfrastructureRolePolicyForManagedInstances 。 | 已更新許可，以維持跨分割區的 EC2 服務名稱一致性。此更新不會變更現有的許可模型或授予其他存取權。                                                                                                                                                                                                                                                                                                                                                                                                                                                                       | 2026 年 2 月 26 日 |
| 將許可更新為 AmazonECSInfrastructureRolePolicyForManagedInstances 。 | AmazonECSInfrastructureRolePolicyForManagedInstances 政策已更新，新增下列許可，以支援 Amazon ECS 受管執行個體上的容量保留： <ul style="list-style-type: none"> <li>resource-groups:ListGroupResources 已新增許可，以允許 Amazon ECS 擷取容量保留的群組。若要執行此命令，也需要下列許可：cloudformation:DescribeStacks、cloudformation:ListStackResources、tag:GetResources</li> <li>ec2:DescribeCapacityReservations 已新增許可，以允許 Amazon ECS 收集與 Amazon ECS 受管執行個體整合所需的容量保留詳細資訊。</li> <li>ec2:RunInstances 許可已修改為包含 Amazon ECS 的 Amazon ResourceGroups 群組資源，以佈建目標 Amazon EC2 容量保留。</li> </ul> | 2026 年 2 月 24 日 |
| 將許可新增至 <a href="#">AmazonECSServiceRolePolicy</a> 受管政策。       | AmazonECSServiceRolePolicy 受管 IAM 政策已更新，包含 ssmmessages:OpenDataChannel                                                                                                                                                                                                                                                                                                                                                                                                                                                   | 2026 年 1 月 20 日 |

| 變更                                                            | 描述                                                                                                                                                                                                                                                                                                                                  | Date             |
|---------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------|
|                                                               | 許可。此許可允許 Amazon ECS 開啟 ECS Exec 工作階段的資料通道。                                                                                                                                                                                                                                                                                          |                  |
| 將許可更新為 AmazonECSInfrastructureRolePolicyForManagedInstances 。 | <p>AmazonECSInfrastructureRolePolicyForManagedInstances 政策已更新，以修改 CreateFleet 許可。子網路、安全群組和 EC2 映像的資源型條件已移除，因為：</p> <ul style="list-style-type: none"> <li>Amazon ECS 不會建立或管理子網路和安全群組，因此不包含必要的 AmazonECSManaged 標籤。</li> <li>雖然 Amazon ECS 擁有和管理 EC2 映像，但客戶看不到這些標籤，導致無法套用以資源為基礎的政策。</li> </ul> <p>此變更可確保政策與缺少預期 ECS 管理標籤的資源正確運作。</p> | 2025 年 12 月 15 日 |
| 正確的 <a href="#">受管執行個體</a> 生命週期政策                             | 更新受管執行個體文件，以釐清執行個體生命週期術語，並將執行個體生命週期上限從 14 天延長為 21 天。                                                                                                                                                                                                                                                                                | 2025 年 12 月 11 日 |
| 將許可新增至 <a href="#">AmazonECSServiceRolePolicy</a> 受管政策。       | AmazonECSServiceRolePolicy 受管 IAM 政策已更新，包含 ec2:DescribeInstanceEventWindows 許可。此許可允許 Amazon ECS 為與事件 Windows 相關聯的服務和叢集擷取 Amazon EC2 事件 Windows。                                                                                                                                                                                     | 2025 年 11 月 20 日 |
| 向 AmazonECS ServiceRolePolicy 新增許可。                           | AmazonECSServiceRolePolicy 已更新，以包含啟用 Amazon ECSto 佈建和取消佈建任務 ENI 的 ec2:AssociateTrunkInterface 和 ec2:DisassociateTrunkInterface 許可。                                                                                                                                                                                                  | 2025 年 11 月 14 日 |



| 變更                                                                                                                   | 描述                                                                                                                                                         | Date             |
|----------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------|
| 更新 <a href="#">the section called “AmazonECS InfrastructureRolePolicyForServiceConnectTransportLayerSecurity”</a> 政策 | 更新 <a href="#">the section called “AmazonECSInfrastructureRolePolicyForServiceConnectTransportLayerSecurity”</a> 政策以允許 Amazon ECS 監控 Amazon ECS 建立秘密的生命週期。 | 2025 年 11 月 13 日 |
| Amazon ECS Canary 部署                                                                                                 | Amazon ECS Canary 部署可讓您將一小部分的流量路由到新版本，同時將大多數流量維持在穩定版本上，藉此測試新的應用程式版本。如需詳細資訊，請參閱 <a href="#">Amazon ECS Canary 部署</a> 。                                      | 2025 年 10 月 28 日 |
| Amazon ECS 線性部署                                                                                                      | Amazon ECS 線性部署可讓您以相等百分比的增量，逐步將流量從目前的服務修訂轉移到新的修訂，在每個步驟提供受控部署與監控。如需詳細資訊，請參閱 <a href="#">Amazon ECS 線性部署</a> 。                                               | 2025 年 10 月 28 日 |
| Amazon ECS 受管執行個體區域                                                                                                  | Amazon ECS 受管執行個體現在可在其他區域使用。如需詳細資訊，請參閱 <a href="#">適用於 Amazon ECS 受管執行個體的架構師</a> 。                                                                         | 2025 年 10 月 27 日 |
| AWS for Fluent Bit 3.0.0 版                                                                                           | AWS for Fluent Bit 3.0.0 版本現已推出，以 Fluent Bit 版本4.1.1和 AL2023 為基礎。如需詳細資訊，請參閱 <a href="#">AWS 適用於 Amazon ECS Fluent Bit的影像儲存庫</a> 。                          | 2025 年 10 月 14 日 |
| FireLens 容器非根使用者支援                                                                                                   | Amazon ECS 支援以非根使用者身分執行 FireLens 容器。如需詳細資訊，請參閱 <a href="#">將 Amazon ECS 日誌傳送至 AWS 服務或 AWS Partner</a> 。                                                    | 2025 年 10 月 13 日 |
| 主控台中的 Amazon ECS 事件擷取                                                                                                | Amazon ECS 主控台提供事件擷取功能，可將服務動作和任務狀態變更等 Amazon ECS 產生的事件儲存到透過 EventBridge 的 CloudWatch Logs。如需詳細資訊，請參閱 <a href="#">任務生命週期事件</a> 。                            | 2025 年 10 月 2 日  |

| 變更                                                                                            | 描述                                                                                                                                                               | Date            |
|-----------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------|
| Amazon ECS 支援用於執行容器化工作負載的 Amazon ECS 受管執行個體                                                   | Amazon ECS 支援用於執行容器化工作負載的 Amazon ECS 受管執行個體。如需詳細資訊，請參閱 <a href="#">Amazon ECS Amazon ECS 受管執行個體</a> 。                                                            | 2025 年 9 月 30 日 |
| 新增 <a href="#">the section called “AmazonECS InstanceRolePolicy ForManagedInstances”</a>      | 新增了 AmazonECSInstanceRolePolicyForManagedInstances 政策，該政策授予 Amazon ECS 受管執行個體向 Amazon ECS 叢集進行註冊所需的許可。                                                           | 2025 年 9 月 30 日 |
| 新增 <a href="#">the section called “AmazonECS InfrastructureRolePolicyForManagedInstances”</a> | 新增了 AmazonECSInfrastructureRolePolicyForManagedInstances 政策，該政策授予 Amazon ECS 建立與管理 Amazon EC2 受管資源的存取權。                                                          | 2025 年 9 月 30 日 |
| 支援以純 IPv6 組態執行 Amazon ECS 工作負載                                                                | Amazon ECS 工作負載現在支援以純 IPv6 組態執行，讓任務只能透過 IPv6 進行通訊。如需詳細資訊，請參閱 <a href="#">Amazon ECS 中 EC2 啟動類型的任務聯網選項</a> ，以及 <a href="#">Amazon ECS 中 Fargate 啟動類型的任務聯網選項</a> 。 | 2025 年 9 月 29 日 |
| 支援使用共用 AWS Cloud Map 命名空間的 Amazon ECS Service Connect                                         | Amazon ECS Service Connect 現在支援使用共用 AWS Cloud Map 命名空間進行跨帳戶服務間通訊。如需詳細資訊，請參閱 <a href="#">搭配共用 AWS Cloud Map 命名空間使用 Amazon ECS Service Connect</a> 。               | 2025 年 9 月 12 日 |
| Amazon ECS 使用 Amazon Q 增強 AWS 主控台內的任務定義編輯                                                     | 如需詳細資訊，請參閱在 <a href="#">Amazon ECS 主控台中使用 Amazon Q 提供任務定義建議</a>                                                                                                  | 2025 年 9 月 10 日 |

| 變更                                                                                           | 描述                                                                                                                                                                                                                                                            | Date            |
|----------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------|
| 支援透過主控台執行 ECS Exec 命令                                                                        | 您現在可以直接透過 Amazon ECS 主控台執行 ECS Exec 命令。如需詳細資訊，請參閱 <a href="#">透過主控台執行 ECS Exec 命令</a> 。                                                                                                                                                                       | 2025 年 9 月 4 日  |
| 新增 <a href="#">the section called “AmazonECSInfrastructureRolePolicyForManagedInstances”</a> | 新增了 AmazonECSInfrastructureRolePolicyForManagedInstances 政策，該政策授予 Amazon ECS 建立與管理 Amazon EC2 受管資源的存取權。                                                                                                                                                       | 2025 年 8 月 31 日 |
| 向 <a href="#">AmazonEC2ContainerServiceforEC2Role</a> 受管政策新增許可。                              | AmazonEC2ContainerServiceforEC2Role 受管 IAM 政策已更新，包含 ecs:ListTagsForResource 許可。此許可允許 Amazon ECS 代理程式透過任務中繼資料端點 ( <code>\${ECS_CONTAINER_METADATA_URI_V4}/taskWithTags</code> ) 擷取任務與容器執行個體標籤。如需詳細資訊，請參閱 <a href="#">AmazonEC2ContainerServiceforEC2Role</a> 。 | 2025 年 8 月 4 日  |
| 向 <a href="#">AmazonECSInfrastructureRolePolicyForLoadBalancers</a> 新增許可。                    | AmazonECSInfrastructureRolePolicyForLoadBalancers 受管 IAM 政策已更新，新增了描述、取消註冊與註冊目標群組的許可。                                                                                                                                                                          | 2025 年 7 月 25 日 |
| 支援更新服務部署控制器                                                                                  | Amazon ECS 支援更新服務部署控制器，讓您可以在現有服務的不同部署類型之間進行變更。如需詳細資訊，請參閱 <a href="#">更新 Amazon ECS 服務參數</a> 。                                                                                                                                                                 | 2025 年 7 月 15 日 |
| Amazon ECS 原生藍/綠部署                                                                           | Amazon ECS 支援原生藍/綠部署，讓您可以在驗證新的服務修訂版後，再將生產流量導向這些修訂版。如需詳細資訊，請參閱 <a href="#">Amazon ECS 藍/綠部署</a> 。                                                                                                                                                              | 2025 年 7 月 15 日 |
| 新增 <a href="#">AmazonECSInfrastructureRolePolicyForLoadBalancers</a> 受管政策。                   | 此政策可讓您存取代表您管理與 Amazon ECS 工作負載相關聯的負載平衡器所需的其他 AWS 服務資源。如需詳細資訊，請參閱 <a href="#">AmazonECSInfrastructureRolePolicyForLoadBalancers</a> 。                                                                                                                          | 2025 年 7 月 15 日 |

| 變更                                                                                            | 描述                                                                                                                                                                                                        | Date            |
|-----------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------|
| SOCI 索引資訊清單 v2 支援                                                                             | 希望使用 SOCI 的 Fargate 客戶，只能使用 SOCI 索引資訊清單 v2。先前在 Fargate 上使用 SOCI 的現有客戶可以繼續使用 SOCI 索引資訊清單 v1，但強烈建議這些客戶遷移至 SOCI 索引資訊清單 v2。如需詳細資訊，請參閱 <a href="#">使用 Seekable OCI (SOCI) 延遲載入容器映像</a> 。                       | 2025 年 7 月 2 日  |
| 新增了因運作狀態檢查失敗而產生的服務動作事件中的任務 ID。                                                                | 如需詳細資訊，請參閱 <a href="#">Amazon ECS 服務運作狀態不良的事件訊息</a> 。                                                                                                                                                     | 2025 年 6 月 30 日 |
| 更新預設日誌驅動程式模式。                                                                                 | Amazon ECS 的預設記錄模式已從 blocking 更新為 non-blocking。您可以透過在容器定義的 logConfiguration 中指定 mode，覆寫此預設模式；也可以透過 Amazon ECS 主控台或 AWS CLI 對帳戶進行 defaultLogDriverMode 設定，設定自訂預設模式。如需詳細資訊，請參閱 <a href="#">預設日誌驅動程式模式</a> 。 | 2025 年 6 月 25 日 |
| 向 <a href="#">AmazonECS ServiceRolePolicy</a> 新增許可。                                           | AmazonECSServiceRolePolicy 受管 IAM 政策已更新為新的 AWS Cloud Map 許可，Amazon ECS 可以更新 AWS Cloud Map Amazon ECS 管理之服務的服務屬性。                                                                                          | 2025 年 6 月 24 日 |
| 支援 Amazon ECS 最佳化 Windows Server 2025 Full AMI 與 Amazon ECS 最佳化 Windows Server 2025 Core AMI。 | 如需詳細資訊，請參閱 <a href="#">Amazon ECS 最佳化 Windows AMI</a> 。                                                                                                                                                   | 2025 年 6 月 20 日 |
| 支援更新服務容量提供者策略。                                                                                | 您現在可以更新服務容量提供者策略。如需詳細資訊，請參閱 <a href="#">更新服務參數</a> 。                                                                                                                                                      | 2025 年 6 月 12 日 |

| 變更                                                                 | 描述                                                                                                                                                          | Date             |
|--------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------|
| 向 <a href="#">AmazonECSInfrastructureRolePolicyForVolumes</a> 新增許可 | AmazonECSInfrastructureRolePolicyForVolumes 政策已更新，新增了 ec2:DescribeInstances 許可。此許可有助於防止連接至相同容器執行個體上執行之 Amazon ECS 任務的 Amazon EBS 磁碟區發生裝置名稱衝突。               | 2025 年 6 月 2 日   |
| 支援連接至 Amazon ECS 任務的 Amazon EBS 磁碟區的磁碟區初始化速率。                      | 您現在可以設定 volumeIntiializationRate，決定從現有 Amazon EBS 磁碟區快照擷取資料以建立新磁碟區並連接至 Amazon ECS 任務的速率 (以 MiB/s 為單位)。如需詳細資訊，請參閱 <a href="#">volumeInitializationRate</a> 。 | 2025 年 5 月 13 日  |
| 支援停止服務部署。                                                          | 您現在可以停止服務部署。如需詳細資訊，請參閱 <a href="#">停止 Amazon ECS 服務部署</a> 。                                                                                                 | 2025 年 5 月 2 日   |
| 支援 Amazon ECS 雙堆疊端點。                                               | 您現在可以使用雙堆疊端點與 Amazon ECS 互動。雙堆疊端點支援透過網際網路通訊協定第 4 版 (IPv4) 與第 6 版 (IPv6) 向 Amazon ECS 提出請求。如需詳細資訊，請參閱 <a href="#">使用 Amazon ECS 雙堆疊端點</a> 。                  | 2025 年 4 月 30 日  |
| 支援在 AWS 區域的帳戶層級設定預設日誌驅動程式模式。                                       | 您現在可以使用 defaultLogDriverMode 帳戶設定，設定從容器到您在 logConfiguration 中指定之日誌驅動程式的日誌訊息預設傳送模式。如需詳細資訊，請參閱 <a href="#">預設日誌驅動程式模式</a> 。                                   | 2025 年 4 月 17 日  |
| 支援將短 ARN 格式的服務遷移至長 ARN 格式。                                         | 您現在可以將短 ARN 格式的服務遷移至長 ARN 格式的封包。如需詳細資訊，請參閱 <a href="#">將 Amazon ECS 短服務 ARN 遷移至長 ARN</a> 。                                                                  | 2025 年 2 月 13 日  |
| 支援對 Amazon ECS 與 Fargate 的三類故障注入。                                  | 您現在可以對 Amazon ECS 與 Fargate 使用網路黑洞連接埠、網路延遲及網路封包遺失故障注入。如需詳細資訊，請參閱 <a href="#">使用 AWS Fault Injection Service 測試 Amazon ECS 工作負載</a> 。                        | 2024 年 12 月 19 日 |

| 變更                                                       | 描述                                                                                                                           | Date             |
|----------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------|------------------|
| 支援具有增強可觀測性的 CloudWatch Container Insights。               | 您可以使用具有增強可觀測性的 CloudWatch Container Insights。如需詳細資訊，請參閱 <a href="#">使用具有增強可觀測性的 Container Insights 監控 Amazon ECS 容器</a> 。    | 2024 年 12 月 2 日  |
| 支援預測擴展。                                                  | 您可以使用預測擴展功能預判未來的資源需求，並視需要主動增加服務中的任務數量。如需詳細資訊，請參閱 <a href="#">使用歷史模式透過預測擴展功能擴展 CloudWatch 服務</a> 。                            | 2024 年 11 月 21 日 |
| 支援服務重新平衡功能。                                              | 您現在可以讓 Amazon ECS 自動在可用區域之間平衡服務任務。如需詳細資訊，請參閱 <a href="#">在可用區域之間平衡 Amazon ECS 服務</a> 。                                       | 2024 年 11 月 19 日 |
| 支援設定將容器映像標籤解析為映像摘要的功能。                                   | 您現在可以為服務中任務內的每個容器，設定將容器映像標籤解析為映像摘要的功能。如需詳細資訊，請參閱 <a href="#">容器映像解析及 versionConsistency</a> 。                                | 2024 年 11 月 19 日 |
| 新增 AmazonECSInfrastructureRolePolicyForVpcLattice IAM 政策 | 此 <a href="#">AmazonECSInfrastructureRolePolicyForVpcLattice</a> 政策可讓您存取代表您管理 Amazon ECS 工作負載中 VPC Lattice 功能所需的其他 AWS 服務資源。 | 2024 年 11 月 18 日 |
| VPC Lattice 現已可用於 Amazon ECS 服務。                         | 您現在可以使用 VPC Lattice 來標準化服務間連線、安全性與可觀測性。如需詳細資訊，請參閱 <a href="#">使用 Amazon VPC Lattice 連接、觀察和保護您的 Amazon ECS 服務</a> 。           | 2024 年 11 月 18 日 |
| 支援透過服務部署與服務修訂版檢視服務歷史記錄。                                  | 您可以透過服務部署與服務修訂版檢視部署歷史記錄。如需詳細資訊，請參閱 <a href="#">使用 Amazon ECS 服務部署檢視服務歷史記錄與 Amazon ECS 服務修訂版</a> 。                            | 2024 年 10 月 30 日 |
| 支援採用 Windows 作業系統的 EC2 啟動類型的 Amazon EBS 磁碟區。             | Amazon ECS 支援連接至在 Windows 作業系統 EC2 執行個體上執行之任務的 Amazon EBS 磁碟區。如需詳細資訊，請參閱 <a href="#">將 Amazon EBS 磁碟區與 Amazon ECS 搭配使用</a> 。 | 2024 年 10 月 24 日 |



| 變更                                                                           | 描述                                                                                                                                                                      | Date             |
|------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------|
| 向 AmazonECSInfrastructureRolePolicyForVolumes 新增許可。                          | AmazonECSInfrastructureRolePolicyForVolumes 政策已更新，允許客戶透過快照建立 Amazon EBS 磁碟區。如需詳細資訊，請參閱 <a href="#">AmazonECSInfrastructureRolePolicyForVolumes</a> 。                    | 2024 年 10 月 10 日 |
| Fargate Spot 現已可用於 Linux ARM。                                                | Amazon ECS 已新增對 Linux ARM 上 Fargate Spot 的支援。                                                                                                                           | 2024 年 9 月 6 日   |
| Amazon ECS 任務中個別容器的容器重新啟動政策。                                                 | 您可以為任務定義中定義的必要與非必要容器啟用重新啟動政策，以更快地克服暫時性故障並維持任務可用性。如需詳細資訊，請參閱 <a href="#">使用容器重新啟動政策在 Amazon ECS 任務中重新啟動個別容器</a> 。                                                        | 2024 年 8 月 15 日  |
| 新增了許可至 <a href="#">the section called “AmazonECS_FullAccess”</a>             | AmazonECS_FullAccess 政策已更新，向名為 ecsInfrastructureRole 的角色新增 IAM 角色的 iam:PassRole 許可。這是建立的預設 IAM 角色 AWS 管理主控台，旨在用作 ECS 基礎設施角色，允許 Amazon ECS 管理連接到 ECS 任務的 Amazon EBS 磁碟區。 | 2024 年 8 月 13 日  |
| ECS Anywhere 新增對 Amazon Linux 2023、CentOS Stream 9、Fedora 40 與 Ubuntu 24 的支援 | ECS Anywhere 現已新增對 Amazon Linux 2023、CentOS Stream 9、Fedora 40 與 Ubuntu 24 等作業系統的支援。如需詳細資訊，請參閱 <a href="#">支援的作業系統和系統架構</a> 。                                           | 2024 年 7 月 23 日  |
| 針對使用滾動更新 (ECS) 部署類型的服務，提供容器映像解析功能。                                           | 為確保使用滾動更新部署控制器的服務中的所有任務皆採用相同的容器映像，Amazon ECS 會將任務定義中指定的容器映像名稱與任何映像標籤，解析為容器映像摘要。如需詳細資訊，請參閱 <a href="#">容器映像解析</a> 。                                                      | 2024 年 7 月 10 日  |
| ECS Anywhere 新增對 Debian 11 與 Debian 12 支援                                    | ECS Anywhere 已新增對 Debian 11 與 Debian 12 作業系統的支援。如需詳細資訊，請參閱 <a href="#">支援的作業系統和系統架構</a> 。                                                                               | 2024 年 3 月 28 日  |

| 變更                                                    | 描述                                                                                                                                                 | Date            |
|-------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------|-----------------|
| 支援在 Fargate 上為 Linux 容器使用 gMSA                        | Amazon ECS 透過名為群組受管服務帳戶 (gMSA) 的特殊類型服務帳戶，支援為 Fargate 上的 Linux 容器提供 Active Directory 身分驗證。如需詳細資訊，請參閱 <a href="#">在 Fargate 上為 Linux 容器使用 gMSA</a> 。 | 2024 年 3 月 5 日  |
| 為連接至任務的 Amazon EBS 磁碟區新增了 CloudWatch 指標               | Amazon ECS 現已針對已連接 Amazon EBS 磁碟區的任務，發布 Amazon EBS 儲存使用率的 CloudWatch 指標。如需詳細資訊，請參閱 <a href="#">Amazon ECS CloudWatch 指標</a> 。                      | 2024 年 2 月 8 日  |
| Service Connect TLS                                   | 您現在可以 <a href="#">搭配 Service Connect 使用 TLS</a> 。                                                                                                  | 2024 年 1 月 22 日 |
| Service Connect TLS 受管政策                              | 新增了 <a href="#">AmazonECSInfrastructureRolePolicyForServiceConnectTransportLayerSecurity</a> 政策。                                                   | 2024 年 1 月 22 日 |
| Service Connect 逾時組態更新                                | Service Connect <a href="#">逾時組態</a> 現已可更新，且包含兩個選用參數：idleTimeout 與 perRequestTimeout 。                                                             | 2024 年 1 月 22 日 |
| Amazon ECS 受管執行個體排空                                   | 您可以使用 Amazon ECS <a href="#">受管執行個體排空</a> 功能，加速 Amazon ECS 執行個體的優雅終止。                                                                              | 2024 年 1 月 19 日 |
| ECS Anywhere 新增對 Ubuntu 22 的支援                        | ECS Anywhere 已新增對 Ubuntu 22 作業系統的支援。如需詳細資訊，請參閱 <a href="#">支援的作業系統和系統架構</a> 。                                                                      | 2024 年 1 月 16 日 |
| 新增 AmazonECSInfrastructureRolePolicyForVolumes IAM 政策 | 已新增 <a href="#">AmazonECSInfrastructureRolePolicyForVolumes</a> 。此政策會授予 Amazon ECS 進行 AWS API 呼叫所需的許可，以管理與 Amazon ECS 工作負載相關聯的 Amazon EBS 磁碟區。     | 2024 年 1 月 11 日 |



| 變更                              | 描述                                                                                                                                                                                           | Date             |
|---------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------|
| Amazon ECS 任務的 Amazon EBS 資料磁碟區 | 您可以在部署期間為每個任務設定 1 個 <a href="#">Amazon EBS 資料磁碟區</a> ，用以連接至獨立的 Amazon ECS 任務或 ECS 服務管理的任務。透過在部署時設定磁碟區，您能建立不受特定磁碟區類型或設定所限制且可重複使用的任務定義。Amazon EBS 磁碟區為資料密集型容器化工作負載，提供高可用性、具成本效益、持久且高效能的區塊儲存服務。 | 2024 年 1 月 11 日  |
| Amazon ECS 傳統主控台已終止服務           | Amazon ECS 主控台已終止服務。                                                                                                                                                                         | 2023 年 12 月 4 日  |
| 更新的 政策                          | <a href="#">AmazonECSServiceRolePolicy</a> 受管 IAM 政策已更新，新增了 events 許可以及額外的 autoscaling 與 autoscaling-plans 許可。                                                                               | 2023 年 12 月 4 日  |
| 執行時期監控支援                        | 您可以使用執行時期監控功能監控 Amazon ECS 工作負載，識別惡意或未經授權的行為。如需詳細資訊，請參閱 <a href="#">執行時期監控</a> 。                                                                                                             | 2023 年 11 月 26 日 |
| 更新的 政策                          | 受 <a href="#">AmazonECSServiceRolePolicy</a> 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 任務淘汰組態              | 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 <a href="#">AWS Fargate 任務維護</a> 。                                                                                                                       | 2023 年 9 月 5 日   |
| 中的其他任務定義參數 AWS Fargate          | AWS Fargate 在 Linux 平台版本 systemControls 中新增對 pidMode 和 的支援 1.4.0。如需詳細資訊，請參閱 <a href="#">任務定義</a> 。                                                                                           | 2023 年 8 月 9 日   |
| Amazon ECS 主控台任務定義頁面重新設計        | Amazon ECS 主控台內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 <a href="#">使用主控台建立任務定義</a> 。                                                                                                                | 2023 年 7 月 26 日  |

| 變更                                | 描述                                                                                                                                                                                                                                                                          | Date            |
|-----------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------|
| Fargate 支援使用 Seekable OCI 索引的延遲載入 | AWS Fargate 正在引入 Seekable 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 <a href="#">使用 Seekable OCI (SOCi) 延遲載入容器映像</a> 。                                                                                                           | 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 <a href="#">Tutorial: Using Windows Containers with Domainless gMSA using the AWS CLI</a> 。如需詳細資訊，請參閱 <a href="#">針對 Linux 容器使用 gMSA</a> 和 <a href="#">針對 Windows 容器使用 gMSA</a> 。 | 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。在 GitHub 上可取得已發行的版本和代理程式的變更。如需詳細資訊，請參閱 <a href="#">Amazon ECS Linux 容器代理程式版本</a> 。                                                                                                          | 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 <a href="#">考量事項</a> 。                                                                                                                                                                                                                 | 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 <a href="#">Amazon ECS 叢集自動擴展</a> 。                                                                                                                                                                     | 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 <code>ecsCreateCluster</code> 。當您建立資源並指定該資源的標籤時，AWS 會執行額外的授權，以驗證是否有建立標籤的許可。如需詳細資訊，請參閱 <a href="#">標記授權</a> 和 <a href="#">建立時授予標記資源的許可</a> 。                                                                                                             | 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 <a href="#">針對 Linux 容器使用 gMSA</a> 。                                                                                                                                                                          | 2023 年 4 月 14 日 |

| 變更                                   | 描述                                                                                                                                      | Date            |
|--------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------|-----------------|
| 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 <a href="#">Fargate 任務儲存</a> 。                                                          | 2023 年 4 月 14 日 |
| AWS Cost Management 支援任務層級 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 <a href="#">任務層級成本和用量報告</a> 。                           | 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 <a href="#">Amazon ECS 最佳化 Linux AMI</a> 。                                  | 2023 年 4 月 10 日 |
| AWS Fargate 聯邦資訊處理標準 (FIPS) 140      | 您可以在 Amazon ECS 上 AWS Fargate 以符合聯邦資訊處理標準 (FIPS) 140 的方式部署工作負載。如需詳細資訊，請參閱 <a href="#">AWS Fargate 聯邦資訊處理標準 (FIPS-140)</a> 。             | 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 <a href="#">使用主控台刪除任務定義修訂版</a> 和 <a href="#">任務定義</a> 。                            | 2023 年 2 月 24 日 |
| AWS Fargate Compute Optimizer 中的服務建議 | AWS Compute Optimizer 會根據 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 <a href="#">檢視針對 Fargate 上 Amazon ECS 服務的建議</a> 。 | 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 <a href="#">新版 Amazon ECS 主控台</a> 。                                                                  | 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 <a href="#">AmazonECS_FullAccess</a> 。                                  | 2023 年 1 月 4 日  |

| 變更                                    | 描述                                                                                                                                          | Date             |
|---------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------|------------------|
| 使用 CloudWatch 警示來偵測 Amazon ECS 服務部署失敗 | 您可以設定 Amazon ECS，以便在其偵測到指定的 CloudWatch 警示已進入 ALARM 狀態時，將部署設定為失敗。如需詳細資訊，請參閱 <a href="#">the section called “滾動更新部署”</a> 。                    | 2022 年 12 月 19 日 |
| 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 <a href="#">the section called “連接埠映射”</a> 。                                                     | 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 <a href="#">the section called “Service Connect”</a> 。                                        | 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 <a href="#">the section called “使用主控台建立任務定義”</a> 。                                         | 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 <a href="#">the section called “使用主控台建立任務定義”</a> 。                                         | 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 <a href="#">the section called “建立滾動更新部署”</a> 及 <a href="#">the section called “將應用程式作為任務執行”</a> 。 | 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 <a href="#">the section called “任務中繼資料端點第 4 版”</a> 。                                             | 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 <a href="#">the section called “任務大小”</a> 。                                 | 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 滿足您的命令列工具需求。如需詳細資訊，請參閱 <a href="#">使用 AWS Copilot 命令列界面建立 Amazon ECS 資源</a> 。                            | 2022 年 9 月 15 日  |

| 變更                                               | 描述                                                                                                                                                                                                                                    | Date             |
|--------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------|
| 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 <a href="#">Amazon ECS endpoints and quotas</a> 。                                                                                                                                           | 2022 年 9 月 8 日   |
| 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 <a href="#">為 Amazon ECS Auto Scaling 群組設定預先初始化的執行個體</a> 。                                                                                                            | 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 <a href="#">外部執行個體的 Amazon ECS 叢集</a> 。                                                                                                                                                     | 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 <a href="#">使用 ECS Exec 監控 Amazon ECS 容器</a> 。                                                                                                                                                          | 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 <a href="#">為 Fargate 工作負載建立 Amazon ECS 叢集</a> 、 <a href="#">刪除 Amazon ECS 叢集</a> 、 <a href="#">使用主控台更新 Amazon ECS 任務定義</a> 和 <a href="#">使用主控台取消註冊 Amazon ECS 任務定義修訂版</a> 。 | 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 <a href="#">使用主控台建立 Amazon ECS 任務定義</a> 。                                                                                                                                                      | 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 <a href="#">the section called "64 位元 ARM 工作負載的任務定義"</a> 。                                                                                                                      | 2021 年 11 月 23 日 |
| Amazon ECS 支援 fluentd log-driver-buffer-limit 選項 | Amazon ECS 支援 fluentd log-driver-buffer-limit 選項。如需詳細資訊，請參閱 <a href="#">將 Amazon ECS 日誌傳送至 AWS 服務或 AWS Partner</a> 。                                                                                                                  | 2021 年 11 月 22 日 |

| 變更                                       | 描述                                                                                                                                          | Date             |
|------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------|------------------|
| 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 <a href="#">Amazon ECS 最佳化 Linux AMI 建置指令碼</a> 。               | 2021 年 11 月 19 日 |
| 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 <a href="#">監控 Amazon ECS 容器執行個體運作狀態</a> 。                                                         | 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 <a href="#">使用 ECS Exec 監控 Amazon ECS 容器</a> 。                                                        | 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 <a href="#">Fargate Windows 平台版本變更日誌</a> 。                                                   | 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 <a href="#">GPU 工作負載的 Amazon ECS 任務定義</a> 及 <a href="#">將外部執行個體註冊到 Amazon ECS 叢集</a> 。 | 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 <a href="#">為 Amazon ECS 任務配置網路介面</a> 。                                                     | 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 <a href="#">Amazon ECS 最佳化 Bottlerocket AMI</a> 。               | 2021 年 6 月 30 日  |
| Amazon ECS 已排程的任務更新                      | 在建立觸發 Amazon ECS 排程任務的規則時，Amazon EventBridge 新增了對其他參數的支援。                                                                                   | 2021 年 6 月 25 日  |
| AWS Amazon ECS 的受管政策                     | Amazon ECS 新增了服務連結角色受 AWS 管政策的文件。如需詳細資訊，請參閱 <a href="#">AWS Amazon Elastic Container Service 的受管政策</a> 。                                    | 2021 年 6 月 8 日   |



| 變更                                                         | 描述                                                                                                                                                      | Date             |
|------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------|------------------|
| 開始使用 AWS CDK                                               | 新增將 AWS CDK 與 Amazon ECS 搭配使用的入門指南。如需詳細資訊，請參閱 <a href="#">使用 建立 Amazon ECS 資源 AWS CDK</a> 。                                                             | 2021 年 5 月 27 日  |
| 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 <a href="#">外部執行個體的 Amazon ECS 叢集</a> 。                                                             | 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 <a href="#">Amazon ECS 最佳化 Linux AMI</a> 。                         | 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 <a href="#">使用 ECS Exec 監控 Amazon ECS 容器</a> 。                                                            | 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 <a href="#">為 Amazon ECS 建立 VPC 端點政策</a> 。                                                                   | 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。如需詳細資訊，請參閱 <a href="#">建立 Amazon ECS 滾動更新部署</a> 及 <a href="#">將應用程式作為 Amazon ECS 任務執行</a> 。                     | 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 <a href="#">搭配 Amazon ECS 使用 FSx for Windows File Server 磁碟區</a> 。 | 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 <a href="#">在雙堆疊模式下使用 VPC</a> 。                                             | 2020 年 11 月 5 日  |

| 變更                               | 描述                                                                                                                                                                                            | Date            |
|----------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------|
| 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 <a href="#">Amazon ECS 任務中繼資料端點第 4 版</a> 。                                                                                                     | 2020 年 11 月 5 日 |
| 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 <a href="#">共用子網路、Local Zone 與 Wavelength Zone 中的 Amazon ECS 應用程式</a> 。                                                         | 2020 年 9 月 4 日  |
| Bottlerocket AMI 的 Amazon ECS 變體 | Bottlerocket 是以 Linux 為基礎的開放原始碼作業系統，由專為執行容器 AWS 而建置。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 <a href="#">Amazon ECS 最佳化 Bottlerocket AMI</a> 。 | 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 <a href="#">Amazon ECS 任務中繼資料端點第 4 版</a> 。                          | 2020 年 8 月 10 日 |
| Fargate 用量指標                     | AWS Fargate 提供 CloudWatch 用量指標，可讓您查看 Fargate 隨需和 Fargate Spot 資源的帳戶用量。如需詳細資訊，請參閱 <a href="#">用量指標</a> 。                                                                                       | 2020 年 8 月 3 日  |
| AWS Copilot 0.1.0 版              | 推出新的 AWS Copilot CLI，提供高階命令，從本機開發環境簡化 Amazon ECS 上的建模、建立、發行和管理容器化應用程式。如需詳細資訊，請參閱 <a href="#">使用 AWS Copilot 命令列界面建立 Amazon ECS 資源</a> 。                                                       | 2020 年 7 月 9 日  |
| AWS Fargate 平台版本棄用排程             | 已新增 Fargate 平台版本取代排程。如需詳細資訊，請參閱 <a href="#">AWS Fargate Linux 平台版本棄用</a> 。                                                                                                                    | 2020 年 7 月 8 日  |
| AWS Fargate 區域擴展                 | AWS Fargate 上的 Amazon ECS 已擴展至歐洲 (米蘭) 區域。                                                                                                                                                     | 2020 年 6 月 25 日 |



| 變更                                             | 描述                                                                                                                                                                  | Date            |
|------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------|
| 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。<br><br>如需詳細資訊，請參閱 <a href="#">Amazon ECS 最佳化 Linux AMI</a> 。                                     | 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 版更新                       | 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 <a href="#">Amazon ECS 的 Fargate 任務暫時性儲存</a> 。 | 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 <a href="#">將個別環境變數傳遞至 Amazon ECS 容器</a> 。                                                                            | 2020 年 5 月 18 日 |
| AWS Fargate 區域擴展                               | AWS Fargate 搭配 Amazon ECS 已擴展至非洲（開普敦）區域。                                                                                                                            | 2020 年 5 月 11 日 |
| 服務配額已更新                                        | <p>下列服務配額已更新：</p> <ul style="list-style-type: none"> <li>每個帳戶的叢集已從 2,000 提高到 10,000。</li> </ul> <p>如需詳細資訊，請參閱 <a href="#">Amazon ECS endpoints and quotas</a>。</p>  | 2020 年 4 月 17 日 |

| 變更                     | 描述                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    | Date           |
|------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------|
| AWS Fargate 平台 1.4.0 版 | <p>AWS Fargate 平台 1.4.0 版已發行，其中包含下列功能：</p> <ul style="list-style-type: none"> <li>• 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<a href="#">將 Amazon EFS 磁碟區與 Amazon ECS 搭配使用</a>。</li> <li>• 暫時性任務儲存體已增加至 20 GB。如需詳細資訊，請參閱<a href="#">Amazon ECS 的 Fargate 任務暫時性儲存</a>。</li> <li>• 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<a href="#">Amazon ECS 中 Fargate 啟動類型的任務聯網選項</a>。</li> <li>• 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。</li> <li>• CloudWatch Container Insights 將包含 Fargate 任務的網路效能指標。如需詳細資訊，請參閱<a href="#">使用具有增強可觀測性的 Container Insights 監控 Amazon ECS</a>。</li> <li>• 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<a href="#">Amazon ECS 任務中繼資料端點第 4 版</a>。</li> </ul> | 2020 年 4 月 8 日 |

| 變更                                  | 描述                                                                                                                                                                                                                                                                                                                                                                                               | Date            |
|-------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------|
|                                     | <ul style="list-style-type: none"> <li>在容器定義中，新增對 <code>SYS_PTRACE</code> Linux 參數的支援。如需詳細資訊，請參閱<a href="#">Linux 參數</a>。</li> <li>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。</li> <li>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。</li> </ul> <p>如需詳細資訊，請參閱<a href="#">適用於 Amazon ECS 的 Fargate 平台版本</a>。</p> |                 |
| 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 <a href="#">將 Amazon EFS 磁碟區與 Amazon ECS 搭配使用</a> 。                                                                                                                                                                                                                                                                                  | 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 <code>ECS_CONTAINER_META_DATA_URI_V4</code> 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 <a href="#">Docker 統計資訊</a> 可供任務使用。如需詳細資訊，請參閱 <a href="#">Amazon ECS 任務中繼資料端點第 4 版</a> 。                                                                                                                                                      | 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 <a href="#">將敏感資料傳遞至 Amazon ECS 容器</a> 。                                                                                                                                                                                                                                                                                                         | 2020 年 2 月 24 日 |
| 已針對藍/綠部署新增額外的 CodeDeploy 部署組態選項。    | CodeDeploy 服務已針對 Amazon ECS 部署類型新增新的 Canary 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 <a href="#">適用於 Amazon ECS 的 CodeDeploy 藍/綠部署</a> 。                                                                                                                                                                                                                                                                   | 2020 年 2 月 6 日  |

| 變更                                | 描述                                                                                                                                                                  | Date             |
|-----------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------|
| 新增 efsVolumeConfiguration 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 <a href="#">將 Amazon EFS 磁碟區與 Amazon ECS 搭配使用</a> 。                       | 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 <a href="#">Amazon ECS 容器代理程式日誌組態參數</a> 。                                                                                  | 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 <a href="#">Fargate 的 Amazon ECS 叢集</a> 。                                                                           | 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 <a href="#">使用叢集自動擴展自動管理 Amazon ECS 容量</a> 。                                                                         | 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 <a href="#">Amazon ECS 叢集</a> 。                                                                                         | 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 <a href="#">the section called “上的 Amazon Elastic Container Service AWS Outposts”</a> 。                              | 2019 年 12 月 3 日  |
| 服務動作事件                            | Amazon ECS 現在會在某些服務動作發生時，將事件傳送至 Amazon EventBridge。如需詳細資訊，請參閱 <a href="#">Amazon ECS 服務動作事件</a> 。                                                                   | 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 <a href="#">GPU 工作負載的 Amazon ECS 任務定義</a> 。                                                      | 2019 年 10 月 8 日  |
| FireLens for Amazon ECS           | FireLens for Amazon ECS 正式發行。FireLens for Amazon ECS 可讓您使用任務定義參數，將日誌路由至 AWS 服務或合作夥伴目的地，以儲存和分析日誌。如需詳細資訊，請參閱 <a href="#">將 Amazon ECS 日誌傳送至 AWS 服務或 AWS Partner</a> 。 | 2019 年 9 月 30 日  |

| 變更                                                         | 描述                                                                                                                                                                      | Date            |
|------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------|
| AWS Fargate 區域擴展                                           | AWS Fargate 搭配 Amazon ECS 已擴展至歐洲（巴黎）、歐洲（斯德哥爾摩）和中東（巴林）區域。                                                                                                                | 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。                                                                                                   | 2019 年 9 月 3 日  |
| FireLens for Amazon ECS                                    | FireLens for Amazon ECS 為公開預覽版。FireLens for 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 <a href="#">將 Amazon ECS 日誌傳送至 AWS 服務或 AWS Partner</a> 。 | 2019 年 8 月 30 日 |
| CloudWatch Container Insights                              | CloudWatch Container Insights 現已正式發行。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 <a href="#">使用具有增強可觀測性的 Container Insights 監控 Amazon ECS</a> 。                           | 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 <a href="#">在 Amazon ECS 上管理容器交換記憶體空間</a> 。  | 2019 年 8 月 16 日 |
| AWS Fargate 區域擴展                                           | AWS Fargate 搭配 Amazon ECS 已擴展至亞太區域（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 <a href="#">援更多 Amazon ECS 容器網路介面的執行個體</a> 。                                                                           | 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 <a href="#">透過 Amazon ECS 服務註冊多個目標群組</a> 。                                                                                              | 2019 年 7 月 30 日 |

| 變更                                       | 描述                                                                                                                                                                                                                                 | Date            |
|------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------|
| 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 <a href="#">在 Amazon ECS 中使用 Secrets Manager 秘密指定敏感資料</a> 。                                                                                                                         | 2019 年 7 月 20 日 |
| CloudWatch Container Insights            | Amazon ECS 已新增對 CloudWatch Container Insights 的支援。如需詳細資訊，請參閱 <a href="#">使用具有增強可觀測性的 Container Insights 監控 Amazon ECS</a> 。                                                                                                        | 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 <a href="#">Amazon Elastic Container Service 如何與 IAM 搭配使用</a> 。                                                                                                               | 2019 年 6 月 27 日 |
| 針對 2019 AWS 年 5 月修補的新 Amazon ECS 最佳化 AMI | Amazon ECS 已更新 Amazon ECS 最佳化 AMI 以解決 <a href="#">AWS-2019-005</a> 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 <code>awsipcTrunking</code> 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 <a href="#">增加 Amazon ECS Linux 容器執行個體網路介面數量</a> 。 | 2019 年 6 月 6 日  |
| AWS Fargate 平台 1.3.0 版更新                 | 從 2019 年 5 月 1 日開始，啟動的任何新 Fargate 任務除了支援 <code>awslogs</code> 日誌驅動程式外，還支援 <code>splunk</code> 日誌驅動程式。如需詳細資訊，請參閱 <a href="#">儲存與記錄</a> 。                                                                                            | 2019 年 5 月 1 日  |
| AWS Fargate 平台 1.3.0 版更新                 | 從 2019 年 5 月 1 日開始，啟動的任何新 Fargate 任務支援使用 <code>secretOptions</code> 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 <a href="#">將敏感資料傳遞至 Amazon ECS 容器</a> 。                                                                                     | 2019 年 5 月 1 日  |

| 變更                       | 描述                                                                                                                                                                                                                                                                                  | Date            |
|--------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------|
| AWS Fargate 平台 1.3.0 版更新 | 從 2019 年 4 月 2 日開始，啟動的任何新 Fargate 任務都支援將敏感資料注入您的容器，方法是將敏感資料存放在 AWS Secrets Manager 秘密或 AWS Systems Manager 參數存放區參數中，然後在容器定義中參考它們。如需詳細資訊，請參閱 <a href="#">將敏感資料傳遞至 Amazon ECS 容器</a> 。                                                                                                | 2019 年 4 月 2 日  |
| AWS Fargate 平台 1.3.0 版更新 | 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 <a href="#">代理組態</a> 、 <a href="#">容器相依性</a> 及 <a href="#">容器逾時</a> 。                                                                                                           | 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 <a href="#">使用第三方控制器部署 Amazon ECS 服務</a> 。                                                                                                                                                                                | 2019 年 3 月 27 日 |
| AWS Amazon ECS 上的深度學習容器  | AWS 深度學習容器是一組 Docker 影像，用於在 Amazon Elastic Container Service (Amazon ECS) 上的 TensorFlow 中訓練和服務模型。Deep Learning Containers 提供最佳化環境與 TensorFlow、Nvidia CUDA (適用於 GPU 執行個體) 和 Intel MKL (適用於 CPU 執行個體) 程式庫，並可在 Amazon ECR 中使用。如需詳細資訊，請參閱 <a href="#">在 Amazon ECS 上使用 AWS 深度學習容器</a> 。 | 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 <a href="#">容器相依性</a> 。                                                                                                                                                                                          | 2019 年 3 月 7 日  |



| 變更                                         | 描述                                                                                                                                                                                          | Date             |
|--------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------|
| Amazon ECS 推出 PutAccountSettingDefault API | <p>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。</p> <p>如需詳細資訊，請參閱<a href="#">Amazon Resource Name (ARN) 和 ID</a>。</p>     | 2019 年 2 月 8 日   |
| Amazon ECS 支援 GPU 工作負載                     | <p>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。</p> <p>如需詳細資訊，請參閱<a href="#">GPU 工作負載的 Amazon ECS 任務定義</a>。</p>                            | 2019 年 2 月 4 日   |
| Amazon ECS 擴充秘密支援                          | <p>Amazon ECS 擴展了對在您的任務定義中直接使用 AWS Secrets Manager 秘密將敏感資料注入容器的支援。</p> <p>如需詳細資訊，請參閱<a href="#">將敏感資料傳遞至 Amazon ECS 容器</a>。</p>                                                             | 2019 年 1 月 21 日  |
| 介面 VPC 端點 (AWS PrivateLink)                | <p>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 Direct Connect 進行存取。</p> <p>如需詳細資訊，請參閱<a href="#">介面 VPC 端點 (AWS PrivateLink)</a>。</p> | 2018 年 12 月 26 日 |



| 變更                     | 描述                                                                                                                                                                                                                                                                                                                                                                                                                                      | Date             |
|------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------|
| AWS Fargate 平台 1.3.0 版 | <p>發行新的 AWS Fargate 平台版本，其中包含：</p> <ul style="list-style-type: none"> <li>新增使用 AWS Systems Manager 參數存放區參數將敏感資料注入容器的支援。</li> </ul> <p>如需詳細資訊，請參閱<a href="#">將敏感資料傳遞至 Amazon ECS 容器</a>。</p> <ul style="list-style-type: none"> <li>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。</li> </ul> <p>如需詳細資訊，請參閱<a href="#">Amazon ECS AWS Fargate 上的 任務淘汰和維護</a>。</p> <p>如需詳細資訊，請參閱<a href="#">適用於 Amazon ECS 的 Fargate 平台版本</a>。</p> | 2018 年 12 月 17 日 |
| 更新服務限制                 | <p>以下服務限制已更新：</p> <ul style="list-style-type: none"> <li>每個帳戶每一區域的叢集數目已從 1000 提高到 2000。</li> <li>每個叢集的容器執行個體數目已從 1000 提高到 2000。</li> <li>每個叢集的服務數目已從 500 提高到 1000。</li> </ul> <p>如需詳細資訊，請參閱<a href="#">Amazon ECS 服務配額</a>。</p>                                                                                                                                                                                                         | 2018 年 12 月 14 日 |
| AWS Fargate 區域擴展       | <p>AWS Fargate 搭配 Amazon ECS 已擴展至亞太區域（孟買）和加拿大（中部）區域。</p> <p>如需詳細資訊，請參閱<a href="#">AWS Fargate 上 Amazon ECS 支援的 區域</a>。</p>                                                                                                                                                                                                                                                                                                              | 2018 年 12 月 7 日  |

| 變更                                            | 描述                                                                                                                                                   | Date             |
|-----------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------|------------------|
| Amazon ECS 藍/綠部署                              | <p>Amazon ECS 已新增使用 CodeDeploy 進行藍/綠部署的支援。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。</p> <p>如需詳細資訊，請參閱<a href="#">適用於 Amazon ECS 的 CodeDeploy 藍/綠部署</a>。</p>   | 2018 年 11 月 27 日 |
| Amazon ECS 最佳化 Amazon Linux 2 (arm64) AMI 已發佈 | <p>Amazon ECS 已發佈 Amazon ECS 最佳化 Amazon Linux 2 AMI，適用於 arm64 架構。</p> <p>如需詳細資訊，請參閱<a href="#">Amazon ECS 最佳化 Linux AMI</a>。</p>                     | 2018 年 11 月 26 日 |
| 在任務定義中增加支援其他 Docker 旗標                        | <p>Amazon ECS 在任務定義中支援下列 Docker 標記：</p> <ul style="list-style-type: none"> <li>• <a href="#">IPC 模式</a></li> <li>• <a href="#">PID 模式</a></li> </ul> | 2018 年 11 月 16 日 |
| 支援 Amazon ECS 秘密                              | <p>Amazon ECS 新增了使用 AWS Systems Manager 參數存放區參數將敏感資料注入容器的支援。</p> <p>如需詳細資訊，請參閱<a href="#">將敏感資料傳遞至 Amazon ECS 容器</a>。</p>                            | 2018 年 11 月 15 日 |
| 資源標記                                          | <p>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。</p> <p>如需詳細資訊，請參閱<a href="#">標記 Amazon ECS 資源</a>。</p>                                           | 2018 年 11 月 15 日 |
| AWS Fargate 區域擴展                              | <p>AWS Fargate 搭配 Amazon ECS 已擴展至美國西部（加利佛尼亞北部）和亞太區域（首爾）區域。</p> <p>如需詳細資訊，請參閱<a href="#">為 Amazon ECS 規劃 AWS Fargate 架構</a>。</p>                      | 2018 年 11 月 7 日  |

| 變更                                    | 描述                                                                                                                                                                                                                                                     | Date             |
|---------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------|
| 更新服務限制                                | <p>以下服務限制已更新：</p> <ul style="list-style-type: none"> <li>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。</li> <li>使用 Fargate 啟動類型之任務的公有 IP 地址數量從 20 增加到 50。</li> </ul> <p>如需詳細資訊，請參閱<a href="#">Amazon ECS 服務配額</a>。</p>                                      | 2018 年 10 月 31 日 |
| AWS Fargate 區域擴展                      | <p>AWS Fargate 搭配 Amazon ECS 已擴展至歐洲（倫敦）區域。</p> <p>如需詳細資訊，請參閱<a href="#">為 Amazon ECS 規劃 AWS Fargate 架構</a>。</p>                                                                                                                                        | 2018 年 10 月 26 日 |
| Amazon ECS 最佳化 Amazon Linux 2 AMI 已發佈 | <p>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 x86_64 為基礎。Amazon ECS 也銷售以 Amazon Linux AMI 為基礎的 AMI，但建議將工作負載遷移至 Amazon Linux 2 變體，因為在 2020 年 6 月 30 日之後將不再支援 Amazon Linux AMI。</p> <p>如需詳細資訊，請參閱<a href="#">Amazon ECS 最佳化 Linux AMI</a>。</p>         | 2018 年 10 月 18 日 |
| Amazon ECS 任務中繼資料端點第 3 版              | <p>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 <a href="#">Docker 統計資訊</a>。如需詳細資訊，請參閱<a href="#">使用 Amazon ECS 中繼資料監控工作負載</a>。</p> | 2018 年 10 月 18 日 |

| 變更                                        | 描述                                                                                                                                                                               | Date            |
|-------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------|
| Amazon ECS 服務探索區域擴展                       | <p>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。</p> <p>如需詳細資訊，請參閱<a href="#">使用服務探索以利用 DNS 名稱連接 Amazon ECS 服務</a>。</p>                              | 2018 年 9 月 27 日 |
| 新增對容器定義中額外 Docker 標記的支援                   | <p>Amazon ECS 支援容器定義中的下列 Docker 標記：</p> <ul style="list-style-type: none"> <li>• <a href="#">系統控制</a></li> <li>• <a href="#">互動性</a></li> <li>• <a href="#">虛擬終端機</a></li> </ul> | 2018 年 9 月 17 日 |
| 使用 AWS Fargate 任務支援 Amazon ECS 的私有登錄檔身分驗證 | <p>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。</p> <p>如需詳細資訊，請參閱<a href="#">在 Amazon ECS 中使用非AWS容器映像</a>。</p>   | 2018 年 9 月 10 日 |
| Amazon ECS 服務探索區域擴展                       | <p>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。</p> <p>如需詳細資訊，請參閱<a href="#">使用服務探索以利用 DNS 名稱連接 Amazon ECS 服務</a>。</p>                            | 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                                                                                                                                                                  | Date            |
|-------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------|
| 使用 AWS Secrets Manager 支援的私有登錄檔身分驗證 | <p>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。</p> <p>如需詳細資訊，請參閱<a href="#">在 Amazon ECS 中使用非AWS 容器映像</a>。</p> | 2018 年 8 月 16 日 |
| 新增 Docker 磁碟區支援                     | <p>Amazon ECS 推出對 Docker 磁碟區的支援。</p> <p>如需詳細資訊，請參閱<a href="#">Amazon ECS 任務的儲存選項</a>。</p>                                                                           | 2018 年 8 月 9 日  |
| AWS Fargate 區域擴展                    | <p>AWS Fargate 搭配 Amazon ECS 已擴展至歐洲（法蘭克福）、亞太區域（新加坡）和亞太區域（雪梨）區域。</p> <p>如需詳細資訊，請參閱<a href="#">為 Amazon ECS 規劃 AWS Fargate 架構</a>。</p>                                | 2018 年 7 月 19 日 |

| 變更                        | 描述                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     | Date            |
|---------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------|
| 已新增 Amazon ECS 服務排程器策略    | <p>Amazon ECS 推出服務排程器策略概念。</p> <p>現已推出兩個服務排程器策略概念：</p> <ul style="list-style-type: none"> <li>• REPLICHA - 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<a href="#">複本排程策略</a>。</li> <li>• 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<a href="#">常駐程式排程策略</a>。</li> </ul> <div style="border: 1px solid #add8e6; border-radius: 10px; padding: 10px; margin-top: 10px;"> <p> <b>Note</b></p> <p>Fargate 任務不支援 DAEMON 排程策略。</p> </div> | 2018 年 6 月 12 日 |
| Amazon ECS 容器代理程式 v1.18.0 | <p>發行了新版 Amazon ECS 容器代理程式，其新增了下列功能：</p> <ul style="list-style-type: none"> <li>• 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<a href="#">Amazon ECS 容器代理程式組態</a>。</li> </ul> <p>如需詳細資訊，請參閱 <a href="#">amazon-ecs-agent github</a>。</p>                                                                                                                                                                                                                                                                                                                                                                                                | 2018 年 5 月 24 日 |

| 變更                                 | 描述                                                                                                                                                                                                                                                                                                                        | Date            |
|------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------|
| 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 <a href="#">使用服務探索以利用 DNS 名稱連接 Amazon ECS 服務</a> 。                                                                                                                                                                                                   | 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 <a href="#">擷取 Amazon ECS 最佳化 Linux AMI 中繼資料</a> 。                                                                                                                                                    | 2018 年 5 月 9 日  |
| AWS Fargate 區域擴展                   | <p>AWS Fargate 搭配 Amazon ECS 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。</p> <p>如需詳細資訊，請參閱<a href="#">為 Amazon ECS 規劃 AWS Fargate 架構</a>。</p>                                                                                                                                                                                    | 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 <a href="#">擷取 Amazon ECS 最佳化 Linux AMI 中繼資料</a> 。                                                                                                                                                                                       | 2018 年 4 月 10 日 |
| AWS Fargate 平台版本                   | <p>發行新的 AWS Fargate 平台版本，其中包含：</p> <ul style="list-style-type: none"> <li>• 新增了對 <a href="#">使用 Amazon ECS 中繼資料監控工作負載</a> 的支援。</li> <li>• 新增了對 <a href="#">運作狀態檢查</a> 的支援。</li> <li>• 新增對 <a href="#">使用服務探索以利用 DNS 名稱連接 Amazon ECS 服務</a> 的支援</li> </ul> <p>如需詳細資訊，請參閱<a href="#">適用於 Amazon ECS 的 Fargate 平台版本</a>。</p> | 2018 年 3 月 26 日 |

| 變更                                        | 描述                                                                                                                                                                            | Date            |
|-------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------|
| 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 <a href="#">使用服務探索以利用 DNS 名稱連接 Amazon ECS 服務</a> 。                                                                            | 2018 年 3 月 22 日 |
| 支援 Docker shm-size 和 tmpfs                | <p>在 Amazon ECS 任務定義中，新增對 Docker shm-size 和 tmpfs 參數的支援。</p> <p>如需更新之 ECS CLI 語法的詳細資訊，請參閱「<a href="#">Linux 參數</a>」。</p>                                                      | 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 <a href="#">運作狀態檢查</a> 。                                                                                                               | 2018 年 3 月 8 日  |
| AWS Fargate                               | 新增使用 AWS Fargate 的 Amazon ECS 概觀。如需詳細資訊，請參閱 <a href="#">為 Amazon ECS 規劃 AWS Fargate 架構</a> 。                                                                                  | 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 <a href="#">Docker 統計資訊</a> 。如需詳細資訊，請參閱 <a href="#">使用 Amazon ECS 中繼資料監控工作負載</a> 。 | 2018 年 2 月 8 日  |
| 使用目標追蹤政策的 Amazon ECS Service Auto Scaling | <p>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<a href="#">使用目標指標擴展 Amazon ECS 服務</a>。</p> <p>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。</p>     | 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 <a href="#">Amazon ECS 最佳化 Linux AMI</a> 。                                                                                                    | 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。記錄了服務任務發生連續失敗時觸發的新服務事件訊息。                                                                                                                                   | 2018 年 1 月 11 日 |



| 變更                                   | 描述                                                                                                                                                                                                                                                                     | Date             |
|--------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------|
| 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 <a href="#">TaskDefinition</a> 。                                                                                                                                                                                               | 2017 年 12 月 12 日 |
| 任務執行 角色                              | <p>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：</p> <ul style="list-style-type: none"> <li>• 呼叫 Amazon ECR 以提取容器映像</li> <li>• 呼叫 CloudWatch 以存放容器應用程式日誌</li> </ul> <p>如需詳細資訊，請參閱<a href="#">Amazon ECS 任務執行 IAM 角色</a>。</p> | 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 <a href="#">Amazon ECS 最佳化 AMI 變體</a> 。                                                                                                                                                                                       | 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 <a href="#">規劃 Amazon ECS 解決方案架構</a> 。                                                                                                                                                                                | 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                                                                                                                                                                                                                                               | Date             |
|---------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------|
| 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 <a href="#">EC2 的 Amazon ECS 任務聯網選項</a> 。 | 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 <a href="#">Amazon ECS 容器中繼資料檔案</a> 。                                                                                                                                     | 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 <a href="#">Amazon ECS 最佳化 Linux AMI</a> 。                                                                                                                                                                       | 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 <a href="#">LinuxParameters</a> 。                                                                                                         | 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 <a href="#">LinuxParameters</a> 。                                                                                                                    | 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 覆寫                      | 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 <a href="#">將應用程式作為 Amazon ECS 任務執行</a> 。                                                                                                                                           | 2017 年 6 月 27 日  |
| 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   |

| 變更                                      | 描述                                                                                                                | Date             |
|-----------------------------------------|-------------------------------------------------------------------------------------------------------------------|------------------|
| Amazon ECS 主控台內的 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 <a href="#">啟動 Amazon ECS Linux 容器執行個體</a> 。                   | 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。                                                                                | 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 <a href="#">排空 Amazon ECS 容器執行個體</a> 。                                | 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 <a href="#">Amazon ECS 最佳化 Linux AMI</a> 。                                         | 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 <a href="#">使用策略來定義 Amazon ECS 任務置放</a> 。                   | 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 <a href="#">Amazon ECS 最佳化 AMI 變體</a> 。                          | 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 <a href="#">在 Amazon ECS 上為容器排程</a> 。                                        | 2016 年 12 月 1 日  |
| 適用於 CloudWatch Events 的 Amazon ECS 事件串流 | Amazon ECS 現在會將容器執行個體和任務狀態變更傳送給 CloudWatch Events。如需詳細資訊，請參閱 <a href="#">使用 EventBridge 自動化因應 Amazon ECS 錯誤</a> 。 | 2016 年 11 月 21 日 |
| Amazon ECS 容器日誌記錄到 CloudWatch Logs      | 新增了 awslogs 驅動程式的支援，以將容器日誌串流傳送至 CloudWatch Logs。如需詳細資訊，請參閱 <a href="#">將 Amazon ECS 日誌傳送至 CloudWatch</a> 。        | 2016 年 9 月 12 日  |

| 變更                                               | 描述                                                                                                                                                                    | Date             |
|--------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------|
| 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 <a href="#">使用負載平衡分佈 Amazon ECS 服務流量</a> 。 | 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 <a href="#">Amazon ECS 任務 IAM 角色</a> 。                                                              | 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 <a href="#">Amazon ECS 最佳化 Linux AMI</a> 。                                                                                             | 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 <a href="#">自動擴展您的 Amazon ECS 服務</a> 。                                                                                        | 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 <a href="#">ListTaskDefinitions</a> 。                                                                                               | 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至 CloudWatch Logs 的能力，以簡化對問題的故障診斷。                                                                                      | 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 <a href="#">Amazon ECS 最佳化 Linux AMI</a> 。                                                                       | 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 <a href="#">Amazon ECS 最佳化 Linux AMI</a> 。                                                                                              | 2015 年 12 月 22 日 |
| 叢集 CPU 和記憶體保留的 CloudWatch 指標                     | Amazon ECS 新增了 CPU 和記憶體保留的自訂 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 |

| 變更                                          | 描述                                                                                                                             | Date            |
|---------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------|-----------------|
| 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 |
| Amazon ECS 叢集和服務的 CloudWatch 指標。            | Amazon ECS 新增了叢集中每個容器執行個體、服務和任務定義系列之 CPU 和記憶體使用率的自訂 CloudWatch 指標。這些新指標可以用來使用 Auto Scaling 群組在叢集中擴展容器執行個體，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                                      | 新增了 deregisterTaskDefinition 和 runTask 之環境變數覆寫的支援。                                                                             | 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。也可以從 AWS 管理主控台 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。