使用者指南

Amazon EKS

Amazon EKS: 使用者指南

Table of Contents

..... xxii
..... xxiii
什麼是 Amazon EKS？ 1
Amazon EKS：簡化的 Kubernetes 管理 1
Amazon EKS 的功能 2
相關服務 3
Amazon EKS 定價 4
常用案例 5
架構 6
控制平台 6
運算 7
Kubernetes 概念 8
為什麼選擇 Kubernetes？ 8
叢集 12
工作負載 15
後續步驟 19
部署選項 20
了解 Amazon EKS 部署選項 20
雲端 Amazon EKS 20
資料中心或邊緣環境中的 Amazon EKS 21
Amazon EKS Anywhere 適用於氣隙隔離環境 22
Amazon EKS 工具 22
設定 23
後續步驟 23
設定 AWS CLI 23
建立存取金鑰 23
設定 AWS CLI 24
取得安全字符 24
若要驗證使用者身分 25
後續步驟 25
設定 kubectl 和 eksctl 25
安裝或更新 kubectl 26
步驟 1：檢查kubectl是否已安裝 26
步驟 2：安裝或更新 kubectl 26

安裝 eksctl 39
後續步驟 39
快速指南 40
於本教學課程中 40
先決條件 40
設定叢集 41
建立 IngressClass 41
部署 2048 遊戲範例應用程式 42
使用 Amazon EKS Auto Mode 保留資料 45
刪除叢集和節點 47
了解 Amazon EKS 48
概觀 48
Amazon EKS 研討會 48
Amazon EKS 實作叢集設定教學課程 49
Amazon EKS 範例 50
AWS 教學課程 51
開發人員研討會 51
Terraform 研討會 51
AWS Amazon EKS 訓練 51
開始使用 53
建立叢集 (EKS Auto 模式) 53
建立叢集 (eksctl) 54
先決條件 54
步驟 1：建立 Amazon EKS 叢集和節點 54
步驟 2：檢視 Kubernetes 資源 56
步驟 3：刪除您的叢集和節點 57
後續步驟 58
建立叢集 （主控台和 CLI) 58
先決條件 58
步驟 1：建立您的 Amazon EKS 叢集 59
步驟 2：將您的電腦設為與叢集進行通訊 62
步驟 3：建立節點 63
步驟 4：檢視資源 64
步驟 5：刪除資源 65
後續步驟 66
EKS 自動模式 67

功能 67
自動化元件 68
組態 69
共同責任模式 69
建立叢集 71
eksctl CLI 71
AWS CLI 73
管理主控台 80
啟用現有叢集 82
遷移參考 83
遷移 EBS 磁碟區 84
遷移負載平衡器 85
在叢集上啟用 85
從 Karpenter 遷移 88
從 MNGs遷移 92
從 Fargate 遷移 92
執行工作負載 97
部署膨脹工作負載 98
部署負載平衡器 101
部署具狀態工作負載 106
設定 111
建立節點類別 113
建立節點集區 120
建立輸入類別 126
建立服務 132
建立 StorageClass 138
停用 EKS Auto 模式 145
更新 Kubernetes 版本 146
檢閱內建節點集區 147
控制部署 149
執行關鍵附加元件 150
使用網路政策 152
標記子網路 154
部署加速工作負載 156
產生 CIS 報告 162
加密根磁碟區 165

更新 AMI 控制項 168
控制 ODCR 部署 172
運作方式 173
受管執行個體 173
身分和存取 176
聯網 181
疑難排解 184
節點監控代理程式 185
使用 EC2 CLI 從 AWS EC2 受管執行個體取得主控台輸出 185
使用偵錯容器和 CLI kubectl 取得節點日誌 186
在 AWS 主控台中檢視與 EKS Auto Mode 相關聯的資源 187
檢視您 AWS 帳戶中的 IAM 錯誤 187
故障診斷無法排程至自動模式節點的 Pod 187
對未加入叢集的節點進行故障診斷 188
跨 Pod 共用磁碟區 190
在控制平面日誌中檢視 Karpenter 事件 191
對自動模式下包含的控制器進行故障診斷 192
版本備註 192
2025 年 8 月 6 日 193
2025 年 6 月 30 日 193
2025 年 6 月 20 日 193
2025 年 6 月 13 日 193
2025 年 4 月 30 日 193
2025 年 4 月 18 日 193
2025 年 4 月 11 日 193
2025 年 4 月 4 日 194
2025 年 3 月 31 日 194
2025 年 3 月 21 日 194
2025 年 3 月 14 日 194
設定叢集 195
建立自動叢集 195
先決條件 40
建立叢集 - AWS 主控台 196
建立叢集 - AWS CLI 200
後續步驟 80
建立叢集 205

先決條件 40
步驟 1：建立叢集 IAM 角色 206
步驟 2：建立叢集 208
步驟 3：更新 kubeconfig 215
步驟 4：叢集設定 216
後續步驟 80
叢集洞察 217
叢集洞見類型 217
考量事項 217
使用案例 218
開始使用 219
檢視叢集洞察 219
更新 Kubernetes 版本 224
Amazon EKS Auto 模式的考量事項 225
Summary 225
步驟 1：準備升級 226
步驟 2：檢閱升級考量事項 226
步驟 3：更新叢集控制平面 227
步驟 4：更新叢集元件 229
降級 Amazon EKS 叢集的 Kubernetes 版本 231
刪除叢集 231
考量事項 84
刪除叢集 (eksctl) 231
刪除叢集AWS （主控台） 232
刪除叢集 (AWS CLI) 233
保護 EKS 叢集免於意外刪除 235
叢集端點存取 236
IPv6 叢集端點格式 236
IPv4 叢集端點格式 236
叢集私有端點 237
修改叢集端點存取 238
存取僅限私有 API 伺服器 239
設定端點存取 240
啟用 Windows 支援 243
考量事項 84
先決條件 40

啟用 Windows 支援 245
部署 Windows Pod 247
在 Windows 節點上支援更高的 Pod 密度 247
停用 Windows 支援 248
私有叢集 248
自動擴展 252
EKS 自動模式 252
其他解決方案 252
了解區域轉移 253
了解 Pod 之間的東西網路流量流程 253
EKS 區域轉移要求 258
常見問答集 265
其他資源 267
啟用區域轉移 268
考量事項 268
什麼是 Amazon 應用程式復原控制器？ 268
什麼是區域轉移？ 269
什麼是區域自動轉移？ 269
EKS 在自動轉移期間會做什麼？ 269
向 Amazon Application Recovery Controller (ARC) 註冊 EKS 叢集AWS （主控台） 269
後續步驟 80
管理存取 271
常見任務 271
背景介紹 154
EKS Auto 模式的考量事項 231
Kubernetes API 存取 272
將 IAM 身分與 Kubernetes 許可建立關聯 273
設定叢集身分驗證模式 274
授予許可 275
aws-auth ConfigMap 312
連結 OIDC 供應商 321
取消連結 OIDC 供應商 325
存取叢集資源 326
所需的許可 327
使用 kubectl 存取叢集 332
自動建立 kubeconfig 檔案 333

工作負載存取 AWS 334
服務帳戶字符 334
叢集附加元件 335
將 AWS Identity and Access Management 許可授予 Amazon Elastic Kubernetes Service 叢集上的工作負載 336
具有 IRSA 的登入資料 339
Pod 身分 361
管理運算 391
比較運算選項 391
受管節點群組 395
受管節點群組概念 396
受管節點群組容量類型 398
建立 400
更新 410
更新行為詳細資訊 414
啟動範本 417
Delete 431
自我管理的節點 433
Amazon Linux 434
Bottlerocket 442
Windows 445
Ubuntu Linux 453
更新方法 456
AWS Fargate 467
AWS Fargate 考量事項 468
Fargate 比較表 470
開始使用 472
定義設定檔 476
刪除設定檔 481
Pod 組態詳細資訊 482
作業系統修補事件 484
收集指標 486
日誌 488
Amazon EC2 執行個體類型 499
Amazon EKS 為每種 Amazon EC2 執行個體類型建議 Pod 數量上限 500
EKS Auto 模式的考量事項 231

預先建置AMIs 502
AL2 AMI 棄用 502
Amazon Linux 507
Bottlerocket 515
Ubuntu Linux 520
Windows 520
節點運作狀態 607
節點監控代理程式 607
節點自動修復 607
節點運作狀態問題 608
檢視節點運作狀態 616
取得節點日誌 618
混合節點 621
功能 621
限制 622
考量事項 622
其他資源 623
先決條件 623
執行混合節點 680
設定 698
運作方式 733
混合節點 nodeadm 771
故障診斷 785
應用程式資料儲存 804
Amazon EBS 804
考量事項 804
先決條件 805
步驟 1：建立 IAM 角色 805
步驟 2：取得 Amazon EBS CSI 驅動程式 813
步驟 3：部署範例應用程式 813
Amazon EFS 813
考量事項 813
先決條件 814
步驟 1：建立 IAM 角色 815
步驟 2：取得 Amazon EFS CSI 驅動程式 820
步驟 3：建立 Amazon EFS 檔案系統 821

步驟 4：部署範例應用程式 821
Amazon FSx for Lustre 821
部署驅動程式 821
最佳化 (EFA) 828
Optimize （非 EBA) 845
Amazon FSx for NetApp ONTAP 852
Amazon FSx for OpenZFS 853
Amazon File Cache 853
適用於 Amazon S3 的掛載點 853
考量事項 854
部署驅動程式 854
移除驅動程式 863
CSI 快照控制器 865
設定聯網 866
從管理主控台將現有的 VPC 子網路新增至 Amazon EKS 叢集 866
VPC 和子網要求 866
VPC 要求和注意事項 867
子網需求和注意事項 868
共用子網路需求和注意事項 873
建立 VPC 874
先決條件 40
公有和私有子網路 874
僅限公有子網路 877
只有私有子網路 878
安全群組要求 879
預設叢集安全群組 879
限制叢集流量 881
共用安全群組 882
管理聯網附加元件 882
內建附加元件 882
選用 AWS 的聯網附加元件 883
Amazon VPC CNI 884
替代 CNI 外掛程式 990
Multus 992
AWS Load Balancer控制器 993
CoreDNS 1009

kube-proxy 1025
工作負載 1030
範例部署 (Linux) 1030
先決條件 40
建立命名空間。 1031
建立 Kubernetes 部署 1031
建立服務 1032
檢閱已建立的資源 1033
在 Pod 上執行 shell 1036
後續步驟 1037
範例部署 (Windows) 1037
先決條件 40
建立命名空間。 1031
建立 Kubernetes 部署 1031
建立服務 1032
檢閱已建立的資源 1040
在 Pod 上執行 shell 1036
後續步驟 1044
Vertical Pod Autoscaler 1044
部署 Vertical Pod Autoscaler 1044
測試您的 Vertical Pod Autoscaler 安裝 1046
Horizontal Pod Autoscaler 1049
執行 Horizontal Pod Autoscaler 測試應用程式 1050
網路負載平衡 1052
先決條件 40
考量事項 84
建立 Network Load Balancer 1055
(選用) 部署範例應用程式 1057
應用程式負載平衡 1061
先決條件 40
將 ALBs與輸入群組重複使用 1063
(選用) 部署範例應用程式 1064
限制服務外部 IPs 1067
將映像複製到儲存庫 1069
檢視 Amazon 映像登錄檔 1072
Amazon EKS 附加元件 1075

考量事項 1076
Amazon EKS Auto 模式的考量事項 1077
支援 1078
AWS 附加元件 1079
社群附加元件 1094
Marketplace 附加元件 1100
建立附加元件 1115
更新 附加元件 1126
驗證相容性 1132
移除附加元件 1134
IAM 角色 1137
您可以自訂的欄位 1143
驗證容器映像 1146
叢集管理 1148
成本監控 1148
依 Pod 檢視成本 1149
安裝 Kubecost 1149
存取 Kubecost 儀表板 1150
進一步了解 Kubecost 1151
指標伺服器 1159
考量事項 84
使用 Amazon EKS 附加元件部署為社群附加元件 1159
使用資訊清單部署 1160
使用 Helm 部署應用程式 1161
標記您的 資源 1162
標籤基本概念 1163
標記您的 資源 1163
標籤限制 1164
標記您的資源以便計費 1164
透過主控台使用標籤 1165
透過 CLI、API 或 eksctl 使用標籤 1166
Service Quotas 1167
在 中檢視 EKS 服務配額 AWS Management Console 1167
使用 CLI 檢視 EKS AWS 服務配額 1168
Amazon EKS 服務配額 1169
AWS Fargate 服務配額 1169

安全 1171
最佳實務 1172
分析漏洞 1172
Amazon EKS 的網際網路安全中心 (CIS) 基準測試 1172
Amazon EKS 平台版本 1173
作業系統漏洞清單 1173
使用 Amazon Inspector 進行節點偵測 1173
使用 Amazon GuardDuty 進行叢集和節點偵測 1173
驗證合規性 1173
EKS 的考量事項 1174
基礎架構安全 1174
恢復能力 1178
Kubernetes 的考量事項 1179
憑證簽署 1179
預設 角色和使用者 1181
啟用秘密加密 1186
AWS Secrets Manager 1190
所有 Kubernetes API 資料的預設信封加密 1191
EKS Auto 的考量 1196
API 安全性和身分驗證 1197
網路安全 1197
EC2 受管執行個體安全性 1197
自動化資源管理 1199
安全最佳實務 1199
IAM 參考 1200
目標對象 1200
使用身分驗證 1200
使用政策管理存取權 1203
Amazon EKS 和 IAM 1204
身分型政策 1208
服務連結角色 1215
Pod 執行 IAM 角色 1228
連接器 IAM 角色 1233
AWS 受管政策 1237
故障診斷 1257
叢集 IAM 角色 1260

節點 IAM 角色 1263
自動模式叢集 IAM 角色 1268
自動模式節點 IAM 角色 1271
監控叢集 1275
在 Amazon EKS 上監控和記錄 1275
Amazon EKS 監控和記錄工具 1276
可觀測性儀表板 1279
Summary 1279
叢集運作狀態 1280
控制平面監控 1280
叢集洞察 1282
節點運作狀態問題 1282
Prometheus 指標 1282
步驟 1：開啟 Prometheus 指標 1283
步驟 2：使用 Prometheus 指標 1285
步驟 3：管理 Prometheus 抓取器 1285
使用 Helm 部署 1285
控制平台 1288
Amazon CloudWatch 1295
Amazon CloudWatch 中的基本指標 1295
Amazon CloudWatch 可觀測性運算子 1302
控制平面日誌 1302
啟用或停用控制平面日誌 1304
檢視叢集控制平面日誌 1306
AWS CloudTrail 1307
參考 1307
日誌檔案項目 1308
Auto Scaling 群組指標 1311
ADOT 運算子 1311
Amazon EKS 儀表板 1311
什麼是 Amazon EKS 儀表板？ 1312
儀表板如何使用 AWS Organizations？ 1312
使用 AWS 主控台啟用 EKS 儀表板 1313
檢視 EKS 儀表板 1314
設定儀表板 1314
使用 AWS 主控台停用 EKS 儀表板 1315

EKS 儀表板故障診斷 1316
設定組織 1316
使用其他 服務 1322
AWS CloudFormation 1322
Amazon EKS 和 AWS CloudFormation 範本 1322
進一步了解 AWS CloudFormation 1323
Amazon Detective 1323
將 Amazon Detective 與 Amazon EKS 搭配使用 1323
Amazon GuardDuty 1324
AWS 彈性中樞 1325
Amazon Security Lake 1325
搭配 Amazon EKS 使用 Security Lake 的優勢 1325
啟用 Security Lake for Amazon EKS 1326
在 Security Lake 中分析 EKS 日誌 1326
Amazon VPC Lattice 1326
AWS 本機區域 1326
Amazon EKS 連接器 1328
Amazon EKS 連接器考量事項 1328
Amazon EKS 連接器的必要 IAM 角色 1329
連接叢集 1329
考量事項 1329
先決條件 1329
步驟 1：註冊叢集 1330
步驟 2：安裝 eks-connector 代理程式 1332
後續步驟 1334
授予對叢集的存取權 1334
先決條件 1334
取消註冊叢集 1336
取消註冊 Kubernetes 叢集 1336
清除 Kubernetes 叢集中的資源 1337
對 EKS 連接器進行故障診斷 1338
基本疑難排解 1338
Helm 問題：403 禁止 1340
主控台錯誤：叢集停留在待定狀態 1340
主控台錯誤：使用者系統：serviceaccount：eks-connector：eks-connector 無法在叢集範圍內模擬 API 群組中的資源使用者 1340

主控台錯誤：【... 】 被禁止：使用者 【... 】 無法在叢集範圍內列出 API 群組中的資源 【... 】 1341
主控台錯誤：Amazon EKS 無法與您的 Kubernetes 叢集 API 伺服器通訊。叢集必須處於作用中狀態，才能成功連接。請過幾分鐘後再試。 1341
Amazon EKS 連接器 Pod 正在損毀迴圈 1342
無法啟動 eks-connector： InvalidActivation 1342
叢集節點遺漏對外連線 1343
Amazon EKS 連接器 Pod 處於 ImagePullBackOff 狀態 1343
常見問答集 1344
安全考量 1345
AWS 責任 1345
客戶責任 1345
AWS Outposts 上的 Amazon EKS 1347
使用每個部署選項的時機 1347
比較部署選項 1348
執行本機叢集 1350
支援 AWS 的區域 1350
部署本機叢集 1351
EKS 平台版本 1360
建立 VPC 和子網路 1370
準備中斷連線 1373
容量考量事項 1377
對叢集進行疑難排解 1379
節點 1387
eksctl 1389
AWS Management Console 1390
EKS 上的 AI/ML 1396
為什麼選擇適用於 AI/ML 的 EKS？ 1396
金鑰使用案例 1396
案例研究 1397
在 EKS 上開始使用Machine Learning 1398
即時推論 1398
建立叢集 1398
叢集組態 1431
設定 Linux GPU AMIs 1432
設定 Windows GPU AMIs 1433

使用 EFA 設定訓練叢集 1439
使用 Inferentia 設定推論叢集 1449
容量管理 1454
為受管節點群組保留 GPUs 1455
為自我管理節點保留 GPUs 1457
配方 1460
防止在特定節點上排程 Pod 1460
資源 1462
研討會 1462
最佳實務 1463
參考架構 1464
教學課程 1464
版本控制 1466
Kubernetes 版本 1466
標準支援的可用版本 1466
延長支援的可用版本 1467
Amazon EKS Kubernetes 發佈日曆 1467
使用 CLI AWS 取得版本資訊 1468
Amazon EKS 版常見問題 1469
Amazon EKS 延伸支援FAQs 1471
標準支援版本 1473
延長支援版本 1477
檢視支援期間 1480
檢視升級政策 1481
啟用延伸支援 1482
停用延伸支援 1483
平台版本 1484
Kubernetes 版本 1.33 1485
Kubernetes 版本 1.32 1486
Kubernetes 版本 1.31 1488
Kubernetes 版本 1.30 1490
Kubernetes 版本 1.29 1494
Kubernetes 版本 1.28 1498
Kubernetes 版本 1.27 1502
Kubernetes 版本 1.26 1507
取得目前的平台版本 1512

變更平台版本 1512
故障診斷 1513
容量不足 1513
節點無法加入叢集 1513
未經授權或存取遭拒 (kubectl) 1515
hostname doesn't match 1516
getsockopt: no route to host 1516
Instances failed to join the Kubernetes cluster 1516
受管節點群組錯誤代碼 1516
Not authorized for images 1521
節點處於 NotReady 狀態 1521
EKS 日誌收集器 1521
容器執行階段網路尚未就緒 1522
TLS 交握逾時 1523
InvalidClientTokenId 1524
節點群組必須符合 Kubernetes 版本，才能升級控制平面 1524
啟動許多節點時，會出現 Too Many Requests 錯誤 1525
針對 Kubernetes API 伺服器請求回應的 HTTP 401 未經授權的錯誤 1525
Amazon EKS 平台版本比目前平台版本落後兩個版本以上 1525
叢集運作狀態FAQs和具有解析路徑的錯誤代碼 1528
與 Amazon EKS 相關的專案 1533
支援部署到 EKS 的軟體 1533
管理工具 1534
eksctl 1534
AWS 適用於 Kubernetes 的 控制器 1534
Flux CD 1534
Kubernetes 專用 CDK 1534
聯網 1535
Kubernetes 專用 Amazon VPC CNI 外掛程式 1535
Kubernetes 的AWS Load Balancer控制器 1535
ExternalDNS 1535
機器學習 1535
Kubeflow 1535
Auto Scaling 1536
Cluster Autoscaler 1536
Karpenter 1536

Escalator 1536
監控 1536
Prometheus 1537
持續整合 / 持續部署 1537
Jenkins X 1537
新功能和藍圖 1538
文件歷史紀錄 1539
貢獻 1579
編輯單一頁面 1579
先決條件 40
程序 87
提取請求概觀 1581
使用 GitHub 編輯檔案 1582
先決條件 40
程序 87
檢視風格意見回饋 1583
安裝 Vale 1584
安裝 VS Code Vale 延伸模組 1584
同步 Vale 1585
在 VS 程式碼中檢視樣式意見回饋 1585
建立頁面 1585
建立頁面 1585
將頁面新增至導覽 1586
插入連結 1586
EKS 使用者指南中的頁面或區段連結 1586
連結至 AWS 文件中的其他指南 1587
外部網頁的連結 1587
使用 Amazon Q 建立 1587
使用 VS 程式碼安裝 Amazon Q 1588
登入 Amazon Q 1588
使用 Amazon Q 建立內容 1589
提示 1589
檢視 PR 預覽 1589
檢視預覽 1590
預覽限制 1591
AsciiDoc 語法 1591

新頁面 1591
標題 1591
文字格式 1592
清單 1592
連結 1592
程式碼範例 1593
映像 1593
資料表 1593
標註 1594
包含其他檔案 1594
屬性 （類似於實體） 1594
程序 1595

協助改善此頁面

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。

若要提供此使用者指南，請選擇位於每個頁面右窗格中的在 GitHub 上編輯此頁面連結。

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。

什麼是 Amazon EKS？

Amazon EKS：簡化的 Kubernetes 管理

Amazon Elastic Kubernetes Service (EKS) 提供全受管 Kubernetes 服務，可消除操作 Kubernetes 叢集的複雜性。透過 EKS，您可以：

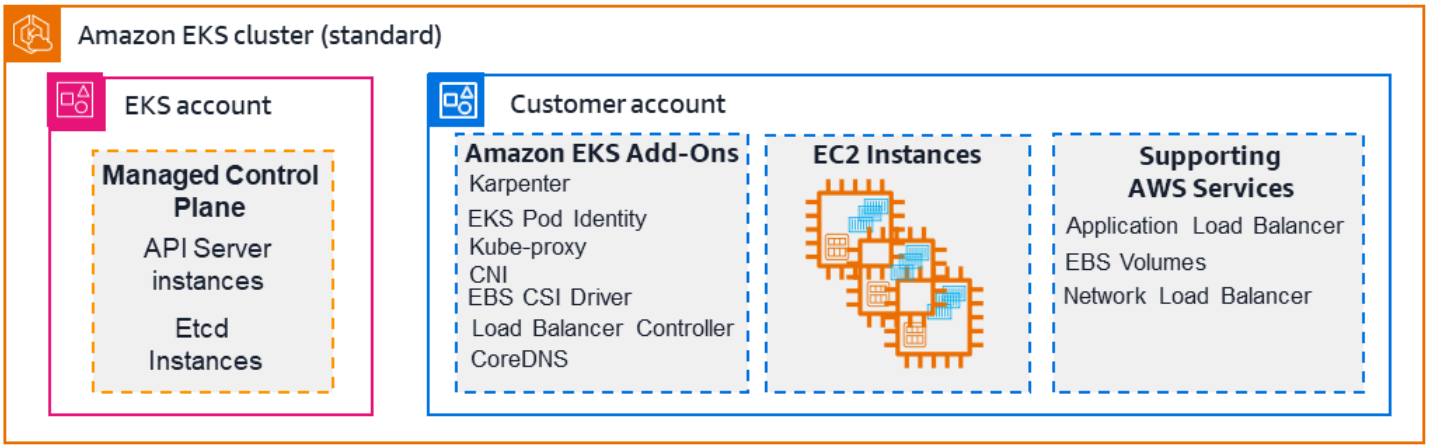
- 以較低的營運開銷更快速地部署應用程式
- 無縫擴展以滿足不斷變化的工作負載需求
- 透過 AWS 整合和自動更新提高安全性
- 選擇標準 EKS 或全自動 EKS Auto 模式

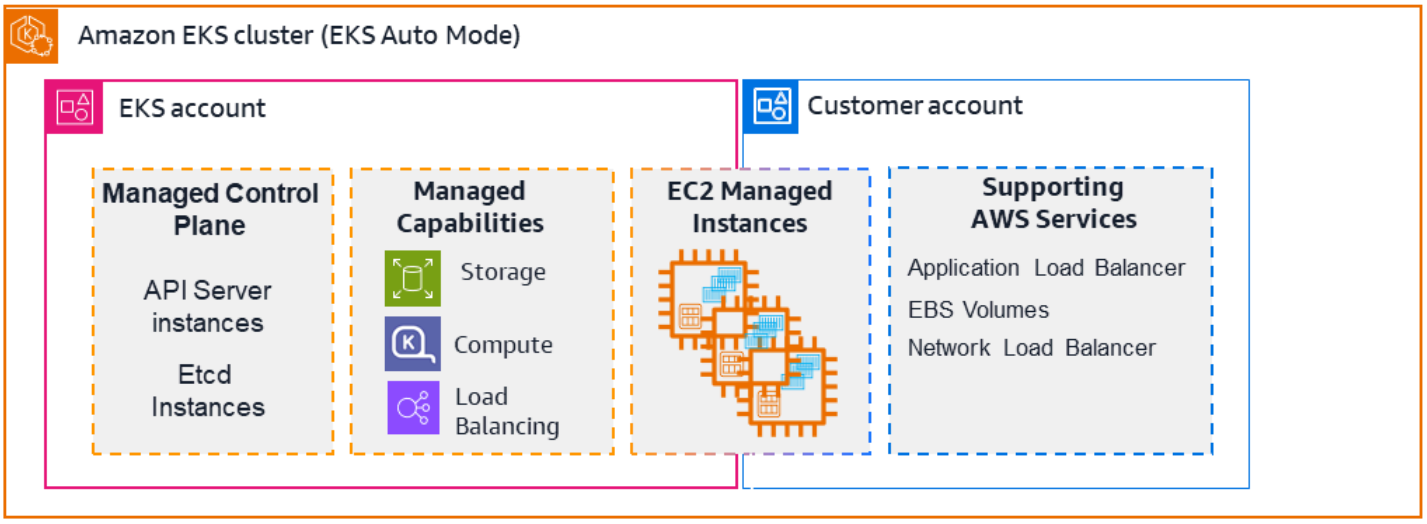
Amazon Elastic Kubernetes Service (Amazon EKS) 是在 Amazon Web Services (AWS) 雲端和您自己的資料中心 (EKS Anywhere 和 Amazon EKS 混合節點) 中執行 Kubernetes 叢集的首要平台。

Amazon EKS 可簡化 Kubernetes 叢集的建置、保護和維護。與維護您自己的資料中心相比，提供足夠的資源來滿足尖峰需求可能更具成本效益。使用 Amazon EKS 的兩種主要方法如下：

- 當您使用 EKS 建立叢集時，EKS standard： AWS 管理 Kubernetes 控制平面。系統會自動為您處理管理節點、排程工作負載、與 AWS 雲端整合，以及存放和擴展控制平面資訊的元件，以保持叢集正常運作。
- EKS 自動模式：EKS 使用 EKS 自動模式功能，擴展其控制以管理節點 (Kubernetes 資料平面）。它透過自動佈建基礎設施、選取最佳運算執行個體、動態擴展資源、持續最佳化成本、修補作業系統，以及與 AWS 安全服務整合，來簡化 Kubernetes 管理。

下圖說明 Amazon EKS 如何根據您選擇的叢集建立方法，將 Kubernetes 叢集與 AWS 雲端整合：





Amazon EKS 可協助您加速生產時間、改善效能、可用性和彈性，並增強系統安全性。如需詳細資訊，請參閱 Amazon Elastic Kubernetes Service。

Amazon EKS 的功能

Amazon EKS 提供下列高階功能：

管理介面

EKS 提供多個介面來佈建、管理和維護叢集，包括 AWS Management Console Amazon EKS API/SDKs、CDK、 AWS CLI、eksctl CLI、 AWS CloudFormation 和 Terraform。如需詳細資訊，請參閱開始使用及設定叢集。

存取控制工具

EKS 依賴 Kubernetes 和 AWS Identity and Access Management (AWS IAM) 功能來管理來自使用者和工作負載的存取。如需詳細資訊，請參閱the section called “Kubernetes API 存取”及the section called “工作負載存取 AWS ”。

運算資源

對於運算資源，EKS 允許完整範圍的 Amazon EC2 執行個體類型和 AWS 創新，例如 Nitro 和 Graviton 搭配 Amazon EKS，讓您最佳化工作負載的運算。如需詳細資訊，請參閱管理運算。

儲存

EKS Auto Mode 會使用 EBS 磁碟區自動建立儲存體方案。使用容器儲存介面 (CSI) 驅動程式，您也可以使用 Amazon S3、Amazon EFS、Amazon FSX 和 Amazon File Cache 滿足您的應用程式儲存需求。如需詳細資訊，請參閱應用程式資料儲存。

安全性

採用共同責任模型，因為它與 Amazon EKS 中的安全性相關。如需詳細資訊，請參閱安全性最佳實務、基礎設施安全性和 Kubernetes 安全性。

監控工具

使用可觀測性儀表板來監控 Amazon EKS 叢集。監控工具包括 Prometheus、CloudWatch、Cloudtrail 和 ADOT Operator。如需儀表板、指標伺服器和其他工具的詳細資訊，請參閱 EKS 叢集成本和 Kubernetes 指標伺服器。

Kubernetes 相容性和支援

Amazon EKS 已通過 Kubernetes 合規認證，因此您可以部署與 Kubernetes 相容的應用程式，而無需重構和使用 Kubernetes 社群工具和外掛程式。EKS 為 Kubernetes 提供標準支援和 eks/latest/userguide/kubernetes-versions-extended.html【extended support，type="documentation"】。如需詳細資訊，請參閱 eks/latest/userguide/kubernetes-versions.html【Understand the Kubernetes version lifecycle on EKS，type="documentation"】。

相關服務

要與 Amazon EKS 搭配使用的服務

您可以使用其他 AWS 服務搭配您使用 Amazon EKS 部署的叢集：

Amazon EC2

使用 Amazon EC2 取得隨需、可擴展的運算容量。

Amazon EBS

使用 Amazon EBS 連接可擴展、高效能的區塊儲存資源。

Amazon ECR

使用 Amazon ECR 安全地存放容器映像。

Amazon CloudWatch

使用 Amazon CloudWatch 即時監控 AWS 資源和應用程式。

Amazon Prometheus

使用 Amazon Managed Service for Prometheus 追蹤容器化應用程式的指標。

Elastic Load Balancing

使用 Elastic Load Balancing 將傳入流量分散到多個目標。

Amazon GuardDuty

使用 Amazon GuardDuty 偵測 EKS 叢集的威脅。

AWS 彈性中樞

使用 AWS Resilience Hub 評估 EKS 叢集彈性。

Amazon EKS 定價

Amazon EKS 根據 Kubernetes 叢集版本支援、Amazon EKS Auto Mode 定價，以及 Amazon EKS 混合節點的每個 vCPU 定價，提供每個叢集定價。

使用 Amazon EKS 時，您需要另外支付用於在 Kubernetes 工作者節點上執行應用程式 AWS 的資源。例如，如果您以具有 Amazon EBS 磁碟區和公有 IPv4 地址的 Amazon EC2 執行個體身分執行 Kubernetes 工作者節點，您需要支付透過 Amazon EC2 的執行個體容量、透過 Amazon EBS 的磁碟區容量，以及透過 Amazon VPC 的 IPv4 地址的費用。 IPv4

請造訪您搭配 Kubernetes 應用程式使用 AWS 之服務的個別定價頁面，以取得詳細的定價資訊。

- 如需 Amazon EKS 叢集、Amazon EKS Auto 模式和 Amazon EKS 混合節點定價，請參閱 Amazon EKS 定價。

- 如需 Amazon EC2 定價，請參閱 Amazon EC2 隨需定價和 Amazon EC2 Spot 定價。
- 如需 AWS Fargate 定價，請參閱 AWS Fargate 定價。
- 您可以使用節省成本計劃來計算 Amazon EKS 叢集中使用的運算。如需詳細資訊，請參閱透過 Savings Plans 定價。

Amazon EKS 中的常用案例

Amazon EKS 在 上提供強大的受管 Kubernetes 服務 AWS，旨在最佳化容器化應用程式。以下是一些 Amazon EKS 最常見的使用案例，協助您充分利用其強項來滿足您的特定需求。

部署高可用性應用程式

使用 Elastic Load Balancing，您可以確保您的應用程式在多個可用區域之中均高度可用。

建置微服務架構

搭配 AWS Cloud Map 或 Amazon VPC Lattice 使用 Kubernetes 服務探索功能來建置彈性系統。

自動化軟體版本程序

管理持續整合和持續部署 (CICD) 管道，這些管道可簡化應用程式的自動化建置、測試和部署程序。

執行無伺服器應用程式

使用 AWS Fargate 搭配 Amazon EKS 執行無伺服器應用程式。這表示您可以專注於應用程式開發，而 Amazon EKS 和 Fargate 則可以處理基礎結構。

執行機器學習工作負載

Amazon EKS 與 TensorFlow、MXNet 和 PyTorch 等熱門機器學習架構相容。有了 GPU 支援，您甚至可以有效地處理複雜的機器學習任務。

在內部部署和雲端中一致地部署

若要簡化內部部署環境中的 Kubernetes 執行，您可以使用相同的 Amazon EKS 叢集、功能和工具在 AWS Outpost 上執行自我管理節點，也可以使用 Amazon EKS 混合節點搭配您自己的基礎設施。對於獨立氣隙隔離的環境，您可以使用 Amazon EKS Anywhere 在您自己的基礎設施上自動化 Kubernetes 叢集生命週期管理。

執行符合成本效益的批次處理和大數據工作負載

利用 Spot 執行個體執行批次處理和大數據工作負載，例如 Apache Hadoop 和 Spark，只需一小部分成本。這可讓您以折扣價格充分利用未使用的 Amazon EC2 容量。

保護應用程式並確保合規

實作強大的安全實務，並維持與 Amazon EKS 的合規性，Amazon EKS 與 AWS Identity and Access Management (IAM)、Amazon Virtual Private Cloud (Amazon VPC) 和 AWS Key Management Service (AWS KMS) 等 AWS 安全服務整合。這確保了資料隱私權和保護均依照產業標準實行。

Amazon EKS 架構

Amazon EKS 符合 Kubernetes 的一般叢集架構。如需詳細資訊，請參閱 Kubernetes 文件中的 Kubernetes 元件。下列各節摘要了 Amazon EKS 一些額外架構的詳細資訊。

控制平台

Amazon EKS 可確保每個叢集都有自己的唯一 Kubernetes 控制平面。此設計可讓每個叢集的基礎設施保持個別，而不會在叢集或 AWS 帳戶之間重疊。設定包含：

分散式元件

控制平面會在 AWS 區域內的三個 AWS 可用區域中，定位至少兩個 API 伺服器執行個體和三個等化執行個體。

最佳效能

Amazon EKS 會主動監控和調整控制平面執行個體，以維持尖峰效能。

彈性

如果控制平面執行個體動搖，如有需要，Amazon EKS 會使用其他可用區域迅速取代它。

一致的執行時間

透過在多個可用區域之中執行叢集的方式來達成可靠的 API 伺服器端點可用性服務水準協議 (SLA)。

Amazon EKS 會使用 Amazon Virtual Private Cloud (Amazon VPC) 來限制單一叢集內控制平面元件之間的流量。叢集元件無法檢視或接收來自其他叢集或 AWS 帳戶的通訊，除非 Kubernetes 角色型存取控制 (RBAC) 政策授權。

運算

除了控制平面之外，Amazon EKS 叢集還有一組稱為節點的工作者機器。選取適當的 Amazon EKS 叢集節點類型，對於滿足您特定的需求與最佳化資源使用率而言十分關鍵。Amazon EKS 提供下列主節點類型：

EKS 自動模式

EKS Auto Mode 將 AWS 管理延伸到控制平面之外，以包含資料平面，自動化叢集基礎設施管理。它將核心 Kubernetes 功能整合為內建元件，包括運算自動擴展、聯網、負載平衡、DNS、儲存和 GPU 支援。EKS Auto Mode 會根據工作負載需求動態管理節點，使用不可變 AMIs 搭配增強型安全功能。它會在遵守 Pod 中斷預算時自動化更新和升級，並包含原本需要附加元件管理的受管元件。此選項適合希望利用day-to-day操作 AWS 專業知識、將營運開銷降至最低，並專注於應用程式開發而非基礎設施管理的使用者。

AWS Fargate

Fargate 是適用於容器的無伺服器運算引擎，無需管理基礎執行個體。使用 Fargate，您可以指定應用程式的資源需求，並 AWS 自動佈建、擴展和維護基礎設施。此選項非常適合優先考慮簡單易用的使用者，也相當適合希望專注於應用程式開發和部署，而不是管理基礎結構的使用者。

Karpenter

Karpenter 是一種彈性、高效能的 Kubernetes 叢集自動擴展器，可協助改善應用程式可用性和叢集效率。Karpenter 會啟動適當大小的運算資源，以因應應用程式負載的變化。此選項可佈建符合工作負載要求的即時運算資源。

受管節點群組

受管節點群組是自動化和自訂的混合，用於管理 Amazon EKS 叢集中的 Amazon EC2 執行個體集合。 AWS 負責修補、更新和擴展節點等任務，從而減輕操作層面。同時，也支援自訂 `kubelet` 引數，為進階 CPU 和記憶體管理政策開啟了可能性。此外，它們透過服務帳戶的 AWS Identity and Access Management (IAM) 角色來增強安全性，同時減少每個叢集個別許可的需求。

自我管理節點

自我管理節點提供對 Amazon EKS 叢集中的 Amazon EC2 執行個體的完全控制權。您負責管理、擴展和維護節點，讓您完全掌握基礎結構。此選項適合需要精細控制和自訂其節點的使用者，也適合已經準備好投入時間來管理和維護其基礎架構的使用者。

Amazon EKS 混合節點

透過 Amazon EKS 混合節點，您可以使用內部部署和邊緣基礎設施做為 Amazon EKS 叢集中的節點。Amazon EKS 混合節點整合跨環境的 Kubernetes 管理，並 AWS 針對您的內部部署和邊緣應用程式將 Kubernetes 控制平面管理卸載至 。

Kubernetes 概念

Amazon Elastic Kubernetes Service (Amazon EKS) 是以開放原始碼 Kubernetes 專案為基礎的 AWS 受管服務。雖然您需要了解 Amazon EKS 服務如何與 AWS Cloud 整合 （特別是當您首次建立 Amazon EKS 叢集時），但一旦 Amazon EKS 叢集啟動並執行，您使用 Amazon EKS 叢集的方式就跟任何其他 Kubernetes 叢集一樣。因此，若要開始管理 Kubernetes 叢集和部署工作負載，您至少需要對 Kubernetes 概念有基本的了解。

此頁面將 Kubernetes 概念分為三個部分：the section called “為什麼選擇 Kubernetes？”、 the section called “叢集”和 the section called “工作負載”。第一節說明執行 Kubernetes 服務的價值，特別是 Amazon EKS 等受管服務。工作負載區段涵蓋如何建置、存放、執行和管理 Kubernetes 應用程式。叢集區段會列出組成 Kubernetes 叢集的不同元件，以及您建立和維護 Kubernetes 叢集的責任。

主題

- 為什麼選擇 Kubernetes？
- 叢集
- 工作負載
- 後續步驟

當您瀏覽此內容時，連結將引導您進一步描述 Amazon EKS 和 Kubernetes 文件中的 Kubernetes 概念，以防您想要深入了解我們在此涵蓋的任何主題。如需 Amazon EKS 如何實作 Kubernetes 控制平面和運算功能的詳細資訊，請參閱 the section called “架構”。

為什麼選擇 Kubernetes？

Kubernetes 旨在改善執行關鍵任務、生產品質容器化應用程式的可用性和可擴展性。Kubernetes 不只是在單一機器上執行 Kubernetes （雖然可以），而是可讓您跨可擴展或收縮以滿足需求的一組電腦執行應用程式來實現這些目標。Kubernetes 包含的功能可讓您更輕鬆地：

- 在多部機器上部署應用程式 （使用在 Pod 中部署的容器）

- 監控容器運作狀態並重新啟動失敗的容器
- 根據負載向上和向下擴展容器
- 使用新版本更新容器
- 在容器之間配置資源
- 平衡跨機器的流量

讓 Kubernetes 自動化這些類型的複雜任務，可讓應用程式開發人員專注於建置和改善其應用程式工作負載，而不必擔心基礎設施。開發人員通常會建立組態檔案，格式為 YAML 檔案，描述應用程式的所需狀態。這可能包括要執行的容器、資源限制、Pod 複本數量、CPU/記憶體配置、親和性規則等。

Kubernetes 的屬性

為了實現其目標，Kubernetes 具有下列屬性：

- Containerized - Kubernetes 是一種容器協同運作工具。若要使用 Kubernetes，您必須先將應用程式容器化。視應用程式類型而定，這可以是一組微服務、批次任務或其他形式。然後，您的應用程式可以利用包含巨大工具生態系統的 Kubernetes 工作流程，其中容器可以儲存為容器登錄檔中的映像，部署到 Kubernetes 叢集，並在可用的節點上執行。您可以在本機電腦上使用 Docker 或其他容器執行時間建置和測試個別容器，然後再將其部署到您的 Kubernetes 叢集。
- 可擴展 — 如果應用程式的需求超過這些應用程式執行中執行個體的容量，Kubernetes 可以向上擴展。Kubernetes 可以視需要判斷應用程式是否需要更多 CPU 或記憶體，並透過自動擴展可用容量或使用更多現有容量來回應。如果有足夠的運算可以執行更多應用程式執行個體 (水平 Pod Autoscaling)，或在節點層級執行擴展，如果需要增加更多節點來處理增加的容量 (Cluster Autoscaler 或 Karpenter)，則可以在 Pod 層級完成擴展。由於不再需要容量，這些服務可以刪除不必要的 Pod 並關閉不需要的節點。
- 可用：如果應用程式或節點運作狀態不佳或無法使用，Kubernetes 可以將執行中的工作負載移至另一個可用的節點。您只需刪除工作負載的執行中執行個體或執行工作負載的節點，即可強制問題發生。這裡的底線是，如果工作負載無法再於其他位置執行，則可以在其他位置提出工作負載。
- 宣告：Kubernetes 使用主動對帳來持續檢查您為叢集宣告的狀態是否符合實際狀態。例如，透過將 Kubernetes 物件套用至叢集，通常透過 YAML 格式的組態檔案，您可以要求 啟動要在叢集上執行的工作負載。您可以稍後將組態變更為執行某些動作，例如使用較新版本的容器或配置更多記憶體。Kubernetes 會執行建立所需狀態所需的動作。這可能包括啟動或關閉節點、停止和重新啟動工作負載，或提取更新的容器。
- 可編寫 - 由於應用程式通常由多個元件組成，因此您希望能夠同時管理一組這些元件（通常由多個容器表示）。雖然 Docker Compose 提供直接使用 Docker 的方法，但 Kubernetes Kompose 命令

可協助您使用 Kubernetes 執行此操作。如需如何執行此操作的範例，請參閱將 Docker Compose 檔案轉換為 Kubernetes 資源。

- 可擴展 - 與專屬軟體不同，開放原始碼 Kubernetes 專案旨在讓您以任何您想要的方式擴展 Kubernetes，以滿足您的需求。APIs和組態檔案開放直接修改。建議第三方編寫自己的控制器，以擴展基礎設施和最終使用者 Kubernetes 功能。Webhook 可讓您設定叢集規則，以強制執行政策並適應不斷變化的條件。如需如何擴展 Kubernetes 叢集的更多想法，請參閱擴展 Kubernetes。
- 可攜式 - 許多組織已在 Kubernetes 上標準化其操作，因為它允許他們以相同的方式管理其所有應用程式需求。開發人員可以使用相同的管道來建置和存放容器化應用程式。然後，這些應用程式可以部署到內部部署、雲端、餐廳point-of-sales終端機或分散在公司遠端網站的 IOT 裝置上執行的 Kubernetes 叢集。其開放原始碼性質可讓人員開發這些特殊的 Kubernetes 分佈，以及管理它們所需的工具。

管理 Kubernetes

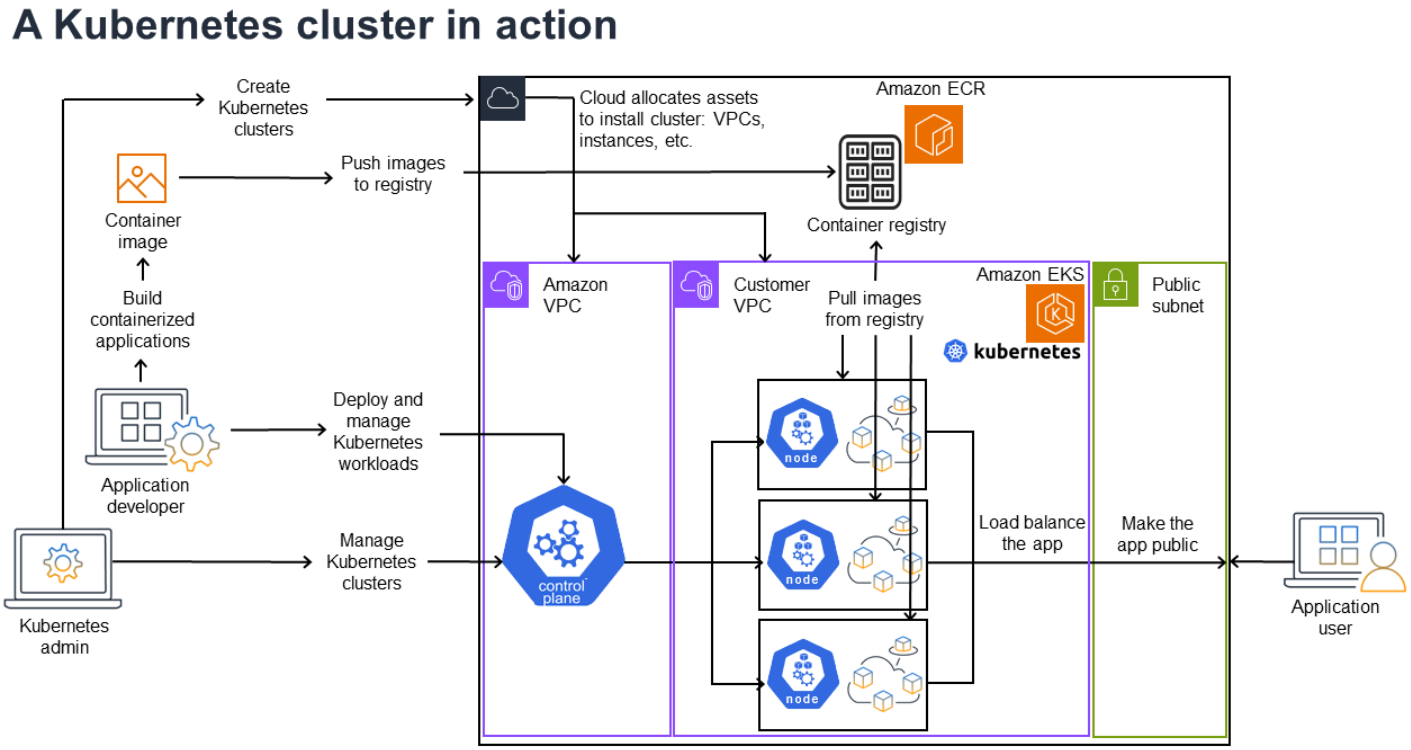
Kubernetes 原始程式碼是免費的，因此您可以使用自己的設備自行安裝和管理 Kubernetes。不過，自我管理 Kubernetes 需要深入的營運專業知識，並需要時間和精力來維護。基於這些原因，大多數部署生產工作負載的人員都會選擇雲端供應商 （例如 Amazon EKS) 或內部部署供應商 （例如 Amazon EKS Anywhere)，並擁有其經過測試的 Kubernetes 分佈和 Kubernetes 專家的支援。這可讓您卸載維護叢集所需的大部分未差異化繁重工作，包括：

- 硬體：如果您沒有可根據您的需求執行 Kubernetes 的硬體，Amazon EKS AWS 等雲端供應商可以節省您的預付成本。使用 Amazon EKS，這表示您可以使用 提供的最佳雲端資源 AWS，包括電腦執行個體 (Amazon Elastic Compute Cloud)、您自己的私有環境 (Amazon VPC)、中央身分和許可管理 (IAM) 和儲存 (Amazon EBS)。 AWS 管理電腦、網路、資料中心，以及執行 Kubernetes 所需的所有其他實體元件。同樣地，您不需要規劃資料中心來處理需求最高的天數的最大容量。對於 Amazon EKS Anywhere 或其他內部部署 Kubernetes 叢集，您需負責管理 Kubernetes 部署中使用的基礎設施，但您仍然可以依賴 AWS 來協助您將 Kubernetes 保持在最新狀態。
- 控制平面管理 - Amazon EKS 管理 AWS託管 Kubernetes 控制平面的安全性和可用性，負責排程容器、管理應用程式的可用性和其他關鍵任務，讓您可以專注於應用程式工作負載。如果您的叢集中斷， AWS 應該具有將叢集還原為執行中狀態的方法。對於 Amazon EKS Anywhere，您可以自行管理控制平面。
- 已測試的升級 - 升級叢集時，您可以依賴 Amazon EKS 或 Amazon EKS Anywhere 來提供其 Kubernetes 分佈的已測試版本。
- 附加元件 - 建置數百個專案來擴展和使用 Kubernetes，您可以將這些專案新增至叢集的基礎設施，或使用這些專案來協助工作負載的執行。 AWS 提供您可以與叢集搭配使用the section called “Amazon EKS 附加元件”的 ，而不是自行建置和管理這些附加元件。Amazon EKS Anywhere 提

供託管套件，其中包含許多熱門開放原始碼專案的建置。因此，您不需要自行建置軟體或管理重要的安全修補程式、錯誤修正或升級。同樣地，如果預設值符合您的需求，則通常只需要極少的這些附加元件組態。the section called “擴展叢集” 如需使用附加元件擴展叢集的詳細資訊，請參閱 。

Kubernetes 運作中

下圖顯示您身為 Kubernetes Admin 或 Application Developer 要建立和使用 Kubernetes 叢集的關鍵活動。在此過程中，它會說明 Kubernetes 元件如何彼此互動，使用 AWS 雲端做為基礎雲端供應商的範例。



Kubernetes Admin 會使用要建置叢集的提供者類型專用的工具來建立 Kubernetes 叢集。此範例使用 AWS 雲端做為提供者，提供稱為 Amazon EKS 的受管 Kubernetes 服務。受管服務會自動配置建立叢集所需的資源，包括為叢集建立兩個新的虛擬私有雲端 (Amazon VPCs)、設定聯網，以及將 Kubernetes 許可直接映射至新的 VPCs以進行雲端資產管理。受管服務也會看到控制平面服務具有執行和配置零個或多個 Amazon EC2 執行個體做為執行工作負載的 Kubernetes 節點的位置。 AWS 會管理控制平面的一個 Amazon VPC 本身，而另一個 Amazon VPC 包含執行工作負載的客戶節點。

許多 Kubernetes Admin 接下來的任務都是使用 Kubernetes 工具來完成，例如 `kubectl`。該工具會直接向叢集的控制平面提出服務請求。然後，對叢集進行查詢和變更的方式與您在任何 Kubernetes 叢集上進行查詢和變更的方式非常類似。

想要將工作負載部署到此叢集的應用程式開發人員可以執行數個任務。開發人員需要將應用程式建置到一或多個容器映像中，然後將這些映像推送到 Kubernetes 叢集可存取的容器登錄檔。 為該目的 AWS 提供 Amazon Elastic Container Registry (Amazon ECR)。

若要執行應用程式，開發人員可以建立 YAML 格式的組態檔案，告訴叢集如何執行應用程式，包括要從登錄檔提取哪些容器，以及如何在 Pod 中包裝這些容器。控制平面 （排程器） 會將容器排程至一或多個節點，而每個節點上的容器執行時間實際上會提取並執行所需的容器。開發人員也可以設定應用程式負載平衡器，以平衡每個節點上執行之可用容器的流量，並公開應用程式，使其可在公有網路上供外界使用。完成後，想要使用應用程式的人可以連線到應用程式端點來存取它。

從 Kubernetes 叢集和工作負載的角度來看，以下各節說明這些功能的詳細資訊。

叢集

如果您的任務是啟動和管理 Kubernetes 叢集，您應該知道如何建立、增強、管理和刪除 Kubernetes 叢集。您也應該知道組成叢集的元件是什麼，以及您需要做什麼來維護這些元件。

用於管理叢集的工具可處理 Kubernetes 服務和基礎硬體提供者之間的重疊。因此，Kubernetes 供應商 （例如 Amazon EKS 或 Amazon EKS Anywhere) 通常會使用供應商專用的工具來完成這些任務的自動化。例如，若要啟動 Amazon EKS 叢集，您可以使用 `eksctl create cluster`，而對於 Amazon EKS Anywhere，您可以使用 `eksctl anywhere create cluster`。請注意，當這些命令建立 Kubernetes 叢集時，它們是提供者特有的，不屬於 Kubernetes 專案本身。

叢集建立和管理工具

Kubernetes 專案提供手動建立 Kubernetes 叢集的工具。因此，如果您想要在單一機器上安裝 Kubernetes，或在機器上執行控制平面並手動新增節點，您可以使用 Kubernetes 安裝工具下列出的 CLI 工具，例如 kind、 minikube 或 kubeadm。為了簡化和自動化叢集建立和管理的完整生命週期，使用已建立的 Kubernetes 供應商支援的工具會更容易，例如 Amazon EKS 或 Amazon EKS Anywhere。

在 AWS 雲端中，您可以使用 CLI 工具建立 Amazon EKS 叢集，例如 eksctl，或更多宣告工具，例如 Terraform （請參閱適用於 Terraform 的 Amazon EKS 藍圖)。您也可以從 建立叢集 AWS Management Console。如需使用 Amazon EKS 取得的清單，請參閱 Amazon EKS 功能。Amazon EKS 為您承擔的 Kubernetes 責任包括：

- 受管控制平面 -AWS 確保 Amazon EKS 叢集可用且可擴展，因為它會為您管理控制平面，並讓它跨 AWS 可用區域使用。
- 節點管理 - 您可以使用受管節點群組 （請參閱the section called “受管節點群組”) 或 Karpenter，讓 Amazon EKS 視需要自動建立節點，而不是手動新增節點。受管節點群組與 Kubernetes Cluster

Autoscaling 整合。使用節點管理工具，您可以利用 Spot 執行個體、節點整合和可用性等功能來節省成本，並使用排程功能來設定部署工作負載和選取節點的方式。

- 叢集聯網 — 使用 CloudFormation 範本，在 Kubernetes 叢集中`eksctl`設定控制平面和資料平面（節點）元件之間的聯網。它也會設定可透過其進行內部和外部通訊的端點。如需詳細資訊，請參閱解密 Amazon EKS 工作者節點的叢集聯網。Amazon EKS 中的 Pod 之間的通訊是使用 Amazon EKS Pod 身分（請參閱 the section called “Pod 身分”) 完成，其提供讓 Pod 利用 AWS 雲端方法來管理登入資料和許可。
- 附加元件 — Amazon EKS 可讓您不必建置和新增通常用於支援 Kubernetes 叢集的軟體元件。例如，當您從 建立 Amazon EKS 叢集時 AWS Management Console，它會自動新增 Amazon EKS kube-proxy (the section called “kube-proxy”)、適用於 Kubernetes 的 Amazon VPC CNI 外掛程式 (the section called “Amazon VPC CNI”) 和 CoreDNS (the section called “CoreDNS”) 附加元件。the section called “Amazon EKS 附加元件” 如需這些附加元件的詳細資訊，請參閱 ，包括可用的 清單。

若要在您自己的現場部署電腦和網路上執行叢集，Amazon 提供 Amazon EKS Anywhere。您可以選擇使用自己的設備，在 VMWare vSphere 上執行 Amazon EKS Anywhere、裸機 (Tinkerbell 提供者)、Snow、CloudStack 或 Nutanix 平台，而不是做為 AWS 提供者。

Amazon EKS Anywhere 是以 Amazon EKS 所使用的相同 Amazon EKS Distro 軟體為基礎。不過，Amazon EKS Anywhere 依賴 Kubernetes 叢集 API (CAPI) 介面的不同實作來管理 Amazon EKS Anywhere 叢集中機器的完整生命週期（例如 vSphere 的 CAPV 和 CloudStack 的 CAPC)。由於整個叢集都在您的設備上執行，因此您需承擔管理控制平面和備份其資料的額外責任（請參閱本文件`etcd`稍後部分）。

叢集元件

Kubernetes 叢集元件分為兩個主要領域：控制平面和工作者節點。控制平面元件會管理叢集，並提供其 APIs存取權。工作者節點（有時又稱為節點）提供實際工作負載執行的位置。節點元件包含在每個節點上執行的服務，以與控制平面和執行容器通訊。叢集的工作者節點集稱為資料平面。

控制平台

控制平面包含一組管理叢集的服務。這些服務可能全都在單一電腦上執行，也可能分散在多部電腦上。在內部，這些稱為控制平面執行個體 (CPIs)。CPIs 的執行方式取決於叢集的大小和高可用性的需求。隨著叢集的需求增加，控制平面服務可以擴展以提供該服務的更多執行個體，並在執行個體之間負載平衡請求。

Kubernetes 控制平面的元件執行的任務包括：

- 與叢集元件 (API 伺服器） 通訊 — API 伺服器 (kube-apiserver) 公開 Kubernetes API，因此可以從叢集內部和外部對叢集提出請求。換言之，新增或變更叢集物件 (Pod、服務、節點等） 的請求可能來自外部命令，例如來自 kubectl執行 Pod 的請求。同樣地，也可以從 API 伺服器向叢集內的元件提出請求，例如查詢 kubelet Pod 狀態的服務。
- 儲存有關叢集的資料 (**etcd** 金鑰值存放區） — etcd服務提供追蹤叢集目前狀態的關鍵角色。如果etcd服務變得無法存取，您將無法更新或查詢叢集的狀態，但工作負載會繼續執行一段時間。因此，關鍵叢集通常會一次執行多個負載平衡etcd的服務執行個體，並在資料遺失或損毀時定期備份etcd金鑰值存放區。請注意，在 Amazon EKS 中，預設會自動為您處理。Amazon EKS Anywhere 提供用於加密備份和還原的說明。請參閱 編碼資料模型，以了解 如何etcd管理資料。
- 排程 Pod 到節點 （ 排程器） — 在 Kubernetes 中啟動或停止 Pod 的請求會導向 Kubernetes 排程器 (kube-scheduler)。由於叢集可能有多個節點能夠執行 Pod，因此由排程器選擇 Pod 應執行的節點（ 如果是複本，則為節點）。如果沒有足夠的可用容量在現有節點上執行請求的 Pod，除非您已進行其他佈建，否則請求將會失敗。這些規定可能包括啟用 受管節點群組 (the section called “受管節點群組”) 或 Karpenter 等服務，這些服務可以自動啟動新節點來處理工作負載。
- 將元件保持在所需狀態 (Controller Manager) — Kubernetes Controller Manager 會以常駐程式程序 (kube-controller-manager) 執行，以監控叢集的狀態，並變更叢集以重新建立預期狀態。特別是，有數個控制器可監看不同的 Kubernetes 物件，其中包括 statefulset-controller、endpoint-controller、node-controller、 cronjob-controller等。
- 管理雲端資源 （ 雲端控制器管理員） — Cloud Controller Manager (cloud-controller-manager) 會處理 Kubernetes 與執行基礎資料中心資源請求的雲端提供者之間的互動。由 Cloud Controller Manager 管理的控制器可包含路由控制器 （ 用於設定雲端網路路由）、服務控制器 （ 用於使用雲端負載平衡服務） 和節點生命週期控制器 （ 用於在整個生命週期中與 Kubernetes 保持節點同步）。

工作者節點 （ 資料平面）

對於單一節點 Kubernetes 叢集，工作負載會在與控制平面相同的機器上執行。不過，較標準的組態是擁有一或多個獨立的電腦系統 (節點)，專用於執行 Kubernetes 工作負載。

當您第一次建立 Kubernetes 叢集時，有些叢集建立工具可讓您設定要新增至叢集的特定數量節點 （ 透過識別現有的電腦系統或讓供應商建立新的節點）。將任何工作負載新增至這些系統之前，服務會新增至每個節點，以實作這些功能：

- 管理每個節點 (**kubelet**) — API 伺服器會與每個節點上執行的 kubelet 服務進行通訊，以確保節點已正確註冊，且排程器請求的 Pod 正在執行中。kubelet 可以讀取 Pod 資訊清單，並在本機系統上設定 Pod 所需的儲存磁碟區或其他功能。它也可以檢查本機執行中容器的運作狀態。

- 在節點上執行容器（容器執行時間）— 每個節點上的容器執行時間會管理指派給節點的每個 Pod 所請求的容器。這表示它可以從適當的登錄檔提取容器映像、執行容器、停止容器，以及回應有關容器的查詢。預設容器執行時間是容器化的。從 Kubernetes 1.24 開始，可做為容器執行時間的 Docker (`dockershim`) 特殊整合已從 Kubernetes 捨棄。雖然您仍然可以使用 Docker 在本機系統上測試和執行容器，但若要搭配 Kubernetes 使用 Docker，您現在必須在每個節點上安裝 Docker 引擎，才能搭配 Kubernetes 使用。
- 管理容器之間的聯網 (**`kube-proxy`**) — 為了支援 Pod 之間的通訊，Kubernetes 使用稱為 Service 的功能來設定 Pod 網路，以追蹤與這些 Pod 相關聯的 IP 地址和連接埠。kube-proxy 服務會在每個節點上執行，以允許 Pod 之間的通訊進行。

擴展叢集

有些服務可以新增至 Kubernetes 以支援叢集，但無法在控制平面中執行。這些服務通常直接在 kube-system 命名空間或其本身命名空間的節點上執行（如同第三方服務供應商一般）。常見的範例是 CoreDNS 服務，它為叢集提供 DNS 服務。如需如何查看叢集上哪些叢集服務正在 kube-system 中執行的資訊，請參閱探索內建服務。

您可以考慮將不同類型的附加元件新增至叢集。為了保持叢集正常運作，您可以新增可觀測性功能（請參閱 監控叢集），讓您執行記錄、稽核和指標等操作。透過此資訊，您通常可以透過相同的可觀測性界面對發生的問題進行故障診斷。這些類型的服務範例包括 Amazon GuardDuty、CloudWatch（請參閱 the section called “Amazon CloudWatch”）、AWS Ditro for OpenTelemetry、Amazon VPC CNI Kubernetes 外掛程式（請參閱 the section called “Amazon VPC CNI”）和 Grafana Kubernetes Monitoring。對於儲存（請參閱 應用程式資料儲存），Amazon EKS 的附加元件包括 Amazon Elastic Block Store CSI 驅動程式（請參閱 the section called “Amazon EBS”）、Amazon Elastic File System CSI 驅動程式（請參閱 the section called “Amazon EFS”），以及數個第三方儲存附加元件，例如 Amazon FSx for NetApp ONTAP CSI 驅動程式 the section called “Amazon FSx for NetApp ONTAP”）。

如需可用 Amazon EKS 附加元件的更完整清單，請參閱 the section called “Amazon EKS 附加元件”。

工作負載

Kubernetes 將工作負載定義為「在 Kubernetes 上執行的應用程式」。該應用程式可以由一組在 Pod 中作為容器執行的微服務組成，也可以作為批次任務或其他類型的應用程式執行。Kubernetes 的任務是確保您為要設定或部署的物件提出的請求已執行。當有人部署應用程式時，您應該了解容器的建置方式、Pod 的定義方式，以及您可以使用哪些方法來部署它們。

容器

您在 Kubernetes 中部署和管理的應用程式工作負載最基本的元素是 Pod。 Pod 代表保存應用程式元件的方式，以及定義描述 Pod 屬性的規格。與 RPM 或 Deb 套件之類的內容形成對比，該套件會將 Linux 系統的軟體封裝在一起，但本身不會以實體的形式執行。

由於 Pod 是最小的可部署單位，因此通常會保留單一容器。不過，在容器緊密耦合的情況下，多個容器可以在 Pod 中。例如，Web 伺服器容器可能封裝在具有附屬類型容器的 Pod 中，該容器可能提供與 Web 伺服器容器緊密連結的記錄、監控或其他服務。在此情況下，在相同的 Pod 中可確保每個執行中的 Pod 執行個體，兩個容器一律在相同的節點上執行。同樣地，Pod 中的所有容器都會共用相同的環境，Pod 中的容器會像在相同的隔離主機中一樣執行。這的影響是容器共用單一 IP 地址，提供 Pod 的存取權，而且容器可以彼此通訊，就像在自己的 localhost 上執行一樣。

Pod 規格 (PodSpec) 定義 Pod 的所需狀態。您可以使用工作負載資源來管理 Pod 範本，以部署個別 Pod 或多個 Pod。工作負載資源包括部署 （管理多個 Pod 複本）StatefulSets （部署需要唯一性的 Pod，例如資料庫 Pod) 和 DaemonSets (Pod 需要在每個節點上持續執行）。有關這些項目的更多信息。

雖然 Pod 是您部署的最小單位，但容器是您建置和管理的最小單位。

建置容器

Pod 實際上只是一或多個容器周圍的結構，每個容器本身都保存檔案系統、可執行檔、組態檔案、程式庫和其他元件，以實際執行應用程式。由於一家名為 Docker Inc. 的公司第一次熱門容器，因此某些人將容器稱為 Docker Containers。不過，Open Container Initiative 自定義以來已為業界定義容器執行時間、映像和分發方法。除此之外，容器是從許多現有的 Linux 功能建立的，有些通常將容器稱為 OCI Containers、Linux Containers 或僅稱為 Containers。

當您建置容器時，通常會以 Dockerfile （原名） 開頭。在該 Dockerfile 中，您可以識別：

- 基礎映像 - 基礎容器映像通常是從作業系統檔案系統的最低版本 （例如 Red Hat Enterprise Linux 或 Ubuntu) 或增強的最小系統建置的容器，以提供軟體來執行特定類型的應用程式 （例如 nodejs 或 python 應用程式）。
- 應用程式軟體 - 您可以將應用程式軟體新增至容器，方法與將其新增至 Linux 系統的方式大致相同。例如，您可以在 Dockerfile 中執行 npm和 `yarn` 安裝 Java 應用程式，或執行 `dnf` `yum`和 安裝 RPM 套件。換句話說，在 Dockerfile 中使用 RUN 命令，您可以執行基礎映像檔案系統中可用的任何命令，以在產生的容器映像內安裝軟體或設定軟體。
- 指示 — Dockerfile 參考說明您在設定 Dockerfile 時可新增至 Dockerfile 的指示。這些包括用來建置容器本身 (`ADD` 或本機系統中`COPY`的檔案） 的指示、識別容器執行時要執行的命令 (`CMD` 或

ENTRYPOINT)，以及將容器連接到其執行所在的系統 （透過識別USER要執行的 、VOLUME要掛載的本機 或連接埠 EXPOSE)。

雖然docker命令和服務傳統上已用於建置容器 (docker build)，但可用於建置容器映像的其他工具包括 Podman 和 nerdctl。請參閱建置更好的容器映像或 Docker Build 概觀，以了解建置容器。

儲存容器

建置容器映像後，您可以將其存放在工作站的容器分佈登錄檔或公有容器登錄檔。在工作站上執行私有容器登錄檔可讓您將容器映像儲存在本機，讓您隨時可用。

若要以更公開的方式存放容器映像，您可以將它們推送至公有容器登錄檔。公有容器登錄檔提供集中位置，用於存放和分發容器映像。公有容器登錄檔的範例包括 Amazon Elastic Container Registry、Red Hat Quay 登錄檔和 Docker Hub 登錄檔。

在 Amazon Elastic Kubernetes Service (Amazon EKS) 上執行容器化工作負載時，我們建議提取存放在 Amazon Elastic Container Registry 中的 Docker 官方映像複本。Amazon ECR 自 2021 以來一直存放這些映像。您可以在 Amazon ECR Public Gallery 中搜尋熱門容器映像，特別是 Docker Hub 映像，您可以搜尋 Amazon ECR Docker Gallery。

執行容器

由於容器是以標準格式建置，因此容器可以在可執行容器執行時間 （例如 Docker) 且其內容符合本機電腦架構 （例如 x86_64或) 的任何機器上執行arm。若要測試容器或在本機桌面上執行容器，您可以使用 docker run或 podman run命令在 localhost 上啟動容器。不過，對於 Kubernetes，每個工作者節點都有部署的容器執行期，並且可以由 Kubernetes 請求節點執行容器。

指派容器在節點上執行後，節點會查看節點上是否已存在請求的容器映像版本。如果沒有，Kubernetes 會告知容器執行期從適當的容器登錄檔提取該容器，然後在本機執行該容器。請記住，容器映像是指在筆記型電腦、容器登錄檔和 Kubernetes 節點之間移動的軟體套件。容器是指該映像的執行中執行個體。

Pod

容器準備就緒後，使用 Pod 包括設定、部署和讓 Pod 可存取。

設定 Pod

當您定義 Pod 時，您會為其指派一組屬性。這些屬性必須至少包含 Pod 名稱和要執行的容器映像。不過，您還需要使用 Pod 定義設定許多其他項目 （請參閱 PodSpec 頁面，了解哪些項目可以進入 Pod 的詳細資訊 ）。其中包含：

- 儲存 - 停止和刪除執行中的容器時，除非您設定更永久的儲存，否則該容器中的資料儲存會消失。Kubernetes 支援許多不同的儲存類型，並在磁碟區保護下抽象化它們。儲存類型包括 CephFS、NFS、iSCSI 等。您甚至可以從本機電腦使用本機區塊裝置。透過叢集提供的其中一種儲存類型，您可以將儲存磁碟區掛載到容器檔案系統中選取的掛載點。持久性磁碟區是在刪除 Pod 後持續存在的磁碟區，而刪除 Pod 時則會刪除暫時性磁碟區。如果您的叢集管理員為叢集建立不同的儲存類別，您可以選擇使用的儲存體屬性，例如使用後是否刪除或回收磁碟區、是否需要更多空間時是否會擴展，甚至是否符合特定效能需求。
- 秘密 - 透過將秘密提供給 Pod 規格中的容器，您可以提供這些容器存取檔案系統、資料庫或其他受保護資產所需的許可。金鑰、密碼和字符是可儲存為秘密的項目之一。使用秘密可讓您不必將此資訊存放在容器映像中，而只需要讓秘密可供執行中的容器使用。類似於秘密是 ConfigMaps。`ConfigMap` 往往會保留較不重要的資訊，例如用於設定服務的鍵/值對。
- 容器資源 — 用於進一步設定容器的物件可以採用資源組態的形式。對於每個容器，您可以請求其可以使用的記憶體和 CPU 數量，以及限制容器可以使用的那些資源總數量。如需範例，請參閱 Pod 和容器的資源管理。
- 中斷 – Pod 可能會非自願中斷（節點故障）或自願中斷（需要升級）。透過設定 Pod 中斷預算，您可以對應用程式在中斷發生時保持的可用性進行一些控制。如需範例，請參閱為您的應用程式指定中斷預算。
- 命名空間 — Kubernetes 提供不同的方法來隔離 Kubernetes 元件和工作負載。在相同命名空間中執行特定應用程式的所有 Pod，是同時保護和管理這些 Pod 的常見方式。您可以建立自己的命名空間來使用，或選擇不指示命名空間（這會導致 Kubernetes 使用`default`命名空間）。Kubernetes 控制平面元件通常會在 kube-system 命名空間中執行。

剛才描述的組態通常會在要套用至 Kubernetes 叢集的 YAML 檔案中收集在一起。對於個人 Kubernetes 叢集，您可能只會將這些 YAML 檔案存放在本機系統上。不過，透過更關鍵的叢集和工作負載，GitOps 是自動化儲存和更新工作負載和 Kubernetes 基礎設施資源的熱門方式。

用來收集和部署 Pod 資訊的物件是由下列其中一種部署方法定義。

部署 Pod

您選擇的部署 Pod 方法取決於您計劃搭配這些 Pod 執行的應用程式類型。以下是您的一些選擇：

- 無狀態應用程式 — 無狀態應用程式不會儲存用戶端的工作階段資料，因此另一個工作階段不需要參考先前工作階段發生的情況。如果 Pod 運作狀態不佳或四處移動而不儲存狀態，這可讓您更輕鬆地將 Pod 取代為新的 Pod。如果您執行的是無狀態應用程式（例如 Web 伺服器），您可以使用 部署來部署 Pod 和 ReplicaSets。ReplicaSet 定義您希望同時執行的 Pod 執行個體數量。雖然您可以直接執行 ReplicaSet，但通常會直接在部署內執行複本，以定義一次應執行多少個 Pod 複本。

- 具狀態應用程式：具狀態應用程式是 Pod 的身分和啟動 Pod 的順序很重要的應用程式。這些應用程式需要穩定且需要以一致方式部署和擴展的持久性儲存。若要在 Kubernetes 中部署具狀態的應用程式，您可以使用 StatefulSets。通常以 StatefulSet 執行的應用程式範例是資料庫。在 StatefulSet 中，您可以定義複本、Pod 及其容器、要掛載的儲存磁碟區，以及資料存放所在容器中的位置。如需部署為 ReplicaSet 的資料庫範例，請參閱執行複寫狀態應用程式。
- 每個節點應用程式 — 有時您想要在 Kubernetes 叢集的每個節點上執行應用程式。例如，您的資料中心可能需要每部電腦執行監控應用程式或特定的遠端存取服務。對於 Kubernetes，您可以使用 DaemonSet 來確保所選應用程式在您叢集中的每個節點上執行。
- 應用程式執行到完成 - 您想要執行一些應用程式來完成特定任務。這可能包括執行每月狀態報告或清除舊資料的報告。任務物件可用來設定應用程式以啟動和執行，然後在任務完成時結束。CronJob 物件可讓您使用 Linux crontab 格式定義的結構，將應用程式設定為在某個特定小時、分鐘、日期、月份或星期幾執行。

讓應用程式可從網路存取

隨著應用程式通常部署為一組移動到不同位置的微服務，Kubernetes 需要一種方法來讓這些微服務能夠互相尋找。此外，若要讓其他人存取 Kubernetes 叢集以外的應用程式，Kubernetes 需要一種方式，以在外部地址和連接埠上公開該應用程式。這些聯網相關功能分別由服務和輸入物件完成：

- 服務 - 由於 Pod 可以移動到不同的節點和地址，另一個需要與第一個 Pod 通訊的 Pod 可能會發現很難找到所在位置。為了解決此問題，Kubernetes 可讓您將應用程式表示為服務。使用 服務，您可以識別具有特定名稱的 Pod 或一組 Pod，然後指出哪些連接埠公開該應用程式的 Pod 服務，以及另一個應用程式可以使用哪些連接埠來聯絡該服務。叢集中的另一個 Pod 可以直接依名稱請求服務，Kubernetes 會將該請求導向執行該服務的 Pod 執行個體的適當連接埠。
- 輸入 - 輸入可讓 Kubernetes Services 表示的應用程式可供叢集外部的用戶端使用。傳入的基本功能包括負載平衡器 （由傳入管理）、傳入控制器，以及將請求從控制器路由至服務的規則。有數個輸入控制器可供您使用 Kubernetes 進行選擇。

後續步驟

了解基本的 Kubernetes 概念以及它們與 Amazon EKS 的關聯，將協助您導覽 Amazon EKS 文件和 Kubernetes 文件，以尋找管理 Amazon EKS 叢集和將工作負載部署到這些叢集所需的資訊。若要開始使用 Amazon EKS，請選擇下列選項：

- the section called “建立叢集 (eksctl)”
- the section called “建立叢集”

- the section called “範例部署 (Linux)”
- 叢集管理

跨雲端和內部部署環境部署 Amazon EKS 叢集

了解 Amazon EKS 部署選項

Amazon Elastic Kubernetes Service (Amazon EKS) 是一項全受管 Kubernetes 服務，可讓您在雲端和內部部署環境中無縫執行 Kubernetes。

在雲端中，Amazon EKS 會自動化 Kubernetes 控制平面和節點的 Kubernetes 叢集基礎設施管理。這對於排程容器、管理應用程式可用性、動態擴展資源、最佳化運算、儲存叢集資料，以及執行其他重要功能至關重要。透過 Amazon EKS，您可以取得 AWS 基礎設施的強大效能、可擴展性、可靠性和可用性，以及與 AWS 聯網、安全性、儲存和可觀測性服務的原生整合。

為了簡化內部部署環境中的 Kubernetes 執行，您可以在自己的基礎設施上使用與 the section called “節點”或 Amazon EKS 混合節點相同的 Amazon EKS 叢集、功能和工具，也可以將 Amazon EKS Anywhere 用於獨立的氣隙隔離環境。

雲端 Amazon EKS

您可以在 AWS 區域、 AWS 本機區域和 Wavelength 區域中使用 Amazon EKS AWS 搭配運算。使用雲端中的 Amazon EKS，Kubernetes 控制平面的安全性、可擴展性和可用性由 AWS 區域中的 完全管理 AWS 。在 AWS 區域中使用運算執行應用程式時，您可以取得 AWS 和 Amazon EKS 功能的完整廣度，包括 Amazon EKS Auto Mode， AWS 只要按一下，即可完全自動化 上的運算、儲存和聯網 Kubernetes 叢集基礎設施管理。在 AWS Local Zones 和 AWS Wavelength Zones 中使用運算執行應用程式時，您可以使用 Amazon EKS 自我管理節點來連接叢集運算的 Amazon EC2 執行個體，並且可以使用 AWS Local Zones 和 AWS Wavelength Zones 中的其他可用 AWS 服務。如需詳細資訊，請參閱 AWS Local Zones 功能和 AWS Wavelength Zones 功能。

	AWS 區域中的 Amazon EKS	本機/Wavelength 區域中的 Amazon EKS
Kuberenetes 控制平面管理	AWS受管	AWS受管
Kubernetes 控制平面位置	AWS 區域	AWS 區域

	AWS 區域中的 Amazon EKS	本機/Wavelength 區域中的 Amazon EKS
Kubernetes 資料平面	• Amazon EKS Auto 模式 • Amazon EKS 受管節點群組 • Amazon EC2 自我管理節點 • AWS Fargate	• Amazon EKS 受管節點群組（僅限本機區域） • Amazon EC2 自我管理節點
Kubernetes 資料平面位置	AWS 區域	AWS 本機或 Wavelength 區域

資料中心或邊緣環境中的 Amazon EKS

如果您需要在自己的資料中心或邊緣環境中執行應用程式，您可以使用 AWS Outposts 上的 Amazon EKS或 Amazon EKS 混合節點。您可以在叢集運算的 AWS Outposts 上使用自我管理的節點搭配 Amazon EC2 執行個體，或者您可以將 Amazon EKS 混合節點搭配自己的現場部署或邊緣基礎設施搭配叢集運算使用。 AWS Outposts 是在資料中心或主機代管設施中執行的 AWS受管基礎設施，而 Amazon EKS 混合節點會在您在現場部署或邊緣環境中管理的實體或虛擬機器上執行。Amazon EKS on AWS Outposts 和 Amazon EKS 混合節點需要從現場部署環境到 AWS 區域的可靠連線，而且您可以使用與雲端中執行應用程式相同的 Amazon EKS 叢集、功能和工具。在 AWS Outposts 上執行時，您也可以在 AWS Outposts 上使用 Outposts 上的 Amazon EKS 本機叢集部署整個 Kubernetes AWS 叢集。

	Amazon EKS 混合節點	AWS Outposts 上的 Amazon EKS
Kuberenetes 控制平面管理	AWS受管	AWS受管
Kubernetes 控制平面位置	AWS 區域	AWS 區域或 AWS Outpost
Kubernetes 資料平面	客戶管理的實體或虛擬機器	Amazon EC2 自我管理節點
Kubernetes 資料平面位置	客戶資料中心或邊緣環境	客戶資料中心或邊緣環境

Amazon EKS Anywhere 適用於氣隙隔離環境

Amazon EKS Anywhere 透過自動化未區分的繁重作業來簡化 Kubernetes 叢集管理，例如內部部署和邊緣環境中的基礎設施設定和 Kubernetes 叢集生命週期操作。與 Amazon EKS 不同，Amazon EKS Anywhere 是客戶管理的產品，客戶負責 Amazon EKS Anywhere 叢集的叢集生命週期操作和維護。Amazon EKS Anywhere 建置於 Kubernetes 子專案叢集 API (CAPI) 上，並支援各種基礎設施，包括 VMware vSphere、裸機、VDI、Apache CloudStack 和 AWS Snow。Amazon EKS Anywhere 可以在氣隙隔離環境中執行，並提供與區域 AWS 服務的選用整合，以實現可觀測性和身分管理。若要取得 Amazon EKS Anywhere 的支援和對 AWS Kubernetes 附加元件的存取，您可以購買 Amazon EKS Anywhere Enterprise Subscriptions。

	Amazon EKS Anywhere
Kuberenetes 控制平面管理	客戶受管
Kubernetes 控制平面位置	客戶資料中心或邊緣環境
Kubernetes 資料平面	客戶管理的實體或虛擬機器
Kubernetes 資料平面位置	客戶資料中心或邊緣環境

Amazon EKS 工具

您可以使用 Amazon EKS 連接器來註冊任何符合規範的 Kubernetes 叢集，並將其連接到 ， AWS 並在 Amazon EKS 主控台中檢視。將叢集連線後，您可以在 Amazon EKS 主控台中查看叢集的狀態、組態和工作負載。您可以使用此功能在 Amazon EKS 主控台中檢視連線的叢集，但 Amazon EKS 連接器不會透過 Amazon EKS 主控台啟用連線叢集的管理或變更操作。

Amazon EKS Distro 是支援所有 Amazon EKS 產品的基礎 Kubernetes 元件的 AWS 分佈。它包含運作中 Kubernetes 叢集所需的核心元件，例如 Kubernetes 控制平面元件 (etcd、kube-apiserver、kube-scheduler、kube-controller-manager) 和聯網元件 (CoreDNS、kube-proxy、CNI 外掛程式) 。Amazon EKS Distro 可用於自行管理 Kubernetes 叢集，並可選擇使用工具。 AWS Support Plans 不涵蓋 Amazon EKS Distro 部署。

設定 以使用 Amazon EKS

若要準備 Amazon EKS 叢集的命令列管理，您需要安裝數種工具。使用下列項目來設定登入資料、建立和修改叢集，並在叢集執行後使用叢集：

- 設定 AWS CLI – 取得 AWS CLI 來設定和管理使用 Amazon EKS 叢集所需的服務。特別是，您需要 AWS CLI 來設定登入資料，但您也需要它搭配其他 AWS 服務。
- 設定 kubectl 和 eksctl – CLI 會與 `eksctl` 互動， AWS 以建立、修改和刪除 Amazon EKS 叢集。叢集啟動後，請使用開放原始碼`kubectl`命令來管理 Amazon EKS 叢集中的 Kubernetes 物件。
- 設定開發環境 （選用） – 考慮新增下列工具：
  - 本機部署工具 – 如果您是初次使用 Kubernetes，請考慮安裝本機部署工具，例如 minikube 或 kind。這些工具可讓您在本機電腦上擁有 Amazon EKS 叢集，以測試應用程式。
  - 套件管理工具 – helm 是 Kubernetes 的熱門套件管理工具，可簡化複雜套件的安裝和管理。使用 Helm，您可以更輕鬆地在 Amazon EKS 叢集上安裝和管理套件，例如 AWS Load Balancer控制器。

後續步驟

- 設定 AWS CLI
- 設定 kubectl 和 eksctl
- 快速入門：部署 Web 應用程式和存放資料

設定 AWS CLI

AWS CLI 是一種命令列工具，可用於 AWS 服務，包括 Amazon EKS。它也用於驗證 IAM 使用者或角色，以便從本機電腦存取 Amazon EKS 叢集和其他 AWS 資源。若要 AWS 從命令列在 中佈建資源，您需要取得 AWS 存取金鑰 ID 和私密金鑰，以便在命令列中使用。然後，您需要在 CLI AWS 中設定這些登入資料。如果您尚未安裝 AWS CLI，請參閱《 AWS 命令列界面使用者指南》中的安裝或更新最新版本的 AWS CLI。

建立存取金鑰

1. 登入 AWS Management Console。
2. 對於單一使用者或多個使用者帳戶：

- 單一使用者帳戶 – : : 在右上角，選擇您的 AWS 使用者名稱以開啟導覽功能表。例如，選擇 **webadmin** 。
- 多使用者帳戶 – : : 從服務清單中選擇 IAM。從 IAM 儀表板中，選取使用者，然後選擇使用者名稱。

3. 選擇 Security credentials (安全登入資料)。
4. 在存取金鑰之下選擇建立存取金鑰。
5. 選擇命令列介面 (CLI)，然後選擇下一步。
6. 選擇 Create access key (建立新的存取金鑰)。
7. 選擇下載 .csv 檔案。

設定 AWS CLI

安裝 CLI AWS 之後，請執行下列步驟來設定它。如需詳細資訊，請參閱《 AWS 命令列界面使用者指南》中的設定 AWS CLI。

1. 在終端機視窗中，輸入以下命令：

```
aws configure
```

   您也可以選擇設定具名設定檔，例如 `--profile cluster-admin`。如果您在 CLI AWS 中設定具名設定檔，則一律必須在後續命令中傳遞此旗標。

2. 輸入您的 AWS 登入資料。例如：

```
Access Key ID [None]: AKIAIOSFODNN7EXAMPLE
Secret Access Key [None]: wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
Default region name [None]: region-code
Default output format [None]: json
```

取得安全字符

如有需要，請執行下列命令，以取得 CLI AWS 的新安全字符。如需詳細資訊，請參閱 AWS CLI 命令參考中的 get-session-token。

依預設，該字符的有效期為 15 分鐘。若要變更預設工作階段逾時，請傳遞 `--duration-seconds` 旗標。例如：

```
aws sts get-session-token --duration-seconds 3600
```

此命令會傳回 CLI AWS 工作階段的臨時安全登入資料。您應該會看到下列回應輸出：

```
{
    "Credentials": {
        "AccessKeyId": "ASIA5FTRU3LOEXAMPLE",
        "SecretAccessKey": "JnKgvwfqUD9mNsPoi9IbxAYEXAMPLE",
        "SessionToken": "VERYLONGSESSIONTOKENSTRING",
        "Expiration": "2023-02-17T03:14:24+00:00"
    }
}
```

若要驗證使用者身分

如有需要，請執行下列命令來驗證終端機工作階段的 IAM 使用者身分 （例如 *ClusterAdmin*) AWS 憑證。

```
aws sts get-caller-identity
```

此命令會傳回為 CLI 設定的 IAM 實體的 Amazon Resource Name AWS (ARN)。您應該會看到類似以下內容的回應輸出：

```
{
    "UserId": "AKIAIOSFODNN7EXAMPLE",
    "Account": "01234567890",
    "Arn": "arn:aws: iam::01234567890:user/ClusterAdmin"
}
```

後續步驟

- 設定 kubectl 和 eksctl
- 快速入門：部署 Web 應用程式和存放資料

設定 kubectl 和 eksctl

安裝 CLI AWS 後，您應該安裝其他兩個工具來建立和管理 Kubernetes 叢集：

- kubectl：kubectl命令列工具是您用來管理 Kubernetes 叢集內資源的主要工具。此頁面說明如何下載和設定與 Kubernetes 叢集版本相符的kubectl二進位檔。請參閱安裝或更新 kubectl。
- eksctl：eksctl命令列工具用於在 AWS 雲端或內部部署（使用 EKS Anywhere) 中建立 EKS 叢集，以及修改和刪除這些叢集。請參閱安裝 eksctl。

安裝或更新 kubectl

本主題將協助您在裝置上下載並安裝或更新 kubectl 二進位檔案。二進位檔案與上游社群版本相同。二進位對 Amazon EKS 或 來說不是唯一的 AWS。使用下列步驟取得kubectl您需要的特定 版本，雖然許多建置器只要執行 brew install kubectl 即可安裝。

> **Note**
>
> 您所使用的 kubectl 版本，必須與 Amazon EKS 叢集控制平面的版本差距在一個版本以內。例如，1.32 kubectl 用戶端搭配 Kubernetes 1.31、1.32 和 1.33 叢集運作。

步驟 1：檢查kubectl是否已安裝

判斷您是否已在裝置上安裝 kubectl。

```
kubectl version --client
```

若您已在裝置路徑中安裝 kubectl，則範例輸出包含下列類似資訊。若您要更新目前使用較新版本安裝的版本，請完成下一個步驟，確定將新版本安裝在目前版本所處的位置。

```
Client Version: v1.31.X-eks-1234567
```

如果您未收到輸出，則表示您尚未kubectl安裝，或未安裝在裝置路徑中的位置。

步驟 2：安裝或更新 kubectl

在下列其中一個作業系統kubectl上安裝或更新：

- the section called “macOS”
- the section called “Linux (amd64)”
- the section called “Linux (arm64)”
- the section called “Windows”

> **Note**
>
> 如果從本節使用的 AWS 區域下載到 AWS 區域的速度緩慢，請考慮設定 CloudFront 來前置內容。如需詳細資訊，請參閱開始使用基本 CloudFront 分佈。

macOS

請依照下列步驟在 macOS `kubectl`上安裝 。步驟包括：

- 選擇並下載所需 Kubernetes 版本的二進位檔。
- 選擇性地檢查二進位的檢查總和。
- 將執行新增至二進位檔的許可。
- 將二進位檔複製到 PATH 中的資料夾。
- 選擇性地將二進位的目錄新增至 PATH。

程序：

1. 從 Amazon S3 下載叢集 Kubernetes 版本的二進位檔。
   - Kubernetes `1.33`

     ```
     curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.33.3/2025-08-03/bin/darwin/amd64/kubectl
     ```

   - Kubernetes `1.32`

     ```
     curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.32.7/2025-08-03/bin/darwin/amd64/kubectl
     ```

   - Kubernetes `1.31`

     ```
     curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.31.11/2025-08-03/bin/darwin/amd64/kubectl
     ```

   - Kubernetes `1.30`

     ```
     curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.30.14/2025-08-03/bin/darwin/amd64/kubectl
     ```

- Kubernetes 1.29

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.29.15/2025-08-03/bin/darwin/amd64/kubectl
```

- Kubernetes 1.28

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.28.15/2025-08-03/bin/darwin/amd64/kubectl
```

- Kubernetes 1.27

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.27.16/2025-01-10/bin/darwin/amd64/kubectl
```

- Kubernetes 1.26

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.26.15/2024-12-12/bin/darwin/amd64/kubectl
```

2. (選用) 使用您的二進位檔案 SHA-256 檢查總和，驗證下載的二進位檔案。

   a. 下載叢集 Kubernetes 版本的SHA-256檢查總和。

   - Kubernetes 1.33

   ```
   curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.33.3/2025-08-03/bin/darwin/amd64/kubectl.sha256
   ```

   - Kubernetes 1.32

   ```
   curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.32.7/2025-08-03/bin/darwin/amd64/kubectl.sha256
   ```

   - Kubernetes 1.31

   ```
   curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.31.11/2025-08-03/bin/darwin/amd64/kubectl.sha256
   ```

   - Kubernetes 1.30

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.30.14/2025-08-03/bin/
darwin/amd64/kubectl.sha256
```

- Kubernetes 1.29

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.29.15/2025-08-03/bin/
darwin/amd64/kubectl.sha256
```

- Kubernetes 1.28

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.28.15/2025-08-03/bin/
darwin/amd64/kubectl.sha256
```

- Kubernetes 1.27

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.27.16/2025-01-10/bin/
darwin/amd64/kubectl.sha256
```

- Kubernetes 1.26

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.26.15/2024-12-12/bin/
darwin/amd64/kubectl.sha256
```

b. 檢查適用您下載之二進位檔案的 SHA-256 檢查總和。

```
openssl sha1 -sha256 kubectl
```

c. 確保輸出中產生的檢查總和與下載 kubectl.sha256 檔案中的檢查總和相符。

3. 將執行許可套用至二進位檔。

```
chmod +x ./kubectl
```

4. 將二進位檔複製到 PATH 中的資料夾。如果您已安裝某一版本的 kubectl，建議您建立 $HOME/bin/kubectl 並確認您的 $PATH 中會先出現 $HOME/bin。

```
mkdir -p $HOME/bin && cp ./kubectl $HOME/bin/kubectl && export PATH=$HOME/bin:$PATH
```

5. (選用) 將 $HOME/bin 路徑新增到 Shell 初始化檔案，因此當您開啟 shell 時，該組態已設定完畢。

```
echo 'export PATH=$HOME/bin:$PATH' >> ~/.bash_profile
```

Linux (amd64)

請依照下列步驟在 Linux (amd64) `kubectl` 上安裝 。步驟包括：

- 選擇並下載所需 Kubernetes 版本的二進位檔。
- 選擇性地檢查二進位的檢查總和。
- 將執行新增至二進位檔的許可。
- 將二進位檔複製到 PATH 中的資料夾。
- 選擇性地將二進位的目錄新增至 PATH。

程序：

1. 從 `kubectl` Amazon S3 下載叢集 Kubernetes 版本的二進位檔。
   - Kubernetes `1.33`

     ```
     curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.33.3/2025-08-03/bin/linux/amd64/kubectl
     ```

   - Kubernetes `1.32`

     ```
     curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.32.7/2025-08-03/bin/linux/amd64/kubectl
     ```

   - Kubernetes `1.31`

     ```
     curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.31.11/2025-08-03/bin/linux/amd64/kubectl
     ```

   - Kubernetes `1.30`

     ```
     curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.30.14/2025-08-03/bin/linux/amd64/kubectl
     ```

   - Kubernetes `1.29`

     ```
     curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.29.15/2025-08-03/bin/linux/amd64/kubectl
     ```

   - Kubernetes `1.28`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.28.15/2025-08-03/bin/linux/amd64/kubectl
```

- Kubernetes 1.27

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.27.16/2024-12-12/bin/linux/amd64/kubectl
```

- Kubernetes 1.26

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.26.15/2024-12-12/bin/linux/amd64/kubectl
```

2. (選用) 使用您的二進位檔案 SHA-256 檢查總和，驗證下載的二進位檔案。

   a. 使用適用於裝置硬體平台的 命令，從 Amazon S3using 下載叢集 Kubernetes 版本的SHA-256檢查總和。

   - Kubernetes 1.33

   ```
   curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.33.3/2025-08-03/bin/linux/amd64/kubectl.sha256
   ```

   - Kubernetes 1.32

   ```
   curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.32.7/2025-08-03/bin/linux/amd64/kubectl.sha256
   ```

   - Kubernetes 1.31

   ```
   curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.31.11/2025-08-03/bin/linux/amd64/kubectl.sha256
   ```

   - Kubernetes 1.30

   ```
   curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.30.14/2025-08-03/bin/linux/amd64/kubectl.sha256
   ```

   - Kubernetes 1.29

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.29.15/2025-08-03/bin/
linux/amd64/kubectl.sha256
```

- Kubernetes 1.28

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.28.15/2025-08-03/bin/
linux/amd64/kubectl.sha256
```

- Kubernetes 1.27

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.27.16/2024-12-12/bin/
linux/amd64/kubectl.sha256
```

- Kubernetes 1.26

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.26.15/2024-12-12/bin/
linux/amd64/kubectl.sha256
```

b. 執行下列其中一個命令，檢查您下載的二進製檔案的 SHA-256 檢查總和。

```
sha256sum -c kubectl.sha256
```

或

```
openssl sha1 -sha256 kubectl
```

c. 首先，您應該會看到 kubectl: OK，第二個是，您可以檢查輸出中產生的檢查總和是否與下載kubectl.sha256檔案中的檢查總和相符。

3. 將執行許可套用至二進位檔。

```
chmod +x ./kubectl
```

4. 將二進位檔複製到 PATH 中的資料夾。如果您已安裝某一版本的 kubectl，建議您建立 $HOME/bin/kubectl 並確認您的 $PATH 中會先出現 $HOME/bin。

```
mkdir -p $HOME/bin && cp ./kubectl $HOME/bin/kubectl && export PATH=$HOME/bin:$PATH
```

5. (選用) 將 $HOME/bin 路徑新增到 Shell 初始化檔案，因此當您開啟 shell 時，該組態已設定完畢。

> **Note**
>
> 本步驟假設您是使用 Bash Shell；若您使用其他 Shell，請將命令更改為使用具體的 Shell 初始化檔案。

```
echo 'export PATH=$HOME/bin:$PATH' >> ~/.bashrc
```

Linux (arm64)

請依照下列步驟在 Linux (arm64) `kubectl`上安裝 。步驟包括：

- 選擇並下載所需 Kubernetes 版本的二進位檔。
- 選擇性地檢查二進位的檢查總和。
- 將執行新增至二進位檔的許可。
- 將二進位檔複製到 PATH 中的資料夾。
- 選擇性地將二進位的目錄新增至 PATH。

程序：

1. 從 `kubectl` Amazon S3 下載叢集 Kubernetes 版本的二進位檔。
   - Kubernetes `1.33`

     ```
     curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.33.3/2025-08-03/bin/linux/arm64/kubectl
     ```

   - Kubernetes `1.32`

     ```
     curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.32.7/2025-08-03/bin/linux/arm64/kubectl
     ```

   - Kubernetes `1.31`

     ```
     curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.31.11/2025-08-03/bin/linux/arm64/kubectl
     ```

- Kubernetes 1.30

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.30.14/2025-08-03/bin/linux/arm64/kubectl
```

- Kubernetes 1.29

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.29.15/2025-08-03/bin/linux/arm64/kubectl
```

- Kubernetes 1.28

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.28.15/2025-08-03/bin/linux/arm64/kubectl
```

- Kubernetes 1.27

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.27.16/2024-12-12/bin/linux/arm64/kubectl
```

- Kubernetes 1.26

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.26.15/2024-12-12/bin/linux/arm64/kubectl
```

2. (選用) 使用您的二進位檔案 SHA-256 檢查總和，驗證下載的二進位檔案。

   a. 使用適用於裝置硬體平台的 命令，從 Amazon S3using 下載叢集 Kubernetes 版本的SHA-256檢查總和。

   - Kubernetes 1.33

     ```
     curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.33.3/2025-08-03/bin/linux/arm64/kubectl.sha256
     ```

   - Kubernetes 1.32

     ```
     curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.32.7/2025-08-03/bin/linux/arm64/kubectl.sha256
     ```

   - Kubernetes 1.31

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.31.11/2025-08-03/bin/
linux/arm64/kubectl.sha256
```

- Kubernetes 1.30

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.30.14/2025-08-03/bin/
linux/arm64/kubectl.sha256
```

- Kubernetes 1.29

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.29.15/2025-08-03/bin/
linux/arm64/kubectl.sha256
```

- Kubernetes 1.28

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.28.15/2025-08-03/bin/
linux/arm64/kubectl.sha256
```

- Kubernetes 1.27

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.27.16/2024-12-12/bin/
linux/arm64/kubectl.sha256
```

- Kubernetes 1.26

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.26.15/2024-12-12/bin/
linux/arm64/kubectl.sha256
```

b. 執行下列其中一個命令，檢查您下載的二進製檔案的 SHA-256 檢查總和。

```
sha256sum -c kubectl.sha256
```

或

```
openssl sha1 -sha256 kubectl
```

c. 首先，您應該會看到 kubectl: OK，第二個是，您可以檢查輸出中產生的檢查總和是否與下載kubectl.sha256檔案中的檢查總和相符。

3. 將執行許可套用至二進位檔。

```
chmod +x ./kubectl
```

4. 將二進位檔複製到 PATH 中的資料夾。如果您已安裝某一版本的 kubectl，建議您建立 $HOME/bin/kubectl 並確認您的 $PATH 中會先出現 $HOME/bin。

```
mkdir -p $HOME/bin && cp ./kubectl $HOME/bin/kubectl && export PATH=$HOME/bin:$PATH
```

5. (選用) 將 $HOME/bin 路徑新增到 Shell 初始化檔案，因此當您開啟 shell 時，該組態已設定完畢。

> **Note**
>
> 本步驟假設您是使用 Bash Shell；若您使用其他 Shell，請將命令更改為使用具體的 Shell 初始化檔案。

```
echo 'export PATH=$HOME/bin:$PATH' >> ~/.bashrc
```

Windows

請依照下列步驟在 Windows kubectl上安裝 。步驟包括：

- 選擇並下載所需 Kubernetes 版本的二進位檔。
- 選擇性地檢查二進位的檢查總和。
- 將二進位檔複製到 PATH 中的資料夾。
- 選擇性地將二進位的目錄新增至 PATH。

程序：

1. 開啟 PowerShell 終端機。
2. 從 kubectl Amazon S3 下載叢集 Kubernetes 版本的二進位檔。
   - Kubernetes 1.33

```
curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.33.3/2025-08-03/bin/windows/amd64/kubectl.exe
```

   - Kubernetes 1.32

```
curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.32.7/2025-08-03/bin/
windows/amd64/kubectl.exe
```

- Kubernetes 1.31

```
curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.31.11/2025-08-03/bin/
windows/amd64/kubectl.exe
```

- Kubernetes 1.30

```
curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.30.14/2025-08-03/bin/
windows/amd64/kubectl.exe
```

- Kubernetes 1.29

```
curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.29.15/2025-08-03/bin/
windows/amd64/kubectl.exe
```

- Kubernetes 1.28

```
curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.28.15/2025-08-03/bin/
windows/amd64/kubectl.exe
```

- Kubernetes 1.27

```
curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.27.16/2024-12-12/bin/
windows/amd64/kubectl.exe
```

- Kubernetes 1.26

```
curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.26.15/2024-12-12/bin/
windows/amd64/kubectl.exe
```

3. (選用) 使用您的二進位檔案 SHA-256 檢查總和，驗證下載的二進位檔案。

   a. 下載適用於 Windows 的叢集 Kubernetes 版本的SHA-256檢查總和。

      - Kubernetes 1.33

```
curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.33.3/2025-08-03/bin/
windows/amd64/kubectl.exe.sha256
```

- Kubernetes 1.32

```
curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.32.7/2025-08-03/bin/windows/amd64/kubectl.exe.sha256
```

- Kubernetes 1.31

```
curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.31.11/2025-08-03/bin/windows/amd64/kubectl.exe.sha256
```

- Kubernetes 1.30

```
curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.30.14/2025-08-03/bin/windows/amd64/kubectl.exe.sha256
```

- Kubernetes 1.29

```
curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.29.15/2025-08-03/bin/windows/amd64/kubectl.exe.sha256
```

- Kubernetes 1.28

```
curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.28.15/2025-08-03/bin/windows/amd64/kubectl.exe.sha256
```

- Kubernetes 1.27

```
curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.27.16/2024-12-12/bin/windows/amd64/kubectl.exe.sha256
```

- Kubernetes 1.26

```
curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.26.15/2024-12-12/bin/windows/amd64/kubectl.exe.sha256
```

b. 檢查適用您下載之二進位檔案的 SHA-256 檢查總和。

```
Get-FileHash kubectl.exe
```

c. 確保輸出中產生的檢查總和與下載 `kubectl.sha256` 檔案中的檢查總和相符。PowerShell 輸出應該是大寫的等效字元字串。

4. 將二進位檔複製到 PATH 中的資料夾。若在 PATH 中已有命令列公用程式專用的現存目錄，請將二進位檔案複製至該目錄。否則，請完成下列步驟。
   a. 建立新目錄以供存放命令列二進位檔，例如 `C:\bin`。
   b. 將 `kubectl.exe` 二進位檔複製到該新目錄。
   c. 編輯您的使用者或系統 PATH 環境變數，將新目錄新增至 PATH。
   d. 關閉 PowerShell 終端，再重新開啟以挑選新的 PATH 變數。
5. 安裝 `kubectl` 後，您可以驗證其版本。

```
kubectl version --client
```

6. 第一次安裝 時`kubectl`，尚未將其設定為與任何伺服器通訊。我們會視需要在其他程序中涵蓋此組態。如果您需要更新組態來與特定叢集進行通訊，那麼您可以執行下列命令。將 *`region-code`* 取代為您的叢集所在的 AWS 區域。使用您叢集的名稱取代 *`my-cluster`*。

```
aws eks update-kubeconfig --region region-code --name my-cluster
```

7. 請考慮設定自動完成，這可讓您在輸入前幾個字母後，使用 Tab 鍵來完成`kubectl`子命令。如需詳細資訊，請參閱 Kubernetes 文件中的 Kubectl 自動完成。

安裝 `eksctl`

CLI `eksctl` 用於使用 EKS 叢集。它會自動化許多個別任務。如需安裝 的說明，請參閱 `eksctl` 文件中的安裝`eksctl`。對於 Linux，請使用 UNIX 說明。

使用您正在使用的 `eksctl` IAM 安全主體時，必須具有使用 Amazon EKS IAM 角色、服務連結角色、AWS CloudFormation、VPC 和相關資源的許可。如需詳細資訊，請參閱《IAM 使用者指南》中的動作和使用服務連結角色。您必須以同一位使用者的身分完成本指南中的所有步驟。若要檢查目前使用者，請執行以下命令：

```
aws sts get-caller-identity
```

後續步驟

- 快速入門：部署 Web 應用程式和存放資料

快速入門：部署 Web 應用程式和存放資料

部署遊戲應用程式並在 Amazon EKS 上保留其資料

本快速入門教學課程示範使用 eksctl 部署 2048 遊戲範例應用程式，並將其資料保留在 Amazon EKS Auto Mode 叢集上的步驟。Amazon EKS Auto Mode 可自動化叢集區塊儲存、聯網、負載平衡和運算自動擴展的例行任務。

在我們進行的過程中，我們會逐步引導您完成叢集設定程序。Amazon EKS Auto Mode 將自動化使用 EC2 受管執行個體建立節點、建立應用程式負載平衡器和建立 EBS 磁碟區的任務。

整體而言，您將部署範例工作負載，其中包含完全與 AWS 服務整合所需的自訂註釋。

於本教學課程中

使用下列`eksctl`叢集範本，您將使用 EKS Auto 模式建置叢集，以進行自動節點佈建。

VPC 組態 使用之後的 eksctl 叢集範本時，eksctl 會自動為叢集建立 IPv4 Virtual Private Cloud (VPC)。根據預設，除了建立公有和私有端點之外，eksctl 還會設定 VPC 來處理所有聯網需求。

執行個體管理 EKS Auto Mode 會根據 Kubernetes 應用程式的需求，動態新增或移除 EKS 叢集中的節點。

資料持久性 使用 EKS Auto Mode 的區塊儲存功能來確保應用程式資料的持久性，即使在涉及 Pod 重新啟動或故障的情況下也是如此。

外部應用程式存取 使用 EKS Auto Mode 的負載平衡功能動態佈建 Application Load Balancer (ALB)。

先決條件

開始之前，請確定您已設定下列先決條件，以使用 Amazon EKS：

- 設定 AWS CLI 並設定登入資料
- 安裝 eksctl
- 安裝 kubectl

如需詳細資訊，請參閱設定。

設定叢集

在本節中，您將使用 EKS Auto Mode 建立叢集，以進行動態節點佈建。

建立 `cluster-config.yaml` 檔案，並將下列內容貼入其中。`region-code` 將 取代為有效的區域，例如 `us-east-1`：

```
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: web-quickstart
  region: <region-code>

autoModeConfig:
  enabled: true
```

現在，我們已準備好建立叢集。

建立 Amazon EKS 叢集：

```
eksctl create cluster -f cluster-config.yaml
```

> **Important**
>
> 如果您不使用 eksctl 建立叢集，則需要手動標記 VPC 子網路。

建立 IngressClass

為 EKS Auto 模式建立 `IngressClass` Kubernetes。IngressClass 定義 EKS Auto Mode 如何處理傳入資源。此步驟會設定 EKS Auto Mode 的負載平衡功能。當您為應用程式建立輸入資源時，EKS Auto Mode 會使用此 IngressClass 來自動佈建和管理負載平衡器，並將 Kubernetes 應用程式與 AWS 負載平衡服務整合。

將下列 yaml 檔案儲存為 `ingressclass.yaml`：

```
apiVersion: networking.k8s.io/v1
```

```
kind: IngressClass
metadata:
  name: alb
  annotations:
    ingressclass.kubernetes.io/is-default-class: "true"
spec:
  controller: eks.amazonaws.com/alb
```

將 IngressClass 套用至您的叢集：

```
kubectl apply -f ingressclass.yaml
```

部署 2048 遊戲範例應用程式

在本節中，我們會逐步引導您部署熱門的「2048 遊戲」作為叢集內的範例應用程式。提供的清單檔案包含 Application Load Balancer (ALB) 的自訂註釋。這些註釋與 整合，並指示 EKS 將傳入的 HTTP 流量視為「面向網際網路」，並使用目標類型「ip」將其路由到`game-2048`命名空間中的適當服務。

> **Note**
>
> 範例中的`docker-2048`映像是`x86_64`容器映像，不會在其他架構上執行。

1. 使用 `--save-config`旗標建立名為 `game-2048`的 Kubernetes 命名空間。

   ```
   kubectl create namespace game-2048 --save-config
   ```

   您應該會看到下列回應輸出：

   ```
   namespace/game-2048 created
   ```

2. 部署 2048 遊戲範例應用程式。

   ```
   kubectl apply -n game-2048 -f https://raw.githubusercontent.com/kubernetes-sigs/aws-load-balancer-controller/v2.8.0/docs/examples/2048/2048_full.yaml
   ```

   此資訊清單會為`game-2048`命名空間設定 Kubernetes 部署、服務和輸入，建立必要的資源以在叢集內部署和公開`game-2048`應用程式。其中包括建立名為 的服務`service-2048`，該服務公開連

接埠 上的部署80，以及名為 的傳入資源ingress-2048，該資源定義傳入 HTTP 流量的路由規則，以及面向網際網路的 Application Load Balancer (ALB) 註釋。您應該會看到下列回應輸出：

```
namespace/game-2048 configured
deployment.apps/deployment-2048 created
service/service-2048 created
ingress.networking.k8s.io/ingress-2048 created
```

3. 執行下列命令以取得game-2048命名空間的輸入資源。

```
kubectl get ingress -n game-2048
```

您應該會看到下列回應輸出：

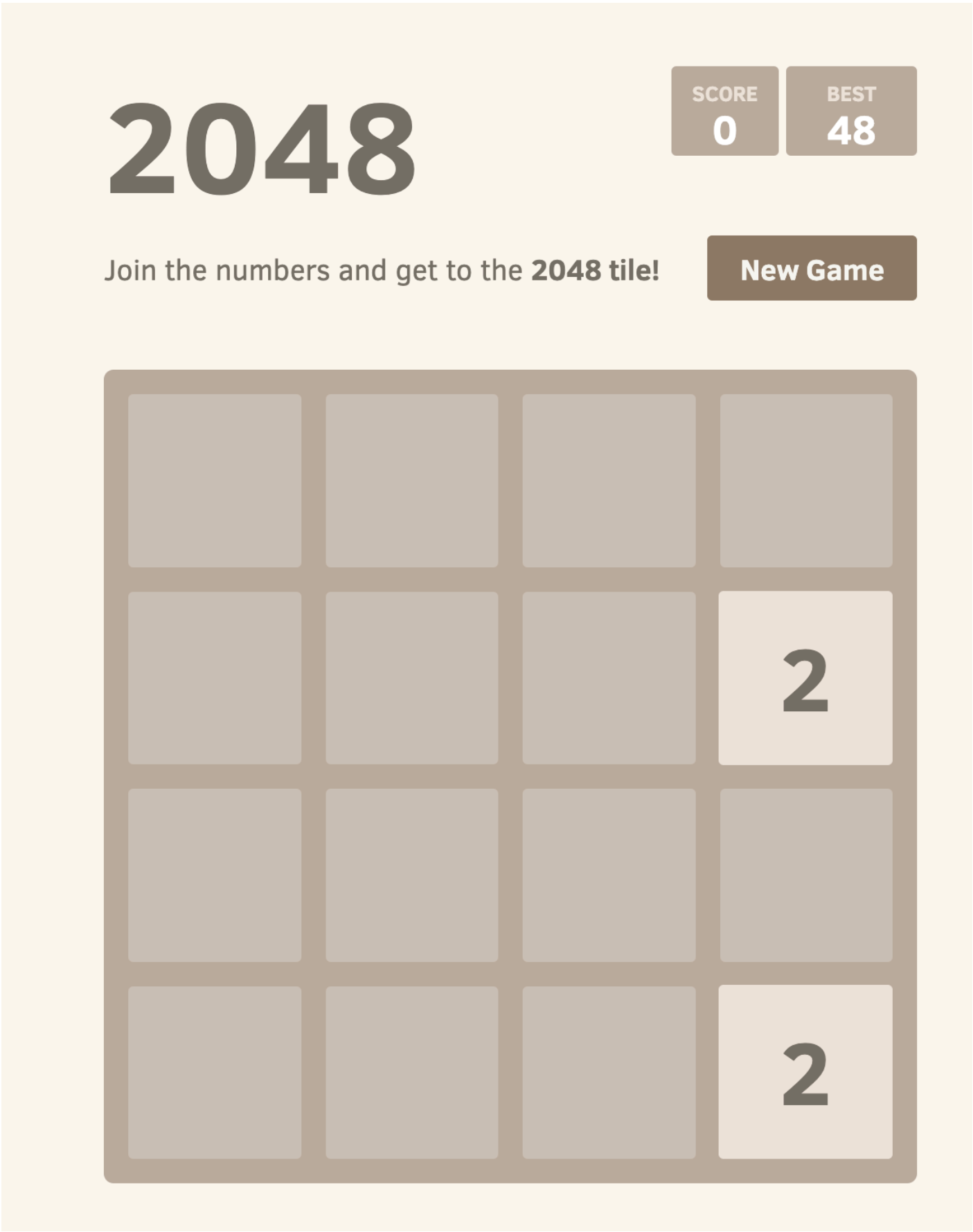
```
NAME           CLASS   HOSTS   ADDRESS
                     PORTS   AGE
ingress-2048   alb     *       k8s-game2048-ingress2-eb379a0f83-378466616.region-
code.elb.amazonaws.com   80      31s
```

您需要等待幾分鐘，Application Load Balancer (ALB) 才能佈建，才能開始下列步驟。

4. 開啟 Web 瀏覽器，然後輸入上ADDRESS一個步驟中的 以存取 Web 應用程式。例如：

```
k8s-game2048-ingress2-eb379a0f83-378466616.region-code.elb.amazonaws.com
```

您應該會在瀏覽器中看到 2048 遊戲。播放！


2048
SCORE
0
BEST
48
Join the numbers and get to the 2048 tile!
New Game
2
2

使用 Amazon EKS Auto Mode 保留資料

現在 2048 遊戲已在您的 Amazon EKS 叢集上啟動並執行，是時候確保使用 Amazon EKS Auto Mode 的區塊儲存功能安全地保留您的遊戲資料了。

1. 建立名為 `storage-class.yaml` 的檔案：

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: auto-ebs-sc
  annotations:
    storageclass.kubernetes.io/is-default-class: "true"
provisioner: ebs.csi.eks.amazonaws.com
volumeBindingMode: WaitForFirstConsumer
parameters:
  type: gp3
  encrypted: "true"
```

2. 套用 `StorageClass`：

```
kubectl apply -f storage-class.yaml
```

3. 建立持久性磁碟區宣告 (PVC) 來請求遊戲資料的儲存。建立名為 的檔案，`ebs-pvc.yaml`並將下列內容新增至其中：

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: game-data-pvc
  namespace: game-2048
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 10Gi
  storageClassName: auto-ebs-sc
```

4. 將 PVC 套用至您的叢集：

```
kubectl apply -f ebs-pvc.yaml
```

您應該會看到下列回應輸出：

```
persistentvolumeclaim/game-data-pvc created
```

5. 現在，您需要更新 2048 遊戲部署，才能使用此 PVC 來存放資料。下列部署設定為使用 PVC 來存放遊戲資料。建立名為 的檔案，ebs-deployment.yaml並將以下內容新增至其中：

```
apiVersion: apps/v1
kind: Deployment
metadata:
  namespace: game-2048
  name: deployment-2048
spec:
  replicas: 3  # Adjust the number of replicas as needed
  selector:
    matchLabels:
      app.kubernetes.io/name: app-2048
  template:
    metadata:
      labels:
        app.kubernetes.io/name: app-2048
    spec:
      containers:
        - name: app-2048
          image: public.ecr.aws/l6m2t8p7/docker-2048:latest
          imagePullPolicy: Always
          ports:
            - containerPort: 80
          volumeMounts:
            - name: game-data
              mountPath: /var/lib/2048
      volumes:
        - name: game-data
          persistentVolumeClaim:
            claimName: game-data-pvc
```

6. 套用更新的部署：

```
kubectl apply -f ebs-deployment.yaml
```

您應該會看到下列回應輸出：

```
deployment.apps/deployment-2048 configured
```

透過這些步驟，您叢集上的 2048 遊戲現在已設定為使用 Amazon EKS Auto Mode 的區塊儲存功能來保留資料。這可確保即使在 Pod 或節點故障時，您的遊戲進度和資料仍然安全。

如果您喜歡此教學課程，請透過提供意見回饋來讓我們知道，以便我們能夠為您提供更多與此相關的使用案例特定快速入門教學課程。

刪除叢集和節點

完成您為本教學課程建立的叢集後，您應該使用以下命令刪除叢集來清除叢集。如果您想要在清除之前對此叢集執行更多操作，請參閱後續步驟。

```
eksctl delete cluster -f ./cluster-config.yaml
```

刪除叢集時，EKS 會自動清除其佈建的任何節點。

依範例了解 Amazon EKS

概觀

本 Amazon EKS 使用者指南包含從命令列或 建立第一個 EKS 叢集的一般用途程序，AWS Management Console以及所有主要 Amazon EKS 元件的可靠參考。不過，身為 Amazon EKS 叢集管理員或開發人員，您可以遵循本指南以外網站中存在的學習路徑，深入了解 Amazon EKS。這些網站可協助您：

- 設定特定類型的叢集。特定叢集類型可以根據您的工作負載類型或安全需求而定。例如，您可能想要調整叢集以執行批次、機器學習或運算密集型工作負載。
- 增強您的叢集。您可以將進階功能新增至叢集，以提供可觀測性、彈性儲存、自動擴展或特殊化叢集聯網等功能。
- 自動化更新。使用 GitOps 等功能，您可以根據 Git 儲存庫中這些元件發生的變更，設定 以自動佈建叢集基礎設施和工作負載。
- 使用進階叢集設定工具。雖然 `eksctl`提供快速建立叢集的方式，但還有其他工具可讓您更輕鬆地設定和升級更複雜的叢集。這些包括 Terraform 和 CloudFormation 等工具。

若要開始您的 Amazon EKS 學習路徑，建議您瀏覽此頁面所述的一些網站。如果您在過程中遇到問題，也有資源可協助您解決這些問題。例如，Re：post 知識中心可讓您搜尋支援資料庫是否有 Amazon EKS 相關的支援問題。此外，Amazon EKS 最佳實務指南也提供設定生產級叢集之最佳方式的秘訣。

Amazon EKS 研討會

從對 Kubernetes 和容器的基本了解開始，Amazon EKS 研討會是一個學習平台，用於引導叢集管理員了解 Amazon EKS 的重要功能。以下是您可以與 Amazon EKS 研討會互動的方式：

- Amazon EKS 基本概念：觀看簡介頁面上的影片，了解 Amazon EKS 如何在 AWS 雲端實作 Kubernetes 功能。如果您需要更基本的 Kubernetes 了解，請觀看什麼是 Kubernetes 影片。
- Amazon EKS 設定：如果您有 AWS 帳戶，設定區段可協助您設定 CloudShell 環境，以建立叢集。它提供 eksctl （簡單的叢集建立命令列） 和 Terraform （建立叢集的更多infrastructure-as-code方法） 的選擇，用於建立 Amazon EKS 叢集。

- Amazon EKS 入門：從範例應用程式區段嘗試簡單的 Web 商店。您可以在其他練習中使用此功能。在本節中，您也可以了解封裝容器映像，以及如何使用 Kubernetes Pod、部署、服務、StatefulSets 和命名空間來管理微服務。然後使用 Kustomize 將變更部署到 Kubernetes 資訊清單。
- Amazon EKS 基礎知識：使用AWS Load Balancer控制器等 AWS 功能，研討會會示範如何向外界公開您的應用程式。對於儲存，研討會展示如何使用 Amazon EBS 進行區塊儲存、使用 Amazon EFS 進行檔案系統儲存，以及使用 Amazon FSx for NetApp ONTAP 管理 ONTAP 檔案系統 AWS。對於節點管理，研討會可協助您設定受管節點群組。
- Amazon EKS 進階功能：Amazon EKS 研討會提供的更多進階功能包括設定實驗室：
  - 自動擴展：這包括節點自動擴展（使用 Cluster Autoscaler 或 Karpenter) 和工作負載自動擴展（使用 Horizontal Pod Autoscaler 和 Cluster proportional Autoscaler)。
  - 可觀測性：了解在一組可觀測性實驗室中記錄、OpenSearch、Amazon EKS 上的 Container Insights，以及 Kubecost 的成本可見性。 https://www.eksworkshop.com/docs/observability/
  - 安全性：這組安全性實驗室可讓您探索 Secrets Management、Amazon GuardDuty、Pod 安全標準和 Kyverno 政策管理。
  - 網路：從網路實驗室了解 Amazon EKS 的網路功能，其中包含 Amazon VPC CNI （支援網路外掛程式）和 Amazon VPC Lattice （用於設定跨 VC 和使用者帳戶的叢集）。
  - 自動化：自動化實驗室會逐步引導您完成管理叢集和專案的 GitOps 方法，例如 AWS Kubernetes 的控制器和用於管理 Amazon EKS 控制平面的跨平面。

Amazon EKS 實作叢集設定教學課程

AWS 社群網站上的一組 Amazon EKS 叢集設定教學課程可協助您建立特殊用途的 Amazon EKS 叢集，並以各種方式增強這些叢集。教學課程分為三種不同的類型：

建置叢集

這些教學課程可協助您建置可用於特殊用途的叢集。這些特殊目的包括執行以下項目的能力：

- 以 IPv6 為基礎的全球可擴展應用程式
- 非同步批次任務
- 高流量微服務
- 在 Fargate 上使用 Karpenter 自動擴展
- 金融工作負載
- Windows 受管節點群組

增強叢集

擁有現有的叢集後，您可以擴展和增強該叢集，以允許它執行專門的工作負載，否則增強叢集。這些教學課程包含以下方法：

- 使用 EFS CSI 提供儲存解決方案
- 使用 EBS CSI 提供動態資料庫儲存
- 使用 AWS Load Balancer控制器在 IPv4 叢集上公開應用程式
- 使用 AWS Load Balancer控制器公開 IPv6 叢集上的應用程式

最佳化 AWS 服務

使用這些教學課程，您可以更好地將叢集與 AWS 服務整合。這些教學課程包括可協助您：

- 使用 ExternalDNS 管理微服務的 DNS 記錄
- 使用 CloudWatch 監控應用程式
- 使用 SQS 和 EFS 儲存體管理非同步任務
- 從工作負載使用 AWS Secrets Manager 秘密
- 使用 Fargate、NGINX 和 ACM PCA 設定 mTLS

Amazon EKS 範例

Amazon EKS 範例儲存庫會存放資訊清單，以便與 Amazon EKS 搭配使用。這些資訊清單可讓您在 Amazon EKS 中嘗試不同類型的應用程式，或建立特定類型的 Amazon EKS 叢集。範例包含資訊清單以：

- 建立 AWS Amazon EKS Fargate 叢集
- 使用現有的 IAM 角色建立叢集
- 將 和 Ubuntu 受管節點群組新增至叢集
- 使用磁碟區快照備份和還原 Pod 儲存
- 復原掛載為具有多個帳戶的 PVCs EBS 磁碟區
- 使用 Classic Load Balancer 啟用 NGINX 傳入控制器的代理通訊協定
- 在 Fargate 上設定記錄至 AWS OpenSearch
- 使用 Web 聯合身分提供者執行 Python SDK

- 在 NFS CSI 控制器上部署範例應用程式
- 使用 StatefulSets 的磁碟區快照
- 在不同可用區域跨節點部署 Pod

請記住，這些範例僅用於學習和測試目的，不適用於生產環境。

AWS 教學課程

AWS 教學課程網站會發佈一些 Amazon EKS 教學課程，但也提供搜尋工具來尋找在 AWS 網站（例如 AWS 社群網站）上發佈的其他教學課程。直接在此網站上發佈的 Amazon EKS 教學課程包括：

- 在 Amazon EKS 上部署容器 Web 應用程式
- 以較低的價格執行 Kubernetes 叢集 (Amazon EKS 和 Spot 執行個體)
- 如何在 Kubernetes 上將 Jenkins 任務成本最佳化

開發人員研討會

如果您是軟體開發人員，想要建立或重構應用程式以在 Amazon EKS 上執行，Amazon EKS 開發人員研討會是很好的開始。研討會不僅可協助您建置容器化應用程式，還可協助您將這些容器部署至容器登錄檔 (ECR)，以及從該處部署至 Amazon EKS 叢集。

從 Amazon EKS Python 研討會開始，逐步重構 Python 應用程式，然後設定開發環境以準備部署應用程式。逐步介紹 Containers、Kubernetes 和 Amazon EKS 的章節，以準備在這些環境中執行容器化應用程式。

Terraform 研討會

雖然 `eksctl` 是建立叢集的簡單工具，但對於更複雜的 infrastructure-as-code類型，Terraform 是熱門的 Amazon EKS 叢集建立和管理工具。Terraform Amazon EKS 研討會會教導如何使用 Terraform 來建置 AWS VPC、建立 Amazon EKS 叢集，以及將選用的增強功能新增至叢集。特別是，有一個用於建立私有 Amazon EKS 叢集的區段

AWS Amazon EKS 訓練

AWS 提供正式訓練，以了解 Amazon EKS。在 Amazon Elastic Kubernetes Service 上執行容器的為期三天的培訓課程教導：

- Kubernetes 和 Amazon EKS 基本概念
- 如何建置 Amazon EKS 叢集
- 使用 IAM 和 Kubernetes RBAC AWS 授權保護 Amazon EKS
- GitOps 自動化工具
- 監控工具
- 改善成本、效率和彈性的技術

開始使用 Amazon EKS

在閱讀此入門指南之前，請確保您已完成 Amazon EKS 的設定並可開始使用。如需詳細資訊，請參閱設定。

對於在 Amazon EKS 中建立含節點的新 Kubernetes 叢集，我們提供兩種入門指南：

- Amazon EKS 入門 – eksctl – 此入門指南可協助您安裝所有必要的資源，以使用 Amazon EKS 入門`eksctl`，這是在 Amazon EKS 上建立和管理 Kubernetes 叢集的簡單命令列公用程式。教學結束時，您將擁有一個執行中的 Amazon EKS 叢集，而您可以在其中部署應用程式。這是 Amazon EKS 入門的最快且最簡易的方式。
- 開始使用 Amazon EKS – AWS Management Console 和 AWS CLI – 此入門指南可協助您使用 AWS Management Console 和 CLI 建立開始使用 Amazon EKS AWS 所需的所有必要資源。教學結束時，您將擁有一個執行中的 Amazon EKS 叢集，而您可以在其中部署應用程式。在本指南中，您可以手動建立 Amazon EKS 叢集所需的各個資源。這些程序可讓您完全了解每個資源的建立方式，以及它們彼此之間的互動方式。

我們也提供下列參考：

- 如需實作教學課程的集合，請參閱 AWS 社群上的 EKS 叢集設定。
- 如需程式碼範例，請參閱使用 AWS SDKs的 Amazon EKS 程式碼範例。

開始使用 Amazon EKS – EKS Auto 模式

如同其他 EKS 入門體驗，使用 EKS Auto Mode 建立您的第一個叢集會將叢集本身的管理委派給 AWS。不過，EKS Auto Mode 會透過處理設定工作負載基礎設施（節點、網路和各種服務）所需的許多基本服務，來擴展 EKS 自動化，讓您更輕鬆地管理節點和擴展以滿足工作負載需求。

選擇下列其中一種方式，以使用 EKS Auto 模式建立叢集：

- the section called “ AWS CLI”：使用`aws`命令列界面來建立叢集。
- the section called “管理主控台”：使用 AWS Management Console 建立叢集。
- the section called “eksctl CLI”：使用`eksctl`命令列界面來建立叢集。

如果您要比較建立第一個 EKS 叢集的不同方法，您應該知道 EKS Auto Mode 已 AWS 接管額外的叢集管理責任，包括設定元件以：

- 隨著工作負載需求增加和減少，啟動和擴展節點。
- 定期升級叢集本身（控制平面）、節點作業系統，以及在節點上執行的服務。
- 選擇預設設定，以決定節點儲存和 Pod 網路組態的大小和速度。

如需 EKS Auto Mode 叢集的詳細資訊，請參閱 EKS 自動模式。

開始使用 Amazon EKS – **eksctl**

> **Note**
>
> 本主題涵蓋不使用 EKS Auto 模式的入門。
> EKS Auto Mode 可自動化叢集運算、儲存和聯網的例行任務。了解如何開始使用 Amazon EKS Auto 模式。

本指南可協助您安裝在將 Amazon Elastic Kubernetes Service (Amazon EKS) 和 `eksctl` 搭配使用所需的所有資源，後者是簡單的命令列公用程式，可用於在 Amazon EKS 上建立和管理 Kubernetes 叢集。此教學結束時，您將擁有一個執行中的 Amazon EKS 叢集，而您可以在其中部署應用程式。

本指南中的程序會為您自動建立數個資源，而您必須在使用 AWS Management Console建立叢集時手動建立這些資源。如果您寧願手動建立大多數資源，以更了解它們如何彼此互動，請使用 AWS Management Console 來建立叢集和運算。如需詳細資訊，請參閱the section called “建立叢集（主控台和 CLI)”。

先決條件

開始本教學課程之前，您必須安裝和設定 AWS CLI、kubectl 和 eksctl 工具，如設定以使用 Amazon EKS 中所述。

步驟 1：建立 Amazon EKS 叢集和節點

> **Important**
>
> 為了盡可能簡單快速地開始使用，本主題包含使用預設設定建立叢集和節點的步驟。建立叢集和節點供生產使用之前，推薦您先熟悉所有設定，並使用符合您需求的設定來部署叢集和節點。如需詳細資訊，請參閱 the section called “建立叢集” 和 管理運算。只有在建立叢集和節點時，才能夠啟用某些設定。

您可以使用下列其中一種節點類型來建立叢集。若要進一步了解每種類型，請參閱 管理運算。叢集部署完成後，您可以新增其他節點類型。

- Fargate – Linux – 如果您想要在 上執行 Linux 應用程式，請選取此類型的節點the section called “ AWS Fargate”。Fargate 是無伺服器運算引擎，可讓您部署 Kubernetes Pod，而無需管理 Amazon EC2 執行個體。
- 受管節點 – Linux：如果您想要在 Amazon EC2 執行個體上執行 Amazon Linux 應用程式，請選取此節點類型。雖然未涵蓋在本指南中，但您也可以將 Windows 自我管理和 Bottlerocket 節點新增至您的叢集。

透過以下命令建立 Amazon EKS 叢集。您可以將 *my-cluster* 取代為您自己的值。此名稱僅能使用英數字元 (區分大小寫) 和連字號。它必須以英數字元開頭，且長度不可超過 100 個字元。名稱在您要建立叢集 AWS 的區域和 AWS 帳戶中必須是唯一的。將 *region-code* 取代為 Amazon EKS 支援的任何 AWS 區域。如需 AWS 區域清單，請參閱《 AWS 一般參考指南》中的 Amazon EKS 端點和配額。

Example

Fargate - Linux

```
eksctl create cluster --name my-cluster --region region-code --fargate
```

Managed nodes - Linux

```
eksctl create cluster --name my-cluster --region region-code
```

叢集建立需要幾分鐘的時間。在建立期間，您會看到幾行輸出。輸出的最後一行類似於下面的範例行。

```
[...]
[#]  EKS cluster "my-cluster" in "region-code" region is ready
```

`eksctl` 在 `kubectl` 中建立組態檔案，`~/.kube/config`或在電腦上 的現有組態檔案中新增新叢集`~/.kube/config`的組態。

叢集建立完成後，請在 AWS CloudFormation 主控台`eksctl-`*`my-cluster`*`-cluster`中檢視名為 的 AWS CloudFormation 堆疊，以查看所有已建立的資源。

步驟 2：檢視 Kubernetes 資源

1. 檢視叢集節點。

```
kubectl get nodes -o wide
```

範例輸出如下。

Example

Fargate - Linux

```
NAME                                                  STATUS   ROLES    AGE
 VERSION              INTERNAL-IP   EXTERNAL-IP   OS-IMAGE         KERNEL-VERSION
                CONTAINER-RUNTIME
fargate-ip-192-0-2-0.region-code.compute.internal   Ready    <none>
 8m3s    v1.2.3-eks-1234567   192.0.2.0     <none>        Amazon Linux 2
 1.23.456-789.012.amzn2.x86_64   containerd://1.2.3
fargate-ip-192-0-2-1.region-code.compute.internal   Ready    <none>
 7m30s   v1.2.3-eks-1234567   192-0-2-1     <none>        Amazon Linux 2
 1.23.456-789.012.amzn2.x86_64   containerd://1.2.3
```

Managed nodes - Linux

```
NAME                                       STATUS   ROLES    AGE    VERSION
        INTERNAL-IP   EXTERNAL-IP   OS-IMAGE         KERNEL-VERSION
   CONTAINER-RUNTIME
ip-192-0-2-0.region-code.compute.internal   Ready    <none>   6m7s
  v1.2.3-eks-1234567   192.0.2.0     192.0.2.2     Amazon Linux 2
 1.23.456-789.012.amzn2.x86_64   containerd://1.2.3
ip-192-0-2-1.region-code.compute.internal   Ready    <none>   6m4s
  v1.2.3-eks-1234567   192.0.2.1     192.0.2.3     Amazon Linux 2
 1.23.456-789.012.amzn2.x86_64   containerd://1.2.3
```

如需您在輸出中看到之內容的詳細資訊，請參閱 the section called “存取叢集資源”。

2. 檢視叢集上執行的工作負載。

```
kubectl get pods -A -o wide
```

範例輸出如下。

Example

Fargate - Linux

```
NAMESPACE     NAME                          READY   STATUS    RESTARTS   AGE   IP
          NODE                                                  NOMINATED NODE
 READINESS GATES
kube-system   coredns-1234567890-abcde   1/1     Running   0          18m
 192.0.2.0   fargate-ip-192-0-2-0.region-code.compute.internal   <none>
 <none>
kube-system   coredns-1234567890-12345   1/1     Running   0          18m
 192.0.2.1   fargate-ip-192-0-2-1.region-code.compute.internal   <none>
 <none>
```

Managed nodes - Linux

```
NAMESPACE     NAME                          READY   STATUS    RESTARTS   AGE     IP
          NODE                                          NOMINATED NODE   READINESS
 GATES
kube-system   aws-node-12345             1/1     Running   0          7m43s
 192.0.2.1   ip-192-0-2-1.region-code.compute.internal   <none>           <none>
kube-system   aws-node-67890             1/1     Running   0          7m46s
 192.0.2.0   ip-192-0-2-0.region-code.compute.internal   <none>           <none>
kube-system   coredns-1234567890-abcde   1/1     Running   0          14m
 192.0.2.3   ip-192-0-2-3.region-code.compute.internal   <none>           <none>
kube-system   coredns-1234567890-12345   1/1     Running   0          14m
 192.0.2.4   ip-192-0-2-4.region-code.compute.internal   <none>           <none>
kube-system   kube-proxy-12345           1/1     Running   0          7m46s
 192.0.2.0   ip-192-0-2-0.region-code.compute.internal   <none>           <none>
kube-system   kube-proxy-67890           1/1     Running   0          7m43s
 192.0.2.1   ip-192-0-2-1.region-code.compute.internal   <none>           <none>
```

如需您在輸出中看到之內容的詳細資訊，請參閱 the section called “存取叢集資源”。

步驟 3：刪除您的叢集和節點

完成您為此教學課程建立的叢集和節點後，您應該使用以下命令刪除叢集和節點來清除 。如果想在清理前使用此叢集執行更多作業，請參閱 the section called “後續步驟”。

```
eksctl delete cluster --name my-cluster --region region-code
```

後續步驟

以下文件主題可協助您擴展叢集的功能。

- 將範例應用程式部署至叢集。
- 建立叢集的 IAM 主體是唯一可以使用 `kubectl`或 呼叫 Kubernetes API 伺服器的主體 AWS Management Console。如果要其他 IAM 主體能夠存取叢集，則需新增這些主體。如需詳細資訊，請參閱 the section called “Kubernetes API 存取” 和 the section called “所需的許可”。
- 部署叢集以供生產使用之前，建議您先熟悉叢集和節點的所有設定。建立叢集時，必須進行某些設定 (例如啟用對 Amazon EC2 節點的 SSH 存取)。
- 若要提高叢集的安全性，可將 Amazon VPC 容器聯網介面外掛程式設定為使用服務帳戶的 IAM 角色。

開始使用 Amazon EKS – AWS Management Console 和 AWS CLI

> **Note**
>
> 本主題涵蓋不使用 EKS Auto 模式的入門。它使用受管節點群組來部署節點。
> EKS Auto Mode 可自動化叢集運算、儲存和聯網的例行任務。了解如何開始使用 Amazon EKS Auto 模式。 EKS Auto Mode 是部署節點的偏好方法。

本指南可協助您建立所有必要的資源，以使用 AWS Management Console 和 CLI 開始使用 Amazon Elastic Kubernetes Service AWS (Amazon EKS)。在本指南中，您可以手動建立每個資源。此教學結束時，您將擁有一個執行中的 Amazon EKS 叢集，而您可以在其中部署應用程式。

本指南中的程序可讓您完全瞭解每個資源的建立方式，以及資源彼此之間的互動方式。如果您想要自動為您建立大部分的資源，請使用 `eksctl` CLI 來建立叢集和節點。如需詳細資訊，請參閱the section called “建立叢集 (eksctl)”。

先決條件

開始此教學之前，您必須先安裝和設定下列在建立和管理 Amazon EKS 叢集所需的工具和資源。

- AWS CLI – 用於使用 AWS 服務的命令列工具，包括 Amazon EKS。如需詳細資訊，請參閱《 AWS 命令列界面使用者指南》中的安裝。安裝 CLI AWS 之後，建議您也進行設定。如需詳細資訊，請參

閱《 AWS 命令列界面使用者指南》中的具有 aws 設定的快速組態。請注意，使用此頁面中顯示的 update-kubeconfig 選項需要 AWS CLI v2。

- **kubectl** – 使用 Kubernetes 叢集的命令列工具。如需詳細資訊，請參閱the section called “設定 kubectl 和 eksctl”。
- 必要的 IAM 許可 – 您使用的 IAM 安全主體必須具有使用 Amazon EKS IAM 角色、服務連結角色、AWS CloudFormation、VPC 和相關資源的許可。如需詳細資訊，請參閱《IAM 使用者指南》中的動作和使用服務連結角色。您必須以同一位使用者的身分完成本指南中的所有步驟。若要檢查目前使用者，請執行以下命令：

```
aws sts get-caller-identity
```

建議您在 Bash shell 中完成本主題中的步驟。如果您不使用 Bash shell，一些指令碼命令，例如行接續字元和變數的設定和使用方式需要調整您的 shell。此外，您的 Shell 的引用及轉義規則可能會有所不同。如需詳細資訊，請參閱《 AWS 命令列界面使用者指南》中的使用引號搭配 AWS CLI 中的字串。

步驟 1：建立您的 Amazon EKS 叢集

> **Important**
>
> 為了盡可能簡單快速地開始使用，本主題包含使用預設設定建立叢集的步驟。建立叢集供生產使用之前，建議您先熟悉所有設定，再使用符合需求的設定來部署叢集。如需詳細資訊，請參閱the section called “建立叢集”。只有在建立叢集時，才能夠啟用某些設定。

1. 使用符合 Amazon EKS 要求的公有和私有子網路建立 Amazon VPC。將 *region-code* 取代為 Amazon EKS 支援的任何 AWS 區域。如需 AWS 區域清單，請參閱《 AWS 一般參考指南》中的 Amazon EKS 端點和配額。您可以使用選擇的任何名稱取代 *my-eks-vpc-stack*。

```
aws cloudformation create-stack \
  --region region-code \
  --stack-name my-eks-vpc-stack \
  --template-url https://s3.us-west-2.amazonaws.com/amazon-eks/
cloudformation/2020-10-29/amazon-eks-vpc-private-subnets.yaml
```

> **Tip**
>
> 如需上一個命令建立的所有資源清單，請開啟位於 https：//https://console.aws.amazon.com/cloudformation/ 的 AWS CloudFormation 主控台。選擇 *my-eks-vpc-stack* 堆疊，然後選擇 Resources (資源) 索引標籤。

2. 建立叢集 IAM 角色，並將所需的 Amazon EKS IAM 受管政策連接到該角色。Amazon EKS 管理的 Kubernetes 叢集會代表您呼叫其他 AWS 服務，以管理您與服務搭配使用的資源。

   a. 將以下內容複製到名為 *eks-cluster-role-trust-policy.json* 的檔案。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "eks.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

   b. 建立角色。

```
aws iam create-role \
  --role-name myAmazonEKSClusterRole \
  --assume-role-policy-document file://"eks-cluster-role-trust-policy.json"
```

   c. 將必要的 Amazon EKS 受管 IAM 政策連接到角色。

```
aws iam attach-role-policy \
  --policy-arn arn:aws: iam::aws:policy/AmazonEKSClusterPolicy \
  --role-name myAmazonEKSClusterRole
```

3. 在 開啟 Amazon EKS 主控台https://console.aws.amazon.com/eks/home#/clusters。

   請確定主控台右上角顯示的 AWS 區域是您要建立叢集 AWS 的區域。如果不是，請選擇 AWS 區域名稱旁的下拉式清單，然後選擇您要使用的 AWS 區域。

4. 選擇 建立叢集。如果您沒有看到此選項，請先在左側導覽窗格中選擇叢集。

5. 在 Configure cluster (設定叢集) 頁面上，執行下列操作：
   a. 選取自訂組態並停用使用 EKS 自動模式。（如果您偏好 EKS Auto Mode 叢集，請改為參考 the section called “管理主控台”。）
   b. 輸入叢集的名稱，例如 *my-cluster*。此名稱僅能使用英數字元 (區分大小寫) 和連字號。它必須以英數字元開頭，且長度不可超過 100 個字元。名稱在您要建立叢集 AWS 的區域和 AWS 帳戶中必須是唯一的。
   c. 對於 Cluster Service Role (叢集服務角色)，選擇 *myAmazonEKSClusterRole*。
   d. 將其餘設定維持在其預設值，然後選取 Next (下一步)。
6. 在 Specify networking (指定聯網) 頁面中，執行下列操作：
   a. 從 VPC 下拉式清單中選擇在先前步驟中建立的 VPC ID。其類似 ** | my-eks-vpc-stack-VPC*。
   b. 從子網路下拉式清單中選擇上一個步驟中建立的子網路。子網路會像是 ** | my-eks-vpc-stack-**。
   c. 從其他安全群組下拉式清單中選擇在上一個步驟中建立的安全群組。其類似 ** | my-eks-vpc-stack-ControlPlaneSecurityGroup-**。
   d. 將其餘設定維持在其預設值，然後選取 Next (下一步)。
7. 在設定可觀測性頁面上，選擇下一步。
8. 在選取附加元件頁面上，選擇下一步。

   如需附加元件的詳細資訊，請參閱 the section called “Amazon EKS 附加元件”。
9. 在設定選取的附加元件設定頁面中，選取下一步：
10. 在 Review and create (檢閱和建立) 頁面上，選取 Create (建立)。

    在叢集名稱的右側，叢集狀態為建立幾分鐘，直到叢集佈建程序完成為止。在狀態為作用中之前，請勿繼續下一個步驟。

> **Note**
>
> 您可能會收到錯誤，告知您請求中的其中一個可用區域沒有足夠的容量來建立 Amazon EKS 叢集。如果發生這種情況，錯誤輸出包含的可用區域可支援新的叢集。使用至少兩個位於帳戶的支援可用區域子網路來建立您的叢集。如需詳細資訊，請參閱the section called “容量不足”。

步驟 2：將您的電腦設為與叢集進行通訊

在此區段中，您會為叢集建立 kubeconfig 檔案。此檔案中的設定會啟用 kubectl CLI 與您的叢集進行通訊。

在繼續之前，請確定您的叢集建立已在步驟 1 中成功完成。

1. 為您的叢集建立或更新 kubeconfig 檔案。將 *region-code* 取代 AWS 為您建立叢集的區域。使用您叢集的名稱取代 *my-cluster*。

   ```
   aws eks update-kubeconfig --region region-code --name my-cluster
   ```

   根據預設， config 檔案會在 中建立，~/.kube或新叢集的組態會新增至 中的現有config檔案~/.kube。
2. 測試組態。

   ```
   kubectl get svc
   ```

   > **Note**
   >
   > 如果您收到任何授權或資源類型錯誤，請參閱故障診斷主題中的the section called “未經授權或存取遭拒 (kubectl)”。

   範例輸出如下。

   ```
   NAME             TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
   svc/kubernetes   ClusterIP   10.100.0.1   <none>        443/TCP   1m
   ```

步驟 3：建立節點

> **Important**
>
> 為了盡可能簡單快速地開始使用，本主題包含以大部分預設設定建立節點的步驟。建立節點供生產使用之前，建議您先熟悉所有設定，再使用符合需求的設定來部署節點。如需詳細資訊，請參閱管理運算。只有在建立節點時，才能夠啟用某些設定。

此程序會將您的叢集設定為使用受管節點群組來建立節點，並指定您在先前步驟中建立的子網路和節點 IAM 角色。它可讓您在 Amazon EC2 執行個體上執行 Amazon Linux 應用程式。

若要進一步了解在 EKS 中設定節點的不同方式，請參閱 管理運算。叢集部署完成後，您可以新增其他節點類型。雖然未涵蓋在本指南中，但您也可以將 Windows 自我管理和 Bottlerocket 節點新增至您的叢集。

建立 EC2 Linux 受管節點群組

1. 建立節點 IAM 角色，並將所需的 Amazon EKS IAM 受管政策連接到該角色。Amazon EKS `kubelet` 節點協助程式會代表您呼叫 AWS APIs。節點透過 IAM 執行個體描述檔和關聯的政策，取得這些 API 呼叫的許可。

   a. 將下列內容複製到名為 `node-role-trust-policy.json` 的檔案。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "ec2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

   b. 建立節點 IAM 角色。

```
aws iam create-role \
  --role-name myAmazonEKSNodeRole \
```

```
  --assume-role-policy-document file://"node-role-trust-policy.json"
```

c. 將所需的受管 IAM 政策連接到角色。

```
aws iam attach-role-policy \
  --policy-arn arn:aws: iam::aws:policy/AmazonEKSWorkerNodePolicy \
  --role-name myAmazonEKSNodeRole
aws iam attach-role-policy \
  --policy-arn arn:aws: iam::aws:policy/AmazonEC2ContainerRegistryReadOnly \
  --role-name myAmazonEKSNodeRole
aws iam attach-role-policy \
  --policy-arn arn:aws: iam::aws:policy/AmazonEKS_CNI_Policy \
  --role-name myAmazonEKSNodeRole
```

d. 在 開啟 Amazon EKS 主控台https://console.aws.amazon.com/eks/home#/clusters。

e. 選擇您在步驟 1：建立 Amazon EKS 叢集中建立的叢集名稱，例如 *my-cluster*。

f. 在 ***my-cluster*** 頁面上，執行下列動作：

g. 選擇 Compute (運算) 索引標籤。

h. 選擇 Add Node Group (新增節點群組)。

2. 在 Configure Node Group (設定節點群組) 頁面上，執行以下操作：

   a. 針對名稱，輸入受管節點群組的唯一名稱，例如 *my-nodegroup*。節點群組名稱不能超過 63 個字元。它必須以字母或數字開頭，但剩餘字元也可以包含連字符和底線。

   b. 對於 Node IAM role name (節點 IAM 角色名稱)，選擇您在前一個步驟建立的 *myAmazonEKSNodeRole* 角色。建議每個節點群組使用其唯一的 IAM 角色。

   c. 選擇下一步。

3. 在 Set compute and scaling configuration (設定運算和擴展組態) 頁面上，接受預設值，然後選取 Next (下一步)。

4. 在 Specify networking (指定聯網) 頁面上，接受預設值，然後選取 Next (下一步)。

5. 在 Review and create (檢閱並建立) 頁面上，檢閱您的受管節點群組組態，然後選擇 Create (建立)。

6. 數分鐘後，Node Group configuration (節點群組組態) 區段中的 Status (狀態) 將從 Creating (建立中) 變更為 Active (作用中)。在狀態為作用中之前，請勿繼續下一個步驟。

步驟 4：檢視資源

您可以檢視節點和 Kubernetes 工作負載。

1. 在左側導覽窗格中選擇叢集。在 Clusters (叢集) 清單中，選擇您建立之叢集的名稱，例如 *my-cluster*。
2. 在 ***my-cluster*** 頁面上，選擇下列項目：
   a. Compute (運算) 標籤 – 您可以看到為叢集部署的 Nodes (節點) 清單。您可以選取節點的名稱，以查看有關節點的詳細資訊。
   b. 資源索引標籤 – 您可以看到預設部署到 Amazon EKS 叢集的所有 Kubernetes 資源。在主控台中選取任何資源類型，以進一步了解其詳細資訊。

步驟 5：刪除資源

完成您為本教學課程建立的叢集和節點之後，您應該刪除您建立的資源。如果想在刪除資源前使用此叢集執行更多作業，請參閱 the section called “後續步驟”。

1. 刪除您建立的任何節點群組設定檔。
   a. 在 開啟 Amazon EKS 主控台https://console.aws.amazon.com/eks/home#/clusters。
   b. 在左側導覽窗格中選擇叢集。在叢集清單中，選取 *my-cluster*。
   c. 選擇 Compute (運算) 索引標籤。
   d. 如果建立了節點群組，請選擇 *my-nodegroup* 節點群組，然後選擇 Delete (刪除)。輸入 *my-nodegroup*，然後選擇刪除。
   e. 在刪除節點群組設定檔之前，請勿繼續。
2. 刪除叢集。
   a. 在左側導覽窗格中選擇叢集。在叢集清單中，選取 *my-cluster*。
   b. 選擇 Delete cluster (刪除叢集)。
   c. 輸入 *my-cluster*，然後選擇刪除。在刪除叢集之前，請勿繼續。
3. 刪除您建立的 VPC AWS CloudFormation 堆疊。
   a. 開啟位在 https://console.aws.amazon.com/cloudformation/ 的 CloudFormation 主控台。
   b. 選取 *my-eks-vpc-stack* 堆疊，然後選擇 Delete (刪除)。
   c. 在刪除 ***my-eks-vpc-stack*** 確認對話方塊中，選擇刪除堆疊。
4. 刪除您建立的 IAM 角色。
   a. 前往 https://console.aws.amazon.com/iam/ 開啟 IAM 主控台。
   b. 在左側導覽窗格中，選擇 Roles (角色)。

c. 從清單中選取您建立的每個角色 (*myAmazonEKSClusterRole* 以及 *myAmazonEKSNodeRole*)。選擇 Delete (刪除)，輸入請求的確認文字，然後選擇 Delete (刪除)。

後續步驟

以下文件主題可協助您擴展叢集的功能。

- 建立叢集的 IAM 主體是唯一可以使用 `kubectl`或 呼叫 Kubernetes API 伺服器的主體 AWS Management Console。如果要其他 IAM 主體能夠存取叢集，則需新增這些主體。如需詳細資訊，請參閱the section called “Kubernetes API 存取”及the section called “所需的許可”。
- 將範例應用程式部署至叢集。
- 部署叢集以供生產使用之前，建議您先熟悉叢集和節點的所有設定。建立叢集時，必須進行某些設定 (例如啟用對 Amazon EC2 節點的 SSH 存取)。
- 若要提高叢集的安全性，可將 Amazon VPC 容器聯網介面外掛程式設定為使用服務帳戶的 IAM 角色。

使用 EKS Auto 模式自動化叢集基礎設施

EKS Auto Mode 會將 Kubernetes 叢集的 AWS 管理延伸到叢集本身之外， AWS 以允許 也設定和管理基礎設施，讓工作負載能夠順暢地運作。您可以委派關鍵基礎設施決策，並利用 的專業知識 AWS 進行day-to-day操作。管理的叢集基礎設施 AWS 包含許多 Kubernetes 功能作為核心元件，而不是附加元件，例如運算自動擴展、Pod 和服務聯網、應用程式負載平衡、叢集 DNS、區塊儲存和 GPU 支援。

若要開始使用，您可以在現有叢集上部署新的 EKS Auto Mode 叢集或啟用 EKS Auto Mode。您可以使用 eksctl、 AWS CLI、、EKS APIs 或您偏好的基礎設施即程式碼工具來部署、升級或修改 AWS Management Console EKS Auto Mode 叢集。 infrastructure-as-code

透過 EKS Auto 模式，您可以繼續使用您偏好的 Kubernetes 相容工具。EKS Auto Mode 會與 Amazon EC2、Amazon EBS 和 ELB 等 AWS 服務整合，利用遵循最佳實務的 AWS 雲端資源。這些資源會自動擴展、成本最佳化並定期更新，以協助將營運成本和開銷降至最低。

功能

EKS Auto Mode 提供下列高階功能：

簡化 Kubernetes 叢集管理：EKS Auto Mode 透過為生產就緒的叢集提供最低的操作負荷，簡化 EKS 管理。使用 EKS Auto Mode，您可以放心地執行要求嚴苛的動態工作負載，而不需要深入的 EKS 專業知識。

應用程式可用性：EKS Auto Mode 會根據 Kubernetes 應用程式的需求，動態新增或移除 EKS 叢集中的節點。這可將手動容量規劃的需求降至最低，並確保應用程式的可用性。

效率：EKS Auto Mode 旨在最佳化運算成本，同時遵守 NodePool 和工作負載需求所定義的彈性。它也會終止未使用的執行個體，並將工作負載合併到其他節點，以提高成本效益。

安全性：EKS Auto Mode 會針對您的節點使用被視為不可變AMIs。這些 AMIs會強制執行鎖定的軟體、啟用 SELinux 強制性存取控制，並提供唯讀根檔案系統。此外，EKS Auto Mode 啟動的節點的生命週期上限為 21 天 （您可以減少），之後會自動將其取代為新的節點。這種方法透過定期循環節點來增強您的安全狀態，並符合許多客戶已採用的最佳實務。

自動化升級：EKS Auto Mode 可讓您的 Kubernetes 叢集、節點和相關元件與最新的修補程式保持最新狀態，同時遵守您設定的 Pod 中斷預算 (PDBs) 和 NodePool 中斷預算 NDBs)。最長可達 21 天的生命週期，如果封鎖 PDBs或其他組態阻止更新，則可能需要介入。

受管元件：EKS Auto Mode 包含 Kubernetes 和 AWS 雲端功能作為核心元件，否則必須作為附加元件進行管理。這包括對 Pod IP 地址指派、Pod 網路政策、本機 DNS 服務、GPU 外掛程式、運作狀態檢查程式和 EBS CSI 儲存體的內建支援。

可自訂的 NodePools 和 NodeClasses：如果您的工作負載需要變更儲存、運算或聯網組態，您可以使用 EKS Auto 模式建立自訂的 NodePools 和 NodeClasses。雖然您不應該編輯預設 NodePools 和 NodeClasses，但您可以將新的自訂 NodePools 或 NodeClasses 與預設組態一起新增，以符合您的特定需求。

自動化元件

EKS Auto Mode 透過自動化關鍵基礎設施元件，簡化 Amazon EKS 叢集的操作。啟用 EKS Auto Mode 可進一步減少管理 EKS 叢集的任務。

以下是自動化的資料平面元件清單：

- 運算：對於許多工作負載，使用 EKS Auto Mode，您可能會忘記 EKS 叢集的許多運算層面。其中包含：
  - 節點：EKS Auto Mode 節點的設計可視為設備。EKS Auto Mode 會執行下列動作：
    - 選擇適當的 AMI，該 AMI 已設定許多執行工作負載所需的服務，無需介入。
    - 使用 SELinux 強制執行模式和唯讀根檔案系統，鎖定對 AMI 上檔案的存取。
    - 禁止 SSH 或 SSM 存取，以防止直接存取節點。
    - 包含 GPU 支援，以及適用於 NVIDIA 和 Neuron GPUs 的個別核心驅動程式和外掛程式，可實現高效能工作負載。
    - 自動處理 EC2 Spot 執行個體中斷通知和 EC2 執行個體運作狀態事件
  - 自動擴展：依賴 Karpenter 自動擴展，EKS Auto Mode 會監控不可排程的 Pod，並使新節點可以部署以執行這些 Pod。隨著工作負載終止，EKS Auto Mode 會在不再需要節點時動態中斷和終止節點，將資源用量最佳化。
  - 升級：控制節點可簡化 EKS Auto Mode 視需要提供安全修補程式、作業系統和元件升級的能力。這些升級旨在將工作負載中斷降至最低。EKS Auto Mode 會強制執行最長 21 天的節點生命週期，以確保up-to-date軟體和 APIs。
- 負載平衡：EKS Auto Mode 與 Amazon 的 Elastic Load Balancing 服務整合，自動化 Kubernetes Services 和輸入資源的負載平衡器佈建和組態，以簡化負載平衡。它支援 Application 和 Network Load Balancer 的進階功能、管理其生命週期，並擴展以符合叢集需求。此整合提供符合 AWS 最佳實務的生產就緒負載平衡解決方案，可讓您專注於應用程式，而非基礎設施管理。

- 儲存：EKS Auto Mode 會在節點終止時設定磁碟區類型、磁碟區大小、加密政策和刪除政策，為您設定暫時性儲存。
- 網路：EKS Auto Mode 可自動化 Pod 和服務連線的關鍵網路任務。這包括 IPv4/IPv6 支援，以及使用次要 CIDR 區塊來擴展 IP 地址空間。
- Identity and Access Management：您不需要在 EKS Auto Mode 叢集上安裝 EKS Pod Identity Agent。

有關這些元件的更多資訊，請參閱 the section called “運作方式”。

組態

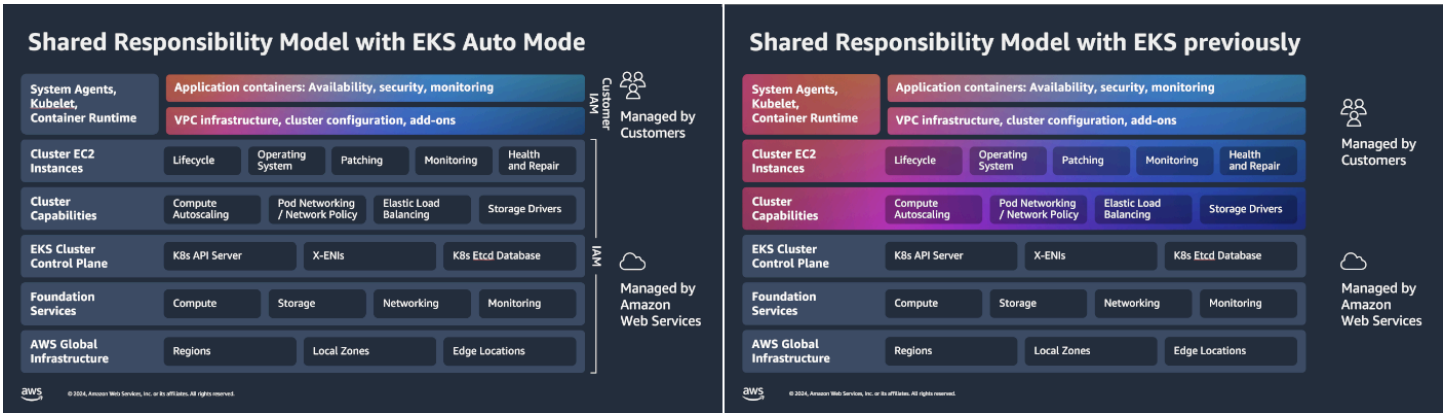
雖然 EKS Auto Mode 會在不介入的情況下有效地管理您的大部分資料平面服務，但有時候您可能會想要變更部分服務的行為。您可以透過下列方式修改 EKS Auto Mode 叢集的組態：

- Kubernetes DaemonSets：您可以改為使用 Kubernetes 協助程式集，而不是修改安裝在節點上的服務。Daemonset 設計為由 Kubernetes 管理，但在叢集中的每個節點上執行。透過這種方式，您可以新增特殊服務來監控或以其他方式監看節點。
- 自訂 NodePools 和 NodeClasses：預設 NodePools 和 NodeClasses 是由 EKS Auto 模式設定，您不應對其進行編輯。若要自訂節點行為，您可以為使用案例建立其他 NodePools 或 NodeClasses，例如：
  - 選取特定執行個體類型（例如，加速處理器或 EC2 Spot 執行個體）。
  - 隔離工作負載以達成安全或成本追蹤目的。
  - 設定暫時性儲存設定，例如 IOPS、大小和輸送量。
- 負載平衡：您可以直接在 EKS Auto Mode 叢集上設定 EKS Auto Mode 作為 Kubernetes 物件執行的某些服務，例如負載平衡。

如需設定 EKS Auto Mode 選項的詳細資訊，請參閱 the section called “設定”。

共同責任模式

AWS 共同責任模型定義 AWS 和 客戶之間的安全與合規責任。以下影像和文字會比較和對比 EKS Auto Mode AWS 和 EKS 標準模式之間的客戶和責任差異。



EKS Auto Mode 會將 Kubernetes 基礎設施的大部分共同責任轉移給客戶 AWS。透過 EKS Auto Mode， AWS 承擔更多雲端安全的責任，這曾經是客戶的責任，現在已共同承擔。客戶現在可以更專注於其應用程式，同時 AWS 管理基礎基礎設施。

客戶責任

在 EKS Auto 模式下，客戶會繼續維護應用程式容器的責任，包括可用性、安全性和監控。它們也會維持對 VPC 基礎設施和 EKS 叢集組態的控制。此模型可讓客戶專注於應用程式特定的考量，同時委派叢集基礎設施管理給 AWS。每個節點的選用功能可以透過 AWS 附加元件包含在叢集中。

AWS 責任

使用 EKS Auto Mode 時， AWS 擴展其責任，包括與已在未使用 Auto Mode 的 EKS 叢集中管理的元件相比，管理幾個額外的關鍵元件。特別是，EKS Auto Mode 會接管 EC2 執行個體啟動的組態、管理、安全性和擴展，以及用於負載平衡、IP 地址管理、聯網政策和區塊儲存的叢集功能。下列元件由 AWS 以 EKS Auto 模式管理：

- 自動模式啟動的 EC2 執行個體：利用 Amazon EC2 受管執行個體 AWS 來處理節點的完整生命週期。EC2 受管執行個體負責作業系統組態、修補、監控和運作狀態維護。在此模型中，執行個體本身和在其上執行的訪客作業系統都是 的責任 AWS。節點使用經最佳化以執行容器的 Bottlerocket AMIs 變體。Bottlerocket AMIs 具有鎖定的軟體、不可變的根檔案系統和安全的網路存取（以防止透過 SSH 或 SSM 直接通訊）。
- Cluster Capabilities： AWS 管理運算自動擴展、具有網路政策強制執行的 Pod 聯網、Elastic Load Balancing 整合和儲存驅動程式組態。
- 叢集控制平面： AWS 繼續管理 Kubernetes API 伺服器、跨帳戶 ENIs 和 等資料庫，如同標準 EKS。
- Foundation Services 和全球基礎設施： AWS 負責基礎運算、儲存、聯網和監控服務，以及區域、本機區域和節點的全球基礎設施。

使用 Amazon EKS Auto 模式建立叢集

本章說明如何使用各種工具和界面建立已啟用自動模式的 Amazon EKS 叢集。自動模式透過自動設定和管理叢集的運算、聯網和儲存基礎設施，簡化叢集建立。您將了解如何使用 AWS CLI AWS Management Console或 eksctl 命令列工具建立自動模式叢集。

> **Note**
>
> EKS Auto Mode 需要 Kubernetes 1.29 版或更新版本。`ap-southeast-7` 或 `mx-central-1` AWS 區域無法使用 EKS Auto Mode。

根據您的需求選擇您偏好的工具： AWS Management Console 提供視覺化界面，非常適合了解 EKS Auto Mode 功能和建立個別叢集。CLI AWS 最適合用於編寫指令碼和自動化任務，特別是將叢集建立整合到現有工作流程或 CI/CD 管道時。eksctl CLI 提供 Kubernetes 原生體驗，建議熟悉 Kubernetes 工具的使用者使用合理的預設值簡化命令列操作。

開始之前，請確定您已安裝並設定必要的先決條件，包括建立 EKS 叢集的適當 IAM 許可。若要了解如何安裝 CLI 工具，例如 `kubectl`、 `aws`和 `eksctl`，請參閱 設定。

您可以使用 AWS CLI AWS Management Console或 eksctl CLI 來建立具有 Amazon EKS Auto 模式的叢集。

主題

- 使用 eksctl CLI 建立 EKS 自動模式叢集
- 使用 CLI AWS 建立 EKS 自動模式叢集
- 使用 建立 EKS 自動模式叢集 AWS Management Console

使用 eksctl CLI 建立 EKS 自動模式叢集

本主題說明如何使用 eksctl 命令列界面 (CLI) 建立 Amazon EKS Auto Mode 叢集。您可以透過執行單一 CLI 命令或套用 YAML 組態檔案來建立自動模式叢集。這兩種方法都提供相同的功能，YAML 方法可更精細地控制叢集設定。

eksctl CLI 透過處理基礎 AWS 資源建立和組態，簡化建立和管理 EKS Auto Mode 叢集的程序。在繼續之前，請確定您已在本機電腦上設定必要的 AWS 登入資料和許可。本指南假設您熟悉基本的 Amazon EKS 概念，並已安裝必要的 CLI 工具。

> **Note**
>
> 您必須安裝 版0.195.0或更新版本的 eksctl。如需詳細資訊，請參閱 GitHub 上的 eksctl 版本。

使用 CLI 命令建立 EKS Auto Mode 叢集

您必須安裝 aws和 eksctl 工具。您必須以足夠的許可登入 AWS CLI，才能管理 AWS 資源，包括：EC2 執行個體、EC2 聯網、EKS 叢集和 IAM 角色。如需詳細資訊，請參閱設定。

執行下列命令，使用 建立新的 EKS Auto Mode 叢集

```
eksctl create cluster --name=<cluster-name> --enable-auto-mode
```

使用 YAML 檔案建立 EKS Auto Mode 叢集

您必須安裝 aws和 eksctl 工具。您必須以足夠的許可登入 AWS CLI，才能管理 AWS 資源，包括：EC2 執行個體、EC2 聯網、EKS 叢集和 IAM 角色。如需詳細資訊，請參閱設定。

檢閱以下 ClusterConfig 資源範例中的 EKS 自動模式組態選項。如需完整的 ClusterConfig 規格，請參閱 eksctl 文件。

AWS 建議啟用 EKS 自動模式。如果這是您第一次建立 EKS Auto Mode 叢集，請保留nodeRoleARN未指定的 ，以建立 EKS Auto Mode 的節點 IAM 角色。如果您的 AWS 帳戶中已有節點 IAM 角色， AWS 建議重複使用它。

AWS 建議不要為 指定任何值nodePools。EKS Auto Mode 將建立預設節點集區。您可以使用 Kubernetes API 來建立其他節點集區。

```
# cluster.yaml
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: <cluster-name>
  region: <aws-region>

iam:
  # ARN of the Cluster IAM Role
  # optional, eksctl creates a new role if not supplied
```

```
  # suggested to use one Cluster IAM Role per account
  serviceRoleARN: <arn-cluster-iam-role>

autoModeConfig:
  # defaults to false
  enabled: boolean
  # optional, defaults to [general-purpose, system].
  # suggested to leave unspecified
  # To disable creation of nodePools, set it to the empty array ([]).
  nodePools: []string
  # optional, eksctl creates a new role if this is not supplied
  # and nodePools are present.
  nodeRoleARN: string
```

將ClusterConfig檔案儲存為 cluster.yaml，並使用下列命令來建立叢集：

```
eksctl create cluster -f cluster.yaml
```

使用 CLI AWS 建立 EKS 自動模式叢集

EKS Auto Mode 叢集可自動化運算、儲存和聯網的例行叢集管理任務。例如，EKS 自動模式叢集會自動偵測何時需要額外節點，並佈建新的 EC2 執行個體以滿足工作負載需求。

本主題會引導您使用 CLI 建立新的 EKS AWS 自動模式叢集，並選擇性地部署範例工作負載。

先決條件

- 在您裝置上安裝和設定的最新版本 AWS 命令列界面 (AWS CLI)。若要檢查您目前的版本，請使用 aws --version。若要安裝最新版本，請參閱《 AWS 命令列界面使用者指南》中的使用 aws 設定安裝 和快速組態。
  - 使用足夠的 IAM 許可登入 CLI，以建立 AWS 資源，包括 IAM 政策、IAM 角色和 EKS 叢集。
- 裝置上安裝的 kubectl 命令列工具。 AWS 建議您使用與 EKS 叢集 Kubernetes 版本相同的 kubectl 版本。若要安裝或升級 kubectl，請參閱 the section called “設定 kubectl 和 eksctl”。

指定 VPC 子網路

Amazon EKS Auto Mode 會將節點部署到 VPC 子網路。建立 EKS 叢集時，您必須指定要部署節點的 VPC 子網路。您可以使用 AWS 帳戶中的預設 VPC 子網路，或為關鍵工作負載建立專用 VPC。

- AWS 建議為您的叢集建立專用 VPC。了解如何 the section called “建立 VPC”。

- EKS 主控台可協助建立新的 VPC。了解如何 the section called “管理主控台”。
- 或者，您可以使用 AWS 您帳戶的預設 VPC。使用下列指示來尋找子網路 IDs。

尋找預設 VPC 的子網路 IDs

使用 AWS CLI：

1. 執行下列命令來列出預設 VPC 及其子網路：

```
aws ec2 describe-subnets --filters "Name=vpc-id,Values=$(aws ec2 describe-vpcs
 --query 'Vpcs[?IsDefault==`true`].VpcId' --output text)" --query 'Subnets[*].
{ID:SubnetId,AZ:AvailabilityZone}' --output table
```

2. 儲存輸出並記下子網路 IDs。

   輸出範例：

```
----------------------------------------
|             DescribeSubnets          |
----------------------------------------
|   SubnetId        |   AvailabilityZone  |
|-------------------|---------------------|
|   subnet-012345678 |   us-west-2a        |
|   subnet-234567890 |   us-west-2b        |
|   subnet-345678901 |   us-west-2c        |
----------------------------------------
```

EKS 自動模式叢集的 IAM 角色

叢集 IAM 角色

EKS Auto Mode 需要叢集 IAM 角色才能在 AWS 帳戶中執行動作，例如佈建新的 EC2 執行個體。您必須建立此角色，才能授予 EKS 必要的許可。 AWS 建議將下列 AWS 受管政策連接至叢集 IAM 角色：

- AmazonEKSComputePolicy
- AmazonEKSBlockStoragePolicy
- AmazonEKSLoadBalancingPolicy
- AmazonEKSNetworkingPolicy

- AmazonEKSClusterPolicy

節點 IAM 角色

當您建立 EKS Auto Mode 叢集時，您可以指定節點 IAM 角色。當 EKS Auto Mode 建立節點來處理待處理的工作負載時，每個新的 EC2 執行個體節點都會指派節點 IAM 角色。此角色允許節點與 EKS 通訊，但節點上執行的工作負載通常無法存取。

如果您想要將許可授予節點上執行的工作負載，請使用 EKS Pod Identity。如需詳細資訊，請參閱the section called “Pod 身分”。

您必須建立此角色並連接下列 AWS 受管政策：

- AmazonEKSWorkerNodeMinimalPolicy
- AmazonEC2ContainerRegistryPullOnly

服務連結角色

EKS Auto Mode 也需要服務連結角色，由 自動建立和設定 AWS。如需詳細資訊，請參閱 AWSServiceRoleForAmazonEKS。

建立 EKS 自動模式叢集 IAM 角色

步驟 1：建立信任政策

建立允許 Amazon EKS 服務擔任角色的信任政策。將政策儲存為 `trust-policy.json`：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "eks.amazonaws.com"
      },
      "Action": [
        "sts:AssumeRole",
        "sts:TagSession"
      ]
    }
  ]
```

```
}
```

步驟 2：建立 IAM 角色

使用信任政策來建立叢集 IAM 角色：

```
aws iam create-role \
    --role-name AmazonEKSAutoClusterRole \
    --assume-role-policy-document file://trust-policy.json
```

步驟 3：記下角色 ARN

擷取並儲存新角色的 ARN，以供後續步驟使用：

```
aws iam get-role --role-name AmazonEKSAutoClusterRole --query "Role.Arn" --output text
```

步驟 4：連接必要的政策

將下列 AWS 受管政策連接至叢集 IAM 角色，以授予必要的許可：

AmazonEKSClusterPolicy：

```
aws iam attach-role-policy \
    --role-name AmazonEKSAutoClusterRole \
    --policy-arn arn:aws: iam::aws:policy/AmazonEKSClusterPolicy
```

AmazonEKSComputePolicy：

```
aws iam attach-role-policy \
    --role-name AmazonEKSAutoClusterRole \
    --policy-arn arn:aws: iam::aws:policy/AmazonEKSComputePolicy
```

AmazonEKSBlockStoragePolicy：

```
aws iam attach-role-policy \
    --role-name AmazonEKSAutoClusterRole \
    --policy-arn arn:aws: iam::aws:policy/AmazonEKSBlockStoragePolicy
```

AmazonEKSLoadBalancingPolicy：

```
aws iam attach-role-policy \
```

```
    --role-name AmazonEKSAutoClusterRole \
    --policy-arn arn:aws: iam::aws:policy/AmazonEKSLoadBalancingPolicy
```

AmazonEKSNetworkingPolicy：

```
aws iam attach-role-policy \
    --role-name AmazonEKSAutoClusterRole \
    --policy-arn arn:aws: iam::aws:policy/AmazonEKSNetworkingPolicy
```

建立 EKS 自動模式節點 IAM 角色

步驟 1：建立信任政策

建立允許 Amazon EKS 服務擔任角色的信任政策。將政策儲存為 `node-trust-policy.json`：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "ec2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

步驟 2：建立節點 IAM 角色

使用上一個步驟中的 node-trust-policy.json 檔案來定義哪些實體可以擔任該角色。執行下列命令來建立節點 IAM 角色：

```
aws iam create-role \
    --role-name AmazonEKSAutoNodeRole \
    --assume-role-policy-document file://node-trust-policy.json
```

步驟 3：記下角色 ARN

建立角色之後，請擷取並儲存節點 IAM 角色的 ARN。在後續步驟中，您將需要此 ARN。使用下列命令來取得 ARN：

```
aws iam get-role --role-name AmazonEKSAutoNodeRole --query "Role.Arn" --output text
```

步驟 4：連接必要的政策

將下列 AWS 受管政策連接至節點 IAM 角色，以提供必要的許可：

AmazonEKSWorkerNodeMinimalPolicy：

```
aws iam attach-role-policy \
    --role-name AmazonEKSAutoNodeRole \
    --policy-arn arn:aws: iam::aws:policy/AmazonEKSWorkerNodeMinimalPolicy
```

AmazonEC2ContainerRegistryPullOnly：

```
aws iam attach-role-policy \
    --role-name AmazonEKSAutoNodeRole \
    --policy-arn arn:aws: iam::aws:policy/AmazonEC2ContainerRegistryPullOnly
```

建立 EKS 自動模式叢集

概觀

若要使用 CLI AWS 建立 EKS 自動模式叢集，您將需要下列參數：

- `cluster-name`：叢集的名稱。
- `k8s-version`：Kubernetes 版本（例如 1.31)。
- `subnet-ids`：在先前步驟中識別IDs。
- `cluster-role-arn`：叢集 IAM 角色的 ARN。
- `node-role-arn`：節點 IAM 角色的 ARN。

預設叢集組態

在建立叢集之前，請檢閱這些預設值和功能：

- `nodePools`：EKS Auto Mode 包含一般用途和系統預設節點集區。進一步了解節點集區。

注意：EKS Auto Mode 中的節點集區與 Amazon EKS 受管節點群組不同，但可以同時存在於同一個叢集中。

- computeConfig.enabled：自動化例行運算任務，例如建立和刪除 EC2 執行個體。
- kubernetesNetworkConfig.elasticLoadBalancing.enabled：自動化負載平衡任務，包括建立和刪除 Elastic Load Balancer。
- storageConfig.blockStorage.enabled：自動化儲存任務，例如建立和刪除 Amazon EBS 磁碟區。
- accessConfig.authenticationMode：需要 EKS 存取項目。進一步了解 EKS 身分驗證模式。

執行 命令

使用下列命令建立叢集：

```
aws eks create-cluster \
  --region ${AWS_REGION} \
  --cli-input-json \
  "{
      \"name\": \"${CLUSTER_NAME}\",
      \"version\": \"${K8S_VERSION}\",
      \"roleArn\": \"${CLUSTER_ROLE_ARN}\",
      \"resourcesVpcConfig\": {
        \"subnetIds\": ${SUBNETS_JSON},
        \"endpointPublicAccess\": true,
        \"endpointPrivateAccess\": true
      },
      \"computeConfig\": {
        \"enabled\": true,
        \"nodeRoleArn\":\"${NODE_ROLE_ARN}\",
        \"nodePools\": [\"general-purpose\", \"system\"]
      },
      \"kubernetesNetworkConfig\": {
        \"elasticLoadBalancing\": {
          \"enabled\": true
        }
      },
      \"storageConfig\": {
        \"blockStorage\": {
          \"enabled\": true
        }
      },
      \"accessConfig\": {
        \"authenticationMode\": \"API\"
      }
```

```
    }
```

檢查叢集狀態

步驟 1：驗證叢集建立

執行下列命令來檢查叢集的狀態。叢集建立通常需要約 15 分鐘：

```
aws eks describe-cluster --name "${CLUSTER_NAME}" --output json
```

步驟 2：更新 kubeconfig

叢集準備就緒後，請更新本機 kubeconfig 檔案，kubectl讓 與叢集通訊。此組態使用 AWS CLI 進行身分驗證。

```
aws eks update-kubeconfig --name "${CLUSTER_NAME}"
```

步驟 3：驗證節點集區

使用下列命令列出叢集中的節點集區：

```
kubectl get nodepools
```

後續步驟

- 了解如何將範例工作負載部署到新的 EKS Auto Mode 叢集。

使用 建立 EKS 自動模式叢集 AWS Management Console

在 中建立 EKS Auto Mode 叢集 AWS Management Console 所需的組態，比其他選項少。EKS 與 AWS IAM 和 VPC Networking 整合，可協助您建立與 EKS 叢集相關聯的資源。

您有兩個選項可在 主控台中建立叢集：

- 快速組態 （ 使用 EKS Auto 模式 ）
- 自訂組態

在本主題中，您將了解如何使用快速組態選項建立 EKS Auto Mode 叢集。

使用快速組態選項建立 EKS Auto 模式

您必須 AWS Management Console 具有足夠的許可來登入 來管理 AWS 資源，包括：EC2 執行個體、EC2 網路、EKS 叢集和 IAM 角色。

1. 導覽至 EKS 主控台
2. 按一下建立叢集
3. 確認已選取快速組態選項
4. 判斷下列值，或使用測試叢集的預設值。
   - 叢集名稱
   - Kubernetes 版本
5. 選取叢集 IAM 角色。如果這是您第一次建立 EKS Auto Mode 叢集，請使用建立建議的角色選項。
   - 或者，您可以為所有 EKS Auto Mode 叢集重複使用 AWS 帳戶中的單一叢集 IAM 角色。
   - 叢集 IAM 角色包含 EKS Auto Mode 所需的許可，以管理資源，包括 EC2 執行個體、EBS 磁碟區和 EC2 負載平衡器。
   - 建立建議的角色選項會預先填入所有欄位的建議值。選取下一步，然後選取建立。角色將使用建議`AmazonEKSAutoClusterRole`的名稱。
   - 如果您最近建立新的角色，請使用重新整理圖示重新載入角色選取下拉式清單。
6. 選取節點 IAM 角色。如果這是您第一次建立 EKS Auto Mode 叢集，請使用建立建議的角色選項。
   - 或者，您可以為所有 EKS Auto Mode 叢集重複使用 AWS 帳戶中的單一節點 IAM 角色。
   - 節點 IAM 角色包含自動模式節點連線到叢集所需的許可。節點 IAM 角色必須包含擷取容器 ECR 映像的許可。
   - 建立建議的角色選項會預先填入所有欄位的建議值。選取下一步，然後選取建立。角色將使用建議`AmazonEKSAutoNodeRole`的名稱。
   - 如果您最近建立新的角色，請使用重新整理圖示重新載入角色選取下拉式清單。
7. 選取 EKS Auto Mode 叢集的 VPC。選擇建立 VPC 以為 EKS 建立新的 VPC，或選擇您先前為 EKS 建立的 VPC。
   - 如果您使用 VPC 主控台建立新的 VPC， AWS 建議您為每個可用區域建立至少一個 NAT Gateway。否則，您可以使用所有其他預設值。
   - 如需 IPv6 叢集需求的詳細資訊和詳細資訊，請參閱the section called “建立 VPC”。
8. （選用） EKS Auto Mode 會自動填入所選 VPC 的私有子網路。您可以移除不需要的子網路。
   - EKS 會自動從 VPC 中選取私有子網路，並遵循最佳實務。您可以選擇從 VPC 中選取其他子網路，例如公有子網路。

9. （選用）選取檢視快速組態預設值，以檢閱新叢集的所有組態值。資料表指出在建立叢集之後，某些值無法編輯。

10選取 Create cluster (建立叢集)。請注意，叢集建立可能需要 15 分鐘才能完成。

後續步驟

- 了解如何將範例工作負載部署到您的 EKS Auto Mode 叢集

在現有的 EKS 叢集上啟用 EKS Auto Mode

您可以在現有的 EKS 叢集上啟用 EKS Auto Mode。

> **Note**
>
> EKS Auto Mode 需要 Kubernetes 1.29 版或更新版本。`ap-southeast-7` 或 `mx-central-1` AWS 區域無法使用 EKS Auto Mode。

AWS 支援下列遷移：

- 從 Karpenter 遷移至 EKS Auto Mode 節點。如需詳細資訊，請參閱the section called “從 Karpenter 遷移”。
- 從 EKS 受管節點群組遷移至 EKS 自動模式節點。如需詳細資訊，請參閱the section called “從 MNGs遷移”。
- 從 EKS Fargate 遷移至 EKS Auto 模式。如需詳細資訊，請參閱the section called “從 Fargate 遷移”。

AWS 不支援下列遷移：

- 將磁碟區從 EBS CSI 控制器（使用 Amazon EKS 附加元件）遷移至 EKS Auto Mode EBS CSI 控制器（由 EKS Auto Mode 管理）。使用一個 製作PVCs 無法由另一個掛載，因為它們使用兩個不同的 Kubernetes 磁碟區佈建器。
  - `eks-auto-mode-ebs-migration-tool` (AWS Labs 專案） 可在標準 EBS CSI StorageClass (`ebs.csi.aws.com`) 和 EKS Auto EBS CSI StorageClass () 之間進行遷移`ebs.csi.eks.amazonaws.com`。請注意，遷移需要刪除並重新建立現有的

PersistentVolumeClaim/PersistentVolume 資源，因此在實作之前，必須在非生產環境中進行驗證。

- 將負載平衡器從 AWS Load Balancer控制器遷移至 EKS Auto 模式

  您可以在 Amazon EKS Auto Mode 叢集上安裝 AWS Load Balancer控制器。使用 `IngressClass`或 `loadBalancerClass`選項，將服務和輸入資源與Load Balancer控制器或 EKS Auto 模式建立關聯。

- 使用替代 CNIs或其他不支援的網路組態遷移 EKS 叢集

遷移參考

使用下列遷移參考，將 Kubernetes 資源設定為由自我管理控制器或 EKS Auto 模式擁有。

功能	資源	欄位	自我管理	EKS 自動模式
區塊儲存	`StorageClass`	`provisioner`	`ebs.csi.aws.com`	`ebs.csi.eks.amazonaws.com`
負載平衡	`Service`	`loadBalancerClass`	`service.k8s.aws/nlb`	`eks.amazonaws.com/nlb`
負載平衡	`IngressClass`	`controller`	`ingress.k8s.aws/alb`	`eks.amazonaws.com/alb`
負載平衡	`IngressClassParams`	`apiversion`	`elbv2.k8s.aws/v1beta1`	`eks.amazonaws.com/v1`
負載平衡	`TargetGroupBinding`	`apiversion`	`elbv2.k8s.aws/v1beta1`	`eks.amazonaws.com/v1`

功能	資源	欄位	自我管理	EKS 自動模式
運算	NodeClass	apiVersion	karpenter.sh/v1	eks.amazonaws.com/v1

遷移 EBS 磁碟區

將工作負載遷移至 EKS Auto Mode 時，由於不同的 CSI 驅動程式佈建器，您需要處理 EBS 磁碟區遷移：

- EKS Auto Mode 佈建器： ebs.csi.eks.amazonaws.com
- 開放原始碼 EBS CSI 佈建器： ebs.csi.aws.com

請依照下列步驟遷移您的持久性磁碟區：

1. 修改磁碟區保留政策：將現有平台版本的 (PV) 變更為 persistentVolumeReclaimPolicyRetain，以確保不會刪除基礎 EBS 磁碟區。
2. 從 Kubernetes 移除 PV：刪除舊 PV 資源，同時保持實際 EBS 磁碟區不變。
3. 使用靜態佈建建立新的 PV：建立新的 PV，參考相同的 EBS 磁碟區，但適用於目標 CSI 驅動程式。
4. 繫結至新的 PVC：建立使用 volumeName 欄位特別參考 PV 的新 PVC。

考量事項

- 在開始此遷移之前，請確定您的應用程式已停止。
- 在開始遷移程序之前備份您的資料。
- 每個持久性磁碟區都需要執行此程序。
- 必須更新工作負載才能使用新的 PVC。

遷移負載平衡器

您無法將現有的負載平衡器從自我管理 AWS 負載平衡器控制器直接轉移到 EKS Auto 模式。相反地，您必須實作藍綠部署策略。這包括在受管控制器下建立新的負載平衡器時，維護現有的負載平衡器組態。

為了將服務中斷降至最低，我們建議採用 DNS 型流量轉移方法。首先，使用 EKS Auto 模式建立新的負載平衡器，同時保持現有組態的運作。然後，使用 DNS 路由（例如 Route 53），逐步將流量從舊負載平衡器轉移到新的負載平衡器。成功遷移流量並驗證新組態後，您就可以停用舊負載平衡器和自我管理控制器。

在現有叢集上啟用 EKS Auto 模式

本主題說明如何在現有的 Amazon EKS 叢集上啟用 Amazon EKS Auto Mode。在現有叢集上啟用自動模式需要更新 IAM 許可並設定核心 EKS 自動模式設定。啟用後，您就可以開始遷移現有的運算工作負載，以利用 Auto Mode 的簡化操作和自動化基礎設施管理。

> **Important**
>
> 在啟用 EKS Auto Mode 之前，請確認您已安裝特定 Amazon EKS 附加元件的最低必要版本。如需詳細資訊，請參閱the section called “必要的附加元件版本”。

開始之前，請確定您擁有 Amazon EKS 叢集的管理員存取權，以及修改 IAM 角色的許可。本主題中的步驟會引導您使用 AWS Management Console 或 CLI AWS 來啟用自動模式。

AWS Management Console

您必須使用管理 IAM、EKS 和 EC2 資源的許可登入 AWS 主控台。

> **Note**
>
> 建立叢集後，無法變更 EKS 叢集的叢集 IAM 角色。EKS Auto Mode 需要此角色的額外許可。您必須將其他政策連接至目前角色。

更新叢集 IAM 角色

1. 在 中開啟叢集概觀頁面 AWS Management Console。

2. 在叢集 IAM 角色 ARN 下，選取在 IAM 中檢視。
3. 從新增許可下拉式清單中，選取連接政策。
4. 使用搜尋方塊尋找並選取下列政策：
   - AmazonEKSComputePolicy
   - AmazonEKSBlockStoragePolicy
   - AmazonEKSLoadBalancingPolicy
   - AmazonEKSNetworkingPolicy
   - AmazonEKSClusterPolicy
5. 選取新增許可
6. 從信任關係索引標籤中，選取編輯信任政策
7. 插入下列叢集 IAM 角色信任政策，然後選取更新政策

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "eks.amazonaws.com"
      },
      "Action": [
        "sts:AssumeRole",
        "sts:TagSession"
      ]
    }
  ]
}
```

啟用 EKS Auto 模式

1. 在 中開啟叢集概觀頁面 AWS Management Console。
2. 在 EKS 自動模式下，選取管理
3. 將 EKS Auto 模式切換為開啟。
4. 從 EKS 節點集區下拉式清單中，選取您要建立的預設節點集區。
   - 進一步了解 EKS Auto 模式的節點集區。如需詳細資訊，請參閱the section called “建立節點集區”。

5. 如果您先前已建立此 AWS 帳戶的 EKS 自動模式節點 IAM 角色，請在節點 IAM 角色下拉式清單中選取該角色。如果您之前尚未建立此角色，請選取建立建議的角色，然後依照步驟執行。

AWS CLI

先決條件

- 現有 EKS 叢集的叢集 IAM 角色必須包含 EKS Auto Mode 的足夠許可，例如下列政策：
  - `AmazonEKSComputePolicy`
  - `AmazonEKSBlockStoragePolicy`
  - `AmazonEKSLoadBalancingPolicy`
  - `AmazonEKSNetworkingPolicy`
  - `AmazonEKSClusterPolicy`
- 叢集 IAM 角色必須具有更新的信任政策，包括 `sts:TagSession`動作。如需建立叢集 IAM 角色的詳細資訊，請參閱 the section called “ AWS CLI”。
- `aws` 安裝、登入 CLI，以及足夠的版本。您必須擁有管理 IAM、EKS 和 EC2 資源的許可。如需詳細資訊，請參閱設定。

程序

使用下列命令在現有叢集上啟用 EKS Auto Mode。

> **Note**
>
> 必須在同一請求中啟用或停用運算、區塊儲存和負載平衡功能。

```
aws eks update-cluster-config \
 --name $CLUSTER_NAME \
 --compute-config enabled=true \
 --kubernetes-network-config '{"elasticLoadBalancing":{"enabled": true}}' \
 --storage-config '{"blockStorage":{"enabled": true}}'
```

必要的附加元件版本

如果您打算在現有叢集上啟用 EKS Auto Mode，您可能需要更新特定附加元件。請注意：

- 這僅適用於轉換至 EKS Auto 模式的現有叢集。
- 啟用 EKS Auto Mode 時建立的新叢集不需要這些更新。

如果您已安裝下列任何附加元件，請確定它們至少為指定的最低版本：

附加元件名稱	最低必要版本
Kubernetes 專用 Amazon VPC CNI 外掛程式	v1.19.0-eksbuild.1
Kube-proxy	• v1.26.15-eksbuild.19 • v1.27.16-eksbuild.14 • v1.28.15-eksbuild.4 • v1.29.10-eksbuild.3 • v1.30.6-eksbuild.3 • v1.31.2-eksbuild.3
Amazon EBS CSI 驅動程式	v1.37.0-eksbuild.1
CSI 快照控制器	v8.1.0-eksbuild.2
EKS Pod 身分識別代理程式	v1.3.4-eksbuild.1

如需詳細資訊，請參閱the section called “更新 附加元件”。

後續步驟

- 若要遷移管理節點群組工作負載，請參閱 the section called “從 MNGs遷移”。
- 若要從自我管理 Karpenter 遷移，請參閱 the section called “從 Karpenter 遷移”。

使用 kubectl 從 Karpenter 遷移至 EKS Auto Mode

本主題將逐步引導您使用 kubectl 將工作負載從 Karpenter 遷移至 Amazon EKS Auto Mode。遷移可以逐步執行，讓您以自己的步調移動工作負載，同時在整個轉換過程中保持叢集穩定性和應用程式可用性。

下面概述step-by-step方法可讓您在遷移期間並排執行 Karpenter 和 EKS Auto Mode。此雙操作策略可讓您在完全停用 Karpenter 之前，先驗證 EKS Auto Mode 上的工作負載行為，以確保順利轉換。您可以個別或群組遷移應用程式，提供彈性以適應特定操作需求和風險承受能力。

先決條件

開始遷移之前，請確定您已：

- 叢集上安裝 Karpenter v1.1 或更新版本。如需詳細資訊，請參閱 Karpenter 文件中的升級至 1.1.0+。
- `kubectl` 已安裝並連接到您的叢集。如需詳細資訊，請參閱設定。

本主題假設您熟悉 Karpenter 和 NodePools。如需詳細資訊，請參閱 Karpenter 文件。

步驟 1：在叢集上啟用 EKS Auto Mode

使用 CLI 或 管理主控台在現有叢集上啟用 AWS EKS Auto Mode。如需詳細資訊，請參閱the section called “在叢集上啟用”。

> **Note**
>
> 啟用 EKS Auto Mode 時，請勿在此階段於轉換期間啟用`general purpose`節點集區。此節點集區不是選擇性的。
> 如需詳細資訊，請參閱the section called “檢閱內建節點集區”。

步驟 2：建立污點 EKS 自動模式 NodePool

使用污點為 EKS 自動模式建立新的 NodePool。這可確保現有的 Pod 不會在新的 EKS Auto Mode 節點上自動排程。此節點集區使用`defaultNodeClass`內建於 EKS Auto 模式的 。如需詳細資訊，請參閱the section called “建立節點類別”。

具有污點的範例節點集區：

```
apiVersion: karpenter.sh/v1
kind: NodePool
metadata:
  name: eks-auto-mode
```

```
spec:
  template:
    spec:
      requirements:
        - key: "eks.amazonaws.com/instance-category"
          operator: In
          values: ["c", "m", "r"]
      nodeClassRef:
        group: eks.amazonaws.com
        kind: NodeClass
        name: default
      taints:
        - key: "eks-auto-mode"
          effect: "NoSchedule"
```

更新節點集區的需求，以符合您要從中遷移的 Karpenter 組態。您需要至少一個需求。

步驟 3：更新工作負載以進行遷移

識別並更新要遷移至 EKS Auto Mode 的工作負載。將公差和節點選擇器新增至這些工作負載：

```
apiVersion: apps/v1
kind: Deployment
spec:
  template:
    spec:
      tolerations:
      - key: "eks-auto-mode"
        effect: "NoSchedule"
      nodeSelector:
        eks.amazonaws.com/compute-type: auto
```

此變更可讓工作負載排程在新的 EKS Auto Mode 節點上。

EKS Auto Mode 使用與 Karpenter 不同的標籤。與 EC2 受管執行個體相關的標籤開頭為 `eks.amazonaws.com`。如需詳細資訊，請參閱the section called “建立節點集區”。

步驟 4：逐漸遷移工作負載

針對您要遷移的每個工作負載重複步驟 3。這可讓您根據您的需求和風險承受能力，個別或群組移動工作負載。

步驟 5：移除原始 Karpenter NodePool

遷移所有工作負載後，您可以移除原始 Karpenter NodePool：

```
kubectl delete nodepool <original-nodepool-name>
```

步驟 6：從 EKS Auto Mode NodePool 移除污點（選用）

如果您希望 EKS Auto Mode 成為新工作負載的預設值，您可以從 EKS Auto Mode NodePool 中移除污點：

```
apiVersion: karpenter.sh/v1
kind: NodePool
metadata:
  name: eks-auto-mode
spec:
  template:
    spec:
      nodeClassRef:
        group: eks.amazonaws.com
        kind: NodeClass
        name: default
      # Remove the taints section
```

步驟 7：從工作負載移除節點選取器（選用）

如果您已從 EKS Auto Mode NodePool 移除污點，您可以選擇從工作負載中移除節點選擇器，因為 EKS Auto Mode 現在是預設值：

```
apiVersion: apps/v1
kind: Deployment
spec:
  template:
    spec:
      # Remove the nodeSelector section
      tolerations:
      - key: "eks-auto-mode"
        effect: "NoSchedule"
```

步驟 8：從您的叢集解除安裝 Karpenter

移除 Karpenter 的步驟取決於您的安裝方式。如需詳細資訊，請參閱 Karpenter 安裝說明。

從 EKS 受管節點群組遷移至 EKS Auto 模式

將 Amazon EKS 叢集轉換為使用 EKS 自動模式時，您可以使用 eksctl CLI 工具從受管節點群組 (MNGs) 順暢遷移現有工作負載。此程序可確保持續的應用程式可用性，同時 EKS 自動模式會最佳化您的運算資源。您可以對執行中的應用程式執行遷移，並將中斷降至最低。

本主題會逐步引導您從現有的受管節點群組安全地耗盡 Pod，並允許 EKS 自動模式在新佈建的執行個體上重新排程它們。透過遵循此程序，您可以利用 EKS 自動模式的智慧型工作負載整合，同時在整個遷移過程中維持應用程式的可用性。

先決條件

- 啟用 EKS Auto Mode 的叢集
- `eksctl` CLI 已安裝並連接到您的叢集。如需詳細資訊，請參閱設定。
- 叢集上未安裝 Karpenter。

程序

使用下列 `eksctl` CLI 命令，從現有的受管節點群組執行個體啟動耗盡 Pod。EKS Auto Mode 將建立新的節點，以傳回置換的 Pod。

```
eksctl delete nodegroup --cluster=<clusterName> --name=<nodegroupName>
```

您需要針對叢集中的每個受管節點群組執行此命令。

如需此命令的詳細資訊，請參閱 eksctl 文件中的刪除和耗盡節點群組。

從 EKS Fargate 遷移至 EKS Auto 模式

本主題將逐步引導您使用 將工作負載從 EKS Fargate 遷移至 Amazon EKS Auto Mode`kubectl`。您可以逐步執行遷移，讓您以自己的步調移動工作負載，同時在整個轉換過程中維持叢集穩定性和應用程式可用性。

下面概述step-by-step方法可讓您在遷移期間並排執行 EKS Fargate 和 EKS Auto Mode。此雙操作策略可讓您在完全停用 EKS Fargate 之前，先驗證 EKS Auto Mode 上的工作負載行為，以確保順利轉換。您可以個別或群組遷移應用程式，提供彈性以滿足您的特定操作需求和風險承受能力。

使用 AWS Fargate 比較 Amazon EKS Auto 模式和 EKS

對於想要執行 EKS 的客戶而言，Amazon EKS 搭配 AWS Fargate 仍然是選項，但 Amazon EKS Auto Mode 是建議的方法。EKS Auto Mode 完全符合 Kubernetes，支援所有上游 Kubernetes 基本概念和平台工具，例如 Fargate 無法支援的 Istio。EKS Auto Mode 也完全支援所有 EC2 執行期購買選項，包括 GPU 和 Spot 執行個體，讓客戶能夠利用交涉的 EC2 折扣和其他節省機制。搭配 Fargate 使用 EKS 時，無法使用這些功能。

此外，EKS Auto Mode 可讓客戶使用標準 Kubernetes 排程功能，確保每個 EC2 執行個體執行單一應用程式容器，達成與 Fargate 相同的隔離模型。透過採用 Amazon EKS Auto Mode，客戶可以釋放在上執行 Kubernetes 的完整優點 AWS ，這是一個完全符合 Kubernetes 標準的平台，提供彈性以利用 EC2 和購買選項的完整廣度，同時保留 Fargate 提供的基礎設施管理的易用性和抽象性。

先決條件

開始遷移之前，請確定您擁有

- 使用 Fargate 設定叢集。如需詳細資訊，請參閱the section called “開始使用”。
- 安裝並`kubectl`連接到您的叢集。如需詳細資訊，請參閱設定。

步驟 1：檢查 Fargate 叢集

1. 檢查具有 Fargate 的 EKS 叢集是否正在執行：

```
kubectl get node
```

```
NAME STATUS ROLES AGE VERSION
fargate-ip-192-168-92-52.ec2.internal Ready <none> 25m v1.30.8-eks-2d5f260
fargate-ip-192-168-98-196.ec2.internal Ready <none> 24m v1.30.8-eks-2d5f260
```

2. 檢查執行中的 Pod：

```
kubectl get pod -A
```

```
NAMESPACE NAME READY STATUS RESTARTS AGE
kube-system coredns-6659cb98f6-gxpjz 1/1 Running 0 26m
kube-system coredns-6659cb98f6-gzzsx 1/1 Running 0 26m
```

3. 在名為 的檔案中建立部署`deployment_fargate.yaml`：

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
      annotations:
        eks.amazonaws.com/compute-type: fargate
    spec:
      containers:
      - name: nginx
        image: nginx
        ports:
        - containerPort: 80
```

4. 套用部署：

```
kubectl apply -f deployment_fargate.yaml
```

```
deployment.apps/nginx-deployment created
```

5. 檢查 Pod 和部署：

```
kubectl get pod,deploy
```

```
NAME                                    READY   STATUS    RESTARTS   AGE
pod/nginx-deployment-5c7479459b-6trtm   1/1     Running   0          61s
pod/nginx-deployment-5c7479459b-g8ssb   1/1     Running   0          61s
pod/nginx-deployment-5c7479459b-mq4mf   1/1     Running   0          61s

NAME                                READY   UP-TO-DATE   AVAILABLE   AGE
```

```
deployment.apps/nginx-deployment   3/3     3            3           61s
```

6. 檢查節點：

```
kubectl get node -owide
```

```
NAME                                       STATUS  ROLES  AGE VERSION
   INTERNAL-IP     EXTERNAL-IP OS-IMAGE       KERNEL-VERSION
 CONTAINER-RUNTIME
fargate-ip-192-168-111-43.ec2.internal  Ready   <none> 31s v1.30.8-eks-2d5f260
 192.168.111.43  <none>      Amazon Linux 2 5.10.234-225.910.amzn2.x86_64
 containerd://1.7.25
fargate-ip-192-168-117-130.ec2.internal Ready   <none> 36s v1.30.8-eks-2d5f260
 192.168.117.130 <none>      Amazon Linux 2 5.10.234-225.910.amzn2.x86_64
 containerd://1.7.25
fargate-ip-192-168-74-140.ec2.internal  Ready   <none> 36s v1.30.8-eks-2d5f260
 192.168.74.140  <none>      Amazon Linux 2 5.10.234-225.910.amzn2.x86_64
 containerd://1.7.25
```

步驟 2：在叢集上啟用 EKS Auto 模式

1. 使用 CLI 或 管理主控台在現有叢集上啟用 AWS EKS Auto Mode。如需詳細資訊，請參閱the section called “在叢集上啟用”。
2. 檢查節點集區：

```
kubectl get nodepool
```

```
NAME              NODECLASS   NODES   READY   AGE
general-purpose   default     1       True    6m58s
system            default     0       True    3d14h
```

步驟 3：更新工作負載以進行遷移

識別並更新您要遷移至 EKS Auto Mode 的工作負載。

若要將工作負載從 Fargate 遷移至 EKS Auto 模式，請套用註釋 `eks.amazonaws.com/compute-type: ec2`。這可確保 Fargate 不會排程工作負載，即使 Fargate 設定檔，EKS Auto Mode NodePool 也會追上工作負載。如需詳細資訊，請參閱the section called “建立節點集區”。

1. 修改您的部署（例如 deployment_fargate.yaml 檔案），將運算類型變更為 ec2：

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
      annotations:
        eks.amazonaws.com/compute-type: ec2
    spec:
      containers:
      - name: nginx
        image: nginx
        ports:
        - containerPort: 80
```

2. 套用部署。此變更可讓工作負載排程在新的 EKS Auto Mode 節點上：

```
kubectl apply -f deployment_fargate.yaml
```

3. 檢查部署是否在 EKS Auto Mode 叢集中執行：

```
kubectl get pod -o wide
```

```
NAME                               READY   STATUS    RESTARTS   AGE     IP
    NODE                   NOMINATED NODE   READINESS GATES
nginx-deployment-97967b68d-ffxxh   1/1     Running   0          3m31s
 192.168.43.240   i-0845aafcb51630ffb   <none>           <none>
nginx-deployment-97967b68d-mbcgj   1/1     Running   0          2m37s
 192.168.43.241   i-0845aafcb51630ffb   <none>           <none>
nginx-deployment-97967b68d-qpd8x   1/1     Running   0          2m35s
 192.168.43.242   i-0845aafcb51630ffb   <none>           <none>
```

4. 確認 EKS Auto Mode 受管節點中沒有執行中的 Fargate 節點和執行中的部署：

```
kubectl get node -owide
```

```
NAME                  STATUS ROLES  AGE   VERSION              INTERNAL-IP     EXTERNAL-
IP OS-IMAGE                                         KERNEL-VERSION CONTAINER-RUNTIME
i-0845aafcb51630ffb Ready  <none> 3m30s v1.30.8-eks-3c20087 192.168.41.125
 3.81.118.95 Bottlerocket (EKS Auto) 2025.3.14 (aws-k8s-1.30) 6.1.129
 containerd://1.7.25+bottlerocket
```

步驟 4：逐漸遷移工作負載

針對您要遷移的每個工作負載重複步驟 3。這可讓您根據您的需求和風險承受能力，個別或群組移動工作負載。

步驟 5：移除原始 Fargate 設定檔

遷移所有工作負載後，您可以移除原始`fargate`設定檔。以 *`Fargate ######## <fargate ######`*：

```
aws eks delete-fargate-profile --cluster-name eks-fargate-demo-cluster --fargate-
profile-name <fargate profile name>
```

步驟 6：縮減 CoreDNS

由於 EKS Auto 模式會處理 CoreDNS，因此您可以將`coredns`部署縮減為 0：

```
kubectl scale deployment coredns -n kube-system —-replicas=0
```

在 EKS Auto Mode 叢集中執行範例工作負載

本章提供範例，說明如何將不同類型的工作負載部署到在 Auto Mode 中執行的 Amazon EKS 叢集。這些範例示範關鍵工作負載模式，包括範例應用程式、負載平衡 Web 應用程式、使用持久性儲存的狀態工作負載，以及具有特定節點置放需求的工作負載。每個範例都包含完整的資訊清單和step-by-step部署說明，您可以用這些說明做為自己應用程式的範本。

繼續進行範例之前，請確定您已讓 EKS 叢集在自動模式下執行，而且已安裝 AWS CLI 和 kubectl。如需詳細資訊，請參閱設定。這些範例假設基本熟悉 Kubernetes 概念和 kubectl 命令。

您可以使用這些使用案例型範例，在 EKS Auto Mode 叢集中執行工作負載。

the section called “部署膨脹工作負載”

示範如何使用 kubectl命令將範例工作負載部署至 EKS Auto Mode 叢集。

the section called “部署負載平衡器”

示範如何在 Amazon EKS 上部署 2048 遊戲的容器化版本。

the section called “部署具狀態工作負載”

示範如何將具狀態應用程式範例部署至 EKS Auto Mode 叢集。

the section called “控制部署”

示範如何使用註釋來控制工作負載是否部署到 EKS Auto Mode 管理的節點。

將範例充滿工作負載部署至 Amazon EKS Auto Mode 叢集

在本教學課程中，您將了解如何將範例工作負載部署至 EKS Auto Mode 叢集，並觀察其如何自動佈建所需的運算資源。您將使用 kubectl 命令來監看叢集的行為，並親自了解自動模式如何簡化 Kubernetes 操作 AWS。在本教學課程結束時，您將了解 EKS Auto Mode 如何透過自動管理基礎運算資源來回應工作負載部署，而不需要手動節點群組組態。

先決條件

- Amazon EKS Auto Mode 叢集。請記下叢集的名稱和 AWS 區域。
- IAM 主體，例如使用者或角色，具有足夠的許可來管理聯網、運算和 EKS 資源。
  - 如需詳細資訊，請參閱《IAM 使用者指南》中的建立角色和連接政策。
- aws CLI 已安裝並設定 IAM 身分。
- kubectl 安裝並連線至叢集的 CLI。
  - 如需詳細資訊，請參閱設定。

步驟 1：檢閱現有的運算資源 （選用）

首先，使用 kubectl列出叢集上的節點集區。

```
kubectl get nodepools
```

範例輸出：

```
general-purpose
```

在本教學課程中，我們將部署設定為使用general-purpose節點集區的工作負載。此節點集區內建於 EKS Auto 模式，並包含一般工作負載的合理預設值，例如微服務和 Web 應用程式。您可以建立自己的節點集區。如需詳細資訊，請參閱the section called “建立節點集區”。

其次，使用 kubectl 列出連接到叢集的節點。

```
kubectl get nodes
```

如果您剛建立 EKS Auto Mode 叢集，則不會有節點。

在本教學課程中，您將部署範例工作負載。如果您沒有節點，或工作負載無法容納現有節點，EKS Auto Mode 會佈建新的節點。

步驟 2：將範例應用程式部署至叢集

檢閱下列 Kubernetes 部署並將其儲存為 inflate.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: inflate
spec:
  replicas: 1
  selector:
    matchLabels:
      app: inflate
  template:
    metadata:
      labels:
        app: inflate
    spec:
      terminationGracePeriodSeconds: 0
      nodeSelector:
        eks.amazonaws.com/compute-type: auto
      securityContext:
        runAsUser: 1000
        runAsGroup: 3000
```

```
        fsGroup: 2000
      containers:
        - name: inflate
          image: public.ecr.aws/eks-distro/kubernetes/pause:3.7
          resources:
            requests:
              cpu: 1
          securityContext:
            allowPrivilegeEscalation: false
```

請注意，eks.amazonaws.com/compute-type: auto選擇器需要在 Amazon EKS Auto Mode 節點上部署工作負載。

將部署套用至您的叢集。

```
kubectl apply -f inflate.yaml
```

步驟 3：觀看 Kubernetes 事件

使用下列命令來監看 Kubernetes 事件，包括建立新的節點。使用 ctrl+c 停止監看事件。

```
kubectl get events -w --sort-by '.lastTimestamp'
```

使用 kubectl 再次列出連接到叢集的節點。請注意新建立的節點。

```
kubectl get nodes
```

步驟 4：在 AWS 主控台中檢視節點和執行個體

您可以在 EKS 主控台中檢視 EKS 自動模式節點，並在 EC2 主控台中檢視相關聯的 EC2 執行個體。

EKS Auto Mode 部署的 EC2 執行個體受到限制。您無法在 EKS Auto Mode 節點上執行任意命令。

步驟 5：刪除部署

使用 kubectl刪除範例部署

```
kubectl delete -f inflate.yaml
```

如果您沒有部署到叢集的其他工作負載，EKS Auto Mode 建立的節點將會是空的。

在預設組態中，EKS Auto Mode 會偵測已空三十秒的節點，並終止它們。

使用 kubectl或 EC2 主控台確認已刪除相關聯的執行個體。

將範例Load Balancer工作負載部署至 EKS Auto 模式

本指南將逐步引導您在 Amazon EKS 上部署 2048 遊戲的容器化版本，以及負載平衡和網際網路可存取性。

先決條件

- EKS Auto Mode 叢集
- kubectl 設定為與您的叢集互動
- 建立 ALB 資源的適當 IAM 許可

步驟 1：建立命名空間

首先，為 2048 遊戲應用程式建立專用命名空間。

建立名為 01-namespace.yaml 的檔案：

```
apiVersion: v1
kind: Namespace
metadata:
  name: game-2048
```

套用命名空間組態：

```
kubectl apply -f 01-namespace.yaml
```

步驟 2：部署應用程式

應用程式會執行 2048 遊戲容器的多個複本。

建立名為 02-deployment.yaml 的檔案：

```
apiVersion: apps/v1
```

```
kind: Deployment
metadata:
  namespace: game-2048
  name: deployment-2048
spec:
  selector:
    matchLabels:
      app.kubernetes.io/name: app-2048
  replicas: 5
  template:
    metadata:
      labels:
        app.kubernetes.io/name: app-2048
    spec:
      containers:
        - image: public.ecr.aws/l6m2t8p7/docker-2048:latest
          imagePullPolicy: Always
          name: app-2048
          ports:
            - containerPort: 80
          resources:
            requests:
              cpu: "0.5"
```

> **Note**
>
> 如果您收到載入映像 的錯誤public.ecr.aws/l6m2t8p7/docker-2048:latest，請確認您的節點 IAM 角色具有從 ECR 提取映像的足夠許可。如需詳細資訊，請參閱the section called “節點 IAM 角色”。此外，範例中的docker-2048映像是x86_64映像，不會在其他架構上執行。

主要元件：

- 部署 5 個應用程式的複本
- 使用公有 ECR 映像
- 每個 Pod 請求 0.5 個 CPU 核心
- 公開 HTTP 流量的連接埠 80

套用部署：

```
kubectl apply -f 02-deployment.yaml
```

步驟 3：建立 服務

服務會將部署公開至叢集網路。

建立名為 03-service.yaml 的檔案：

```
apiVersion: v1
kind: Service
metadata:
  namespace: game-2048
  name: service-2048
spec:
  ports:
    - port: 80
      targetPort: 80
      protocol: TCP
  selector:
    app.kubernetes.io/name: app-2048
```

主要元件：

- 建立 NodePort 服務
- 將連接埠 80 映射至容器的連接埠 80
- 使用標籤選擇器來尋找 Pod

套用服務：

```
kubectl apply -f 03-service.yaml
```

步驟 4：設定負載平衡

您將設定輸入，將應用程式公開至網際網路。

首先，建立 IngressClass。建立名為 04-ingressclass.yaml 的檔案：

```
apiVersion: networking.k8s.io/v1
kind: IngressClass
```

```
metadata:
  labels:
    app.kubernetes.io/name: LoadBalancerController
  name: alb
spec:
  controller: eks.amazonaws.com/alb
```

> **Note**
>
> EKS Auto Mode 需要子網路標籤來識別公有和私有子網路。
> 如果您使用 建立叢集`eksctl`，表示您已經有這些標籤。
> 了解如何 the section called “標記子網路”。

然後建立輸入資源。建立名為 `05-ingress.yaml` 的檔案：

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  namespace: game-2048
  name: ingress-2048
  annotations:
    alb.ingress.kubernetes.io/scheme: internet-facing
    alb.ingress.kubernetes.io/target-type: ip
spec:
  ingressClassName: alb
  rules:
    - http:
        paths:
          - path: /
            pathType: Prefix
            backend:
              service:
                name: service-2048
                port:
                  number: 80
```

主要元件：

- 建立面向網際網路的 ALB
- 使用 IP 目標類型進行直接 Pod 路由

- 將所有流量 (/) 路由至遊戲服務

套用輸入組態：

```
kubectl apply -f 04-ingressclass.yaml
kubectl apply -f 05-ingress.yaml
```

步驟 5：驗證部署

1. 檢查所有 Pod 是否正在執行：

   ```
   kubectl get pods -n game-2048
   ```

2. 確認已建立服務：

   ```
   kubectl get svc -n game-2048
   ```

3. 取得 ALB 端點：

   ```
   kubectl get ingress -n game-2048
   ```

輸入輸出中的 ADDRESS 欄位會顯示您的 ALB 端點。等待 2-3 分鐘讓 ALB 佈建和註冊所有目標。

步驟 6：存取遊戲

開啟您的 Web 瀏覽器，並瀏覽至先前步驟的 ALB 端點 URL。您應該會看到 2048 遊戲界面。

步驟 7：清除

若要移除在本教學課程中建立的所有資源：

```
kubectl delete namespace game-2048
```

這將刪除命名空間中的所有資源，包括部署、服務和輸入資源。

幕後的情況

1. 部署會建立執行 2048 遊戲的 5 個 Pod
2. 服務提供對這些 Pod 的穩定網路存取

3. EKS 自動模式：
   - 在 中建立 Application Load Balancer AWS
   - 設定 Pod 的目標群組
   - 設定路由規則，將流量導向服務

故障診斷

如果遊戲未載入：

- 確保所有 Pod 正在執行： `kubectl get pods -n game-2048`
- 檢查輸入狀態： `kubectl describe ingress -n game-2048`
- 驗證 ALB 運作狀態檢查：在 AWS 主控台中檢查目標群組運作狀態

將具狀態工作負載範例部署至 EKS Auto 模式

本教學課程將引導您將具狀態應用程式範例部署至 EKS Auto Mode 叢集。應用程式會將時間戳記寫入持久性磁碟區，示範 EKS Auto Mode 的自動 EBS 磁碟區佈建和持久性功能。

先決條件

- EKS Auto Mode 叢集
- 設定適當許可的 AWS CLI
- `kubectl` 已安裝和設定
  - 如需詳細資訊，請參閱設定。

步驟 1：設定您的環境

1. 設定您的環境變數：

```
export CLUSTER_NAME=my-auto-cluster
export AWS_REGION="us-west-2"
```

2. 更新您的 kubeconfig：

```
aws eks update-kubeconfig --name "${CLUSTER_NAME}"
```

步驟 2：建立儲存體方案

StorageClass 定義 EKS Auto Mode 如何佈建 EBS 磁碟區。

EKS Auto Mode 不會StorageClass為您建立 。您必須建立StorageClass參考，ebs.csi.eks.amazonaws.com才能使用 EKS Auto Mode 的儲存功能。

1. 建立名為 storage-class.yaml 的檔案：

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: auto-ebs-sc
  annotations:
    storageclass.kubernetes.io/is-default-class: "true"
provisioner: ebs.csi.eks.amazonaws.com
volumeBindingMode: WaitForFirstConsumer
parameters:
  type: gp3
  encrypted: "true"
```

2. 套用 StorageClass：

```
kubectl apply -f storage-class.yaml
```

主要元件：

- provisioner: ebs.csi.eks.amazonaws.com - 使用 EKS Auto 模式
- volumeBindingMode: WaitForFirstConsumer - 延遲磁碟區建立，直到 Pod 需要它
- type: gp3 - 指定 EBS 磁碟區類型
- encrypted: "true" - EBS 將使用預設aws/ebs金鑰來加密使用此類別建立的磁碟區。此為選用操作，但建議您採用。
- storageclass.kubernetes.io/is-default-class: "true" - Kubernetes 預設會使用此儲存類別，除非您在持久性磁碟區宣告上指定不同的磁碟區類別。如果您要從另一個儲存控制器遷移，請謹慎設定此值。（選用）

步驟 3：建立持久性磁碟區宣告

PVC 會從 請求儲存體StorageClass。

1. 建立名為 pvc.yaml 的檔案：

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: auto-ebs-claim
spec:
  accessModes:
    - ReadWriteOnce
  storageClassName: auto-ebs-sc
  resources:
    requests:
      storage: 8Gi
```

2. 套用 PVC：

```
kubectl apply -f pvc.yaml
```

主要元件：

- accessModes: ReadWriteOnce - 磁碟區一次可由一個節點掛載
- storage: 8Gi - 請求 8 GiB 磁碟區
- storageClassName: auto-ebs-sc - 參考StorageClass我們建立的

步驟 4：部署應用程式

部署會執行將時間戳記寫入持久性磁碟區的容器。

1. 建立名為 deployment.yaml 的檔案：

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: inflate-stateful
spec:
  replicas: 1
  selector:
    matchLabels:
      app: inflate-stateful
  template:
```

```
    metadata:
      labels:
        app: inflate-stateful
    spec:
      terminationGracePeriodSeconds: 0
      nodeSelector:
        eks.amazonaws.com/compute-type: auto
      containers:
        - name: bash
          image: public.ecr.aws/docker/library/bash:4.4
          command: ["/usr/local/bin/bash"]
          args: ["-c", "while true; do echo $(date -u) >> /data/out.txt; sleep 60;
 done"]
          resources:
            requests:
              cpu: "1"
          volumeMounts:
            - name: persistent-storage
              mountPath: /data
      volumes:
        - name: persistent-storage
          persistentVolumeClaim:
            claimName: auto-ebs-claim
```

2. 套用部署：

```
kubectl apply -f deployment.yaml
```

主要元件：

- 將時間戳記寫入檔案的簡易 bash 容器
- 在 掛載 PVC /data
- 請求 1 個 CPU 核心
- 使用 EKS 受管節點的節點選取器

步驟 5：驗證設定

1. 檢查 Pod 是否正在執行：

```
kubectl get pods -l app=inflate-stateful
```

2. 確認 PVC 已繫結：

```
kubectl get pvc auto-ebs-claim
```

3. 檢查 EBS 磁碟區：

```
# Get the PV name
PV_NAME=$(kubectl get pvc auto-ebs-claim -o jsonpath='{.spec.volumeName}')
# Describe the EBS volume
aws ec2 describe-volumes \
  --filters Name=tag:CSIVolumeName,Values=${PV_NAME}
```

4. 確認資料正在寫入：

```
kubectl exec "$(kubectl get pods -l app=inflate-stateful \
  -o=jsonpath='{.items[0].metadata.name}')" -- \
  cat /data/out.txt
```

步驟 6：清除

執行下列命令來移除在本教學課程中建立的所有資源：

```
# Delete all resources in one command
kubectl delete deployment/inflate-stateful pvc/auto-ebs-claim storageclass/auto-ebs-sc
```

幕後的情況

1. PVC 從 請求儲存 `StorageClass`
2. 排定 Pod 時：
   a. EKS Auto Mode 佈建 EBS 磁碟區
   b. 建立PersistentVolume
   c. 將磁碟區連接至節點
3. Pod 掛載磁碟區並開始寫入時間戳記

快照控制器

EKS Auto Mode 與 Kubernetes CSI Snapshotter 相容，也稱為快照控制器。不過，EKS Auto Mode 不包含快照控制器。您負責安裝和設定快照控制器。如需詳細資訊，請參閱the section called “CSI 快照控制器”。

檢閱以下`VolumeSnapshotClass`參考 EKS Auto Mode 儲存功能的文章。

```
apiVersion: snapshot.storage.k8s.io/v1
kind: VolumeSnapshotClass
metadata:
  name: auto-ebs-vsclass
driver: ebs.csi.eks.amazonaws.com
deletionPolicy: Delete
```

進一步了解 Kubernetes CSI Snapshotter。

設定 EKS 自動模式設定

本章說明如何設定 Amazon Elastic Kubernetes Service (EKS) Auto Mode 叢集的特定層面。雖然 EKS Auto Mode 會自動管理大多數基礎設施元件，但您可以自訂特定功能以符合工作負載需求。

使用本主題中所述的組態選項，您可以修改聯網設定、運算資源和負載平衡行為，同時維持自動化基礎設施管理的優勢。在進行任何組態變更之前，請檢閱下列各節中的可用選項，以判斷哪種方法最適合您的需求。

您想要設定哪些功能？	組態選項
節點聯網和儲存 • 設定公有和私有子網路之間的節點放置 • 定義節點存取控制的自訂安全群組 • 自訂網路位址轉譯 (SNAT) 政策 • 啟用 Kubernetes 網路政策、詳細的網路政策記錄和監控 • 設定暫時性儲存參數（大小、IOPS、輸送量） • 使用自訂 KMS 金鑰設定加密的暫時性儲存	the section called “建立節點類別”

您想要設定哪些功能？	組態選項
• 將不同子網路中的 Pod 流量與節點隔離	
節點運算資源 • 選取特定的 EC2 執行個體類型和系列 • 定義 CPU 架構 (x86_64、ARM64) • 設定容量類型 （隨需、Spot) • 指定可用區域 • 設定節點污點和標籤 • 設定最小和最大節點計數	the section called “建立節點集區”
Application Load Balancer 設定 • 部署內部或面向網際網路的負載平衡器 • 設定跨區域負載平衡 • 設定閒置逾時期間 • 啟用 HTTP/2 和 WebSocket 支援 • 設定運作狀態檢查參數 • 指定 TLS 憑證設定 • 定義目標群組屬性 • 設定 IP 地址類型 (IPv4、雙堆疊)	the section called “建立輸入類別”
Network Load Balancer 設定 • 設定直接 Pod IP 路由 • 啟用跨區域負載平衡 • 設定連線閒置逾時 • 設定運作狀態檢查參數 • 指定子網路置放 • 設定 IP 地址類型 (IPv4、雙堆疊) • 設定保留用戶端來源 IP • 定義目標群組屬性	the section called “建立服務”

您想要設定哪些功能？	組態選項
儲存體方案設定 • 定義 EBS 磁碟區類型 (gp3、io1、io2 等） • 設定磁碟區加密和 KMS 金鑰用量 • 設定 IOPS 和輸送量參數 • 設定為預設儲存類別 • 定義佈建磁碟區的自訂標籤	the section called “建立 StorageClass”
控制 ODCR 用量 • 將工作負載部署設定為 EC2 隨需容量預留 • 針對目標容量用量，依 ID 明確選取特定 ODCRs • 使用標籤型選取來鎖定 ODCRs 群組 • 透過擁有跨 AWS 帳戶案例的帳戶來篩選 ODCRs • 控制工作負載是否自動使用開啟ODCRs	the section called “控制 ODCR 部署”

建立 Amazon EKS 的節點類別

Amazon EKS 節點類別是範本，可讓您精細控制 EKS Auto Mode 受管節點的組態。節點類別定義適用於 EKS 叢集中節點群組的基礎設施層級設定，包括網路組態、儲存設定和資源標記。本主題說明如何建立和設定節點類別，以符合您的特定操作需求。

當您需要自訂 EKS Auto Mode 在預設設定之外佈建和設定 EC2 執行個體的方式時，建立節點類別可讓您精確控制關鍵基礎設施參數。例如，您可以為增強安全性指定私有子網路置放、為效能敏感工作負載設定執行個體暫時性儲存，或為成本分配套用自訂標記。

建立節點類別

若要建立 `NodeClass`，請依照下列步驟進行：

1. 使用節點類別組態建立 YAML 檔案（例如 `nodeclass.yaml`）
2. 使用 將組態套用至您的叢集 `kubectl`

3. 參考節點集區組態中的節點類別。如需詳細資訊，請參閱the section called “建立節點集區”。

您需要kubectl安裝並設定 。如需詳細資訊，請參閱設定。

基本節點類別範例

以下是節點類別範例：

```
apiVersion: eks.amazonaws.com/v1
kind: NodeClass
metadata:
  name: private-compute
spec:
  subnetSelectorTerms:
    - tags:
        Name: "private-subnet"
        kubernetes.io/role/internal-elb: "1"
  securityGroupSelectorTerms:
    - tags:
        Name: "eks-cluster-sg"
  ephemeralStorage:
    size: "160Gi"
```

此 NodeClass 會增加節點上的暫時性儲存量。

使用 套用此組態：

```
kubectl apply -f nodeclass.yaml
```

接著，在您的節點集區組態中參考節點類別。如需詳細資訊，請參閱the section called “建立節點集區”。

建立節點類別存取項目

如果您建立自訂節點類別，則需要建立 EKS 存取項目，以允許節點加入叢集。當您使用內建節點類別和節點集區時，EKS 會自動建立存取項目。

如需存取項目如何運作的詳細資訊，請參閱 the section called “授予許可”。

為 EKS Auto Mode 節點類別建立存取項目時，您需要使用EC2存取項目類型。

使用 CLI 建立存取項目

若要建立 EC2 節點的存取項目並關聯 EKS Auto Node 政策：

使用叢集名稱和節點角色 ARN 更新下列 CLI 命令。節點角色 ARN 是在節點類別 YAML 中指定。

```
# Create the access entry for EC2 nodes
aws eks create-access-entry \
  --cluster-name <cluster-name> \
  --principal-arn <node-role-arn> \
  --type EC2

# Associate the auto node policy
aws eks associate-access-policy \
  --cluster-name <cluster-name> \
  --principal-arn <node-role-arn> \
  --policy-arn arn:aws: eks::aws:cluster-access-policy/AmazonEKSAutoNodePolicy \
  --access-scope type=cluster
```

使用 CloudFormation 建立存取項目

若要建立 EC2 節點的存取項目並關聯 EKS Auto Node 政策：

使用叢集名稱和節點角色 ARN 更新下列 CloudFormation。節點角色 ARN 是在節點類別 YAML 中指定。

```
EKSAutoNodeRoleAccessEntry:
  Type: AWS::EKS::AccessEntry
  Properties:
    ClusterName: <cluster-name>
    PrincipalArn: <node-role-arn>
    Type: "EC2"
    AccessPolicies:
      - AccessScope:
          Type: cluster
        PolicyArn: arn:aws: eks::aws:cluster-access-policy/AmazonEKSAutoNodePolicy
  DependsOn: [ <cluster-name> ] # previously defined in CloudFormation
```

如需部署 CloudFormation 堆疊的相關資訊，請參閱 CloudFormation 入門

節點類別規格

```
apiVersion: eks.amazonaws.com/v1
```

```
kind: NodeClass
metadata:
  name: my-node-class
spec:
  # Required fields
  role: MyNodeRole  # IAM role for EC2 instances

  subnetSelectorTerms:
    - tags:
        Name: "private-subnet"
        kubernetes.io/role/internal-elb: "1"
    # Alternative using direct subnet ID
    # - id: "subnet-0123456789abcdef0"

  securityGroupSelectorTerms:
    - tags:
        Name: "eks-cluster-sg"
    # Alternative approaches:
    # - id: "sg-0123456789abcdef0"
    # - name: "eks-cluster-security-group"

  # Optional: Pod subnet selector for advanced networking
  podSubnetSelectorTerms:
    - tags:
        Name: "pod-subnet"
        kubernetes.io/role/pod: "1"
    # Alternative using direct subnet ID
    # - id: "subnet-0987654321fedcba0"
# must include pod security group selector also
 podSecurityGroupSelectorTerms:
  - tags:
      Name: "eks-pod-sg"
    # Alternative using direct security group ID
    # - id: "sg-0123456789abcdef0"

  # Optional: Selects on-demand capacity reservations and capacity blocks
  # for EKS Auto Mode to prioritize.
  capacityReservationSelectorTerms:
    - id: cr-56fac701cc1951b03
    # Alternative Approaches
    - tags:
        Name: "targeted-odcr"
      # Optional owning account ID filter
      owner: "012345678901"
```

```
  # Optional fields
  snatPolicy: Random  # or Disabled

  networkPolicy: DefaultAllow  # or DefaultDeny
  networkPolicyEventLogs: Disabled  # or Enabled

  ephemeralStorage:
    size: "80Gi"    # Range: 1-59000Gi or 1-64000G or 1-58Ti or 1-64T
    iops: 3000      # Range: 3000-16000
    throughput: 125 # Range: 125-1000
    # Optional KMS key for encryption
    kmsKeyID: "arn:aws: kms:region:account:key/key-id"
    # Accepted formats:
    # KMS Key ID
    # KMS Key ARN
    # Key Alias Name
    # Key Alias ARN

  advancedNetworking:
    # Optional: Controls whether public IP addresses are assigned to instances that are
 launched with the nodeclass.
    # If not set, defaults to the MapPublicIpOnLaunch setting on the subnet.
    associatePublicIPAddress: false

    # Optional: Forward proxy, commonly requires certificateBundles as well
    # for EC2, see https://repost.aws/knowledge-center/eks-http-proxy-containerd-
automation
    httpsProxy: http://192.0.2.4:3128 #commonly port 3128 (Squid) or 8080 (NGINX) #Max
 255 characters
    #httpsProxy: http://[2001:db8::4]:3128 # IPv6 address with port, use []
    noProxy: #Max 50 entries
        - localhost #Max 255 characters each
        - 127.0.0.1
        #- ::1 # IPv6 localhost
        #- 0:0:0:0:0:0:0:1 # IPv6 localhost
        - 169.254.169.254 # EC2 Instance Metadata Service
        #- [fd00:ec2::254] # IPv6 EC2 Instance Metadata Service
        # Domains to exclude, put all VPC endpoints here
        - .internal
        - .eks.amazonaws.com

  # Optional: Custom certificate bundles.
  certificateBundles:
```

```
    - name: "custom-cert"
      data: "base64-encoded-cert-data"

  # Optional: Additional EC2 tags (with restrictions)
  tags:
    Environment: "production"
    Team: "platform"
    # Note: Cannot use restricted tags like:
    # - kubernetes.io/cluster/*
    # - karpenter.sh/provisioner-name
    # - karpenter.sh/nodepool
    # - karpenter.sh/nodeclaim
    # - karpenter.sh/managed-by
    # - eks.amazonaws.com/nodeclass
```

考量事項

- 如果您想要驗證執行個體有多少本機儲存體，您可以描述節點以查看暫時性儲存資源。
- 磁碟區加密 - EKS 使用設定的自訂 KMS 金鑰來加密執行個體的唯讀根磁碟區和讀取/寫入資料磁碟區。
- 取代節點 IAM 角色 - 如果您變更與 相關聯的節點 IAM 角色`NodeClass`，您將需要建立新的存取項目。在叢集建立期間，EKS 會自動為節點 IAM 角色建立存取項目。節點 IAM 角色需要 `AmazonEKSAutoNodePolicy` EKS 存取政策。如需詳細資訊，請參閱the section called “授予許可”。
- 最大 Pod 密度 - EKS 會將節點上的 Pod 數量上限限制為 110。此限制會在現有的最大 Pod 計算之後套用。如需詳細資訊，請參閱the section called “Amazon EC2 執行個體類型”。
- 標籤 - 如果您想要將標籤從 Kubernetes 傳播到 EC2，則需要設定其他 IAM 許可。如需詳細資訊，請參閱the section called “身分和存取”。
- 預設節點類別 - 請勿將您的自訂節點類別命名為 `default`。這是因為 EKS Auto Mode 包含`NodeClass`稱為 的 `default`，會在您啟用至少一個內建 時自動佈建`NodePool`。如需啟用內建的詳細資訊`NodePools`，請參閱 the section called “檢閱內建節點集區”。
- **`subnetSelectorTerms` 具有多個子網路的行為** - 如果有多個子網路符合`subnetSelectorTerms`條件或您透過 ID 提供，EKS Auto Mode 會建立分佈於子網路的節點。
  - 如果子網路位於不同的可用區域 (AZ)，您可以使用 Kubernetes 功能，例如 Pod 拓撲分散限制條件和拓撲感知路由，分別將 Pod 和流量分散到各個區域。
  - 如果同一個 AZ 中有多個子網路符合 `subnetSelectorTerms`，EKS Auto Mode 會在分佈在該 AZ 中子網路的每個節點上建立 Pod。EKS Auto Mode 會在相同 AZ 中其他子網路中的每個節點上

建立次要網路介面。它會根據每個子網路中的可用 IP 地址數量進行選擇，以更有效率地使用子網路。不過，您無法指定 EKS Auto Mode 為每個 Pod 使用的子網路；如果您需要 Pod 在特定子網路中執行，請the section called “Pod 的子網路選擇”改用 。

Pod 的子網路選擇

podSubnetSelectorTerms 和 podSecurityGroupSelectorTerms 欄位允許 Pod 在與其節點不同的子網路中執行，以啟用進階聯網組態。此區隔可增強對網路流量路由和安全政策的控制。請注意，podSecurityGroupSelectorTerms 是 的必要項目podSubnetSelectorTerms。

使用案例

當您需要以下項目podSubnetSelectorTerms時使用：

- 將基礎設施流量 node-to-node通訊） 與應用程式流量 (pod-to-pod通訊） 分開
- 將不同的網路組態套用至節點子網路，而非 Pod 子網路。
- 針對節點和 Pod 實作不同的安全政策或路由規則。
- 針對節點流量設定反向代理或網路篩選，而不會影響 Pod 流量。使用 advancedNetworking和 certificateBundles 來定義反向代理，以及代理的任何自我簽署或私有憑證。

範例組態

```
apiVersion: eks.amazonaws.com/v1
kind: NodeClass
metadata:
  name: advanced-networking
spec:
  role: MyNodeRole

  # Subnets for EC2 instances (nodes)
  subnetSelectorTerms:
    - tags:
        Name: "node-subnet"
        kubernetes.io/role/internal-elb: "1"

  securityGroupSelectorTerms:
    - tags:
        Name: "eks-cluster-sg"

  # Separate subnets for pods
```

```
  podSubnetSelectorTerms:
    - tags:
        Name: "pod-subnet"
        kubernetes.io/role/pod: "1"

  podSecurityGroupSelectorTerms:
  - tags:
      Name: "eks-pod-sg"
```

Pod 子網路選擇器的考量事項

- 降低 Pod 密度：使用 時，每個節點可以執行較少的 PodpodSubnetSelectorTerms，因為節點的主要網路介面位於節點子網路中，且無法用於 Pod 子網路中的 Pod。
- 子網路選擇器限制：標準subnetSelectorTerms和securityGroupSelectorTerms組態不適用於 Pod 子網路選擇。
- 網路規劃：確保節點和 Pod 子網路有足夠的 IP 地址空間來支援工作負載需求。
- 路由組態：確認已正確設定 Pod 子網路的路由表和網路存取控制清單 (ACL)，以便在節點和 Pod 子網路之間進行通訊。

為 EKS 自動模式建立節點集區

Amazon EKS 節點集區提供彈性的方式來管理 Kubernetes 叢集中的運算資源。本主題示範如何使用 Karpenter 建立和設定節點集區，這是有助於最佳化叢集擴展和資源使用率的節點佈建工具。透過 Karpenter 的 NodePool 資源，您可以定義運算資源的特定需求，包括執行個體類型、可用區域、架構和容量類型。

您無法修改內建system和general-purpose節點集區。您只能啟用或停用它們。如需詳細資訊，請參閱the section called “檢閱內建節點集區”。

NodePool 規格允許透過各種支援的標籤和需求，精細控制 EKS 叢集的運算資源。這些選項包括指定 EC2 執行個體類別、CPU 組態、可用區域、架構 (ARM64/AMD64) 和容量類型 （點或隨需）。您也可以設定 CPU 和記憶體用量的資源限制，確保您的叢集保持在所需的操作範圍內。

EKS Auto Mode 利用眾所周知的 Kubernetes 標籤，提供一致且標準化的節點特性識別方式。這些標籤，例如topology.kubernetes.io/zone可用區域和 CPU kubernetes.io/arch 架構，請遵循已建立的 Kubernetes 慣例。此外，EKS 特定的標籤 （字首為 eks.amazonaws.com/) 會使用 AWS 特定屬性來擴展此功能，例如執行個體類型、CPU 製造商、GPU 功能和聯網規格。此標準化標籤系統可無縫整合現有的 Kubernetes 工具，同時提供深度 AWS 基礎設施整合。

建立 NodePool

請依照下列步驟為您的 Amazon EKS 叢集建立 NodePool：

1. 使用所需的 NodePool 組態建立名為 `nodepool.yaml`的 YAML 檔案。您可以使用以下範例組態。
2. 將 NodePool 套用至您的叢集：

```
kubectl apply -f nodepool.yaml
```

3. 確認 NodePool 已成功建立：

```
kubectl get nodepools
```

4. （選用）監控 NodePool 狀態：

```
kubectl describe nodepool default
```

確保您的 NodePool 參考存在於叢集中的有效 NodeClass。NodeClass 會為您的運算資源定義 AWS特定的組態。如需詳細資訊，請參閱the section called “建立節點類別”。

範例 NodePool

```
apiVersion: karpenter.sh/v1
kind: NodePool
metadata:
  name: my-node-pool
spec:
  template:
    metadata:
      labels:
        billing-team: my-team
    spec:
      nodeClassRef:
        group: eks.amazonaws.com
        kind: NodeClass
        name: default

      requirements:
        - key: "eks.amazonaws.com/instance-category"
          operator: In
```

```
          values: ["c", "m", "r"]
        - key: "eks.amazonaws.com/instance-cpu"
          operator: In
          values: ["4", "8", "16", "32"]
        - key: "topology.kubernetes.io/zone"
          operator: In
          values: ["us-west-2a", "us-west-2b"]
        - key: "kubernetes.io/arch"
          operator: In
          values: ["arm64", "amd64"]

  limits:
    cpu: "1000"
    memory: 1000Gi
```

EKS 自動模式支援的標籤

EKS Auto Mode 支援下列眾所周知的標籤。

> **Note**
>
> EKS Auto Mode 使用與 Karpenter 不同的標籤。與 EC2 受管執行個體相關的標籤開頭為 `eks.amazonaws.com`。

標籤	範例	描述
topology.kubernetes.io/zone	us-east-2a	AWS 區域
node.kubernetes.io/instance-type	g4dn.8xlarge	AWS 執行個體類型
kubernetes.io/arch	amd64	架構由執行個體上的 GOARCH 值定義
karpenter.sh/capacity-type	Spot	容量類型包括 `spot`、 `on-demand`
eks.amazonaws.com/instance-hypervisor	nitro	使用特定 Hypervisor 的執行個體類型

標籤	範例	描述
eks.amazonaws.com/compute-type	auto	識別 EKS Auto Mode 受管節點
eks.amazonaws.com/instance-encryption-in-transit-supported	true	支援（或不支援）傳輸中加密的執行個體類型
eks.amazonaws.com/instance-category	g	相同類別的執行個體類型，通常是產生編號之前的字串
eks.amazonaws.com/instance-generation	4	執行個體類別內的執行個體類型產生編號
eks.amazonaws.com/instance-family	g4dn	類似屬性但資源數量不同的執行個體類型
eks.amazonaws.com/instance-size	8xlarge	類似資源數量但不同屬性的執行個體類型
eks.amazonaws.com/instance-cpu	32	執行個體上的 CPUs 數量
eks.amazonaws.com/instance-cpu-manufacturer	aws	CPU 製造商的名稱
eks.amazonaws.com/instance-memory	131072	執行個體上記憶體的 MB 數
eks.amazonaws.com/instance-ebs-bandwidth	9500	執行個體上可用的 EBS 最大百萬位元數
eks.amazonaws.com/instance-network-bandwidth	131072	執行個體上可用的基準百萬位元數
eks.amazonaws.com/instance-gpu-name	t4	執行個體上的 GPU 名稱，如果有的話
eks.amazonaws.com/instance-gpu-manufacturer	nvidia	GPU 製造商的名稱

標籤	範例	描述
eks.amazonaws.com/instance-gpu-count	1	執行個體上的 GPUs 數量
eks.amazonaws.com/instance-gpu-memory	16384	GPU 上記憶體的 MB 數
eks.amazonaws.com/instance-local-nvme	900	執行個體上本機 nvme 儲存體的 GB 數

> **Note**
>
> EKS Auto Mode 僅支援特定執行個體，且具有最小大小需求。如需詳細資訊，請參閱the section called “EKS Auto Mode 支援的執行個體參考”。

不支援的 EKS Auto 模式標籤

EKS Auto Mode 不支援下列標籤。

- EKS Auto Mode 僅支援 Linux
  - `node.kubernetes.io/windows-build`
  - `kubernetes.io/os`

停用內建節點集區

如果您建立自訂節點集區，您可以停用內建節點集區。如需詳細資訊，請參閱the section called “檢閱內建節點集區”。

沒有內建節點集區的叢集

您可以建立沒有內建節點集區的叢集。當您的組織建立自訂節點集區時，這會很有幫助。

> **Note**
>
> 當您建立不含內建節點集區的叢集時，不會自動佈建 `default` NodeClass。您需要建立自訂 NodeClass。如需詳細資訊，請參閱the section called “建立節點類別”。

概觀：

1. 建立 EKS 叢集，其中 `nodePools`和 `nodeRoleArn`值皆為空白。
   - 範例 eksctl`autoModeConfig`：

     ```
     autoModeConfig:
       enabled: true
       nodePools: []
       # Do not set a nodeRoleARN
     ```

     如需詳細資訊，請參閱the section called “eksctl CLI”
2. 使用節點角色 ARN 建立自訂節點類別
   - 如需詳細資訊，請參閱the section called “建立節點類別”
3. 建立自訂節點類別的存取項目
   - 如需詳細資訊，請參閱the section called “建立節點類別存取項目”
4. 建立自訂節點集區，如上所述。

中斷

您可以設定 EKS 自動模式，以多種方式透過 NodePool 中斷節點。您可以使用 `spec.disruption.consolidationPolicy`、 `spec.disruption.consolidateAfter`或 `spec.template.spec.expireAfter`。您也可以透過 NodePool 的 對限制 EKS Auto Mode 的中斷進行評分`spec.disruption.budgets`。您也可以控制時段和同時節點中斷的數量。如需設定此行為的指示，請參閱 Karpenter 文件中的中斷。

您可以將節點集區的中斷設定為：

- 識別執行個體未充分利用的時間，並合併工作負載。
- 建立節點集區中斷預算，以評定由於漂移、清空和合併而導致的節點終止限制。

根據預設，EKS Auto 模式：

- 合併未充分利用的執行個體。
- 在 336 小時後終止執行個體。
- 設定 10% 節點的單一中斷預算。
- 允許在發行新的自動模式 AMI 時，因偏離而取代節點，大約每週發生一次。

終止寬限期

在 `terminationGracePeriod` EKS Auto NodePool 上未明確定義 時，系統會自動將預設的 24 小時終止寬限期套用至相關聯的 NodeClaim。雖然 EKS Auto 客戶在其自訂 NodePool 組態中不會看到`terminationGracePeriod`預設值，但他們會在 NodeClaim 上觀察此預設值。無論寬限期是在 NodePool 上明確設定或在 NodeClaim 上預設，此功能都保持一致，以確保整個叢集的可預測節點終止行為。

建立 IngressClass 以設定 Application Load Balancer

EKS Auto Mode 會自動執行負載平衡的例行任務，包括將叢集應用程式公開至網際網路。

AWS 建議使用 Application Load Balancer (ALB) 來提供 HTTP 和 HTTPS 流量。Application Load Balancer 可以根據請求的內容路由請求。如需 Application Load Balancer 的詳細資訊，請參閱什麼是 Elastic Load Balancing？

EKS Auto Mode 會建立和設定 Application Load Balancer (ALBs)。例如，EKS Auto Mode 會在您建立 `Ingress` Kubernetes 物件時建立負載平衡器，並將其設定為將流量路由至叢集工作負載。

概觀

1. 建立您要向網際網路公開的工作負載。
2. 建立 `IngressClassParams` 資源，指定 AWS 特定組態值，例如用於 SSL/TLS 和 VPC 子網路的憑證。
3. 建立 `IngressClass` 資源，指定 EKS Auto Mode 將是資源的控制器。
4. 建立將 HTTP 路徑和連接埠與叢集工作負載建立關聯的`Ingress`資源。

EKS Auto Mode 將使用 `Ingress` 資源中指定的負載平衡器組態，建立指向 資源中指定之工作負載的 Application Load Balancer。 `IngressClassParams`

先決條件

- 在 Amazon EKS 叢集上啟用 EKS 自動模式
- 設定為連線至叢集的 Kubectl
  - 您可以使用 將下面的範例組態 YAML 檔案`kubectl apply -f <filename>`套用至您的叢集。

> **Note**
>
> EKS Auto Mode 需要子網路標籤來識別公有和私有子網路。
> 如果您使用 建立叢集`eksctl`，表示您已經有這些標籤。
> 了解如何 the section called “標記子網路”。

步驟 1：建立工作負載

建立您要向網際網路公開的工作負載。這可以是任何提供 HTTP 流量的 Kubernetes 資源，例如部署或服務。在此範例中，我們使用名為 的簡單 HTTP 服務`service-2048`，在連接埠 80 上提供 HTTP 流量。

步驟 2：建立 IngressClassParams

建立 `IngressClassParams` 物件以指定 Application Load Balancer AWS 的特定組態選項。在此範例中，我們會建立名為 `IngressClassParams` 的資源 `alb`（您將在下一個步驟中使用），指定名為`internet-facing`的檔案中的負載平衡器方案`alb-ingressclassparams.yaml`。

```
apiVersion: eks.amazonaws.com/v1
kind: IngressClassParams
metadata:
  name: alb
spec:
  scheme: internet-facing
```

將組態套用至您的叢集：

```
kubectl apply -f alb-ingressclassparams.yaml
```

步驟 3：建立 IngressClass

建立 IngressClass，參考名為 的檔案中IngressClassParams資源中設定 AWS 的特定組態值alb-ingressclass.yaml。請記下 的名稱IngressClass。在此範例中， IngressClass和 都命名IngressClassParams為 alb。

使用is-default-class註釋來控制Ingress資源是否預設應使用此類別。

```
apiVersion: networking.k8s.io/v1
kind: IngressClass
metadata:
  name: alb
  annotations:
    # Use this annotation to set an IngressClass as Default
    # If an Ingress doesn't specify a class, it will use the Default
    ingressclass.kubernetes.io/is-default-class: "true"
spec:
  # Configures the IngressClass to use EKS Auto Mode
  controller: eks.amazonaws.com/alb
  parameters:
    apiGroup: eks.amazonaws.com
    kind: IngressClassParams
    # Use the name of the IngressClassParams set in the previous step
    name: alb
```

如需組態選項的詳細資訊，請參閱 the section called “IngressClassParams 參考”。

將組態套用至您的叢集：

```
kubectl apply -f alb-ingressclass.yaml
```

步驟 4：建立輸入

在名為 的檔案中建立 Ingress 資源alb-ingress.yaml。此資源的目的是將 Application Load Balancer 上的路徑和連接埠與叢集中的工作負載建立關聯。在此範例中，我們會建立名為 Ingress的資源2048-ingress，將流量路由到連接埠 80 service-2048上名為 的服務。

如需設定此資源的詳細資訊，請參閱 Kubernetes 文件中的傳入。

```
apiVersion: networking.k8s.io/v1
kind: Ingress
```

```
metadata:
  name: 2048-ingress
spec:
  # this matches the name of IngressClass.
  # this can be omitted if you have a default ingressClass in cluster: the one with
 ingressclass.kubernetes.io/is-default-class: "true"  annotation
  ingressClassName: alb
  rules:
    - http:
        paths:
          - path: /*
            pathType: ImplementationSpecific
            backend:
              service:
                name: service-2048
                port:
                  number: 80
```

將組態套用至您的叢集：

```
kubectl apply -f alb-ingress.yaml
```

步驟 5：檢查狀態

使用 尋找 kubectl的狀態Ingress。負載平衡器可能需要幾分鐘的時間才能使用。

使用您在上一個步驟中設定Ingress的資源名稱。例如：

```
kubectl get ingress 2048-ingress
```

資源準備就緒後，請擷取負載平衡器的網域名稱。

```
kubectl get ingress 2048-ingress -o
 jsonpath='{.status.loadBalancer.ingress[0].hostname}'
```

若要在 Web 瀏覽器中檢視服務，請檢閱Ingress救援中指定的連接埠和路徑。

步驟 6：清除

若要清除負載平衡器，請使用下列命令：

```
kubectl delete ingress 2048-ingress
kubectl delete ingressclass alb
kubectl delete ingressclassparams alb
```

EKS Auto Mode 會自動刪除您 AWS 帳戶中相關聯的負載平衡器。

IngressClassParams 參考

下表是常用組態選項的快速參考。

欄位	描述	範例值
scheme	定義 ALB 是內部還是面向網際網路	internet-facing
namespaceSelector	限制哪些命名空間可以使用此 IngressClass	environment: prod
group.name	將多個輸入分組以共用單一 ALB	retail-apps
ipAddressType	設定 ALB 的 IP 地址類型	dualstack
subnets.ids	ALB 部署IDs 清單	subnet-xxxx, subnet-yyyy
subnets.tags	標記篩選條件以選取 ALB 的子網路	Environment: prod
certificateARNs	要使用的 SSL 憑證 ARNs	arn:aws: acm:region:account:certificate/id
tags	AWS 資源的自訂標籤	Environment: prod, Team: platform
loadBalancerAttributes	負載平衡器特定屬性	idle_timeout.timeout_seconds: 60

考量事項

- 您無法在 IngressClass 上使用註釋來設定具有 EKS Auto 模式的負載平衡器。
- 您必須更新叢集 IAM 角色，才能啟用從 Kubernetes 到 AWS Load Balancer資源的標籤傳播。如需詳細資訊，請參閱the section called “EKS Auto 資源的自訂 AWS 標籤”。
- 如需將資源與 EKS Auto Mode 或 self-managed AWS Load Balancer Controller 建立關聯的資訊，請參閱 the section called “遷移參考”。
- 如需修正負載平衡器問題的相關資訊，請參閱 the section called “疑難排解”。
- 如需有關使用 EKS Auto Mode 負載平衡功能的更多考量，請參閱 the section called “負載平衡”。

下表提供 IngressClassParams、Ingress 註釋和 TargetGroupBinding 組態中 EKS Auto Mode 變更的詳細比較。這些資料表強調了 EKS Auto Mode 的負載平衡功能與開放原始碼負載平衡器控制器之間的主要差異，包括 API 版本變更、已棄用的功能和更新的參數名稱。

IngressClassParams

上一個	新增	描述
`elbv2.k8s.aws/v1beta1`	`eks.amazonaws.com/v1`	API 版本變更
`spec.certificateArn`	`spec.certificateARNs`	支援多個憑證 ARNs
`spec.subnets.tags`	`spec.subnets.matchTags`	變更子網路比對結構描述
`spec.listeners.listenerAttributes`	`spec.listeners.attributes`	簡化屬性命名

輸入註釋

上一個	新增	描述
`kubernetes.io/ingress.class`	不支援	在輸入物件`spec.ingressClassName` 上使用

上一個	新增	描述
alb.ingress.kubernetes.io/group.name	不支援	僅在 IngressClass 中指定群組
alb.ingress.kubernetes.io/waf-acl-id	不支援	請改用 WAF v2
alb.ingress.kubernetes.io/web-acl-id	不支援	請改用 WAF v2
alb.ingress.kubernetes.io/shield-advanced-protection	不支援	停用 Shield 整合
alb.ingress.kubernetes.io/auth-type: oidc	不支援	目前不支援 OIDC 驗證類型

TargetGroupBinding

上一個	新增	描述
elbv2.k8s.aws/v1beta1	eks.amazonaws.com/v1	API 版本變更
spec.targetType 選用	spec.targetType 必要	明確目標類型規格
spec.networking.ingress.from	不支援	不再支援沒有安全群組的 NLB

使用服務註釋來設定 Network Load Balancer

了解如何使用 Kubernetes 服務註釋在 Amazon EKS 中設定 Network Load Balancer (NLB)。本主題說明 EKS Auto Mode 支援用於自訂 NLB 行為的註釋，包括網際網路可存取性、運作狀態檢查、SSL/TLS 終止和 IP 目標模式。

當您在 EKS Auto Mode `LoadBalancer`中建立 類型的 Kubernetes 服務時，EKS 會根據您指定的註釋自動佈建和設定 AWS Network Load Balancer。這種宣告式方法可讓您直接透過 Kubernetes 資訊清單管理負載平衡器組態，將基礎設施維護為程式碼實務。

根據預設，EKS Auto Mode 會針對 LoadBalancer 類型的所有服務處理 Network Load Balancer 佈建 - 不需要額外的控制器安裝或組態。 LoadBalancer `loadBalancerClass: eks.amazonaws.com/nlb` 規格會自動設定為叢集預設值，簡化部署程序，同時維持與現有 Kubernetes 工作負載的相容性。

> **Note**
>
> EKS Auto Mode 需要子網路標籤來識別公有和私有子網路。
> 如果您使用 建立叢集`eksctl`，表示您已經有這些標籤。
> 了解如何 the section called “標記子網路”。

範例服務

如需 Kubernetes `Service` 資源的詳細資訊，請參閱 Kubernetes 文件。

檢閱以下範例`Service`資源：

```
apiVersion: v1
kind: Service
metadata:
  name: echoserver
  annotations:
    # Specify the load balancer scheme as internet-facing to create a public-facing
 Network Load Balancer (NLB)
    service.beta.kubernetes.io/aws-load-balancer-scheme: internet-facing
spec:
  selector:
    app: echoserver
  ports:
    - port: 80
      targetPort: 8080
      protocol: TCP
  type: LoadBalancer
  # Specify the new load balancer class for NLB as part of EKS Auto Mode feature
  # For clusters with Auto Mode enabled, this field can be omitted as it's the default
  loadBalancerClass: eks.amazonaws.com/nlb
```

常用的註釋

下表列出 EKS Auto Mode 支援的常用註釋。請注意，EKS Auto Mode 可能不支援所有註釋。

> **Tip**
> 下列所有註釋都需要加上字首 `service.beta.kubernetes.io/`

欄位	描述	範例
`aws-load-balancer-type`	指定負載平衡器類型。使用 `external` 進行新的部署。	`external`
`aws-load-balancer-nlb-target-type`	指定要將流量路由到節點執行個體還是直接路由到 Pod IPs。將 `instance`用於標準部署或`ip`直接 Pod 路由。	`instance`
`aws-load-balancer-scheme`	控制負載平衡器是內部還是面向網際網路。	`internet-facing`
`aws-load-balancer-healthcheck-protocol`	目標群組的運作狀態檢查通訊協定。常見選項為 TCP（預設）或 HTTP。	`HTTP`
`aws-load-balancer-healthcheck-path`	使用 HTTP/HTTPS 通訊協定時運作狀態檢查的 HTTP 路徑。	`/healthz`
`aws-load-balancer-healthcheck-port`	用於運作狀態檢查的連接埠。可以是特定的連接埠號碼或 `traffic-port` 。	`traffic-port`
`aws-load-balancer-subnets`	指定要在其中建立負載平衡器的子網路。可以使用子網路 IDs或名稱。	`subnet-xxxx, subnet-yyyy`

欄位	描述	範例
aws-load-balancer-ssl-cert	來自 AWS Certificate Manager for HTTPS/TLS 的 SSL 憑證 ARN。	arn:aws: acm:region:account:certificate/cert-id
aws-load-balancer-ssl-ports	指定哪些連接埠應使用 SSL/TLS。	443, 8443
load-balancer-source-ranges	允許存取負載平衡器的 CIDR 範圍。	10.0.0.0/24, 192.168.1.0/24
aws-load-balancer-additional-resource-tags	要套用至負載平衡器和相關資源的其他 AWS 標籤。	Environment=prod,Team=platform
aws-load-balancer-ip-address-type	指定負載平衡器是使用 IPv4 還是雙堆疊 (IPv4 + IPv6)。	ipv4 或 dualstack

考量事項

- 您必須更新叢集 IAM 角色，才能啟用從 Kubernetes 到 AWS Load Balancer資源的標籤傳播。如需詳細資訊，請參閱the section called “EKS Auto 資源的自訂 AWS 標籤”。
- 如需有關將資源與 EKS Auto Mode 或 self-managed AWS Load Balancer Controller 建立關聯的資訊，請參閱 the section called “遷移參考”。
- 如需修正負載平衡器問題的相關資訊，請參閱 the section called “疑難排解”。
- 如需有關使用 EKS Auto Mode 負載平衡功能的更多考量，請參閱 the section called “負載平衡”。

遷移至 EKS Auto Mode 進行負載平衡時，服務註釋和資源組態中需要進行數項變更。下表概述了先前實作和新實作之間的主要差異，包括不支援的選項和建議的替代方案。

服務註釋

上一個	新增	描述
service.beta.kubernetes.io/load-balancer-source-ranges	不支援	在 服務spec.loadBalancerSourceRanges 上使用
service.beta.kubernetes.io/aws-load-balancer-type	不支援	在 服務spec.loadBalancerClass 上使用
service.beta.kubernetes.io/aws-load-balancer-internal	不支援	使用 service.beta.kubernetes.io/aws-load-balancer-scheme
各種負載平衡器屬性	不支援	使用 service.beta.kubernetes.io/aws-load-balancer-attributes
service.beta.kubernetes.io/aws-load-balancer-proxy-protocol	不支援	請service.beta.kubernetes.io/aws-load-balancer-attributes 改用
service.beta.kubernetes.io/aws-load-balancer-access-log-enabled	不支援	請service.beta.kubernetes.io/aws-load-balancer-attributes 改用
service.beta.kubernetes.io/aws-load-balancer-access-log-s3-bucket-name	不支援	請service.beta.kubernetes.io/aws-load-balancer-attributes 改用
service.beta.kubernetes.io/aws-load-	不支援	請service.beta.kubernetes.io/aws-load-

上一個	新增	描述
balancer-access-log-s3-bucket-prefix		balancer-attributes 改用
service.beta.kubernetes.io/aws-load-balancer-cross-zone-load-balancing-enabled	不支援	請service.beta.kubernetes.io/aws-load-balancer-attributes 改用

若要從已棄用的負載平衡器屬性註釋遷移，請將這些設定合併到service.beta.kubernetes.io/aws-load-balancer-attributes註釋中。此註釋接受各種負載平衡器屬性的鍵值對逗號分隔清單。例如，若要指定代理通訊協定、存取記錄和跨區域負載平衡，請使用下列格式：

```
service.beta.kubernetes.io/aws-load-balancer-attributes:
 access_logs.s3.enabled=true,access_logs.s3.bucket=my-bucket,access_logs.s3.prefix=my-prefix,load_balancing.cross_zone.enabled=true
```

此合併格式提供更一致且靈活的方法來設定負載平衡器屬性，同時減少所需的個別註釋數量。檢閱您現有的服務組態，並更新它們以使用此合併格式。

TargetGroupBinding

上一個	新增	描述
elbv2.k8s.aws/v1beta1	eks.amazonaws.com/v1	API 版本變更
spec.targetType 選用	spec.targetType 必要	明確目標類型規格
spec.networking.ingress.from	不支援	不再支援沒有安全群組的 NLB

建立儲存體方案

Amazon EKS Auto Mode StorageClass中的 定義了應用程式請求持久性儲存時自動佈建 Amazon EBS 磁碟區的方式。此頁面說明如何建立和設定StorageClass可搭配 Amazon EKS Auto 模式使用的 來佈建 EBS 磁碟區。

透過設定 StorageClass，您可以為 EBS 磁碟區指定預設設定，包括磁碟區類型、加密、IOPS 和其他儲存參數。您也可以StorageClass將 設定為使用 AWS KMS 金鑰進行加密管理。

EKS Auto Mode 不會StorageClass為您建立 。您必須建立StorageClass參考，ebs.csi.eks.amazonaws.com才能使用 EKS Auto Mode 的儲存功能。

首先，建立名為 的檔案storage-class.yaml：

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: auto-ebs-sc
  annotations:
    storageclass.kubernetes.io/is-default-class: "true"
provisioner: ebs.csi.eks.amazonaws.com
volumeBindingMode: WaitForFirstConsumer
parameters:
  type: gp3
  encrypted: "true"
```

其次，將儲存體方案套用至您的叢集。

```
kubectl apply -f storage-class.yaml
```

關鍵元件：

- provisioner: ebs.csi.eks.amazonaws.com - 使用 EKS Auto 模式
- volumeBindingMode: WaitForFirstConsumer - 延遲磁碟區建立，直到 Pod 需要它為止
- type: gp3 - 指定 EBS 磁碟區類型
- encrypted: "true" - EBS 會加密使用 建立的任何磁碟區StorageClass。EBS 將使用預設aws/ebs金鑰別名。如需詳細資訊，請參閱《Amazon EBS 使用者指南》中的 Amazon EBS 加密如何運作。此值是選用的，但建議使用。

- `storageclass.kubernetes.io/is-default-class: "true"` - Kubernetes 預設會使用此儲存類別，除非您在持久性磁碟區宣告上指定不同的磁碟區類別。此值是選用的。如果您要從不同的儲存控制器遷移，請謹慎設定此值。

使用自我管理 KMS 金鑰加密 EBS 磁碟區

若要使用自我管理 KMS 金鑰來加密 EKS Auto Mode 自動化的 EBS 磁碟區，您需要：

1. 建立自我管理的 KMS 金鑰。
   - 如需詳細資訊，請參閱《KMS 使用者指南》中的建立對稱加密 KMS 金鑰或 Amazon Elastic Block Store (Amazon EBS) 如何使用 KMS。
2. 建立新的政策，以允許存取 KMS 金鑰。
   - 使用以下範例 IAM 政策來建立政策。插入新自我管理 KMS 金鑰的 ARN。如需詳細資訊，請參閱《IAM 使用者指南》中的建立角色和連接政策（主控台）。 AWS
3. 將政策連接至 EKS 叢集角色。
   - 使用 AWS 主控台尋找 EKS 叢集角色的 ARN。角色資訊會顯示在概觀區段中。如需詳細資訊，請參閱the section called “叢集 IAM 角色”。
4. 更新 `StorageClass`以參考 `parameters.kmsKeyId` 欄位的 KMS 金鑰 ID。

自我管理 KMS IAM 政策範例

在下列政策中更新下列值：

- `<account-id>` – AWS 您的帳戶 ID，例如 `111122223333`
- `<aws-region>` – 叢集 AWS 的區域，例如 `us-west-2`

```
{
  "Version": "2012-10-17",
  "Id": "key-auto-policy-3",
  "Statement": [
      {
          "Sid": "Enable IAM User Permissions",
          "Effect": "Allow",
          "Principal": {
              "AWS": "arn:aws: iam::<account-id>:root"
          },
          "Action": "kms:*",
```

```
          "Resource": "*"
      },
      {
        "Sid": "Allow access through EBS for all principals in the account that are
 authorized to use EBS",
        "Effect": "Allow",
        "Principal": {
            "AWS": "*"
        },
        "Action": [
            "kms:Encrypt",
            "kms:Decrypt",
            "kms:ReEncrypt*",
            "kms:GenerateDataKey*",
            "kms:CreateGrant",
            "kms:DescribeKey"
        ],
        "Resource": "*",
        "Condition": {
            "StringEquals": {
                "kms:CallerAccount": "<account-id>",
                "kms:ViaService": "ec2.<aws-region>.amazonaws.com"
            }
        }
    }
  ]
}
```

自我管理 KMS 範例 **StorageClass**

```
parameters:
  type: gp3
  encrypted: "true"
  kmsKeyId: <custom-key-arn>
```

StorageClass 參數參考

如需 Kubernetes StorageClass 資源的一般資訊，請參閱 Kubernetes 文件中的儲存類別。

StorageClass 資源的 THe parameters區段是特定的 AWS。使用下表來檢閱可用的選項。

參數	值	預設	描述
"csi.storage.k8s.io/fstype"	xfs、ext2、ext3、ext4	ext4	在磁碟區建立期間格式化的檔案系統類型。此參數區分大小寫！
「類型」	io1、io2、gp2、gp3、sc1、st1、Standard、sbp1、sbg1	gp3	EBS 磁碟區類型。
「iopsPerGB」			每秒每個 GiB 的 I/O 操作。可以為 IO1, IO2和 GP3 磁碟區指定。
「allowAutoIOPSPerGBIncrease」	true、false	false	當 時"true"，當 iopsPerGB * <volume size> 太低而無法符合 支援的 IOPS 範圍時，CSI 驅動程式會增加磁碟區的 IOPS AWS。這可讓動態佈建一律成功，即使使用者指定太小的 PVC 容量或iopsPerGB 值也一樣。另一方面，它可能會帶來額外的成本，因為這類磁碟區的 IOPS 高於 中請求的 IOPSiopsPerGB 。

參數	值	預設	描述
"iops"			每秒的 I/O 操作。可以為 IO1, IO2和 GP3 磁碟區指定。
「輸送量」		125	以 MiB/s 為單位的輸送量。只有在指定 gp3 磁碟區類型時才有效。
「加密」	true、false	false	磁碟區是否應該加密。有效值為「true」或「false」。
"blockExpress"	true、false	false	啟用建立 io2 Block Express 磁碟區。
"kmsKeyId"			加密磁碟區時要使用之金鑰的完整 ARN。如果未指定， AWS 將針對磁碟區所在的區域使用預設 KMS 金鑰。如果未變更/aws/ebs，這將是一個自動產生的金鑰，稱為 。
"blockSize"			格式化基礎檔案系統時要使用的區塊大小。僅 linux 節點和具有 fstype ext2、ext4、ext3或 的 支援xfs。

參數	值	預設	描述
「inodeSize」			格式化基礎檔案系統時要使用的節點大小。僅 linux 節點和具有 fstype ext2、ext4、ext3或 的 支援xfs。
"bytesPerInode"			格式化基礎檔案系統時bytes-per-inode要使用的 。僅 linux 節點和具有 fstype ext2、ext3、的 支援ext4。
「numberOfInodes」			格式化基礎檔案系統時number-of-inodes要使用的 。僅 linux 節點和具有 fstype ext2、ext3、的 支援ext4。
"ext4BigAlloc"	true、false	false	透過啟用bigalloc格式化選項，將ext4檔案系統變更為使用叢集區塊配置。警告：bigalloc 可能未完全支援節點的 Linux 核心。
"ext4ClusterSize"			啟用 bigalloc功能時，格式化 ext4 檔案系統時要使用的叢集大小。注意：ext4BigAlloc 參數必須設定為 true。

如需詳細資訊，請參閱 GitHub 上的 AWS EBS CSI 驅動程式。

考量事項

> **Note**
>
> 您只能根據 EKS Auto Mode 節點上的 EKS Auto Mode StorageClasses 部署工作負載。如果您有具有混合類型節點的叢集，您需要將工作負載設定為僅在 EKS Auto Mode 節點上執行。如需詳細資訊，請參閱the section called “控制部署”。

EKS Auto Mode 的區塊儲存功能與 EBS CSI 驅動程式不同。

- 靜態佈建
  - 如果您想要搭配 EKS Auto 模式使用外部建立的 EBS 磁碟區，則需要手動新增具有 金鑰`eks:eks-cluster-name`和叢集名稱值的 AWS 標籤。
- 節點啟動標記
  - 您無法使用節點啟動污點功能，在儲存功能就緒之前防止 Pod 排程
- 動態佈建磁碟區上的自訂標籤
  - 您無法使用額外標籤 CLI 旗標在動態佈建的 EBS 磁碟區上設定自訂標籤
  - 您可以使用`StorageClass`標記來新增自訂標籤。EKS Auto Mode 會將標籤新增至相關聯的 AWS 資源。您需要更新自訂標籤的叢集 IAM 角色。如需詳細資訊，請參閱the section called “EKS Auto 資源的自訂 AWS 標籤”。
- EBS 詳細效能指標
  - 您無法存取 EBS 詳細效能的 Prometheus 指標

安裝 CSI 快照控制器附加元件

EKS Auto Mode 與 CSI 快照控制器 Amazon EKS 附加元件相容。

AWS 建議您將此附加元件設定為在內建`system`節點集區上執行。

如需詳細資訊，請參閱：

- the section called “執行關鍵附加元件”
- the section called “檢閱內建節點集區”

- the section called “CSI 快照控制器”

在系統節點集區中安裝快照控制器

1. 在 AWS 主控台中開啟您的 EKS 叢集
2. 從附加元件索引標籤中，選取取得更多附加元件
3. 選取 CSI 快照控制器，然後選取下一步
4. 在設定選取的附加元件設定頁面上，選取選用組態設定以檢視附加元件組態結構描述
   a. 插入下列 yaml，將快照控制器與`system`節點集區建立關聯。快照控制器包含`CriticalAddonsOnly`污點的容錯。

```
{
        "nodeSelector": {
            "karpenter.sh/nodepool": "system"
        }
}
```

   b. 選取下一步
5. 檢閱附加元件組態，然後選取建立

停用 EKS Auto 模式

您可以在現有的 EKS 叢集上停用 EKS 自動模式。這是破壞性操作。

- EKS 將終止所有由 EKS Auto Mode 操作的 EC2 執行個體。
- EKS 會刪除由 EKS Auto Mode 操作的所有負載平衡器。
- EKS 不會刪除 EKS Auto 模式佈建的 EBS 磁碟區。

EKS Auto Mode 旨在完整管理其建立的資源。手動介入可能會導致 EKS Auto Mode 在停用時無法完全清除這些資源。例如，如果您從外部安全群組規則參考受管安全群組，並忘記在停用叢集的 EKS Auto Mode 之前移除該參考，則受管安全群組將會洩漏（未刪除）。以下步驟說明如何在發生這種情況時移除洩漏的安全群組。

停用 EKS Auto 模式AWS（主控台）

1. 在 中開啟叢集概觀頁面 AWS Management Console。

2. 在 EKS 自動模式下，選取管理
3. 將 EKS Auto 模式切換到 off。

如果此程序結束時未刪除任何受管安全群組，您可以使用刪除安全群組中的描述手動刪除。

停用 EKS 自動模式 (AWS CLI)

使用下列命令來停用現有叢集上的 EKS Auto Mode。

您需要安裝 aws CLI，並使用足夠的許可來登入以管理 EKS 叢集。如需詳細資訊，請參閱設定。

> **Note**
>
> 運算、區塊儲存和負載平衡功能都必須在相同的請求中啟用或停用。

```
aws eks update-cluster-config \
 --name $CLUSTER_NAME \
 --compute-config enabled=false \
 --kubernetes-network-config '{"elasticLoadBalancing":{"enabled": false}}' \
 --storage-config '{"blockStorage":{"enabled": false}}'
```

您可以在停用 EKS Auto Mode 後，檢查洩漏的 EKS Auto Mode 安全群組是否無法刪除，如下所示：

```
aws ec2 describe-security-groups \
    --filters Name=tag:eks:eks-cluster-name,Values=<cluster-Name> Name=tag-
key,Values=ingress.eks.amazonaws.com/resource,service.eks.amazonaws.com/resource --
query "SecurityGroups[*].[GroupName]"
```

若要接著刪除安全群組：

```
aws ec2 delete-security-group --group-name=<sg-name>
```

更新 EKS Auto Mode 叢集的 Kubernetes 版本

本主題說明如何更新自動模式叢集的 Kubernetes 版本。自動模式透過節點替換處理控制平面更新的協調，同時透過 Pod 中斷預算維持工作負載可用性，簡化版本更新程序。

升級自動模式叢集時，許多傳統上需要手動更新的元件現在會做為服務的一部分進行管理。了解升級程序的自動化層面和您的責任，有助於確保叢集的版本順利轉換。

了解 EKS Auto 模式的更新

啟動控制平面升級後，EKS Auto Mode 會升級叢集中的節點。隨著節點過期，EKS Auto Mode 會將它們取代為新的節點。新節點具有對應的新 Kubernetes 版本。升級節點時，EKS Auto Mode 會觀察 Pod 中斷預算。

此外，您不再需要更新下列元件：

- Amazon VPC CNI
- AWS Load Balancer控制器
- CoreDNS
- `kube-proxy`
- Karpenter
- AWS EBS CSI 驅動程式

EKS Auto Mode 會將這些元件取代為服務功能。

您仍需負責更新：

- 部署到您的叢集的應用程式和工作負載
- 自我管理的附加元件和控制器
- Amazon EKS 附加元件
  - 了解如何 the section called “更新 附加元件”

了解叢集升級的最佳實務

啟動叢集更新

若要開始叢集更新，請參閱 the section called “更新 Kubernetes 版本”。

啟用或停用內建 NodePools

EKS Auto Mode 有兩個內建的 NodePools 您可以使用 AWS 主控台、CLI 或 API 啟用或停用這些 NodePools。

內建 NodePool 參考

- `system`

  - 此 NodePool `CriticalAddonsOnly` 有污點。許多 EKS 附加元件，例如 CoreDNS，可容忍此污點。使用此系統節點集區來分隔叢集關鍵應用程式。
  - 同時支援 `amd64`和 `arm64`架構。
- `general-purpose`
  - 此 NodePool 支援在您的叢集中啟動一般用途工作負載的節點。
  - 僅使用`amd64`架構。

兩個內建的 NodePools：

- 使用預設 EKS NodeClass
- 僅使用隨需 EC2 容量
- 使用 C、M 和 R EC2 執行個體系列
- 需要第 5 代或更新版本的 EC2 執行個體

> **Note**
>
> EKS 需要啟用至少一個內建 NodePool 才能佈建 "default" NodeClass。如果您停用所有內建 NodePools，您將需要建立自訂 NodeClass 並設定 NodePool 來使用它。如需 NodeClasses 的詳細資訊，請參閱 the section called “建立節點類別”。

程序

先決條件

- 在您裝置上安裝和設定的最新版本 AWS 命令列界面 (AWS CLI)。若要檢查您目前的版本，請使用 `aws --version`。若要安裝最新版本，請參閱《 AWS 命令列界面使用者指南》中的使用 aws 設定安裝 和 快速組態。
  - 使用足夠的 IAM 許可登入 CLI，以建立 AWS 資源，包括 IAM 政策、IAM 角色和 EKS 叢集。

使用 CLI AWS 啟用

使用下列命令來啟用兩個內建的 NodePools：

```
aws eks update-cluster-config \
  --name <cluster-name> \
```

```
  --compute-config '{
    "nodeRoleArn": "<node-role-arn>",
    "nodePools": ["general-purpose", "system"],
    "enabled": true
  }' \
  --kubernetes-network-config '{
  "elasticLoadBalancing":{"enabled": true}
  }' \
  --storage-config '{
  "blockStorage":{"enabled": true}
  }'
```

您可以修改命令以選擇性地啟用 NodePools。

使用 AWS CLI 停用

使用下列命令來停用兩個內建的 NodePools：

```
aws eks update-cluster-config \
  --name <cluster-name> \
  --compute-config '{
  "enabled": true,
  "nodePools": []
  }' \
  --kubernetes-network-config '{
  "elasticLoadBalancing":{"enabled": true}}' \
  --storage-config '{
  "blockStorage":{"enabled": true}
  }'
```

控制工作負載是否部署在 EKS Auto Mode 節點上

在具有 EKS Auto Mode 的 EKS 叢集中執行工作負載時，您可能需要控制特定工作負載是否在 EKS Auto Mode 節點或其他運算類型上執行。本主題說明如何使用節點選擇器和親和性規則，以確保您的工作負載已排程在預期的運算基礎設施上。

本主題中的範例示範如何使用 `eks.amazonaws.com/compute-type`標籤來要求或防止 EKS Auto Mode 節點上的工作負載部署。這特別適用於同時執行 EKS Auto Mode 和其他運算類型的混合模式叢集，例如自我管理 Karpenter 佈建器或 EKS 受管節點群組。

EKS Auto Mode 節點已將標籤的值設定為 `eks.amazonaws.com/compute-type auto`。您可以使用此標籤來控制工作負載是否部署到由 EKS Auto Mode 管理的節點。

需要將工作負載部署到 EKS Auto Mode 節點

> **Note**
>
> EKS Auto Mode 不需要此`nodeSelector`值。此`nodeSelector`值只有在您以混合模式執行叢集時才相關，節點類型不是由 EKS Auto Mode 管理。例如，您可能已使用 EKS 受管節點群組將靜態運算容量部署至叢集，並具有由 EKS Auto Mode 管理的動態運算容量。

您可以將此`nodeSelector`項目新增至部署或其他工作負載，以要求 Kubernetes 將它們排程到 EKS Auto Mode 節點。

```
apiVersion: apps/v1
kind: Deployment
spec:
  template:
    nodeSelector:
      eks.amazonaws.com/compute-type: auto
```

需要工作負載不會部署到 EKS Auto Mode 節點

您可以將此新增至`nodeAffinity`部署或其他工作負載，以要求 Kubernetes 不將它們排程到 EKS Auto Mode 節點。

```
  affinity:
    nodeAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        nodeSelectorTerms:
        - matchExpressions:
          - key: eks.amazonaws.com/compute-type
            operator: NotIn
            values:
            - auto
```

在專用執行個體上執行關鍵附加元件

在本主題中，您將了解如何部署具有`CriticalAddonsOnly`容錯能力的工作負載，因此 EKS Auto Mode 會將工作負載排程到`system`節點集區。

EKS Auto Mode 的內建system節點集區專為在專用執行個體上執行關鍵附加元件而設計。此隔離可確保基本元件具有專用資源，並與一般工作負載隔離，從而增強整體叢集穩定性和效能。

本指南示範如何使用CriticalAddonsOnly容錯和適當的system節點選取器，將附加元件部署至節點集區。透過遵循這些步驟，您可以確保將關鍵應用程式排程在專用system節點上，利用 EKS Auto Mode 專用節點集區結構提供的隔離和資源配置優勢。

EKS Auto Mode 有兩個內建節點集區： general-purpose和 system。如需詳細資訊，請參閱the section called “檢閱內建節點集區”。

system 節點集區的目的是將關鍵附加元件隔離到不同的節點。節點集區佈建的system節點具有 CriticalAddonsOnly Kubernetes 污點。Kubernetes 只會在這些節點具有對應的容錯能力時，將 Pod 排程到這些節點上。如需詳細資訊，請參閱 Kubernetes 文件中的 標記和容錯。

先決條件

- 已啟用內建system節點集區的 EKS 自動模式叢集。如需詳細資訊，請參閱 the section called “檢閱內建節點集區”
- kubectl 已安裝並設定。如需詳細資訊，請參閱設定。

程序

檢閱下方的 yaml 範例。請注意下列組態：

- nodeSelector— 這會將工作負載與內建system節點集區建立關聯。此節點集區必須使用 AWS API 啟用。如需詳細資訊，請參閱the section called “檢閱內建節點集區”。
- tolerations— 此容錯能力可克服節點集區中system節點上的CriticalAddonsOnly污點。

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: sample-app
spec:
  replicas: 3
  selector:
    matchLabels:
      app: sample-app
  template:
    metadata:
```

```
      labels:
        app: sample-app
    spec:
      nodeSelector:
        karpenter.sh/nodepool: system
      tolerations:
      - key: "CriticalAddonsOnly"
        operator: "Exists"
      containers:
      - name: app
        image: nginx:latest
        resources:
          requests:
            cpu: "500m"
            memory: "512Mi"
```

若要更新要在system節點集區上執行的工作負載，您需要：

1. 更新現有的工作負載以新增上述組態：
   - nodeSelector
   - tolerations
2. 使用 將更新的工作負載部署到您的叢集 kubectl apply

更新工作負載之後，它將在專用節點上執行。

搭配 EKS Auto 模式使用網路政策

網路政策可讓您控制 Amazon EKS 叢集內 IP 地址或連接埠層級的流量流程。本主題說明如何透過 EKS Auto 模式啟用和使用網路政策。

先決條件

- 啟用 EKS Auto 模式的 Amazon EKS 叢集
- 設定為連線至叢集的 kubectl

步驟 1：啟用網路政策控制器

若要搭配 EKS Auto Mode 使用網路政策，您必須先將 ConfigMap 套用至叢集，以啟用網路政策控制器。

1. 建立名為 enable-network-policy.yaml 且具有下列內容的檔案：

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: amazon-vpc-cni
  namespace: kube-system
data:
  enable-network-policy-controller: "true"
```

2. 將 ConfigMap 套用至您的叢集：

```
kubectl apply -f enable-network-policy.yaml
```

步驟 2：在節點類別中啟用網路政策

在使用網路政策之前，您需要確保您的節點類別已設定為支援它們。請遵循下列步驟：

1. 使用下列內容建立或編輯節點類別 YAML 檔案（例如 nodeclass-network-policy.yaml)：

```
apiVersion: eks.amazonaws.com/v1
kind: NodeClass
metadata:
  name: network-policy-enabled
spec:
  # Enables network policy support
  networkPolicy: DefaultAllow
  # Optional: Enables logging for network policy events
  networkPolicyEventLogs: Enabled
  # Include other Node Class configurations as needed
```

2. 將節點類別組態套用至您的叢集：

```
kubectl apply -f nodeclass-network-policy.yaml
```

3. 確認節點類別已建立：

```
kubectl get nodeclass network-policy-enabled
```

4. 更新您的節點集區以使用此節點類別。如需詳細資訊，請參閱the section called “建立節點集區”。

節點使用此節點類別後，將能夠強制執行網路政策。您現在可以繼續建立和套用網路政策，以控制叢集內的流量。如需所有節點類別組態選項，請參閱 the section called “建立節點類別”。

步驟 3：建立和測試網路政策

EKS Auto Mode 叢集現在已設定為支援 Kubernetes 網路政策。您可以使用 來測試此項目the section called “星政策示範”。

標記 EKS Auto Mode 的子網路

如果您使用 EKS Auto Mode 的負載平衡功能，則需要將 AWS 標籤新增至 VPC 子網路。

背景介紹

這些標籤會識別與叢集相關聯的子網路，更重要的是子網路是公有或私有。

公有子網路可透過網際網路閘道直接存取網際網路。它們用於需要公開存取的資源，例如負載平衡器。

私有子網路沒有直接網際網路存取，並針對傳出流量使用 NAT 閘道。它們用於不需要公有 IPs 的內部資源，例如 EKS 節點。

若要進一步了解 NAT 閘道和網際網路閘道，請參閱《Amazon Virtual Private Cloud (VPC) 使用者指南》中的將您的 VPC 連線至其他網路。

需求

目前，用於 EKS Auto Mode 負載平衡的子網路需要有下列其中一個標籤。

公有子網路

公有子網路用於面向網際網路的負載平衡器。這些子網路必須具有下列標籤：

金鑰	值
`kubernetes.io/role/elb`	1 或 ``

私有子網路

私有子網路用於內部負載平衡器。這些子網路必須具有下列標籤：

金鑰	值
kubernetes.io/role/internal-elb	1 或 ``

程序

開始之前，請識別哪些子網路是公有（具有網際網路閘道存取）和哪些是私有（使用 NAT 閘道）。您需要許可才能修改 VPC 資源。

AWS Management Console

1. 開啟 Amazon VPC 主控台並導覽至子網路
2. 選取要標記的子網路
3. 選擇標籤索引標籤，然後選取新增標籤
4. 新增適當的標籤：
   - 對於公有子網路：Key=kubernetes.io/role/elb
   - 對於私有子網路：Key=kubernetes.io/role/internal-elb
5. 將值設定為 1或保持空白
6. 儲存並重複剩餘的子網路

AWS CLI

對於公有子網路：

```
aws ec2 create-tags \
    --resources subnet-ID \
    --tags Key=kubernetes.io/role/elb,Value=1
```

對於私有子網路：

```
aws ec2 create-tags \
    --resources subnet-ID \
    --tags Key=kubernetes.io/role/internal-elb,Value=1
```

將 取代為您實際subnet-ID的子網路 ID。

部署加速工作負載

本教學課程示範 Amazon EKS Auto Mode 如何簡化啟動加速工作負載。Amazon EKS Auto Mode 透過自動化關鍵基礎設施元件，立即提供運算、聯網、負載平衡、儲存以及身分存取和管理功能，簡化叢集本身以外的操作。

Amazon EKS Auto Mode 包含特定執行個體類型所需的驅動程式和裝置外掛程式，例如 NVIDIA 和 AWS Neuron 驅動程式。您不需要安裝或更新這些元件。

EKS Auto Mode 會自動管理這些加速器的驅動程式：

- AWS Trainium
- AWS 推論
- Amazon EC2 加速執行個體上的 NVIDIA GPUs

> **Note**
>
> EKS Auto Mode 包含適用於 Kubernetes 的 NVIDIA 裝置外掛程式。此外掛程式會自動執行，不會在您的叢集中顯示為協助程式集。

其他聯網支援：

- Elastic Fabric Adapter (EFA)

Amazon EKS Auto Mode 可消除加速器驅動程式和裝置外掛程式管理。

您也可以透過將叢集擴展到零，從節省成本中獲益。您可以設定 EKS Auto 模式，在沒有工作負載執行時終止執行個體。這適用於批次型推論工作負載。

以下提供如何使用 Amazon EKS Auto Mode 啟動加速工作負載的範例。

先決條件

- 已設定 Amazon EKS Auto 模式的 Kubernetes 叢集。
- 啟用 `default` `general-purpose`或 `system`受管節點集區時建立的 EKS 節點類別。

步驟 1：部署 GPU 工作負載

在此範例中，您將為需要 45GB GPU 記憶體的 NVIDIA 工作負載建立 NodePool。使用 EKS Auto Mode 時，您可以使用 Kubernetes 排程限制來定義執行個體需求。

若要部署 Amazon EKS Auto Mode NodePool和範例 workload，請檢閱下列 NodePool 和 Pod 定義，並儲存為 nodepool-gpu.yaml和 pod.yaml：

nodepool-gpu.yaml

```
apiVersion: karpenter.sh/v1
kind: NodePool
metadata:
  name: gpu
spec:
  disruption:
    budgets:
    - nodes: 10%
    consolidateAfter: 1h
    consolidationPolicy: WhenEmpty
  template:
    metadata: {}
    spec:
      nodeClassRef:
        group: eks.amazonaws.com
        kind: NodeClass
        name: default
      requirements:
        - key: "karpenter.sh/capacity-type"
          operator: In
          values: ["on-demand"]
        - key: "kubernetes.io/arch"
          operator: In
          values: ["amd64"]
        - key: "eks.amazonaws.com/instance-family"
          operator: In
          values:
          - g6e
          - g6
      taints:
        - key: nvidia.com/gpu
          effect: NoSchedule
      terminationGracePeriod: 24h0m0s
```

pod.yaml

```
apiVersion: v1
kind: Pod
metadata:
  name: nvidia-smi
spec:
  nodeSelector:
    eks.amazonaws.com/instance-gpu-name: l40s
    eks.amazonaws.com/compute-type: auto
  restartPolicy: OnFailure
  containers:
  - name: nvidia-smi
    image: public.ecr.aws/amazonlinux/amazonlinux:2023-minimal
    args:
    - "nvidia-smi"
    resources:
      requests:
        memory: "30Gi"
        cpu: "3500m"
        nvidia.com/gpu: 1
      limits:
        memory: "30Gi"
        nvidia.com/gpu: 1
  tolerations:
  - key: nvidia.com/gpu
    effect: NoSchedule
    operator: Exists
```

請注意，`eks.amazonaws.com/compute-type: auto`選取器需要將工作負載部署在 Amazon EKS Auto Mode 節點上。NodePool 也會設定一個污點，僅允許排程具有 Nvidia GPUs 容錯能力的 Pod。

將 NodePool 和工作負載套用至您的叢集。

```
kubectl apply -f nodepool-gpu.yaml
kubectl apply -f pod.yaml
```

您應該會看到下列輸出：

```
nodepool.karpenter.sh/gpu configured created
pod/nvidia-smi created
```

等待幾秒鐘，然後檢查叢集中的節點。您現在應該會在 Amazon EKS Auto Mode 叢集中看到佈建的新節點：

```
> kubectl get nodes

NAME        TYPE           CAPACITY    ZONE         NODE                   READY   AGE
gpu-dnknr   g6e.2xlarge    on-demand   us-west-2b   i-02315c7d7643cdee6   True    76s
```

步驟 2：驗證

您可以看到 Amazon EKS Auto Mode 啟動 g6e.2xlarge而不是 ，g6.2xlarge因為工作負載需要具有 l40s 的執行個體GPU，根據下列 Kubernetes 排程限制：

```
...
  nodeSelector:
    eks.amazonaws.com/instance-gpu-name: l40s
...
    requests:
        memory: "30Gi"
        cpu: "3500m"
        nvidia.com/gpu: 1
      limits:
        memory: "30Gi"
        nvidia.com/gpu: 1
```

現在，執行下列命令來查看容器日誌：

```
kubectl logs nvidia-smi
```

輸出範例：

```
+-----------------------------------------------------------------------------------------
+
| NVIDIA-SMI 535.230.02             Driver Version: 535.230.02   CUDA Version: 12.2
 |
|-----------------------------------------+----------------------
+----------------------+
| GPU  Name                 Persistence-M | Bus-Id        Disp.A | Volatile Uncorr. ECC
 |
| Fan  Temp   Perf          Pwr:Usage/Cap |         Memory-Usage | GPU-Util  Compute M.
 |
```

```
|                                         |                      |               MIG M.
 |
|=========================================+======================
+======================|
|   0  NVIDIA L40S                    On  | 00000000:30:00.0 Off |                    0
 |
| N/A   27C    P8              23W / 350W |      0MiB / 46068MiB |      0%      Default
 |
|                                         |                      |                  N/A
 |
+-----------------------------------------+----------------------
+----------------------+

+---------------------------------------------------------------------------------------
+
| Processes:
 |
|  GPU   GI   CI        PID   Type   Process name                          GPU Memory
 |
|        ID   ID                                                           Usage
 |
|
=========================================================================================|
|  No running processes found
 |
+---------------------------------------------------------------------------------------
+
```

您可以看到容器偵測到它正在具有 NVIDIA GPU 的執行個體上執行，而且您不需要安裝任何裝置驅動程式，因為這是由 Amazon EKS Auto Mode 管理。

步驟 3：清除

若要移除建立的所有物件，請使用 kubectl刪除範例部署和 NodePool，以便終止節點：

```
kubectl delete -f nodepool-gpu.yaml
kubectl delete -f pod.yaml
```

NodePools 參考範例

建立 NVIDIA NodePool

下列 NodePool 定義：

- 僅啟動 g6e和 g6 系列的執行個體
- 閒置 1 小時時合併節點
  - 的 1 小時值consolodateAfter支援尖峰工作負載並減少節點流失。您可以consolidateAfter根據您的工作負載需求進行調整。

具有 GPU 執行個體系列和整合的範例 NodePool

```
apiVersion: karpenter.sh/v1
kind: NodePool
metadata:
  name: gpu
spec:
  disruption:
    budgets:
    - nodes: 10%
    consolidateAfter: 1h
    consolidationPolicy: WhenEmpty
  template:
    metadata: {}
    spec:
      nodeClassRef:
        group: eks.amazonaws.com
        kind: NodeClass
        name: default
      requirements:
        - key: "karpenter.sh/capacity-type"
          operator: In
          values: ["on-demand"]
        - key: "kubernetes.io/arch"
          operator: In
          values: ["amd64"]
        - key: "eks.amazonaws.com/instance-family"
          operator: In
          values:
          - g6e
          - g6
      terminationGracePeriod: 24h0m0s
```

而不是設定eks.amazonaws.com/instance-gpu-name您可能eks.amazonaws.com/instance-family用來指定執行個體系列的 。如需影響排程檢閱的其他已知標籤，請參閱 the section called “EKS 自動模式支援的標籤”。

如果您有特定的儲存需求，您可以調整節點暫時性儲存 iops，sizethroughput並透過建立自己的 NodeClass 以在 NodePool 中參考。進一步了解可設定的 NodeClass 選項。

NodeClass 的範例儲存組態

```
apiVersion: eks.amazonaws.com/v1
kind: NodeClass
metadata:
  name: gpu
spec:
  ephemeralStorage:
    iops: 3000
    size: 80Gi
    throughput: 125
```

定義 AWS Trainium 和 AWS Inferentia NodePool

下列 NodePool 具有 eks.amazonaws.com/instance-category集合，其中僅啟動 Inferentia 和 Trainium 系列的執行個體：

```
        - key: "eks.amazonaws.com/instance-category"
          operator: In
          values:
            - inf
            - trn
```

使用 kubectl 除錯從 Kubernetes 節點產生 CIS 合規報告

本主題說明如何使用 kubectl debug命令產生 Amazon EKS 節點的 CIS （網際網路安全中心） 合規報告。命令可讓您暫時在 Kubernetes 節點上建立偵錯容器，並使用 apiclient工具執行 CIS 合規檢查。apiclient 工具是 Bottlerocket OS 的一部分，這是 EKS Auto Mode 節點所使用的作業系統。

先決條件

開始之前，請確定您已：

- 存取已kubectl設定 的 Amazon EKS 叢集 （版本必須至少為 v1.32.0；輸入 kubectl version 進行檢查）。
- 偵錯節點的適當 IAM 許可。
- 允許偵錯操作的有效設定檔 （例如 sysadmin)。

如需搭配 使用偵錯設定檔的詳細資訊kubectl，請參閱 Kubernetes 文件中的在套用設定檔時對 Pod 或節點進行偵錯。

程序

1. 決定您要執行報告的節點 AWS 執行個體 ID。使用下列命令列出叢集中的節點。執行個體 ID 位於名稱欄中，開頭為 i-：

```
kubectl get nodes
```

```
NAME                  STATUS   ROLES    AGE   VERSION
i-0ea0ba0f8ef9ad609   Ready    <none>   62s   v1.30.10-eks-1a9dacd
```

2. 執行下列命令，<instance-id>將 取代為您要查詢之節點的執行個體 ID：

```
kubectl debug node/<instance-id> -it --profile=sysadmin --image=public.ecr.aws/
amazonlinux/amazonlinux:2023 -- bash -c "yum install -q -y util-linux-core; nsenter -
t 1 -m apiclient report cis --level 1 --format text"
```

此命令的元件包括：

- kubectl debug node/<instance-id>— 在指定的 EC2 執行個體 ID 上建立偵錯工作階段。
- -it— 配置 TTY（命令列 shell），並保持 stdin 開啟以供互動式使用。
- --profile=sysadmin— 使用具有適當許可的指定kubectl設定檔。
- --image=public.ecr.aws/amazonlinux/amazonlinux:2023— 使用 amazonlinux:2023做為偵錯的容器映像。
- bash -c "…" — 在 bash shell 中執行下列命令：
  - yum install -q -y util-linux-core— 安靜地安裝必要的公用程式套件。
  - nsenter -t 1 -m— 執行 nsenter以輸入主機程序的命名空間 (PID 1)。
  - apiclient report cis --level 1 --format text— 使用文字輸出，在層級 1 執行 CIS 合規報告。

3. 檢閱報告文字輸出。

解譯輸出

命令會產生文字型報告，顯示各種 CIS 控制項的合規狀態。輸出包括：

- 個別 CIS IDs
- 每個控制項的描述
- 每個檢查的通過、失敗或略過狀態
- 說明任何合規問題的詳細資訊

以下是 Bottlerocket 執行個體上執行之報告輸出的範例：

```
Benchmark name:  CIS Bottlerocket Benchmark
Version:         v1.0.0
Reference:       https://www.cisecurity.org/benchmark/bottlerocket
Benchmark level: 1
Start time:      2025-04-11T01:40:39.055623436Z

[SKIP] 1.2.1     Ensure software update repositories are configured (Manual)
[PASS] 1.3.1     Ensure dm-verity is configured (Automatic)[PASS] 1.4.1     Ensure
 setuid programs do not create core dumps (Automatic)
[PASS] 1.4.2     Ensure address space layout randomization (ASLR) is enabled
 (Automatic)
[PASS] 1.4.3     Ensure unprivileged eBPF is disabled (Automatic)
[PASS] 1.5.1     Ensure SELinux is configured (Automatic)
[SKIP] 1.6       Ensure updates, patches, and additional security software are
 installed (Manual)
[PASS] 2.1.1.1   Ensure chrony is configured (Automatic)
[PASS] 3.2.5     Ensure broadcast ICMP requests are ignored (Automatic)
[PASS] 3.2.6     Ensure bogus ICMP responses are ignored (Automatic)
[PASS] 3.2.7     Ensure TCP SYN Cookies is enabled (Automatic)
[SKIP] 3.4.1.3   Ensure IPv4 outbound and established connections are configured
 (Manual)
[SKIP] 3.4.2.3   Ensure IPv6 outbound and established connections are configured
 (Manual)
[PASS] 4.1.1.1   Ensure journald is configured to write logs to persistent disk
 (Automatic)
[PASS] 4.1.2     Ensure permissions on journal files are configured (Automatic)

Passed:          11
Failed:          0
Skipped:         4
Total checks:    15
```

如需基準的相關資訊，請參閱網際網路安全中心 (CIS) 中的 Kubernetes 基準。

相關資源

- Bottlerocket 作業系統文件中的 Bottlerocket CIS 基準。
- Kubernetes 文件中的偵錯執行 Pod。
- 來自網際網路安全中心 (CIS) 的 Kubernetes 基準

使用 EKS Auto 模式的客戶受管 KMS 金鑰啟用 EBS 磁碟區加密

您可以使用客戶受管 KMS 金鑰加密 EKS Auto Mode 執行個體的暫時性根磁碟區。

Amazon EKS Auto Mode 使用服務連結角色，在管理 Kubernetes 叢集的加密 EBS 磁碟區 AWS 時，將許可委派給其他服務。本主題說明如何設定使用 EKS Auto Mode 為 Amazon EBS 加密指定客戶受管金鑰時所需的金鑰政策。

考量：

- EKS Auto Mode 不需要額外的授權，即可使用預設 AWS 受管金鑰來保護您帳戶中的加密磁碟區。
- 本主題涵蓋加密暫時性磁碟區，也就是 EC2 執行個體的根磁碟區。如需加密工作負載所用資料磁碟區的詳細資訊，請參閱 the section called “建立 StorageClass”。

概觀

當 EKS Auto Mode 啟動執行個體時，下列 AWS KMS 金鑰可用於 Amazon EBS 根磁碟區加密：

- AWS 受管金鑰 – Amazon EBS 在您帳戶中建立、擁有和管理的加密金鑰。這是新帳戶的預設加密金鑰。
- 客戶受管金鑰 – 您建立、擁有和管理的自訂加密金鑰。

> **Note**
>
> 金鑰必須是對稱的。Amazon EBS 不支援非對稱客戶受管金鑰。

步驟 1：設定金鑰政策

您的 KMS 金鑰必須具有金鑰政策，允許 EKS Auto Mode 使用客戶受管金鑰加密的 Amazon EBS 磁碟區啟動執行個體。

使用下列結構設定您的金鑰政策：

> **Note**
>
> 此政策僅包含 EKS Auto 模式的許可。如果其他身分需要使用金鑰或管理授予，金鑰政策可能需要額外的許可。

```
{
    "Version": "2012-10-17",
    "Id": "MyKeyPolicy",
    "Statement": [
        {
            "Sid": "Allow use of the key",
            "Effect": "Allow",
            "Principal": {
                "AWS": [
                    "arn:aws: iam::<account-id>:role/ClusterServiceRole"
                ]
            },
            "Action": [
                "kms:Encrypt",
                "kms:Decrypt",
                "kms:ReEncrypt*",
                "kms:GenerateDataKey*",
                "kms:DescribeKey"
            ],
            "Resource": "*"
        },
        {
            "Sid": "Allow attachment of persistent resources",
            "Effect": "Allow",
            "Principal": {
                "AWS": [
                    "arn:aws: iam::<account-id>:role/ClusterServiceRole"
                ]
            },
            "Action": [
                "kms:CreateGrant",
                "kms:ListGrants",
                "kms:RevokeGrant"
            ],
```

```
            "Resource": "*",
            "Condition": {
                "Bool": {
                    "kms:GrantIsForAWSResource": "true"
                }
            }
        }
    ]
}
```

請務必將 取代<account-id>為您實際 AWS 的帳戶 ID。

設定金鑰政策時：

- ClusterServiceRole 必須有必要的 IAM 許可，才能使用 KMS 金鑰進行加密操作
- kms:GrantIsForAWSResource 條件可確保只能為 AWS 服務建立授予

步驟 2：使用客戶受管金鑰設定 NodeClass

設定金鑰政策後，請在 EKS Auto Mode NodeClass 組態中參考 KMS 金鑰：

```
apiVersion: eks.amazonaws.com/v1
kind: NodeClass
metadata:
  name: my-node-class
spec:
  # Insert existing configuration

  ephemeralStorage:
    size: "80Gi"  # Range: 1-59000Gi or 1-64000G or 1-58Ti or 1-64T
    iops: 3000    # Range: 3000-16000
    throughput: 125  # Range: 125-1000

    # KMS key for encryption
    kmsKeyID: "arn:aws: kms:<region>:<account-id>:key/<key-id>"
```

將預留位置值取代為您的實際值：

- <region> 您的 AWS 區域
- <account-id> 使用 AWS 您的帳戶 ID
- <key-id> 使用您的 KMS 金鑰 ID

您可以使用下列任何格式指定 KMS 金鑰：

- KMS 金鑰 ID： `1a2b3c4d-5e6f-1a2b-3c4d-5e6f1a2b3c4d`
- KMS 金鑰 ARN： `arn:aws: kms:us-west-2:111122223333:key/1a2b3c4d-5e6f-1a2b-3c4d-5e6f1a2b3c4d`
- 金鑰別名名稱： `alias/eks-auto-mode-key`
- 金鑰別名 ARN： `arn:aws: kms:us-west-2:111122223333:alias/eks-auto-mode-key`

使用 kubectl 套用 NodeClass 組態：

```
kubectl apply -f nodeclass.yaml
```

相關資源

- 建立 Amazon EKS 的節點類別
- 在 AWS Key Management Service 開發人員指南中檢視詳細資訊
  - 金鑰政策中 AWS 服務的許可
  - 變更金鑰政策
  - AWS KMS 中的授予

更新 EKS Auto Mode 的組織控制

有些組織控制可防止 EKS Auto Mode 正常運作。若是如此，您必須更新這些控制項，以允許 EKS Auto Mode 擁有代表您管理 EC2 執行個體所需的許可。

EKS Auto Mode 使用服務角色來啟動傳回 EKS Auto Mode 節點的 EC2 執行個體。服務角色是在您的帳戶中建立的 IAM 角色，服務會擔任該角色來代表您執行動作。服務控制政策 SCPs) 一律適用於使用服務角色執行的動作。這可讓 SCP 抑制自動模式的操作。最常見的情況是使用 SCP 來限制可啟動的 Amazon Machine Image (AMIs)。若要允許 EKS Auto Mode 運作，請修改 SCP 以允許從 EKS Auto Mode 帳戶啟動 AMIs。

您也可以使用 EC2 允許 AMIs功能來限制其他帳戶中 AMIs的可見性。如果您使用此功能，則必須展開影像條件，在感興趣的區域中也包含 EKS Auto Mode AMI 帳戶。

封鎖所有 AMIs SCP 範例，EKS Auto Mode 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 EKS Auto Mode AMI 帳戶。

> **Note**
>
> 請務必不要使用`ec2:Owner`內容索引鍵。Amazon 擁有 EKS Auto Mode AMI 帳戶，此金鑰的值一律為 `amazon`。建構允許啟動 AMIs SCP，如果 `ec2:Owner amazon`將允許啟動任何 Amazon 擁有的 AMI，而不只是 EKS Auto Mode 的 AMI。*

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "DenyAMI",
      "Effect": "Deny",
      "Action": "ec2:RunInstances",
      "Resource": "arn:*:ec2:*::image/ami-*",
      "Condition": {
        "StringNotEquals": {
          "aws:ResourceAccount": [
            "767397842682",
            "992382739861"
          ]
        }
      }
    }
  ]
}
```

EKS Auto Mode AMI 帳戶

AWS 帳戶因區域主機 EKS Auto Mode 公有 AMIs 而異。

AWS 區域	帳戶
af-south-1	471112993317

ap-east-1	590183728416
ap-northeast-1	851725346105
ap-northeast-2	992382805010
ap-northeast-3	891377407544
ap-south-1	975049899075
ap-south-2	590183737426
ap-southeast-1	339712723301
ap-southeast-2	58264376476
ap-southeast-3	471112941769
ap-southeast-4	590183863144
ap-southeast-5	654654202513
ap-southeast-7	533267217478
ca-central-1	992382439851
ca-west-1	767397959864
eu-central-1	891376953411
eu-central-2	381492036002
eu-north-1	339712696471
eu-south-1	975049955519
eu-south-2	471112620929
eu-west-1	381492008532
eu-west-2	590184142468

eu-west-3	891376969258
il-central-1	590183797093
me-central-1	637423494195
me-south-1	905418070398
mx-central-1	211125506622
sa-east-1	339712709251
us-east-1	992382739861
us-east-2	975050179949
us-west-1	975050035094
us-west-2	767397842682

關聯公有 IP 地址

當 `ec2:RunInstances` 稱為執行個體啟動的 `AssociatePublicIpAddress` 欄位時，會自動由執行個體啟動所在的子網路類型決定。SCP 可用來強制將此值明確設定為 false，無論要啟動的子網路類型為何。在此情況下，NodeClass 欄位`spec.advancedNetworking.associatePublicIPAddress`也可以設定為 false，以滿足 SCP 的需求。

```
{
      "Sid": "DenyPublicEC2IPAddesses",
      "Effect": "Deny",
      "Action": "ec2:RunInstances",
      "Resource": "arn:aws:ec2:*:*:network-interface/*",
      "Condition": {
          "BoolIfExists": {
              "ec2:AssociatePublicIpAddress": "true"
          }
      }
  }
```

使用 EKS Auto 模式控制工作負載部署至 EC2 隨需容量預留

EC2 隨需容量保留 ODCRs) 可讓您為特定可用區域中的 Amazon EC2 執行個體保留任何持續時間的運算容量。使用 EKS Auto Mode 時，您可能想要控制 Kubernetes 工作負載是否部署在這些預留執行個體上，以最大限度地利用預先購買的容量，或確保關鍵工作負載能夠存取保證的資源。

根據預設，EKS Auto Mode 會自動在開啟ODCRs 中啟動。不過，透過在 NodeClass `capacityReservationSelectorTerms`上設定 ，您可以明確控制工作負載使用的 ODCRs。使用設定的 ODCRs節點將具有並優先於隨需`karpenter.sh/capacity-type: reserved`和 Spot。啟用此功能後，EKS Auto Mode 將不再自動使用開啟ODCRs - 它們必須由 NodeClass 明確選取，可讓您精確控制整個叢集的容量保留用量。

> **Warning**
>
> 如果您在叢集中的 NodeClass `capacityReservationSelectorTerms`上設定 ，EKS Auto Mode 將不再自動為叢集中的任何 NodeClass 使用開啟ODCRs。

範例 NodeClass

```
apiVersion: eks.amazonaws.com/v1
kind: NodeClass
spec:
  # Optional: Selects upon on-demand capacity reservations and capacity blocks
  # for EKS Auto Mode to prioritize.
  capacityReservationSelectorTerms:
    - id: cr-56fac701cc1951b03
    # Alternative Approaches
    - tags:
        app: "my-app"
      # Optional owning account ID filter
      owner: "012345678901"
```

此範例 NodeClass 示範兩種選取 ODCRs 的方法。第一種方法會依其 ID () 直接參考特定 ODCR`cr-56fac701cc1951b03`。第二個方法使用以標籤為基礎的選擇，以具有標籤 ODCRs 為目標`Name: "targeted-odcr"`。您也可以選擇性地依擁有保留 AWS 的帳戶進行篩選，這在跨帳戶案例或處理共用容量保留時特別有用。

了解 EKS Auto Mode 的運作方式

使用本章來了解 Amazon EKS Auto Mode 叢集的元件如何運作。

主題

- 了解 Amazon EKS Auto Mode 受管執行個體
- 了解 EKS Auto Mode 中的身分和存取
- 了解 EKS Auto 模式的 VPC 網路和負載平衡

了解 Amazon EKS Auto Mode 受管執行個體

本主題說明 Amazon EKS Auto Mode 如何管理 EKS 叢集中的 Amazon EC2 執行個體。當您啟用 EKS Auto Mode 時，叢集的運算資源會自動由 EKS 佈建和管理，變更您與做為叢集中節點的 EC2 執行個體互動的方式。

了解 Amazon EKS Auto Mode 如何管理執行個體對於規劃工作負載部署策略和操作程序至關重要。與傳統 EC2 執行個體或受管節點群組不同，這些執行個體遵循不同的生命週期模型，其中 EKS 負責許多操作層面，同時限制特定類型的存取和自訂。

Amazon EKS Auto Mode 會自動執行建立新 EC2 執行個體的例行任務，並將其做為節點連接至 EKS 叢集。EKS Auto Mode 會偵測工作負載何時無法適應現有節點，並建立新的 EC2 執行個體。

Amazon EKS Auto Mode 負責建立、刪除和修補 EC2 執行個體。您要負責執行個體上部署的容器和 Pod。

EKS Auto Mode 建立的 EC2 執行個體與其他 EC2 執行個體不同，它們是受管執行個體。這些受管執行個體由 EKS 擁有，且受到更多限制。您無法在 EKS Auto Mode 管理的執行個體上直接存取或安裝軟體。

AWS 建議執行 EKS Auto Mode 或自我管理 Karpenter。您可以在遷移期間或在進階組態中安裝 。如果兩者都已安裝，請設定節點集區，讓工作負載與 Karpenter 或 EKS Auto 模式相關聯。

如需詳細資訊，請參閱《Amazon EC2 使用者指南》中的 Amazon EC2 受管執行個體。 Amazon EC2

比較表

標準 EC2 執行個體	EKS Auto Mode 受管執行個體
您負責修補和更新執行個體。	AWS 會自動修補和更新執行個體。

標準 EC2 執行個體	EKS Auto Mode 受管執行個體
EKS 不負責執行個體上的軟體。	EKS 負責執行個體上的特定軟體，例如 `kubelet`、容器執行時間和作業系統。
您可以使用 EC2 API 刪除 EC2 執行個體。	EKS 會決定您帳戶中部署的執行個體數量。如果您刪除工作負載，EKS 將減少您帳戶中的執行個體數量。
您可以使用 SSH 存取 EC2 執行個體。	您可以將 Pod 和容器部署到受管執行個體。
您可以判斷作業系統和映像 (AMI)。	AWS 會決定作業系統和映像。
您可以部署依賴 Windows 或 Ubuntu 功能的工作負載。	您可以根據 Linux 部署容器，但不需要特定的作業系統相依性。
您可以決定要啟動的執行個體類型和系列。	AWS 決定要啟動的執行個體類型和系列。您可以使用節點集區來限制 EKS Auto Mode 選取的執行個體類型。

下列功能適用於受管執行個體和標準 EC2 執行個體：

- 您可以在 AWS 主控台中檢視執行個體。
- 您可以使用執行個體儲存體做為工作負載的暫時性儲存體。

AMI 支援

使用 EKS Auto Mode， AWS 決定運算節點所使用的映像 (AMI)。 AWS 會監控推出新的 EKS Auto Mode AMI 版本。如果您遇到與 AMI 版本相關的工作負載問題，請建立支援案例。如需詳細資訊，請參閱《支援使用者指南》中的建立支援案例和案例管理。 AWS

一般而言，EKS 每週都會發行新的 AMI，其中包含 CVE 和安全性修正。

EKS Auto Mode 支援的執行個體參考

EKS Auto Mode 只會建立支援類型且符合最小大小需求的執行個體。

EKS Auto Mode 支援下列執行個體類型：

系列	執行個體類型
運算最佳化 (C)	c8g、c7a、c7g、c7gn、c7gd、c7i、c7i-flex、c6a、c6g、c6i、c6gn、c6id、c6in、c6gd、c5、c5a、c5d、c5ad、c5n、c4
一般用途 (M)	m8g、m7i、m7a、m7g、m7gd、m7i-flex、m6a、m6i、m6in、m6g、m6idn、m6id、m6gd、m5、m5a、m5ad、m5n、m5dn、m5dn、m5d、m5zn、m4
記憶體最佳化 (R)	r8g、r7a、r7iz、r7gd、r7i、r7g、r6a、r6i、r6id、r6in、r6idn、r6g、r6gd、r5、r5n、r5a、r5dn、r5b、r5ad、r5d、r4
爆量 (T)	t4g、t3、t3a、t2
高記憶體 (Z/X)	z1d、x8g、x2gd
儲存最佳化 (I/D)	i8g、i7ie、i4g、i4i、i3、i3en、is4gen、d3、d3en、im4gn
加速運算 (P/G/Inf/Trn)	p5、p4d、p4de、p3、p3dn、gr6、g6、g6e、g5g、g5、g4dn、inf2、inf1、trn1、trn1n
高效能運算 (X2)	x2iezn、x2iedn、x2idn

此外，EKS Auto Mode 只會建立符合下列要求的 EC2 執行個體：

- 超過 1 個 CPU
- 執行個體大小不是奈米、微型或小型

如需詳細資訊，請參閱 Amazon EC2 執行個體類型命名慣例。

執行個體中繼資料服務

- EKS Auto Mode 預設會強制執行 IMDSv2，跳轉限制為 1，並遵循 AWS 安全最佳實務。
- 此預設組態無法在自動模式中修改。
- 對於通常需要 IMDS 存取的附加元件，請在安裝期間提供參數（例如 AWS 區域），以避免 IMDS 查詢。如需詳細資訊，請參閱the section called “您可以自訂的欄位”。

- 如果 Pod 在自動模式下執行時絕對需要 IMDS 存取，則必須將 Pod 設定為使用 執行`hostNetwork: true`。這可讓 Pod 直接存取執行個體中繼資料服務。
- 授予 Pod 存取執行個體中繼資料時，請考慮安全性影響。

如需 Amazon EC2 執行個體中繼資料服務 (IMDS) 的詳細資訊，請參閱《Amazon EC2 使用者指南》中的設定執行個體中繼資料服務選項。

考量事項

- 如果 NodeClass 中設定的暫時性儲存體小於執行個體的 NVMe 本機儲存體，EKS Auto Mode 會透過自動採取下列動作來消除手動組態的需求：
  - 使用較小的 (20 GiB) Amazon EBS 資料磁碟區來降低成本。
  - 格式化和設定 NVMe 本機儲存體以供暫時性資料使用。這包括如果有多個 NVMe 磁碟機，請設定 RAID 0 陣列。
- Amazon EKS Auto Mode 不支援 AWS Fault Injection Service。如需詳細資訊，請參閱 AWS《Resilience Hub 使用者指南》中的管理 Fault Injection Service 實驗。
- 您不需要在 EKS Auto Mode 節點`Neuron Device Plugin`上安裝 。

  如果您的叢集中有其他類型的節點，您需要將 Neuron 裝置外掛程式設定為不在自動模式節點上執行。如需詳細資訊，請參閱the section called “控制部署”。

了解 EKS Auto Mode 中的身分和存取

本主題說明使用 EKS Auto Mode 所需的 Identity and Access Management (IAM) 角色和許可。EKS Auto Mode 使用兩個主要 IAM 角色：叢集 IAM 角色和節點 IAM 角色。這些角色可與 EKS Pod Identity 和 EKS 存取項目搭配使用，為您的 EKS 叢集提供全面的存取管理。

當您設定 EKS Auto Mode 時，您將需要設定這些具有特定許可的 IAM 角色，以允許 AWS 服務與您的叢集資源互動。這包括管理運算資源、儲存磁碟區、負載平衡器和聯網元件的許可。了解這些角色組態對於適當的叢集操作和安全性至關重要。

在 EKS Auto 模式中， AWS IAM 角色會透過 EKS 存取項目自動映射至 Kubernetes 許可，無需手動設定 `aws-auth` ConfigMaps 或自訂繫結。當您建立新的自動模式叢集時，EKS 會使用存取項目自動建立對應的 Kubernetes 許可，確保 AWS 服務和叢集元件在 AWS 和 Kubernetes 授權系統中都具有適當的存取層級。這種自動化整合可減少組態複雜性，並有助於防止管理 EKS 叢集時經常發生的許可相關問題。

叢集 IAM 角色

叢集 IAM 角色是 Amazon EKS 用來管理 Kubernetes 叢集許可的 AWS Identity and Access Management (IAM) 角色。此角色會授予 Amazon EKS 必要的許可，以代表您叢集與其他 AWS 服務互動，並使用 EKS 存取項目自動設定 Kubernetes 許可。

- 您必須將 AWS IAM 政策連接至此角色。
- EKS Auto Mode 會使用 EKS 存取項目自動將 Kubernetes 許可連接到此角色。
- 使用 EKS Auto 模式時， AWS 建議為每個 AWS 帳戶建立單一叢集 IAM 角色。
- AWS 建議將此角色命名為 `AmazonEKSAutoClusterRole`。
- 此角色需要多個 AWS 服務的許可，才能管理 資源，包括 EBS 磁碟區、Elastic Load Balancer 和 EC2 執行個體。
- 此角色的建議組態包含多個 AWS 受管 IAM 政策，與 EKS Auto Mode 的不同功能相關。
  - `AmazonEKSComputePolicy`
  - `AmazonEKSBlockStoragePolicy`
  - `AmazonEKSLoadBalancingPolicy`
  - `AmazonEKSNetworkingPolicy`
  - `AmazonEKSClusterPolicy`

如需叢集 IAM 角色和 AWS 受管 IAM 政策的詳細資訊，請參閱：

- the section called “ AWS 受管政策”
- the section called “叢集 IAM 角色”

如需 Kubernetes 存取的詳細資訊，請參閱：

- the section called “檢閱存取政策”

節點 IAM 角色

節點 IAM 角色是 Amazon EKS 用來管理 Kubernetes 叢集中工作者節點許可的 AWS Identity and Access Management (IAM) 角色。此角色會授予以 Kubernetes 節點身分執行的 EC2 執行個體與 AWS 服務和資源互動的必要許可，並使用 EKS 存取項目自動設定 Kubernetes RBAC 許可。

- 您必須將 AWS IAM 政策連接至此角色。

- EKS Auto Mode 會使用 EKS 存取項目自動將 Kubernetes RBAC 許可連接到此角色。
- AWS 建議將此角色命名為 `AmazonEKSAutoNodeRole`。
- 使用 EKS Auto Mode， AWS 建議為每個 AWS 帳戶建立單一節點 IAM 角色。
- 此角色的許可有限。關鍵許可包括擔任 Pod 身分角色，以及從 ECR 提取映像。
- AWS 建議下列 AWS 受管 IAM 政策：
  - `AmazonEKSWorkerNodeMinimalPolicy`
  - `AmazonEC2ContainerRegistryPullOnly`

如需叢集 IAM 角色和 AWS 受管 IAM 政策的詳細資訊，請參閱：

- the section called “ AWS 受管政策”
- the section called “節點 IAM 角色”

如需 Kubernetes 存取的詳細資訊，請參閱：

- the section called “檢閱存取政策”

服務連結角色

Amazon EKS 針對特定操作使用服務連結角色 (SLR)。服務連結角色是直接連結至 Amazon EKS 的一種特殊 IAM 角色類型。服務連結角色是由 Amazon EKS 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。

AWS 會自動建立和設定 SLR。您必須先刪除 SLR 的相關資源，才能刪除該 SLR。這可保護您的 Amazon EKS 資源，因為您不會不小心移除存取資源的許可。

SLR 政策授予 Amazon EKS 許可，以觀察和刪除核心基礎設施元件：EC2 資源 （執行個體、網路介面、安全群組）、ELB 資源 （負載平衡器、目標群組）、CloudWatch 功能 （記錄和指標），以及具有 "eks" 字首的 IAM 角色。它還透過 VPC/託管區域關聯啟用私有端點聯網，並包含 EventBridge 監控和清除 EKS 標記資源的許可。

如需詳細資訊，請參閱：

- the section called “AWS 受管政策：AmazonEKSServiceRolePolicy”
- the section called “Amazon EKS 的服務連結角色許可”

EKS Auto 資源的自訂 AWS 標籤

根據預設，與 EKS Auto Mode 相關的受管政策不允許將使用者定義的標籤套用至 Auto Mode 佈建 AWS 的資源。如果您想要將使用者定義的標籤套用至 AWS 資源，則必須將其他許可連接到叢集 IAM 角色，並具有足夠的許可來建立和修改 AWS 資源上的標籤。以下是允許不受限制標記存取的政策範例：

檢視自訂標籤政策範例

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Compute",
            "Effect": "Allow",
            "Action": [
                "ec2:CreateFleet",
                "ec2:RunInstances",
                "ec2:CreateLaunchTemplate"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:RequestTag/eks:eks-cluster-name": "${aws:PrincipalTag/eks:eks-
cluster-name}"
                },
                "StringLike": {
                    "aws:RequestTag/eks:kubernetes-node-class-name": "*",
                    "aws:RequestTag/eks:kubernetes-node-pool-name": "*"
                }
            }
        },
        {
            "Sid": "Storage",
            "Effect": "Allow",
            "Action": [
                "ec2:CreateVolume",
                "ec2:CreateSnapshot"
            ],
            "Resource": [
                "arn:aws: ec2:*:*:volume/*",
                "arn:aws: ec2:*:*:snapshot/*"
            ],
```

```
            "Condition": {
                "StringEquals": {
                    "aws:RequestTag/eks:eks-cluster-name": "${aws:PrincipalTag/eks:eks-
cluster-name}"
                }
            }
        },
        {
            "Sid": "Networking",
            "Effect": "Allow",
            "Action": "ec2:CreateNetworkInterface",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:RequestTag/eks:eks-cluster-name": "${aws:PrincipalTag/eks:eks-
cluster-name}"
                },
                "StringLike": {
                    "aws:RequestTag/eks:kubernetes-cni-node-name": "*"
                }
            }
        },
        {
            "Sid": "LoadBalancer",
            "Effect": "Allow",
            "Action": [
                "elasticloadbalancing:CreateLoadBalancer",
                "elasticloadbalancing:CreateTargetGroup",
                "elasticloadbalancing:CreateListener",
                "elasticloadbalancing:CreateRule",
                "ec2:CreateSecurityGroup"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:RequestTag/eks:eks-cluster-name": "${aws:PrincipalTag/eks:eks-
cluster-name}"
                }
            }
        },
        {
            "Sid": "ShieldProtection",
            "Effect": "Allow",
            "Action": [
```

```
                "shield:CreateProtection"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:RequestTag/eks:eks-cluster-name": "${aws:PrincipalTag/eks:eks-
cluster-name}"
                }
            }
        },
        {
            "Sid": "ShieldTagResource",
            "Effect": "Allow",
            "Action": [
                "shield:TagResource"
            ],
            "Resource": "arn:aws: shield::*:protection/*",
            "Condition": {
                "StringEquals": {
                    "aws:RequestTag/eks:eks-cluster-name": "${aws:PrincipalTag/eks:eks-
cluster-name}"
                }
            }
        }
    ]
}
```

存取政策參考

如需 EKS Auto Mode 所用 Kubernetes 許可的詳細資訊，請參閱 the section called “檢閱存取政策”。

了解 EKS Auto 模式的 VPC 網路和負載平衡

本主題說明如何在 EKS Auto 模式中設定虛擬私有雲端 (VPC) 網路和負載平衡功能。雖然 EKS Auto Mode 會自動管理大多數聯網元件，您仍然可以透過`NodeClass`資源和負載平衡器註釋自訂叢集聯網組態的某些層面。

當您使用 EKS Auto Mode 時， 會 AWS 管理叢集的 VPC 容器網路界面 (CNI) 組態和負載平衡器佈建。您可以透過定義`NodeClass`物件並將特定註釋套用至服務和輸入資源來影響聯網行為，同時維護 EKS Auto Mode 提供的自動化操作模型。

網路功能

EKS Auto Mode 具有新的聯網功能，可處理節點和 Pod 網路。您可以透過建立 `NodeClass` Kubernetes 物件進行設定。

先前 AWS VPC CNI 的組態選項不適用於 EKS Auto 模式。

使用 設定聯網 **`NodeClass`**

EKS Auto Mode 中的`NodeClass`資源可讓您自訂聯網功能的某些層面。透過 `NodeClass`，您可以指定安全群組選擇、控制跨 VPC 子網路的節點放置、設定 SNAT 政策、設定網路政策，以及啟用網路事件記錄。此方法會維護 EKS Auto Mode 的自動化操作模型，同時提供網路自訂的彈性。

您可以使用 `NodeClass`來：

- 選取節點的安全群組
- 控制節點在 VPC 子網路上的放置方式
- 將節點 SNAT 政策設定為 `random`或 `disabled`
- 啟用 Kubernetes 網路政策，包括：
  - 將網路政策設定為預設拒絕或預設允許
  - 啟用檔案的網路事件記錄。
- 透過將 Pod 連接到不同的子網路，將 Pod 流量與節點流量隔離。

了解如何建立 Amazon EKS NodeClass。

考量事項

EKS Auto Mode 支援：

- EKS 網路政策。
- Kubernetes Pod 的 `HostPort`和 `HostNetwork`選項。
- 公有或私有子網路中的節點和 Pod。
- 在節點上快取 DNS 查詢。

EKS Auto Mode 不支援：

- 每個 Pod 的安全群組 (SGPP)。

- 中的自訂聯網`ENIConfig`。您可以將 Pod 放在多個子網路中，或使用 將其與節點流量完全隔離the section called “Pod 的子網路選擇”。
- 暖 IP、暖字首和暖 ENI 組態。
- IP 目標組態下限。
- 啟用或停用字首委派。
- 開放原始碼 AWS VPC CNI 支援的其他組態。
- 網路政策組態，例如連接計時器自訂 （ 預設為 300 秒 ）。
- 將網路事件日誌匯出至 CloudWatch。

網路資源管理

EKS Auto Mode 透過監控網路組態的 NodeClass 資源來處理字首、IP 定址和網路介面管理。服務會自動執行數個金鑰操作：

字首委派

EKS Auto Mode 會將 /28 IPv4 字首佈建至節點的主要網路界面，並維護預先定義的暖集區，以根據排程的 Pod 數量進行擴展。必要時，它會佈建次要網路介面，其具有與節點子網路中主要介面相同的安全群組。如果子網路中不再提供字首，服務會回到次要 IPv4 地址。

冷卻時間管理

服務會針對不再使用的字首或次要 IPv4 地址實作冷卻集區。在冷卻時間到期後，這些資源會釋出回 VPC。不過，如果 Pod 在冷卻期間重複使用這些資源，則會從冷卻集區還原這些資源。

IPv6 支援

對於 IPv6 叢集，EKS Auto Mode 會在主要網路界面上為每個節點佈建 /80 IPv6 字首。

該服務也確保對所有網路介面進行適當的管理和垃圾收集。

負載平衡

您可以在服務和輸入資源上使用註釋來設定 EKS Auto Mode 佈建的 AWS Elastic Load Balancer。

如需詳細資訊，請參閱 the section called “建立輸入類別” 或 the section called “建立服務” 。

使用 EKS Auto Mode 進行負載平衡的考量

- 預設目標模式是 IP 模式，而不是執行個體模式。

- EKS Auto Mode 僅支援 Network Load Balancer 的安全群組模式。
- AWS 不支援將負載平衡器從自我管理 AWS 負載平衡器控制器遷移至 EKS Auto Mode 管理。
- 不支援`TargetGroupBinding`規格 中的 `networking.ingress.ipBlock` 欄位。
- 如果您的工作者節點使用自訂安全群組 `eks-cluster-sg-` （而非命名模式），您的叢集角色需要額外的 IAM 許可。預設 EKS 受管政策僅允許 EKS 修改名為 的安全群組`eks-cluster-sg-`。如果沒有修改自訂安全群組的許可，EKS 無法新增允許 ALB/NLB 流量到達 Pod 的必要輸入規則。

EKS Auto 模式故障診斷

使用 EKS Auto Mode， 會 AWS 對您 AWS 帳戶中的 EC2 執行個體承擔更多責任。EKS 負責節點上的容器執行期、節點上的作業系統，以及特定控制器。這包括區塊儲存控制器、負載平衡控制器和運算控制器。

您必須使用 AWS 和 Kubernetes APIs 對節點進行故障診斷。您可以：

- 使用 Kubernetes `NodeDiagnostic` 資源來擷取節點日誌，方法是使用 the section called “節點監控代理程式”。如需更多步驟，請參閱 the section called “取得節點日誌”。
- 使用 AWS EC2 CLI 命令從節點`get-console-output`擷取主控台輸出。如需更多步驟，請參閱 the section called “使用 EC2 CLI 從 AWS EC2 受管執行個體取得主控台輸出”。
- 使用 Kubernetes 除錯容器來擷取節點日誌。如需更多步驟，請參閱 the section called “使用偵錯容器和 CLI `kubectl` 取得節點日誌”。

> **Note**
>
> EKS Auto Mode 使用 EC2 受管執行個體。您無法直接存取 EC2 受管執行個體，包括 SSH。

您可能會遇到下列問題，這些問題具有 EKS Auto Mode 元件特定的解決方案：

- Pod 停滯在 `Pending` 狀態，但未排程到自動模式節點。如需解決方案，請參閱 the section called “故障診斷無法排程至自動模式節點的 Pod”。
- 未將叢集加入為 Kubernetes 節點的 EC2 受管執行個體。如需解決方案，請參閱 the section called “對未加入叢集的節點進行故障診斷”。
- `Services` 使用 EKS Auto Mode 中包含之控制器的 `NodePools`、 `PersistentVolumes`和 的錯誤和問題。如需解決方案，請參閱 the section called “對自動模式下包含的控制器進行故障診斷”。

- 增強的 Pod 安全性可防止跨 Pod 共用磁碟區。如需解決方案，請參閱 the section called “跨 Pod 共用磁碟區”。

您可以使用下列方法來疑難排解 EKS Auto Mode 元件：

- the section called “使用 EC2 CLI 從 AWS EC2 受管執行個體取得主控台輸出”
- the section called “使用偵錯容器和 CLI kubectl 取得節點日誌”
- the section called “在 AWS 主控台中檢視與 EKS Auto Mode 相關聯的資源”
- the section called “檢視您 AWS 帳戶中的 IAM 錯誤”
- the section called “偵測 的節點連線問題 VPC Reachability Analyzer”

節點監控代理程式

EKS Auto Mode 包含 Amazon EKS 節點監控代理程式。您可以使用此代理程式來檢視節點的疑難排解和偵錯資訊。節點監控代理程式會發佈 Kubernetes `events`和節點 `conditions`。如需詳細資訊，請參閱the section called “節點運作狀態”。

使用 EC2 CLI 從 AWS EC2 受管執行個體取得主控台輸出

此程序有助於疑難排解開機時間或核心層級的問題。

首先，您需要判斷與工作負載相關聯之執行個體的 EC2 執行個體 ID。其次，使用 AWS CLI 擷取主控台輸出。

1. 確認您`kubectl`已安裝並連線至您的叢集
2. （選用）使用 Kubernetes 部署的名稱來列出相關聯的 Pod。

```
kubectl get pods -l app=<deployment-name>
```

3. 使用 Kubernetes Pod 的名稱來判斷相關聯節點的 EC2 執行個體 ID。

```
kubectl get pod <pod-name> -o wide
```

4. 使用 EC2 執行個體 ID 擷取主控台輸出。

```
aws ec2 get-console-output --instance-id <instance id> --latest --output text
```

使用偵錯容器和 CLI **kubectl** 取得節點日誌

從 EKS Auto Mode 節點擷取日誌的建議方法是使用 NodeDiagnostic 資源。如需這些步驟，請參閱 the section called “取得節點日誌”。

不過，您可以使用 kubectl debug node命令，從執行個體即時串流日誌。此命令會在您要偵錯的節點上啟動新的 Pod，然後以互動方式使用。

1. 啟動偵錯容器。下列命令使用 i-01234567890123456 做為節點的執行個體 ID，-it配置 tty並連接 stdin以供互動式使用，並使用 kubeconfig 檔案的 sysadmin設定檔。

   ```
   kubectl debug node/i-01234567890123456 -it --profile=sysadmin --image=public.ecr.aws/
   amazonlinux/amazonlinux:2023
   ```

   範例輸出如下。

   ```
   Creating debugging pod node-debugger-i-01234567890123456-nxb9c with container
    debugger on node i-01234567890123456.
   If you don't see a command prompt, try pressing enter.
   bash-5.2#
   ```

2. 您現在可以從 shell 安裝 util-linux-core 來提供 nsenter命令。使用 在主機上nsenter輸入 PID 1 (init) 的掛載命名空間，然後執行 journalctl命令從 串流日誌kubelet：

   ```
   yum install -y util-linux-core
   nsenter -t 1 -m journalctl -f -u kubelet
   ```

為了安全起見，Amazon Linux 容器映像預設不會安裝許多二進位檔。您可以使用 yum whatprovides命令來識別必須安裝的套件，以提供指定的二進位檔。

```
yum whatprovides ps
```

```
Last metadata expiration check: 0:03:36 ago on Thu Jan 16 14:49:17 2025.
procps-ng-3.3.17-1.amzn2023.0.2.x86_64 : System and process monitoring utilities
Repo        : @System
Matched from:
Filename    : /usr/bin/ps
Provide    : /bin/ps
```

```
procps-ng-3.3.17-1.amzn2023.0.2.x86_64 : System and process monitoring utilities
Repo        : amazonlinux
Matched from:
Filename    : /usr/bin/ps
Provide    : /bin/ps
```

在 AWS 主控台中檢視與 EKS Auto Mode 相關聯的資源

您可以使用 AWS 主控台來檢視與 EKS Auto Mode 叢集相關聯的資源狀態。

- EBS 磁碟區
  - 搜尋標籤索引鍵來檢視 EKS 自動模式磁碟區 `eks:eks-cluster-name`
- 負載平衡器
  - 搜尋標籤索引鍵來檢視 EKS Auto Mode 負載平衡器 `eks:eks-cluster-name`
- EC2 執行個體
  - 搜尋標籤索引鍵來檢視 EKS Auto Mode 執行個體 `eks:eks-cluster-name`

檢視您 AWS 帳戶中的 IAM 錯誤

1. 導覽至 CloudTrail 主控台
2. 從左側導覽窗格中選取「事件歷史記錄」
3. 套用錯誤碼篩選條件：
   - AccessDenied
   - UnauthorizedOperation
   - InvalidClientTokenId

尋找與您的 EKS 叢集相關的錯誤。使用錯誤訊息來更新您的 EKS 存取項目、叢集 IAM 角色或節點 IAM 角色。您可能需要將新的政策連接至具有 EKS Auto Mode 許可的這些角色。

故障診斷無法排程至自動模式節點的 Pod

如果 Pod 保持 `Pending` 狀態且未排程到自動模式節點，請確認您的 Pod 或部署資訊清單是否具有 `nodeSelector`。如果`nodeSelector`存在 ，請確保在 EKS Auto Mode 建立的節點上使用 `eks.amazonaws.com/compute-type: auto`來排程它。如需 EKS Auto Mode 所用節點標籤的詳細資訊，請參閱 the section called “控制部署”。

對未加入叢集的節點進行故障診斷

EKS Auto Mode 會自動設定具有正確資訊的新 EC2 執行個體來加入叢集，包括叢集端點和叢集憑證授權單位 (CA)。不過，這些執行個體仍然無法將 EKS 叢集加入為節點。執行下列命令來識別未加入叢集的執行個體：

1. 執行 `kubectl get nodeclaim` 以檢查 `NodeClaims` 是否為 `Ready = False`。

```
kubectl get nodeclaim
```

2. 在狀態下執行`kubectl describe nodeclaim <node_claim>`並查看，以尋找阻止節點加入叢集的任何問題。

```
kubectl describe nodeclaim <node_claim>
```

常見錯誤訊息：

`Error getting launch template configs`

如果您使用預設叢集 IAM 角色許可在 中設定自訂標籤`NodeClass`，則可能會收到此錯誤。請參閱 the section called “身分和存取”。

`Error creating fleet`

從 `RunInstances` EC2 API 呼叫 時，可能會有一些授權問題。Check AWS CloudTrail 是否有錯誤，請參閱 the section called “自動模式叢集 IAM 角色” 以取得所需的 IAM 許可。

偵測 的節點連線問題 `VPC Reachability Analyzer`

> **Note**
>
> 每個執行 VPC Reachability Analyzer 的分析都會向您收取費用。如需定價詳細資訊，請參閱 Amazon VPC 定價。

執行個體未加入叢集的一個原因是網路連線問題，導致執行個體無法連線到 API 伺服器。若要診斷此問題，您可以使用 VPC Reachability Analyzer 來分析無法加入叢集和 API 伺服器之節點之間的連線。您將需要兩個資訊：

- 無法加入叢集之節點的執行個體 ID
- Kubernetes API 伺服器端點的 IP 地址

若要取得執行個體 ID，您需要在叢集上建立工作負載，讓 EKS Auto 模式啟動 EC2 執行個體。這也會在您的叢集中建立具有執行個體 ID 的NodeClaim物件。執行 `kubectl get nodeclaim -o yaml`以列印叢集NodeClaims中的所有 。每個 都NodeClaim包含執行個體 ID 做為欄位，並在 providerID 中再次包含：

```
kubectl get nodeclaim -o yaml
```

範例輸出如下。

```
    nodeName: i-01234567890123456
    providerID: aws:///us-west-2a/i-01234567890123456
```

您可以執行 來判斷 Kubernetes API 伺服器端點`kubectl get endpoint kubernetes -o yaml`。地址位於地址欄位中：

```
kubectl get endpoints kubernetes -o yaml
```

範例輸出如下。

```
apiVersion: v1
kind: Endpoints
metadata:
  name: kubernetes
  namespace: default
subsets:
- addresses:
  - ip: 10.0.143.233
  - ip: 10.0.152.17
  ports:
  - name: https
    port: 443
    protocol: TCP
```

透過這兩項資訊，您可以執行 分析。首先導覽至 中的 VPC Reachability Analyzer AWS Management Console。

1. 按一下「建立和分析路徑」
2. 提供分析的名稱（例如「節點聯結失敗」）
3. 針對「來源類型」，選取「執行個體」
4. 輸入失敗節點的執行個體 ID 做為「來源」
5. 對於「路徑目的地」，選取「IP 地址」
6. 輸入 API 伺服器的其中一個 IP 地址做為「目的地地址」
7. 展開「額外封包標頭組態區段」
8. 輸入 443 的「目的地連接埠」
9. 如果尚未選取，請選取「通訊協定」做為 TCP
10. 按一下「建立和分析路徑」
11. 完成分析可能需要幾分鐘的時間。如果分析結果指出連線失敗，則會指出故障在網路路徑中的位置，讓您可以解決問題。

跨 Pod 共用磁碟區

EKS 自動模式節點是以強制執行模式使用 SELinux 設定，可在相同節點上執行的 Pod 之間提供更多隔離。啟用 SELinux 時，大多數非特殊權限 Pod 會自動套用自己的多類別安全 (MCS) 標籤。此 MCS 標籤每個 Pod 都是唯一的，旨在確保一個 Pod 中的程序無法操作任何其他 Pod 或主機上的程序。即使已標記的 Pod 以根目錄執行並可存取主機檔案系統，也無法操作檔案、對主機進行敏感系統呼叫、存取容器執行時間，或取得 kubelet 的私密金鑰材料。

因此，您在嘗試在 Pod 之間共用資料時可能會遇到問題。例如，`PersistentVolumeClaim`具有 存取模式的 `ReadWriteOnce`仍不允許多個 Pod 同時存取磁碟區。

若要在 Pod 之間啟用此共用，您可以使用 Pod 的 `seLinuxOptions`在這些 Pod 上設定相同的 MCS 標籤。在此範例中，我們將三個類別指派給 `c123,c456,c789` Pod。這不會與節點上指派給 Pod 的任何類別發生衝突，因為只會指派兩個類別。

```
securityContext:
  seLinuxOptions:
    level: "s0:c123,c456,c789"
```

在控制平面日誌中檢視 Karpenter 事件

對於已啟用控制平面日誌的 EKS 叢集，您可以透過查詢日誌來深入了解 Karpenter 的動作和決策程序。這對於疑難排解與節點佈建、擴展和終止相關的 EKS Auto Mode 問題特別有用。若要檢視 Karpenter 相關事件，請使用下列 CloudWatch Logs Insights 查詢：

```
fields @timestamp, @message
| filter @logStream like /kube-apiserver-audit/
| filter @message like 'DisruptionBlocked'
or @message like 'DisruptionLaunching'
or @message like 'DisruptionTerminating'
or @message like 'DisruptionWaitingReadiness'
or @message like 'Unconsolidatable'
or @message like 'FailedScheduling'
or @message like 'NoCompatibleInstanceTypes'
or @message like 'NodeRepairBlocked'
or @message like 'Disrupted'
or @message like 'Evicted'
or @message like 'FailedDraining'
or @message like 'TerminationGracePeriodExpiring'
or @message like 'TerminationFailed'
or @message like 'FailedConsistencyCheck'
or @message like 'InsufficientCapacityError'
or @message like 'UnregisteredTaintMissing'
or @message like 'NodeClassNotReady'
sort @timestamp desc
```

此查詢會篩選 kube-apiserver 稽核日誌中的特定 Karpenter 相關事件。這些事件包括各種中斷狀態、排程失敗、容量問題和節點相關問題。透過分析這些日誌，您可以更深入了解：

- Karpenter 為什麼要採取特定動作。
- 防止正確佈建、擴展或終止節點的任何問題。
- 執行個體類型的潛在容量或相容性問題。
- 節點生命週期事件，例如中斷、移出或終止。

若要使用此查詢：

1. 導覽至 CloudWatch 主控台
2. 從左側導覽窗格中選取「Logs Insights」

3. 選擇 EKS 叢集控制平面日誌的日誌群組
4. 將查詢貼到查詢編輯器
5. 視需要調整時間範圍
6. 執行查詢

結果會顯示 Karpenter 相關事件的時間表，協助您疑難排解問題，並了解叢集中 EKS Auto Mode 的行為。若要檢閱特定節點上的 Karpenter 動作，您可以將下列指定執行個體 ID 的篩選條件新增至上述查詢：

```
|filter @message like /[.replaceable]`i-12345678910123456`/
```

> **Note**
>
> 若要使用此查詢，必須在 EKS 叢集上啟用控制平面記錄。如果您尚未這麼做，請參閱 the section called “控制平面日誌”。

對自動模式下包含的控制器進行故障診斷

如果控制器發生問題，您應該研究：

- 如果與該控制器相關聯的資源格式正確且有效。
- 如果已為您的叢集正確設定 AWS IAM 和 Kubernetes RBAC 資源。如需詳細資訊，請參閱the section called “身分和存取”。

檢閱 EKS Auto Mode 版本備註

此頁面記錄 Amazon EKS Auto 模式的更新。您可以定期查看此頁面，以取得有關功能、錯誤修正、已知問題和已棄用功能的公告。

若要接收此特定文件頁面所有來源檔案變更的通知，您可以使用 RSS 讀取器訂閱下列 URL：

```
https://github.com/awsdocs/amazon-eks-user-guide/commits/mainline/latest/ug/automode/auto-change.adoc.atom
```

2025 年 8 月 6 日

功能：在 NodeClass 上新增了新組態`spec.advancedNetworking.associatePublicIPAddress`，可用於防止公有 IP 地址指派給 EKS Auto Mode Nodes

2025 年 6 月 30 日

功能：自動模式 NodeClass 現在除了讀取/寫入資料磁碟區之外，還會使用設定的自訂 KMS 金鑰來加密執行個體的唯讀根磁碟區。先前，自訂 KMS 金鑰僅用於加密資料磁碟區。

2025 年 6 月 20 日

功能：支援控制工作負載部署至 EC2 隨需容量保留 ODCRs)。這會將選用金鑰新增至`capacityReservationSelectorTerms` NodeClass，讓您明確控制工作負載使用的 ODCRs。如需詳細資訊，請參閱the section called “控制 ODCR 部署”。

2025 年 6 月 13 日

功能：支援 中的個別 Pod 子網路`NodeClass`。這會新增選用金鑰 `podSubnetSelectorTerms`和`podSecurityGroupSelectorTerms`，以設定 Pod 的子網路和安全群組。如需詳細資訊，請參閱the section called “Pod 的子網路選擇”。

2025 年 4 月 30 日

功能：支援 中的轉送網路代理`NodeClass`。這會新增選用金鑰`advancedNetworking`來設定 HTTPS 代理。如需詳細資訊，請參閱the section called “節點類別規格”。

2025 年 4 月 18 日

功能：支援透過單點傳送 DNS 解析 .local 網域 （通常保留給多點傳送 DNS)。

2025 年 4 月 11 日

功能：已將 `certificateBundles`和 `ephemeralStorage.kmsKeyID`新增至 `NodeClass`。如需詳細資訊，請參閱the section called “節點類別規格”。

功能：改善映像提取速度，特別是具有本機執行個體儲存體的執行個體類型，可以利用更快的映像解壓縮。

錯誤修正：已解決導致 FailedCreatePodSandBox 的競爭條件，撥號時發生錯誤：撥打 tcp 127.0.0.1：50051：連線：Pod 在啟動後立即排程至節點時發生連線遭拒。

2025 年 4 月 4 日

功能：`registryPullQPS`從 5 增加到 25，`registryBurst`從 10 增加到 50，以減少用戶端強制執行的映像提取限流 (`Failed to pull image xyz: pull QPS exceeded`)

2025 年 3 月 31 日

錯誤修正：修正以下問題：如果 Core DNS Pod 在自動模式節點上執行，節點上 Pod 的 DNS 查詢會命中該 Core DNS Pod，而不是節點本機 DNS 伺服器。自動模式節點上來自 Pod 的 DNS 查詢一律會移至節點本機 DNS。

2025 年 3 月 21 日

錯誤修正：當叢集中未安裝`kube-dns`服務時，自動模式節點現在可`kube-dns.kube-system.svc.cluster.local`正確解析。解決 GitHub 問題 #2546。

2025 年 3 月 14 日

功能：在`IPv6`叢集中啟用`IPv4`輸出。從`IPv6`自動模式叢集輸出的`IPv4`流量現在會自動轉譯為節點主要 ENI `v4`的地址。

Amazon EKS 叢集生命週期和組態

本節提供有關 Kubernetes 叢集完整生命週期管理的深入指導，以及設定這些叢集的不同方式。它涵蓋叢集建立、監控叢集運作狀態、更新 Kubernetes 版本，以及刪除叢集。它還包含進階主題，說明如何設定端點存取、啟用/停用 Windows 支援、設定私有叢集、實作自動擴展，以及如何使用區域轉移增強彈性，以在多可用區設定中重新導向流量。

主題

- 建立 Amazon EKS Auto Mode 叢集
- 建立 Amazon EKS 叢集
- 準備進行 Kubernetes 版本升級，並使用叢集洞察對錯誤設定進行疑難排解
- 將現有叢集更新為新的 Kubernetes 版本
- 刪除叢集
- 叢集 API 伺服器端點
- 在 EKS 叢集上部署 Windows 節點
- 停用 Windows 支援
- 部署網際網路存取受限的私有叢集
- 使用 Karpenter 和 Cluster Autoscaler 擴展叢集運算
- 了解 Amazon EKS 中的 Amazon Application Recovery Controller (ARC) 區域轉移
- 啟用 EKS 區域轉移以避免可用區域受損

建立 Amazon EKS Auto Mode 叢集

本主題提供使用進階組態選項建立 Amazon EKS Auto Mode 叢集的詳細說明。它涵蓋先決條件、聯網選項和附加元件組態。程序包括設定 IAM 角色、設定叢集設定、指定網路參數，以及選取附加元件。使用者可以使用 AWS Management Console 或 AWS CLI 建立叢集，並提供兩種方法的step-by-step指引。

對於尋求較不複雜的設定程序的使用者，請參閱下列的簡化叢集建立步驟：

- the section called “eksctl CLI”
- the section called “ AWS CLI”
- the section called “管理主控台”

此進階組態指南適用於需要更精細控制其 EKS Auto Mode 叢集設定，並熟悉 Amazon EKS 概念和要求的使用者。繼續進階組態之前，請確定您符合所有先決條件，並徹底了解 EKS Auto Mode 叢集的網路和 IAM 需求。

EKS Auto Mode 需要額外的 IAM 許可。如需詳細資訊，請參閱：

- the section called “EKS 自動模式叢集的 IAM 角色”
- the section called “身分和存取”

> **Note**
>
> 如果您想要在沒有 EKS Auto 模式的情況下建立叢集，請參閱 the section called “建立叢集”。本主題涵蓋進階組態。如果您想要開始使用 EKS Auto 模式，請參閱 the section called “建立叢集”。

先決條件

- 現有 VPC 和子網符合 Amazon EKS 要求。部署叢集以供生產使用之前，建議您先徹底了解 VPC 和子網要求。如果您沒有 VPC 和子網路，您可以使用 Amazon EKS provided AWS CloudFormation 範本建立它們。
- `kubectl` 命令列工具安裝在您的裝置或 AWS CloudShell 上。版本可以與叢集的 Kubernetes 版本相同，也可以比叢集的 Kubernetes 版本更早或更晚一個次要版本。例如，如果您的叢集版本為 `1.29`，則可以搭配使用 `kubectl` `1.28`、`1.29` 或 `1.30` 版。若要安裝或升級 `kubectl`，請參閱 the section called “設定 kubectl 和 eksctl”。
- 在您的裝置或 AWS CloudShell 上安裝和設定的 AWS 命令列界面 (AWS CLI) 版本 `1.27.160` `2.12.3`或更新版本。若要檢查您目前的版本，請使用 `aws --version`。若要安裝最新版本，請參閱《 AWS 命令列界面使用者指南》中的使用 aws 設定安裝 和 快速組態。 https://docs.aws.amazon.com/cli/latest/userguide/cli-configure-quickstart.html#cli-configure-quickstart-config
- 具有建立和修改 EKS 和 IAM 資源許可的 IAM 主體。

建立叢集 - AWS 主控台

1. 開啟 Amazon EKS 主控台。
2. 選取 Add cluster (新增叢集)，然後選取 Create (建立)。

3. 在組態選項下，選取自訂組態。
   - 本主題涵蓋自訂組態。如需快速組態的詳細資訊，請參閱 the section called “管理主控台”。
4. 確認已啟用使用 EKS 自動模式。
   - 本主題涵蓋使用 EKS Auto 模式建立叢集。如需不使用 EKS Auto Mode 建立叢集的詳細資訊，請參閱 the section called “建立叢集”。
5. 在 Configure cluster (設定叢集) 頁面上，輸入下列欄位：
   - Name (名稱)：叢集的名稱。名稱只能包含英數字元 （區分大小寫）、連字號和底線。它必須以英數字元開頭，且長度不可超過 100 個字元。名稱在您要建立叢集 AWS 的區域和 AWS 帳戶中必須是唯一的。
   - 叢集 IAM 角色 – 選擇您建立的 Amazon EKS 叢集 IAM 角色，以允許 Kubernetes 控制平面代表您管理 AWS 資源。如果您先前尚未為 EKS Auto Mode 建立叢集 IAM 角色，請選取建立建議的角色按鈕，在 IAM 主控台中建立具有必要許可的角色。
   - Kubernetes version (Kubernetes 版本) – 您的叢集使用的 Kubernetes 版本。我們建議選取最新版本，除非您需要較早版本。
   - 升級政策 — 您要為叢集設定的 Kubernetes 版本政策。如果您希望叢集只在標準支援版本上執行，您可以選擇標準。如果您希望叢集在版本的標準支援結束時輸入延伸支援，您可以選擇延伸。如果您選取目前正在擴充支援的 Kubernetes 版本，則無法選取標準支援做為選項。
6. 在設定叢集頁面的自動模式運算區段中，輸入下列欄位：
   - 節點集區 — 判斷您是否要在節點集區中使用建置。如需詳細資訊，請參閱the section called “檢閱內建節點集區”。
   - 節點 IAM 角色 — 如果您啟用任何內建節點集區，則需要選取節點 IAM 角色。EKS Auto Mode 會將此角色指派給新節點。您無法在建立叢集之後變更此值。如果您先前尚未為 EKS Auto Mode 建立節點 IAM 角色，請選取建立建議的角色按鈕，以建立具有必要許可的角色。如需有關此角色的詳細資訊，請參閱 the section called “身分和存取”。
7. 在設定叢集頁面的叢集存取區段中，輸入下列欄位：
   - 引導叢集管理員存取 - 叢集建立者會自動成為 Kubernetes 管理員。如果您想要停用此功能，請選取不允許叢集管理員存取。
   - 叢集身分驗證模式 — EKS Auto Mode 需要 EKS 存取項目，即 EKS API 身分驗證模式。您可以選擇選擇 EKS API 和 ConfigMap 來啟用`ConfigMap`身分驗證模式。
8. 在設定叢集頁面上輸入其餘欄位：
   - Secrets encryption (秘密加密)：(選用) 選擇使用 KMS 金鑰啟用 Kubernetes 秘密的秘密加密。您也可以在建立叢集後啟用此功能。啟用此功能之前，請確定您已熟悉 中的資訊the section called “啟用秘密加密”。

- ARC 區域轉移 — EKS Auto 模式不支援電弧區域轉移。
- Tags (標籤) – (選用) 將任何標籤新增到您的叢集。如需詳細資訊，請參閱the section called “標記您的 資源”。

  完成此頁面後，請選擇下一步。

9. 在 Specify networking (指定網路) 頁面上，選取下列欄位的值：
   - VPC：選擇符合 Amazon EKS VPC 要求的現有 VPC 來建立您的叢集。在選擇 VPC 之前，建議您先熟悉 中的所有要求和考量事項the section called “VPC 和子網要求”。您無法在叢集建立後變更要使用的 VPC。如果沒有列出 VPC，則您需要先建立一個。如需詳細資訊，請參閱 the section called “建立 VPC”。
   - Subnets (子網路)：根據預設，前一個欄位指定的 VPC 中的所有可用子網路會預先選取。您必須選取至少兩個。

     您選擇的子網必須符合 Amazon EKS 子網要求。在選取子網路之前，建議您先熟悉所有 Amazon EKS VPC 和子網路需求和考量事項。

     安全群組:(選用) 指定一或多個您希望 Amazon EKS 與其建立的網路介面相關聯的安全群組。

     無論您是否選擇任何安全群組，Amazon EKS 都會建立一個安全群組，以支援您的叢集和 VPC 之間的通訊。Amazon EKS 將此安全群組及您選擇的任何群組與其建立的網路介面相關聯。如需有關 Amazon EKS 建立的叢集安全群組的詳細資訊，請參閱 the section called “安全群組要求”。您可以修改 Amazon EKS 建立的叢集安全群組中的規則。
   - 選擇叢集 IP 地址系列：您可以選擇 IPv4 或 IPv6。

     根據預設，Kubernetes 會將IPv4地址指派給 Pod 和服務。在決定使用 IPv6 系列之前，請確定您已熟悉 VPC 要求和考量、、 和 主題中的所有考量和要求。 the section called “子網需求和注意事項” the section called “安全群組要求” the section called “IPv6”如果您選擇 IPv6 系列，則無法指定 Kubernetes 的地址範圍，以便從 為 IPv4 系列指派IPv6服務地址。Kubernetes 會從唯一的本機地址範圍 () 指派服務地址fc00::/7。
   - （選用）選擇設定 Kubernetes 服務 IP 地址範圍，並指定服務**IPv4**範圍。

     指定您自己的範圍有助於防止 Kubernetes 服務與對等或連接到 VPC 的其他網路之間發生衝突。在 CIDR 標記法中輸入範圍。例如：10.2.0.0/16。

     CIDR 區塊必須符合下列需求：
     - 處於以下範圍之一：10.0.0.0/8、172.16.0.0/12 或 192.168.0.0/16。
     - 具有最小尺寸的 /24 和最大尺寸的 /12。

- 與您的 Amazon EKS 資源的 VPC 範圍不重疊。

您只能在使用 IPv4 地址系列時指定此選項，並且只能在建立叢集時指定。如果您未指定此項目，則 Kubernetes 會從 10.100.0.0/16或 172.20.0.0/16 CIDR 區塊指派服務 IP 地址。

- 針對 Cluster endpoint access (叢集端點存取)，選取一個選項。建立叢集後，您可以變更此選項。在選取非預設選項之前，請務必熟悉這些選項及其含義。如需詳細資訊，請參閱the section called “叢集端點存取”。

完成此頁面後，請選擇下一步。

10.(選用) 在設定可觀測性頁面上，選擇要開啟的指標和控制平面日誌記錄選項。根據預設，系統會關閉每個日誌類型。

- 如需 Prometheus 指標選項的詳細資訊，請參閱 the section called “步驟 1：開啟 Prometheus 指標”。
- 如需控制平面日誌記錄選項的詳細資訊，請參閱 the section called “控制平面日誌”。
- 完成此頁面後，請選擇下一步。

11.在 Select add-ons (選取附加元件) 頁面上，選擇您要新增至叢集的附加元件。您可以視需要選擇任意數量的 Amazon EKS 附加元件和 AWS Marketplace 附加元件。如果您想要安裝的 AWS Marketplace 附加元件未列出，您可以按一下頁面編號以檢視其他頁面結果，或在搜尋方塊中輸入文字來搜尋可用的 AWS Marketplace 附加元件。您也可以依類別、廠商或定價模型進行篩選，然後從搜尋結果中選擇附加元件。建立叢集時，您可以檢視、選取和安裝支援 EKS Pod 身分的任何附加元件，如 中所述the section called “Pod 身分”。

- EKS Auto Mode 會將特定附加元件的功能自動化。如果您計劃將 EKS 受管節點群組部署到 EKS 自動模式叢集，請選取其他 Amazon EKS 附加元件並檢閱選項。您可能需要安裝附加元件，例如 CoreDNS 和 kube-proxy。EKS 只會在自我管理節點和節點群組上安裝本節中的附加元件。
- 完成此頁面後，請選擇下一步。

12.在設定選取的附加元件設定頁面上，選取您要安裝的版本。建立叢集後，您隨時皆可更新至更新版本。

對於支援 EKS Pod 身分的附加元件，您可以使用 主控台自動產生角色，其中包含專為附加元件預先填入的名稱、 AWS 受管政策和信任政策。您可以重複使用現有角色，或為支援的附加元件建立新的角色。如需使用主控台為支援 EKS Pod 身分的附加元件建立角色的步驟，請參閱 the section called “建立附加元件AWS （主控台）”。如果附加元件不支援 EKS Pod 身分，則會顯示一則訊息，其中包含在叢集建立後使用精靈建立服務帳戶 (IRSA) 的 IAM 角色的指示。

您可以在建立叢集後更新每個附加元件的組態。如需有關設定附加元件的詳細資訊，請參閱the section called “更新 附加元件”。完成此頁面後，請選擇下一步。

13.在 Review and create (檢閱並建立) 頁面上，檢閱您在先前頁面上輸入或選取的資訊。如需變更，請選擇 Edit (編輯)。當您滿意時，請選擇建立。在佈建叢集時，Status (狀態) 欄位顯示 CREATING (正在建立)。

> **Note**
>
> 您可能會收到錯誤，告知您請求中的其中一個可用區域沒有足夠的容量來建立 Amazon EKS 叢集。如果發生這種情況，錯誤輸出包含的可用區域可支援新的叢集。使用至少兩個位於帳戶的支援可用區域子網路來建立您的叢集。如需詳細資訊，請參閱the section called “容量不足”。

叢集佈建需要幾分鐘的時間。

建立叢集 - AWS CLI

下列 CLI 指示涵蓋建立 IAM 資源和建立叢集。

建立 EKS 自動模式叢集 IAM 角色

步驟 1：建立信任政策

建立允許 Amazon EKS 服務擔任角色的信任政策。將政策儲存為 `trust-policy.json`：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "eks.amazonaws.com"
      },
      "Action": [
        "sts:AssumeRole",
        "sts:TagSession"
      ]
    }
  ]
}
```

步驟 2：建立 IAM 角色

使用信任政策來建立叢集 IAM 角色：

```
aws iam create-role \
    --role-name AmazonEKSAutoClusterRole \
    --assume-role-policy-document file://trust-policy.json
```

步驟 3：記下角色 ARN

擷取並儲存新角色的 ARN，以供後續步驟使用：

```
aws iam get-role --role-name AmazonEKSAutoClusterRole --query "Role.Arn" --output text
```

步驟 4：連接必要的政策

將下列 AWS 受管政策連接至叢集 IAM 角色，以授予必要的許可：

AmazonEKSClusterPolicy：

```
aws iam attach-role-policy \
    --role-name AmazonEKSAutoClusterRole \
    --policy-arn arn:aws: iam::aws:policy/AmazonEKSClusterPolicy
```

AmazonEKSComputePolicy：

```
aws iam attach-role-policy \
    --role-name AmazonEKSAutoClusterRole \
    --policy-arn arn:aws: iam::aws:policy/AmazonEKSComputePolicy
```

AmazonEKSBlockStoragePolicy：

```
aws iam attach-role-policy \
    --role-name AmazonEKSAutoClusterRole \
    --policy-arn arn:aws: iam::aws:policy/AmazonEKSBlockStoragePolicy
```

AmazonEKSLoadBalancingPolicy：

```
aws iam attach-role-policy \
    --role-name AmazonEKSAutoClusterRole \
    --policy-arn arn:aws: iam::aws:policy/AmazonEKSLoadBalancingPolicy
```

AmazonEKSNetworkingPolicy：

```
aws iam attach-role-policy \
    --role-name AmazonEKSAutoClusterRole \
    --policy-arn arn:aws: iam::aws:policy/AmazonEKSNetworkingPolicy
```

建立 EKS 自動模式節點 IAM 角色

步驟 1：建立信任政策

建立允許 Amazon EKS 服務擔任角色的信任政策。將政策儲存為 `node-trust-policy.json`：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "ec2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

步驟 2：建立節點 IAM 角色

使用上一個步驟中的 node-trust-policy.json 檔案來定義哪些實體可以擔任該角色。執行下列命令來建立節點 IAM 角色：

```
aws iam create-role \
    --role-name AmazonEKSAutoNodeRole \
    --assume-role-policy-document file://node-trust-policy.json
```

步驟 3：記下角色 ARN

建立角色之後，請擷取並儲存節點 IAM 角色的 ARN。在後續步驟中，您將需要此 ARN。使用下列命令來取得 ARN：

```
aws iam get-role --role-name AmazonEKSAutoNodeRole --query "Role.Arn" --output text
```

步驟 4：連接必要的政策

將下列 AWS 受管政策連接至節點 IAM 角色，以提供必要的許可：

AmazonEKSWorkerNodeMinimalPolicy：

```
aws iam attach-role-policy \
    --role-name AmazonEKSAutoNodeRole \
    --policy-arn arn:aws: iam::aws:policy/AmazonEKSWorkerNodeMinimalPolicy
```

AmazonEC2ContainerRegistryPullOnly：

```
aws iam attach-role-policy \
    --role-name AmazonEKSAutoNodeRole \
    --policy-arn arn:aws: iam::aws:policy/AmazonEC2ContainerRegistryPullOnly
```

建立叢集

1. 使用下列命令建立您的叢集。執行命令之前，請執行下列替換：
   - 將 *region-code* 取代為您要建立叢集 AWS 的區域。
   - 以叢集的名稱取代 *my-cluster*。名稱只能包含英數字元（區分大小寫）、連字號和底線。它必須以英數字元開頭，且長度不可超過 100 個字元。在您建立叢集 AWS 的區域和 AWS 帳戶中，名稱必須是唯一的。
   - 將 *1.30* 取代為任何 eks/latest/userguide/kubernetes-versions.html【Amazon EKS 支援的版本，type="documentation"】。
   - 將 *111122223333* 取代為您的帳戶 ID
   - 如果您已為叢集和節點角色建立不同的命名 IAM 角色，請取代 ARNs。
   - 以您自己的值取代 `subnetIds` 值。您也可以新增其他 ID。您必須指定至少兩個子網路 ID。

     您選擇的子網必須符合 Amazon EKS 子網要求。在選取子網路之前，建議您先熟悉所有 Amazon EKS VPC 和子網路需求和考量事項。
   - 如果您不想指定安全群組 ID，`,securityGroupIds=sg-<ExampleID1>`請從命令中移除 。如果要指定一或多個安全群組 ID，請將 `securityGroupIds` 取代為您自己的值。您也可以新增其他 ID。

     無論您是否選擇任何安全群組，Amazon EKS 都會建立一個安全群組，以支援您的叢集和 VPC 之間的通訊。Amazon EKS 將此安全群組及您選擇的任何群組與其建立的網路介面相關聯。如需

有關 Amazon EKS 建立的叢集安全群組的詳細資訊，請參閱 the section called “安全群組要求”。您可以修改 Amazon EKS 建立的叢集安全群組中的規則。

```
aws eks create-cluster \
  --region region-code \
  --name my-cluster \
  --kubernetes-version 1.30 \
  --role-arn arn:aws: iam::111122223333:role/AmazonEKSAutoClusterRole \
  --resources-vpc-config '{"subnetIds": ["subnet-ExampleID1","subnet-ExampleID2"],
 "securityGroupIds": ["sg-ExampleID1"], "endpointPublicAccess": true,
 "endpointPrivateAccess": true}' \
  --compute-config '{"enabled": true, "nodeRoleArn": "arn:aws:
 iam::111122223333:role/AmazonEKSAutoNodeRole", "nodePools": ["general-purpose",
 "system"]}' \
  --kubernetes-network-config '{"elasticLoadBalancing": {"enabled": true}}' \
  --storage-config '{"blockStorage": {"enabled": true}}' \
  --access-config '{"authenticationMode": "API"}'
```

> **Note**
>
> 您可能會收到錯誤，告知您請求中的其中一個可用區域沒有足夠的容量來建立 Amazon EKS 叢集。如果發生這種情況，錯誤輸出包含的可用區域可支援新的叢集。使用至少兩個位於帳戶的支援可用區域子網路來建立您的叢集。如需詳細資訊，請參閱the section called “容量不足”。

以下是選用設定，如有需要，必須將其新增至上一個命令中。您只能在建立叢集時啟用這些選項，而不能在建立叢集之後啟用。

- 如果您想指定 Kubernetes 要從哪個 IPv4 無類別域間路由 (CIDR) 區塊中指派服務 IP 地址，您必須將 `--kubernetes-network-config serviceIpv4Cidr=<cidr-block>` 新增至下列命令來指定。

  指定您自己的範圍有助於防止 Kubernetes 服務與對等或連接到 VPC 的其他網路之間發生衝突。在 CIDR 標記法中輸入範圍。例如：`10.2.0.0/16`。

  CIDR 區塊必須符合下列需求：

  - 處於以下範圍之一：`10.0.0.0/8`、`172.16.0.0/12` 或 `192.168.0.0/16`。
  - 具有最小尺寸的 `/24` 和最大尺寸的 `/12`。

- 與您的 Amazon EKS 資源的 VPC 範圍不重疊。

  您只能在使用 IPv4 地址系列時指定此選項，並且只能在建立叢集時指定。如果您未指定此項目，則 Kubernetes 會從 10.100.0.0/16或 172.20.0.0/16 CIDR 區塊指派服務 IP 地址。

- 如果您要建立叢集，並希望叢集將IPv6地址指派給 Pod 和服務，而不是IPv4地址，請將 --kubernetes-network-config ipFamily=ipv6新增至下列命令。

  根據預設，Kubernetes 會將IPv4地址指派給 Pod 和服務。在決定使用 IPv6 系列之前，請確定您已熟悉 VPC 要求和考量、、 和 主題中的所有考量和要求。 the section called “子網需求和注意事項” the section called “安全群組要求” the section called “IPv6”如果您選擇 IPv6 系列，則無法指定 Kubernetes 的地址範圍，以便像針對 IPv4 系列一樣從 指派IPv6服務地址。Kubernetes 會從唯一的本機地址範圍 () 指派服務地址fc00::/7。

2. 佈建叢集需要幾分鐘才能完成。您可以使用下列命令來查詢叢集的狀態。

```
aws eks describe-cluster --region region-code --name my-cluster --query
 "cluster.status"
```

後續步驟

- the section called “使用 kubectl 存取叢集”
- the section called “授予許可”
- 為叢集啟用密鑰加密。
- 設定叢集的日誌。
- 將節點新增至叢集。

建立 Amazon EKS 叢集

> **Note**
>
> 本主題涵蓋在沒有 EKS Auto 模式的情況下建立 EKS 叢集。
> 如需建立 EKS Auto Mode 叢集的詳細說明，請參閱 the section called “建立自動叢集”。
> 若要開始使用 EKS Auto 模式，請參閱 the section called “建立叢集 (EKS Auto 模式) ”。

本主題提供可用選項的概觀，並介紹您建立 Amazon EKS 叢集時需要考慮的問題。如果您需要使用現場部署基礎設施建立叢集做為節點的運算，請參閱 the section called “建立叢集”。如果這是您第一次建立 Amazon EKS 叢集，我們建議您遵循 中的其中一個指南開始使用。這些指南可協助您建立一個簡單的預設叢集，而無需擴展到所有可用選項。

先決條件

- 現有 VPC 和子網符合 Amazon EKS 要求。部署叢集以供生產使用之前，建議您先徹底了解 VPC 和子網要求。如果您沒有 VPC 和子網路，您可以使用 Amazon EKS provided AWS CloudFormation 範本建立它們。
- `kubectl` 命令列工具安裝在您的裝置或 AWS CloudShell 上。版本可以與叢集的 Kubernetes 版本相同，也可以比叢集的 Kubernetes 版本更早或更晚一個次要版本。若要安裝或升級 `kubectl`，請參閱 the section called “設定 kubectl 和 eksctl”。
- 在您的裝置或 AWS CloudShell 上安裝和設定的 AWS 命令列界面 (AWS CLI) 版本 `1.27.160` `2.12.3`或更新版本。若要檢查您目前的版本，請使用 `aws --version | cut -d / -f2 | cut -d ' ' -f1`。適用於 macOS 的 `yum`、 `apt-get`或 Homebrew 等套件管理員通常是最新版本 CLI AWS 後面的幾個版本。若要安裝最新版本，請參閱《 AWS 命令列界面使用者指南》中的使用 aws 設定安裝 和 快速組態。 https://docs.aws.amazon.com/cli/latest/userguide/cli-configure-quickstart.html#cli-configure-quickstart-config在 AWS CloudShell 中安裝的 AWS CLI 版本也可能是最新版本後面的幾個版本。若要更新它，請參閱《CloudShell AWS 使用者指南》中的將 CLI 安裝到您的主目錄。 AWS CloudShell
- 具有對 Amazon EKS 叢集 `create` 和 `describe` 的許可的 IAM 主體。如需詳細資訊，請參閱the section called “在 Outpost 上建立本機 Kubernetes 叢集”及the section called “所有叢集的清單或描述”。

步驟 1：建立叢集 IAM 角色

1. 如果您已有叢集 IAM 角色，或者您要使用 建立叢集`eksctl`，則可以略過此步驟。根據預設，`eksctl` 會為您建立角色。
2. 執行下列命令以建立 IAM 信任政策 JSON 檔案。

```
cat >eks-cluster-role-trust-policy.json <<EOF
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
```

```
      "Principal": {
        "Service": "eks.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF
```

3. 建立 Amazon EKS 叢集 IAM 角色。如有必要，請在 *eks-cluster-role-trust-policy.json* 前面加上您在上一步中寫入檔案的電腦的路徑。命令會將您在上一步驟中建立的信任策略與角色相關聯。若要建立 IAM 角色，必須為建立角色的 IAM 主體指派以下 iam:CreateRole 動作 (許可)。

```
aws iam create-role --role-name myAmazonEKSClusterRole --assume-role-policy-document
 file://"eks-cluster-role-trust-policy.json"
```

4. 您可以指派 Amazon EKS 受管政策或建立自己的自訂政策。如需了解在自訂政策中必須使用的最低許可，請參閱 the section called “叢集 IAM 角色”。

   將名為 AmazonEKSClusterPolicy 的 Amazon EKS 受管政策連接至角色。若要將 IAM 政策連接至 IAM 主體，必須為連接政策的 IAM 實體指派以下 IAM 動作之一 (許可)：iam:AttachUserPolicy 或 iam:AttachRolePolicy。

```
aws iam attach-role-policy --policy-arn arn:aws: iam::aws:policy/
AmazonEKSClusterPolicy --role-name myAmazonEKSClusterRole
```

服務連結角色

Amazon EKS 會自動建立名為 的服務連結角色AWSServiceRoleForAmazonEKS。

這是叢集 IAM 角色以外的項目。服務連結角色是直接連結至 Amazon EKS 的一種特殊 IAM 角色類型。此角色允許 Amazon EKS 管理您帳戶中的叢集。如需詳細資訊，請參閱the section called “叢集角色”。

您用來建立 EKS 叢集的 IAM Identity 必須具有建立服務連結角色的許可。這包括 iam:CreateServiceLinkedRole許可。

如果服務連結角色尚未存在，且您目前的 IAM 角色沒有足夠的許可來建立，叢集建立操作將會失敗。

步驟 2：建立叢集

您可以使用下列方式建立叢集：

- eksctl
- 的 AWS Management Console
- AWS CLI

建立叢集 - eksctl

1. 您需要在裝置0.210.0或 AWS CloudShell 上安裝版本 或更新版本的eksctl命令列工具。如需有關安裝或更新 eksctl 的指示，請參閱 eksctl 文件中的安裝一節。
2. 在預設區域中使用 Amazon EKS 預設 Kubernetes 版本建立 Amazon EKS IPv4叢集 AWS 。執行命令之前，請執行下列替換：
3. 將 *region-code* 取代為您要在其中建立叢集 AWS 的區域。
4. 以叢集的名稱取代 *my-cluster*。此名稱僅能使用英數字元 (區分大小寫) 和連字號。它必須以英數字元開頭，且長度不可超過 100 個字元。在您建立叢集 AWS 的區域和 AWS 帳戶中，名稱必須是唯一的。
5. 將 *1.33* 取代為任何 eks/latest/userguide/kubernetes-versions.html【Amazon EKS 支援的版本，type="documentation"】。
6. 變更 vpc-private-subnets 的值以符合您的要求。您也可以新增其他 ID。您必須指定至少兩個子網路 ID。如果您想要指定公有子網路，您可以將 --vpc-private-subnets變更為 --vpc-public-subnets。公有子網路具有與網際網路閘道路由相關聯的路由表，但私有子網路沒有相關聯的路由表。我們建議盡可能使用私有子網。

   您選擇的子網必須符合 Amazon EKS 子網要求。在選取子網路之前，建議您先熟悉所有 Amazon EKS VPC 和子網路需求和考量事項。
7. 執行以下命令：

```
eksctl create cluster --name my-cluster --region region-code --version 1.33 --vpc-
private-subnets subnet-ExampleID1,subnet-ExampleID2 --without-nodegroup
```

   叢集佈建需要幾分鐘的時間。建立叢集時，會出現幾行輸出。輸出的最後一行類似於下面的範例行。

```
[#]  EKS cluster "my-cluster" in "region-code" region is ready
```

8. 繼續 the section called “步驟 3：更新 kubeconfig”

選用設定

若要查看使用 eksctl 建立叢集時可指定的大部分選項，請使用 eksctl create cluster --help 命令。若要查看所有可用的選項，您可使用 config 檔案。如需詳細資訊，請參閱 eksctl 文件中的使用組態檔與組態檔結構描述。您可以在 GitHub 上找到組態檔範例。

以下是選用設定，如有需要，必須將其新增至上一個命令中。您只能在建立叢集時啟用這些選項，而不能在建立叢集之後啟用。如果您需要指定這些選項，您必須使用 eksctl 組態檔案建立叢集並指定設定，而不是使用先前的命令。

- 如果您想要指定 Amazon EKS 指派給其建立之網路介面的一或多個安全群組，請指定 securityGroup 選項。

  無論您是否選擇任何安全群組，Amazon EKS 都會建立一個安全群組，以支援您的叢集和 VPC 之間的通訊。Amazon EKS 將此安全群組及您選擇的任何群組與其建立的網路介面相關聯。如需有關 Amazon EKS 建立的叢集安全群組的詳細資訊，請參閱 the section called “安全群組要求”。您可以修改 Amazon EKS 建立的叢集安全群組中的規則。
- 如果您想要指定 Kubernetes 從哪個無IPv4類別網域間路由 (CIDR) 區塊指派服務 IP 地址，請指定 serviceIPv4CIDR 選項。

  指定您自己的範圍有助於防止 Kubernetes 服務與對等或連接到 VPC 的其他網路之間發生衝突。在 CIDR 標記法中輸入範圍。例如：10.2.0.0/16。

  CIDR 區塊必須符合下列需求：

  - 處於以下範圍之一：10.0.0.0/8、172.16.0.0/12 或 192.168.0.0/16。
  - 具有最小尺寸的 /24 和最大尺寸的 /12。
  - 與您的 Amazon EKS 資源的 VPC 範圍不重疊。

    您只能在使用 IPv4 地址系列時指定此選項，並且只能在建立叢集時指定。如果您未指定此項目，則 Kubernetes 會從 10.100.0.0/16或 172.20.0.0/16 CIDR 區塊指派服務 IP 地址。
- 如果您要建立叢集，並希望叢集將IPv6地址指派給 Pod 和服務，而不是IPv4地址，請指定 ipFamily 選項。

根據預設，Kubernetes 會將IPv4地址指派給 Pod 和服務。在決定使用 IPv6 系列之前，請確定您已熟悉 the section called “VPC 要求和注意事項”、the section called “安全群組要求”、 the section called “子網需求和注意事項”和 the section called “IPv6”主題中的所有考量事項和要求。如果您選擇 IPv6 系列，則無法指定 Kubernetes 的地址範圍，以便從 為 IPv4 系列指派IPv6服務地址。Kubernetes 會從唯一的本機地址範圍 () 指派服務地址fc00::/7。

建立叢集 - AWS 主控台

1. 開啟 Amazon EKS 主控台。
2. 選取 Add cluster (新增叢集)，然後選取 Create (建立)。
3. 在組態選項下，選取自訂組態
   - 如需使用 EKS Auto Mode 快速建立叢集的詳細資訊，請參閱 the section called “管理主控台”。
4. 在 EKS Auto 模式下，關閉使用 EKS Auto 模式。
   - 如需使用自訂組態建立 EKS Auto Mode 叢集的詳細資訊，請參閱 the section called “建立自動叢集”。
5. 在 Configure cluster (設定叢集) 頁面上，輸入下列欄位：
   - Name (名稱)：叢集的名稱。名稱只能包含英數字元 （區分大小寫）、連字號和底線。它必須以英數字元開頭，且長度不可超過 100 個字元。在您建立叢集 AWS 的區域和 AWS 帳戶中，名稱必須是唯一的。
   - 叢集 IAM 角色 – 選擇您建立的 Amazon EKS 叢集 IAM 角色，以允許 Kubernetes 控制平面代表您管理 AWS 資源。
   - Kubernetes version (Kubernetes 版本) – 您的叢集使用的 Kubernetes 版本。我們建議選取最新版本，除非您需要較早版本。
   - 支援類型 — 您要為叢集設定的 Kubernetes 版本政策。如果您希望叢集只在標準支援版本上執行，您可以選擇標準支援。如果您希望叢集在版本的標準支援結束時輸入延伸支援，您可以選擇延伸支援。如果您選取目前處於延伸支援的 Kubernetes 版本，則無法選取標準支援做為選項。
   - Secrets encryption (秘密加密)：(選用) 選擇使用 KMS 金鑰啟用 Kubernetes 秘密的秘密加密。您也可以在建立叢集後啟用此功能。啟用此功能之前，請確定您已熟悉 中的資訊the section called “啟用秘密加密”。
   - Tags (標籤) – (選用) 將任何標籤新增到您的叢集。如需詳細資訊，請參閱the section called “標記您的 資源”。
   - ARC 區域轉移 - （選用） 您可以使用 Route53 Application Recovery 控制器來緩解受損的可用區域。如需詳細資訊，請參閱the section called “了解區域轉移”。

6. 在設定叢集頁面的叢集存取區段中，輸入下列欄位：

   - 引導叢集管理員存取 - 叢集建立者會自動成為 Kubernetes 管理員。如果您想要停用此功能，請選取不允許叢集管理員存取。
   - 叢集身分驗證模式 — 決定如何授予 IAM 使用者和角色對 Kubernetes APIs存取權。如需詳細資訊，請參閱the section called “設定叢集身分驗證模式”。

     完成此頁面後，請選擇下一步。

7. 在 Specify networking (指定網路) 頁面上，選取下列欄位的值：

   - VPC：選擇符合 Amazon EKS VPC 要求的現有 VPC 來建立您的叢集。在選擇 VPC 之前，建議您先熟悉 中的所有需求和考量事項the section called “VPC 和子網要求”。您無法在叢集建立後變更要使用的 VPC。如果沒有列出 VPC，則您需要先建立一個。如需詳細資訊，請參閱 the section called “建立 VPC”。
   - Subnets (子網路)：根據預設，前一個欄位指定的 VPC 中的所有可用子網路會預先選取。您必須選取至少兩個。

     您選擇的子網必須符合 Amazon EKS 子網要求。在選取子網路之前，建議您先熟悉所有 Amazon EKS VPC 和子網路需求和考量事項。

     安全群組:(選用) 指定一或多個您希望 Amazon EKS 與其建立的網路介面相關聯的安全群組。

     無論您是否選擇任何安全群組，Amazon EKS 都會建立一個安全群組，以支援您的叢集和 VPC 之間的通訊。Amazon EKS 將此安全群組及您選擇的任何群組與其建立的網路介面相關聯。如需有關 Amazon EKS 建立的叢集安全群組的詳細資訊，請參閱 the section called “安全群組要求”。您可以修改 Amazon EKS 建立的叢集安全群組中的規則。
   - 選擇叢集 IP 地址系列：您可以選擇 IPv4 或 IPv6。

     根據預設，Kubernetes 會將IPv4地址指派給 Pod 和服務。在決定使用 IPv6 系列之前，請確定您已熟悉 VPC 要求和考量、、 和 主題中的所有考量和要求。 the section called “子網需求和注意事項” the section called “安全群組要求” the section called “IPv6”如果您選擇 IPv6 系列，則無法指定 Kubernetes 的地址範圍，以便像針對 IPv4 系列一樣從 指派IPv6服務地址。Kubernetes 會從唯一的本機地址範圍 () 指派服務地址fc00::/7。
   - （選用）選擇設定 Kubernetes 服務 IP 地址範圍，並指定服務**IPv4**範圍。

     指定您自己的範圍有助於防止 Kubernetes 服務與對等或連接到 VPC 的其他網路之間發生衝突。在 CIDR 標記法中輸入範圍。例如：10.2.0.0/16。

     CIDR 區塊必須符合下列需求：

- 處於以下範圍之一：10.0.0.0/8、172.16.0.0/12 或 192.168.0.0/16。
- 具有最小尺寸的 /24 和最大尺寸的 /12。
- 與您的 Amazon EKS 資源的 VPC 範圍不重疊。

您只能在使用 IPv4 地址系列時指定此選項，並且只能在建立叢集時指定。如果您未指定此項目，則 Kubernetes 會從 10.100.0.0/16或 172.20.0.0/16 CIDR 區塊指派服務 IP 地址。

- 針對 Cluster endpoint access (叢集端點存取)，選取一個選項。建立叢集後，您可以變更此選項。在選取非預設選項之前，請務必熟悉這些選項及其含義。如需詳細資訊，請參閱the section called “叢集端點存取”。

完成此頁面後，請選擇下一步。

8. (選用) 在設定可觀測性頁面上，選擇要開啟的指標和控制平面日誌記錄選項。根據預設，系統會關閉每個日誌類型。
   - 如需 Prometheus 指標選項的詳細資訊，請參閱 the section called “步驟 1：開啟 Prometheus 指標”。
   - 如需控制平面日誌記錄選項的詳細資訊，請參閱 the section called “控制平面日誌”。

   完成此頁面後，請選擇下一步。
9. 在 Select add-ons (選取附加元件) 頁面上，選擇您要新增至叢集的附加元件。某些附加元件會預先選取。您可以視需要選擇任意數量的 Amazon EKS 附加元件和 AWS Marketplace 附加元件。如果您想要安裝的 AWS Marketplace 附加元件未列出，您可以按一下頁面編號以檢視其他頁面結果，或在搜尋方塊中輸入文字來搜尋可用的 AWS Marketplace 附加元件。您也可以依類別、廠商或定價模型進行篩選，然後從搜尋結果中選擇附加元件。建立叢集時，您可以檢視、選取和安裝支援 EKS Pod 身分的任何附加元件，如 中所述the section called “Pod 身分”。

   完成此頁面後，請選擇下一步。

   預設會安裝某些附加元件，例如 Amazon VPC CNI、CoreDNS 和 kube-proxy。如果您停用任何預設附加元件，這可能會影響您執行 Kubernetes 應用程式的能力。
10. 在設定選取的附加元件設定頁面上，選取您要安裝的版本。建立叢集後，您隨時皆可更新至更新版本。

    對於支援 EKS Pod 身分的附加元件，您可以使用 主控台自動產生角色，其中包含專為附加元件預先填入的名稱、 AWS 受管政策和信任政策。您可以重複使用現有角色，或為支援的附加元件建立新的角色。如需使用主控台為支援 EKS Pod 身分的附加元件建立角色的步驟，請參閱 the section called “建立附加元件AWS （主控台）”。如果附加元件不支援 EKS Pod 身分，則會顯示訊息，其中包含在叢集建立後使用精靈建立服務帳戶 (IRSA) 的 IAM 角色的指示。

您可以在建立叢集後更新每個附加元件的組態。如需有關設定附加元件的詳細資訊，請參閱the section called “更新 附加元件”。完成此頁面後，請選擇下一步。

11. 在 Review and create (檢閱並建立) 頁面上，檢閱您在先前頁面上輸入或選取的資訊。如需變更，請選擇 Edit (編輯)。當您滿意時，請選擇建立。在佈建叢集時，Status (狀態) 欄位顯示 CREATING (正在建立)。

> **Note**
>
> 您可能會收到錯誤，告知您請求中的其中一個可用區域沒有足夠的容量來建立 Amazon EKS 叢集。如果發生這種情況，錯誤輸出包含的可用區域可支援新的叢集。使用至少兩個位於帳戶的支援可用區域子網路來建立您的叢集。如需詳細資訊，請參閱the section called “容量不足”。

叢集佈建需要幾分鐘的時間。

12. 繼續 the section called “步驟 3：更新 kubeconfig”

建立叢集 - AWS CLI

1. 使用下列命令建立您的叢集。執行命令之前，請執行下列替換：
   - 將 *region-code* 取代為您要在其中建立叢集 AWS 的區域。
   - 使用叢集的名稱取代 *my-cluster*。名稱只能包含英數字元（區分大小寫）、連字號和底線。它必須以英數字元開頭，且長度不可超過 100 個字元。在您建立叢集 AWS 的區域和 AWS 帳戶中，名稱必須是唯一的。
   - 將 *1.33* 取代為任何 eks/latest/userguide/kubernetes-versions.html【Amazon EKS 支援的版本，type="documentation"】。
   - 將 *111122223333* 取代為您的 帳戶 ID，並將 *myAmazonEKSClusterRole* 取代為您的叢集 IAM 角色名稱。
   - 以您自己的值取代 `subnetIds` 值。您也可以新增其他 ID。您必須指定至少兩個子網路 ID。

     您選擇的子網必須符合 Amazon EKS 子網要求。在選取子網路之前，建議您先熟悉所有 Amazon EKS VPC 和子網路需求和考量事項。
   - 如果您不想指定安全群組 ID，，`securityGroupIds=sg-<ExampleID1>`請從命令中移除 。如果要指定一或多個安全群組 ID，請將 `securityGroupIds` 取代為您自己的值。您也可以新增其他 ID。

無論您是否選擇任何安全群組，Amazon EKS 都會建立一個安全群組，以支援您的叢集和 VPC 之間的通訊。Amazon EKS 將此安全群組及您選擇的任何群組與其建立的網路介面相關聯。如需有關 Amazon EKS 建立的叢集安全群組的詳細資訊，請參閱 the section called “安全群組要求”。您可以修改 Amazon EKS 建立的叢集安全群組中的規則。

```
aws eks create-cluster --region region-code --name my-cluster --kubernetes-version
 1.33 \
   --role-arn arn:aws: iam::111122223333:role/myAmazonEKSClusterRole \
   --resources-vpc-config subnetIds=subnet-ExampleID1,subnet-
ExampleID2,securityGroupIds=sg-ExampleID1
```

> **Note**
>
> 您可能會收到錯誤，告知您請求中的其中一個可用區域沒有足夠的容量來建立 Amazon EKS 叢集。如果發生這種情況，錯誤輸出包含的可用區域可支援新的叢集。使用至少兩個位於帳戶的支援可用區域子網路來建立您的叢集。如需詳細資訊，請參閱the section called “容量不足”。

以下是選用設定，如有需要，必須將其新增至上一個命令中。您只能在建立叢集時啟用這些選項，而不能在建立叢集之後啟用。

- 根據預設，EKS 會在叢集建立期間安裝多個聯網附加元件。這包括 Amazon VPC CNI、CoreDNS 和 kube-proxy。

  如果您想要停用這些預設聯網附加元件的安裝，請使用下列參數。這可用於替代 CNIs，例如 Cilium。如需詳細資訊，請參閱 EKS API 參考。

  ```
  aws eks create-cluster --bootstrapSelfManagedAddons false
  ```

- 如果您想指定 Kubernetes 要從哪個 IPv4 無類別域間路由 (CIDR) 區塊中指派服務 IP 地址，您必須將 `--kubernetes-network-config serviceIpv4Cidr=<cidr-block>` 新增至下列命令來指定。

  指定您自己的範圍有助於防止 Kubernetes 服務與對等或連接到 VPC 的其他網路之間發生衝突。在 CIDR 標記法中輸入範圍。例如：`10.2.0.0/16`。

  CIDR 區塊必須符合下列需求：

  - 處於以下範圍之一：`10.0.0.0/8`、`172.16.0.0/12` 或 `192.168.0.0/16`。

- 具有最小尺寸的 /24 和最大尺寸的 /12。
- 與您的 Amazon EKS 資源的 VPC 範圍不重疊。

您只能在使用 IPv4 地址系列時指定此選項，並且只能在建立叢集時指定。如果您未指定此項目，則 Kubernetes 會從 10.100.0.0/16或 172.20.0.0/16 CIDR 區塊指派服務 IP 地址。

- 如果您要建立叢集，並希望叢集將IPv6地址指派給 Pod 和服務，而不是IPv4地址，請將 --kubernetes-network-config ipFamily=ipv6新增至下列命令。

  根據預設，Kubernetes 會將IPv4地址指派給 Pod 和服務。在決定使用 IPv6 系列之前，請確定您已熟悉 VPC 要求和考量、、 和 主題中的所有考量和要求。 the section called “子網需求和注意事項” the section called “安全群組要求” the section called “IPv6”如果您選擇 IPv6 系列，則無法指定 Kubernetes 的地址範圍，以便像針對 IPv4 系列一樣從 指派IPv6服務地址。Kubernetes 會從唯一的本機地址範圍 () 指派服務地址fc00::/7。

2. 佈建叢集需要幾分鐘才能完成。您可以使用下列命令來查詢叢集的狀態。

```
aws eks describe-cluster --region region-code --name my-cluster --query
 "cluster.status"
```

在傳回的輸出為 之前，請勿繼續進行下一個步驟ACTIVE。

3. 繼續 the section called “步驟 3：更新 kubeconfig”

步驟 3：更新 kubeconfig

1. 如果您使用 eksctl 建立叢集，則可以略過此步驟。這是因為 eksctl 已為您完成此步驟。透過向 kubectl config 檔案新增內容，使 kubectl 能夠與您的叢集通訊。如需建立和更新檔案的詳細資訊，請參閱 the section called “使用 kubectl 存取叢集”。

```
aws eks update-kubeconfig --region region-code --name my-cluster
```

範例輸出如下。

```
Added new context arn:aws: eks:region-code:111122223333:cluster/my-cluster to /home/
username/.kube/config
```

2. 透過執行以下命令確認與叢集的通訊。

```
kubectl get svc
```

範例輸出如下。

```
NAME         TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
kubernetes   ClusterIP   10.100.0.1   <none>        443/TCP   28h
```

步驟 4：叢集設定

1. （建議）若要使用一些 Amazon EKS 附加元件，或讓個別 Kubernetes 工作負載具有特定的 AWS Identity and Access Management (IAM) 許可，請為您的叢集建立 IAM OpenID Connect (OIDC) 供應商。您只需要為叢集建立 IAM OIDC 提供商一次。若要進一步了解 Amazon EKS 附加元件，請參閱 the section called “Amazon EKS 附加元件”。若要進一步了解如何將特定 IAM 許可指派給您的工作負載，請參閱 the section called “具有 IRSA 的登入資料”。
2. （建議）在將 Amazon EC2 節點部署至叢集之前，為 Kubernetes 外掛程式的 Amazon VPC CNI 外掛程式設定叢集。預設情況下，外掛程式已隨叢集一起安裝。將 Amazon EC2 節點新增到叢集時，外掛程式會自動部署至您新增的每個 Amazon EC2 節點。外掛程式需要您將下列其中一個 IAM 政策連接至 IAM 角色。如果您的叢集使用 `IPv4` 系列，請使用 AmazonEKS_CNI_Policy 受管 IAM 政策。如果您的叢集使用 `IPv6` 系列，請使用您建立的 IAM 政策。

   您連接政策的 IAM 角色可以是節點 IAM 角色，也可以是僅用於外掛程式的專用角色。我們建議將政策連接至此角色。如需建立角色的詳細資訊，請參閱 the section called “為 IRSA 設定”或 Amazon EKS 節點 IAM 角色。
3. 如果您使用 部署叢集 AWS Management Console，則可以略過此步驟。根據預設， AWS Management Console 部署適用於 Kubernetes、CoreDNS 和 Amazon EKS 附加元件`kube-proxy`的 Amazon VPC CNI 外掛程式。

   如果您使用 `eksctl`或 AWS CLI 部署叢集，則會部署適用於 Kubernetes、CoreDNS 和`kube-proxy`自我管理附加元件的 Amazon VPC CNI 外掛程式。您可以將與叢集一起部署的 Kubernetes、CoreDNS 和`kube-proxy`自我管理附加元件的 Amazon VPC CNI 外掛程式遷移至 Amazon EKS 附加元件。如需詳細資訊，請參閱the section called “Amazon EKS 附加元件”。
4. （選用）如果您尚未這麼做，您可以為叢集啟用 Prometheus 指標。如需詳細資訊，請參閱《Amazon Managed Service for Prometheus 使用者指南》中的建立湊集器。
5. 如果您計劃將工作負載部署到使用 Amazon EBS 磁碟區的叢集，則必須在部署工作負載之前將 Amazon EBS CSI 安裝到您的叢集。

後續步驟

- 建立叢集的 IAM 主體是唯一可以存取叢集的 IAM 主體。將許可授予其他 IAM 主體，讓他們可以存取您的叢集。
- 如果建立叢集的 IAM 主體僅具有先決條件中所參考的最低 IAM 許可，那麼您可能需要為該主體新增其他 Amazon EKS 許可。如需將 Amazon EKS 許可授予 IAM 主體的詳細資訊，請參閱 the section called “IAM 參考”。
- 如果您希望建立叢集的 IAM 主體或任何其他主體在 Amazon EKS 主控台中檢視 Kubernetes 資源，請將必要許可授予實體。
- 如果您希望節點和 IAM 主體從 VPC 內存取您的叢集，請啟用叢集的私有端點。根據預設，公有端點為啟用狀態。如有需要，您可以在啟用私有端點後停用公有端點。如需詳細資訊，請參閱the section called “叢集端點存取”。
- 為叢集啟用密鑰加密。
- 設定叢集的日誌。
- 將節點新增至叢集。

準備進行 Kubernetes 版本升級，並使用叢集洞察對錯誤設定進行疑難排解

Amazon EKS 叢集洞察可提供問題偵測和建議，以解決這些問題，協助您管理叢集。每個 Amazon EKS 叢集都會根據 Amazon EKS 精選洞察清單定期進行自動檢查。這些洞見檢查由 Amazon EKS 完整管理，並提供如何解決任何問題清單的建議。

叢集洞見類型

- 組態洞察：識別 EKS 混合節點設定中可能影響叢集或工作負載功能的錯誤組態。
- 升級洞見：識別可能影響您升級到新 Kubernetes 版本的問題。

考量事項

- 頻率：Amazon EKS 每 24 小時重新整理一次叢集洞察。您無法手動重新整理叢集洞見。如果您修正叢集問題，則需要一些時間才能更新叢集洞察。若要判斷修正是否成功，請將變更部署到洞見檢查的「上次重新整理時間」的時間進行比較。

- 許可：Amazon EKS 會自動為每個 EKS 叢集中的叢集洞察建立叢集存取項目。此項目提供 EKS 檢視叢集相關資訊的許可。EKS 會使用此資訊來產生洞見。如需詳細資訊，請參閱the section called "AmazonEKSClusterInsightsPolicy"。

使用案例

Amazon EKS 中的叢集洞察提供自動檢查，以協助維護 Kubernetes 叢集的運作狀態、可靠性和最佳組態。以下是叢集洞察的主要使用案例，包括升級準備和組態疑難排解。

升級洞見

升級洞見是叢集洞見中特定類型的洞見檢查。這些檢查會傳回與 Kubernetes 版本升級準備度相關的洞見。Amazon EKS 會在每個 EKS 叢集上執行升級洞見檢查。

> **Important**
>
> Amazon EKS 已暫時復原一項功能，在發生特定叢集洞見問題時，會要求您使用`--force`旗標來升級叢集。如需詳細資訊，請參閱 GitHub 上對更新叢集版本強制執行升級洞察的暫時轉返。
> 如需更新叢集的詳細資訊，請參閱 the section called "步驟 3：更新叢集控制平面"。

在更新叢集 Kubernetes 版本之前，您可以使用 Amazon EKS 主控台中可觀測性儀表板的叢集洞察索引標籤。如果您的叢集已識別問題，請檢閱問題並進行適當的修正。問題包括 Amazon EKS 和 Kubernetes 文件的連結。修正問題後，請等待叢集洞察重新整理。如果所有問題都已解決，請更新您的叢集。

Amazon EKS 會傳回與 Kubernetes 版本升級準備度相關的洞見。升級洞見可識別可能影響 Kubernetes 叢集升級的可能問題。這可最大限度地減少管理員為升級做準備所花費的精力，並提高較新 Kubernetes 版本上應用程式的可靠性。Amazon EKS 會根據可能影響問題的 Kubernetes 版本升級清單自動掃描叢集。Amazon EKS 會根據對每個 Kubernetes 版本發行版本所做的變更的審核，經常更新洞見檢查清單。

Amazon EKS 升級洞察有助於加快新版本的測試和驗證流程。它們還允許叢集管理員和應用程式開發人員透過重點關注問題並提供修復建議，利用最新的 Kubernetes 功能。

組態洞察

EKS 叢集洞察會自動使用混合節點掃描 Amazon EKS 叢集，以識別影響 Kubernetes 控制plane-to-webhook通訊、Exec 和 Logs 等 kubectl 命令的組態問題。組態洞見會發現問題並提供修補建議，加速完成正常運作混合節點設定的時間。

開始使用

若要查看已執行的洞見檢查清單，以及 Amazon EKS 已識別的任何相關問題，您可以使用 AWS Management Console、 AWS CLI、 AWS SDKs和 Amazon EKS `ListInsights` API 操作。若要開始使用，請參閱the section called “檢視叢集洞察”。

檢視叢集洞察

Amazon EKS 提供兩種類型的洞見：組態洞見和升級洞見。組態洞察可識別 EKS 混合節點設定中的錯誤組態，這可能會影響叢集或工作負載的功能。升級洞見會識別可能影響您升級至新 Kubernetes 版本的問題。

若要查看已執行的洞見檢查清單，以及 Amazon EKS 已識別的任何相關問題，您可以呼叫 中的查詢 AWS Management Console、 AWS CLI、 AWS SDKs和 Amazon EKS `ListInsights` API 操作。

檢視組態洞察 （主控台）

1. 開啟 Amazon EKS 主控台。
2. 從叢集清單中，選擇您想要查看其相關洞察之 Amazon EKS 叢集的名稱。
3. 選擇監控叢集。
4. 選擇叢集運作狀態索引標籤。
5. 在組態洞察資料表中，您會看到下列資料欄：
   - 名稱：Amazon EKS 對叢集執行的檢查。
   - 洞見狀態 – 狀態為 的洞見`Error`表示組態錯誤可能會影響叢集功能。狀態為 的洞見`Warning`表示組態與記錄的方法不符，但如果您刻意設定叢集功能，該功能可能會運作。狀態為 的洞見`Passing`表示 Amazon EKS 在叢集中找不到與此洞見檢查相關的任何問題。
   - 版本 – 適用的版本。
   - 上次重新整理時間 – 此叢集的洞見狀態上次重新整理的時間。
   - 描述：來自洞察檢查的資訊，其中包括提醒和建議的補救措施。

檢視升級洞見（主控台）

1. 開啟 Amazon EKS 主控台。
2. 從叢集清單中，選擇您想要查看其相關洞察之 Amazon EKS 叢集的名稱。
3. 選擇監控叢集。
4. 選擇升級洞見索引標籤。
5. 在升級洞見表格中，您會看到下列資料欄：
   - 名稱：Amazon EKS 對叢集執行的檢查。
   - 洞見狀態 – 狀態為「錯誤」的洞見通常表示受影響的 Kubernetes 版本是目前叢集版本的 N+1，而狀態為「警告」表示洞見適用於未來的 Kubernetes 版本 N+2 或更高版本。「通過」狀態意味著 Amazon EKS 在此洞察檢查中未發現叢集中存在任何問題。「不明」狀態意味著 Amazon EKS 無法確定叢集是否受到此洞察檢查的影響。
   - 版本 – 洞見檢查是否有可能問題的 Kubernetes 版本。
   - 上次重新整理時間 – 此叢集的洞見狀態上次重新整理的時間。
   - 上次轉換時間 – 此洞見的狀態上次變更的時間。
   - 描述：來自洞察檢查的資訊，其中包括提醒和建議的補救措施。

檢視叢集洞見 (AWS CLI)

1. 確定您想要檢查哪個叢集以取得洞察。以下命令會列出針對指定叢集的洞察。視需要對命令進行下列修改，然後執行修改後的命令：
   - 使用您區域的程式碼取代 AWS *region-code*。
   - 使用您叢集的名稱取代 *my-cluster*。

   ```
   aws eks list-insights --region region-code --cluster-name my-cluster
   ```

   範例輸出如下。

   ```
   {
   "insights":
       [
           {
               "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
               "name": "Deprecated APIs removed in Kubernetes vX.XX",
   ```

```
            "category": "UPGRADE_READINESS",
            "kubernetesVersion": "X.XX",
            "lastRefreshTime": 1734557315.000,
            "lastTransitionTime": 1734557309.000,
            "description": "Checks for usage of deprecated APIs that are scheduled
for removal in Kubernetes vX.XX. Upgrading your cluster before migrating to the
updated APIs supported by vX.XX could cause application impact.",
            "insightStatus":
            {
                "status": "PASSING",
                "reason": "No deprecated API usage detected within the last 30
days.",
            },
        },
        {
            "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
            "name": "Kubelet version skew",
            "category": "UPGRADE_READINESS",
            "kubernetesVersion": "X.XX",
            "lastRefreshTime": 1734557309.000,
            "lastTransitionTime": 1734557309.000,
            "description": "Checks for kubelet versions of worker nodes in the
cluster to see if upgrade would cause non compliance with supported Kubernetes
kubelet version skew policy.",
            "insightStatus":
            {
                "status": "UNKNOWN",
                "reason": "Unable to determine status of node kubelet versions.",
            },
        },
        {
            "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE33333",
            "name": "Deprecated APIs removed in Kubernetes vX.XX",
            "category": "UPGRADE_READINESS",
            "kubernetesVersion": "X.XX",
            "lastRefreshTime": 1734557315.000,
            "lastTransitionTime": 1734557309.000,
            "description": "Checks for usage of deprecated APIs that are scheduled
for removal in Kubernetes vX.XX. Upgrading your cluster before migrating to the
updated APIs supported by vX.XX could cause application impact.",
            "insightStatus":
            {
                "status": "PASSING",
```

```
                "reason": "No deprecated API usage detected within the last 30
days.",
            },
        },
        {
            "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLEaaaaa",
            "name": "Cluster health issues",
            "category": "UPGRADE_READINESS",
            "kubernetesVersion": "X.XX",
            "lastRefreshTime": 1734557314.000,
            "lastTransitionTime": 1734557309.000,
            "description": "Checks for any cluster health issues that prevent
successful upgrade to the next Kubernetes version on EKS.",
            "insightStatus":
            {
                "status": "PASSING",
                "reason": "No cluster health issues detected.",
            },
        },
        {
            "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLEbbbbb",
            "name": "EKS add-on version compatibility",
            "category": "UPGRADE_READINESS",
            "kubernetesVersion": "X.XX",
            "lastRefreshTime": 1734557314.000,
            "lastTransitionTime": 1734557309.000,
            "description": "Checks version of installed EKS add-ons to ensure they
are compatible with the next version of Kubernetes. ",
            "insightStatus": { "status": "PASSING", "reason": "All installed EKS
add-on versions are compatible with next Kubernetes version."},
        },
        {
            "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLEccccc",
            "name": "kube-proxy version skew",
            "category": "UPGRADE_READINESS",
            "kubernetesVersion": "X.XX",
            "lastRefreshTime": 1734557314.000,
            "lastTransitionTime": 1734557309.000,
            "description": "Checks version of kube-proxy in cluster to see if
upgrade would cause non compliance with supported Kubernetes kube-proxy version
skew policy.",
            "insightStatus":
            {
                "status": "PASSING",
```

```
                "reason": "kube-proxy versions match the cluster control plane
 version.",
            },
        },
        {
            "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLEddddd",
            "name": "Deprecated APIs removed in Kubernetes vX.XX",
            "category": "UPGRADE_READINESS",
            "kubernetesVersion": "X.XX",
            "lastRefreshTime": 1734557315.000,
            "lastTransitionTime": 1734557309.000,
            "description": "Checks for usage of deprecated APIs that are scheduled
 for removal in Kubernetes vX.XX. Upgrading your cluster before migrating to the
 updated APIs supported by vX.XX could cause application impact.",
            "insightStatus":
            {
                "status": "PASSING",
                "reason": "No deprecated API usage detected within the last 30
 days.",
            },
        },
    ],
"nextToken": null,
}
```

2. 如需洞察的描述性資訊，請執行以下命令。視需要對命令進行下列修改，然後執行修改後的命令：
   - 以您區域的程式碼取代 AWS *region-code*。
   - 將 *a1b2c3d4-5678-90ab-cdef-EXAMPLE22222* 取代為從列出叢集洞見中擷取的洞見 ID。
   - 使用您叢集的名稱取代 *my-cluster*。

   ```
   aws eks describe-insight --region region-code --id a1b2c3d4-5678-90ab-cdef-
   EXAMPLE22222 --cluster-name my-cluster
   ```

   範例輸出如下。

   ```
   {
     "insight":
       {
         "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
         "name": "Kubelet version skew",
         "category": "UPGRADE_READINESS",
         "kubernetesVersion": "1.27",
   ```

```
      "lastRefreshTime": 1734557309.000,
      "lastTransitionTime": 1734557309.000,
      "description": "Checks for kubelet versions of worker nodes in the cluster
 to see if upgrade would cause non compliance with supported Kubernetes kubelet
 version skew policy.",
      "insightStatus":
        {
          "status": "UNKNOWN",
          "reason": "Unable to determine status of node kubelet versions.",
        },
      "recommendation": "Upgrade your worker nodes to match the Kubernetes version
 of your cluster control plane.",
      "additionalInfo":
        {
          "Kubelet version skew policy": "https://kubernetes.io/releases/version-
skew-policy/#kubelet",
          "Updating a managed node group": "https://docs.aws.amazon.com/eks/latest/
userguide/update-managed-node-group.html",
        },
      "resources": [],
      "categorySpecificSummary":
        { "deprecationDetails": [], "addonCompatibilityDetails": [] },
    },
}
```

將現有叢集更新為新的 Kubernetes 版本

Amazon EKS 中提供新的 Kubernetes 版本時，您可以將 Amazon EKS 叢集更新至最新版本。

> **Important**
>
> 升級叢集後，您就無法降級至先前的版本。更新至新的 Kubernetes 版本之前，建議您檢閱 eks/latest/userguide/kubernetes-versions.html【Understand the Kubernetes version lifecycle on EKS，type="documentation"】 中的資訊，以及本主題中的更新步驟。

新的 Kubernetes 版本有時會加入重大變更。新的 Kubernetes 版本加入了重大變更，因此推薦您在生產叢集執行更新之前，先針對新的 Kubernetes 版本測試您應用程式的行為。若要達成此目標，您可以在移至新的 Kubernetes 版本之前，先建置持續的整合工作流程來測試應用程式行為。

更新程序包括 Amazon EKS 啟動新的 API 伺服器節點，並使用更新的 Kubernetes 版本取代現有的節點。Amazon EKS 會針對這些新節點上的網路流量執行標準基礎設施和整備運作狀態檢查，以確認它們是否如預期般運作。不過，一旦開始叢集升級，您就無法暫停或停止它。如果檢查有失敗，Amazon EKS 會還原基礎設施部署，之前的 Kubernetes 版本仍然保留您的叢集。執行中的應用程式不會受到影響，而且您的叢集永遠不會處於非確定性或無法復原的狀態。Amazon EKS 會定期備份所有受管叢集，並且具備在必要時復原叢集的機制。我們會持續評估並改善 Kubernetes 基礎設施管理程序。

若要更新叢集，Amazon EKS 最多需要五個來自子網的可用 IP 地址，子網是在您建立叢集時指定的。Amazon EKS 會在您指定的任何子網中建立新的叢集彈性網路介面（網路介面）。網路介面可能會在與您現有網路介面所在的子網不同的子網中建立，因此請確認您的安全群組規則針對您建立叢集時指定的任何子網，允許所需的叢集通訊 如果您在建立叢集時指定的任何子網路不存在、沒有足夠的可用 IP 地址，或沒有允許必要叢集通訊的安全群組規則，則更新可能會失敗。

為了確保叢集的 API 伺服器端點始終可存取，Amazon EKS 提供高可用性的 Kubernetes 控制平面，並在更新操作期間執行 API 伺服器執行個體的滾動更新。為了考慮變更支援 Kubernetes API 伺服器端點之 API 伺服器執行個體的 IP 地址，您必須確保您的 API 伺服器用戶端有效管理重新連線。最近版本的 `kubectl`和正式支援的 Kubernetes 用戶端程式庫，以透明的方式執行此重新連線程序。

> **Note**
>
> 若要進一步了解叢集更新的內容，請參閱《EKS 最佳實務指南》中的叢集升級最佳實務。此資源可協助您規劃升級，並了解升級叢集的策略。

Amazon EKS Auto 模式的考量事項

- Amazon EKS Auto Mode 的運算功能可控制節點的 Kubernetes 版本。升級控制平面之後，EKS Auto 模式會開始逐步更新受管節點。EKS Auto Mode 遵守 Pod 中斷預算。
- 您不需要手動升級 Amazon EKS Auto Mode 的功能，包括運算自動擴展、區塊儲存和負載平衡功能。

Summary

Amazon EKS 叢集升級程序的高階摘要如下：

1. 確保您的叢集處於支援升級的狀態。這包括檢查部署至叢集的資源所使用的 Kubernetes APIs，確保叢集沒有任何運作狀態問題。評估叢集的升級準備程度時，您應該使用 Amazon EKS 升級洞察。

2. 將控制平面升級至下一個次要版本（例如，從 1.32 升級至 1.33)。
3. 升級資料平面中的節點，以符合控制平面的節點。
4. 升級在叢集上執行的任何其他應用程式（例如 `cluster-autoscaler`)。
5. 升級 Amazon EKS 提供的附加元件，例如預設包含的附加元件：
   - Amazon VPC CNI 建議的版本
   - CoreDNS 建議版本
   - `kube-proxy` 建議的版本
6. 升級任何與叢集通訊的用戶端（例如 `kubectl`)。

步驟 1：準備升級

將叢集控制平面的 Kubernetes 版本與節點的 Kubernetes 版本進行比較。

- 取得叢集控制平面的 Kubernetes 版本。

```
kubectl version
```

- 取得節點的 Kubernetes 版本。此命令會傳回所有自我管理和受管 Amazon EC2、Fargate 和混合節點。每個 Fargate Pod 都會列為自己的節點。

```
kubectl get nodes
```

將控制平面更新為新的 Kubernetes 版本之前，請確定叢集中受管節點和 Fargate 節點的 Kubernetes 次要版本與控制平面的版本相同。例如，如果您的控制平面正在執行版本 `1.29`，且其中一個節點正在執行版本 `1.28`，則您必須先將節點更新為版本 `1.29`，才能將控制平面更新為 1.30。我們也建議您在更新控制平面之前，將自我管理節點和混合節點更新為與控制平面相同的版本。如需詳細資訊，請參閱the section called “更新”、the section called “更新方法”及the section called “升級混合節點”。如果您有 Fargate 節點的次要版本低於控制平面版本，請先刪除由節點表示的 Pod。接著更新您的控制平面。任何剩餘的 Pod 都會在您重新部署後更新為新版本。

步驟 2：檢閱升級考量事項

Amazon EKS 叢集洞察會根據影響已棄用 Kubernetes API 用量等問題的潛在 Kubernetes 版本升級清單自動掃描叢集。Amazon EKS 會根據 Kubernetes 專案中的變更評估，定期更新要執行的洞見檢查清單。隨著 Amazon EKS 服務以及新版本引入變更，Amazon EKS 也會更新洞見檢查清單。如需詳細資訊，請參閱the section called “叢集洞察”。

檢閱 Kubernetes 文件中的已棄用 API 遷移指南。

檢閱升級洞見

使用 Amazon EKS 升級洞察來識別問題。如需詳細資訊，請參閱the section called “檢視升級洞見（主控台）”。

詳細考量

- 由於 Amazon EKS 執行高可用性的控制平面，因此您一次只能更新一個次要版本。如需此要求的詳細資訊，請參閱 Kubernetes Version and Version Skew Support Policy (Kubernetes 版本和 版本差異支援政策)。假設您目前的叢集版本是版本 `1.28`，並且您想要將其更新為版本 `1.30`。您必須先將您的版本 `1.28` 叢集更新為版本 `1.29`，再將版本 `1.29` 叢集更新為版本 `1.30`。
- 檢閱`kubelet`節點上 Kubernetes `kube-apiserver`和 之間的版本扭曲。
  - 從 Kubernetes 版本 開始`1.28`，最多`kubelet`可以有三個早於 的次要版本`kube-apiserver`。請參閱 Kubernetes upstream version skew policy 一節。
  - 如果受管節點和 Fargate 節點`kubelet`上的 是 Kubernetes 版本`1.25`或更新版本，您可以提前更新叢集最多三個版本，而無需更新`kubelet`版本。例如，如果 `kubelet` 的版本是 `1.25`，在保持 `kubelet` 的版本為 `1.25` 的情況下，您可以將您的 Amazon EKS 叢集版本從 `1.25` 更新為 `1.26`、`1.27` 或 `1.28`
- 作為開始更新前的最佳實務，請確定節點`kubelet`上的 與控制平面的 Kubernetes 版本相同。
- 如果您的叢集使用早於 的 Kubernetes 的 Amazon VPC CNI 外掛程式版本設定`1.8.0`，建議您在更新叢集之前將外掛程式更新至最新版本。若要更新外掛程式，請參閱 the section called “Amazon VPC CNI”。

步驟 3：更新叢集控制平面

> **Important**
>
> Amazon EKS 已暫時復原一項功能，在發生特定叢集洞見問題時，會要求您使用`--force`旗標來升級叢集。如需詳細資訊，請參閱 GitHub 上對更新叢集版本強制執行升級洞察的暫時復原。
> Amazon EKS 會在「上次重新整理時間」的 24 小時後重新整理叢集洞見。您可以比較您解決問題的時間與叢集洞見的「上次重新整理時間」。
> 此外，在解決取代的 API 用量之後，洞見狀態最多可能需要 30 天才能更新。升級洞見一律會在滾動的 30 天時段內尋找已棄用的 API 用量。

您可以使用下列方式提交升級 EKS 控制平面版本的請求：

- eksctl
- AWS 主控台
- AWS CLI

更新叢集 - eksctl

此程序需要 `eksctl` 版本 `0.210.0` 或更新版本。您可使用以下命令檢查您的版本：

```
eksctl version
```

如需有關安裝和更新 `eksctl` 的指示，請參閱 `eksctl` 文件中的 Installation 一節。

更新 Amazon EKS 控制平面的 Kubernetes 版本。使用您的叢集名稱取代 `<cluster-name>`。`<version-number>` 將 取代為您要更新叢集的 Amazon EKS 支援版本編號。如需支援的版本編號清單，請參閱 eks/latest/userguide/kubernetes-versions.html【Amazon EKS 支援的版本，type="documentation"】。

```
eksctl upgrade cluster --name <cluster-name> --version <version-number> --approve
```

此更新需要幾分鐘的時間來完成。

繼續進行the section called “步驟 4：更新叢集元件”。

更新叢集 - AWS 主控台

1. 開啟 Amazon EKS 主控台。
2. 針對您要升級的叢集選擇立即升級。
3. 選取要更新叢集的版本，然後選擇升級。
4. 此更新需要幾分鐘的時間來完成。繼續進行the section called “步驟 4：更新叢集元件”。

更新叢集 - AWS CLI

1. 確認已安裝 AWS CLI 且您已登入。如需詳細資訊，請參閱安裝或更新至最新版本的 AWS CLI。
2. 使用以下 CLI 命令更新您的 Amazon EKS AWS 叢集。取代您要升級之叢集`<region-code>`的 `<cluster-name>`和 。`<version-number>` 將 取代為您要更新叢集的 Amazon EKS 支援版本

編號。如需支援的版本編號清單，請參閱 eks/latest/userguide/kubernetes-versions.html【Amazon EKS 支援的版本，type="documentation"】。

```
aws eks update-cluster-version --name <cluster-name> \
  --kubernetes-version <verion-number> --region <region-code>
```

範例輸出如下。

```
{
    "update": {
        "id": "<update-id>",
        "status": "InProgress",
        "type": "VersionUpdate",
        "params": [
            {
                "type": "Version",
                "value": "<version-number>"
            },
            {
                "type": "PlatformVersion",
                "value": "eks.1"
            }
        ],
[...]
        "errors": []
    }
```

3. 此更新需要幾分鐘的時間來完成。使用下列命令來監控叢集更新的狀態。除了使用相同的 <cluster-name>和 之外<region-code>，請使用上一個命令傳回<update-id>的 。

```
aws eks describe-update --name <cluster-name> \
   --region <region-code> --update-id <update-id>
```

顯示 Successful 狀態時，即表示更新已完成。

4. 繼續進行the section called “步驟 4：更新叢集元件”。

步驟 4：更新叢集元件

1. 叢集更新完成之後，請將您的節點更新至與已更新叢集相同的 Kubernetes 最低版本。如需詳細資訊，請參閱the section called “更新方法”、the section called “更新”及the section called “升級混合

節點”。在 Fargate 上啟動的任何新 Pod 都有符合您叢集kubelet版本的版本。現有的 Fargate Pod 不會變更。

2. (選用) 如果您在更新叢集之前已將 Kubernetes Cluster Autoscaler 部署至叢集，則請將 Cluster Autoscaler 更新為符合您最後更新之 Kubernetes 主要和次要版本的最新版本。

   a. 在 Web 瀏覽器中開啟 Cluster Autoscaler 版本頁面，並尋找符合您叢集 Kubernetes 主要和次要版本的最新 Cluster Autoscaler 版本。例如，如果您叢集的 Kubernetes 版本1.30找到開頭為 的最新 Cluster Autoscaler 版本1.30。記錄該版本的語意版本編號 (例如，1.30.n)，以在下一步中使用。

   b. 使用下列命令，將 Cluster Autoscaler 映像標籤設定為您在前一個步驟中記錄的版本。如有必要，請使用您自己的值取代 X.XX.X。

   ```
   kubectl -n kube-system set image deployment.apps/cluster-autoscaler cluster-
   autoscaler=registry.k8s.io/autoscaling/cluster-autoscaler:vX.XX.X
   ```

3. （僅限具有 GPU 節點的叢集） 如果您的叢集具有具有 GPU 支援的節點群組 （例如 p3.2xlarge)，您必須在叢集上更新適用於 KubernetesDaemonSet 的 NVIDIA 裝置外掛程式。DaemonSet 請先以您想要的 NVIDIA/k8s-device-plugin 版本來取代 <vX.X.X>，再執行下列命令。

   ```
   kubectl apply -f https://raw.githubusercontent.com/NVIDIA/k8s-device-plugin/<vX.X.X>/
   deployments/static/nvidia-device-plugin.yml
   ```

4. 更新 Kubernetes、CoreDNS 和kube-proxy附加元件的 Amazon VPC CNI 外掛程式。建議您將附加元件更新為服務帳戶字符中列出的最低版本。

   - 如果您正在使用 Amazon EKS 附加元件，請在 Amazon EKS 主控台中選取 Clusters (叢集)，然後在左導覽窗格中選取您已更新的叢集名稱。通知會顯示在主控台。它們會通知您每個有可用更新的附加元件有新的可用版本。若要更新附加元件，請選擇 Add-ons (附加元件) 標籤。在具有可用更新之附加元件的其中一個方塊中，選取 Update now (立即更新)，選取可用的版本，然後選取 Update (更新)。
   - 或者，您可以使用 AWS CLI 或 eksctl 來更新附加元件。如需詳細資訊，請參閱the section called “更新 附加元件”。

5. 如有必要，請更新您的 kubectl 版本。您所使用的 kubectl 版本，必須與 Amazon EKS 叢集控制平面的版本差距在一個版本以內。

降級 Amazon EKS 叢集的 Kubernetes 版本

您無法降級 Amazon EKS 叢集的 Kubernetes。相反地，請在先前的 Amazon EKS 版本上建立新的叢集，並遷移工作負載。

刪除叢集

使用 Amazon EKS 叢集完成後，您應該刪除與其相關聯的資源，才不會產生任何不必要的成本。

您可以使用 `eksctl`、 AWS Management Console或 CLI AWS 刪除叢集。

考量事項

- 如果由於已移除叢集建立者而收到錯誤，請參閱此篇文章來解決問題。
- Amazon Managed Service for Prometheus 資源不在叢集生命週期內，需要獨立於叢集進行維護。當您刪除叢集時，請務必同時刪除任何適用的抓取器，以停止適用的成本。如需詳細資訊，請參閱《Amazon Managed Service for Prometheus 使用者指南》中的尋找和刪除抓取器。
- 若要移除連接的叢集，請參閱 the section called “取消註冊叢集”

EKS Auto 模式的考量事項

- 將刪除任何 EKS 自動模式節點，包括 EC2 受管執行個體
- 將刪除所有負載平衡器

如需詳細資訊，請參閱the section called “停用 EKS Auto 模式”。

刪除叢集 (eksctl)

此程序需要 `eksctl` 版本 `0.210.0` 或更新版本。您可使用以下命令檢查您的版本：

```
eksctl version
```

如需有關安裝或更新 `eksctl` 的指示，請參閱 `eksctl` 文件中的 Installation 一節。

1. 列出所有在叢集中執行的服務。

```
kubectl get svc --all-namespaces
```

a. 刪除任何與 EXTERNAL-IP 值相關的服務。這些服務都是由 Elastic Load Balancing 負載平衡器所朝向的，且您必須在 Kubernetes 中刪除它們，以允許負載平衡器與相關資源可正確釋出。將 *service-name* 取代為列出的每個服務的名稱，如所述。

```
kubectl delete svc service-name
```

2. 使用下列命令刪除叢集及其相關聯的節點，將 *prod* 取代為您的叢集名稱。

```
eksctl delete cluster --name prod
```

輸出：

```
[#]  using region region-code
[#]  deleting EKS cluster "prod"
[#]  will delete stack "eksctl-prod-nodegroup-standard-nodes"
[#]  waiting for stack "eksctl-prod-nodegroup-standard-nodes" to get deleted
[#]  will delete stack "eksctl-prod-cluster"
[#]  the following EKS cluster resource(s) for "prod" will be deleted: cluster. If in
 doubt, check CloudFormation console
```

刪除叢集AWS （主控台）

1. 列出所有在叢集中執行的服務。

```
kubectl get svc --all-namespaces
```

2. 刪除任何與 EXTERNAL-IP 值相關的服務。這些服務都是由 Elastic Load Balancing 負載平衡器所朝向的，且您必須在 Kubernetes 中刪除它們，以允許負載平衡器與相關資源可正確釋出。將 *service-name* 取代為列出的每個服務的名稱，如所述。

```
kubectl delete svc service-name
```

3. 刪除所有節點群組和 Fargate 描述檔。

a. 開啟 Amazon EKS 主控台。

b. 在左側導覽窗格中，選擇 Amazon EKS Clusters (叢集)，然後在叢集的標籤式清單中，選擇您要刪除的叢集名稱。

c. 選擇 Compute (運算) 索引標籤，然後選取要刪除的節點群組。選擇 Delete (刪除)，輸入節點群組的名稱，然後選擇 Delete (刪除)。刪除叢集中的所有節點群組。

> **Note**
>
> 列出的節點群組僅為受管節點群組。

d. 選擇要刪除的 Fargate Profile (Fargate 描述檔)，選取 Delete (刪除)，輸入描述檔的名稱，然後選擇 Delete (刪除)。刪除叢集中的所有 Fargate 描述檔。

4. 刪除所有自我管理的 node AWS CloudFormation 堆疊。

a. 開啟 AWS CloudFormation 主控台。

b. 選擇要刪除的節點堆疊，然後選擇 Delete (刪除)。

c. 在 Delete stack (刪除堆疊) 確認對話方塊中，選擇 Delete stack (刪除堆疊)。刪除叢集中的所有自我管理節點堆疊。

5. 刪除叢集。

a. 開啟 Amazon EKS 主控台。

b. 選擇要刪除的叢集並選擇 Delete (刪除)。

c. 在刪除叢集確認畫面上，選擇 Delete (刪除)。

6. （選用）刪除 VPC AWS CloudFormation 堆疊。

a. 開啟 AWS CloudFormation 主控台。

b. 選取要刪除的 VPC 堆疊，然後選擇 Delete (刪除)。

c. 在 Delete stack (s刪除堆疊) 確認對話方塊中，選擇 Delete stack (刪除堆疊)。

刪除叢集 (AWS CLI)

1. 列出所有在叢集中執行的服務。

```
kubectl get svc --all-namespaces
```

2. 刪除任何與 EXTERNAL-IP 值相關的服務。這些服務都是由 Elastic Load Balancing 負載平衡器所朝向的，且您必須在 Kubernetes 中刪除它們，以允許負載平衡器與相關資源可正確釋出。將 *service-name* 取代為列出的每個服務的名稱，如所述。

```
kubectl delete svc service-name
```

3. 刪除所有節點群組和 Fargate 描述檔。

   a. 使用下列命令列出叢集中的節點群組。

   ```
   aws eks list-nodegroups --cluster-name my-cluster
   ```

   > **Note**
   >
   > 列出的節點群組僅為受管節點群組。

   b. 使用下列命令來刪除每個節點群組。刪除叢集中的所有節點群組。

   ```
   aws eks delete-nodegroup --nodegroup-name my-nodegroup --cluster-name my-cluster
   ```

   c. 使用下列命令列出您叢集中的 Fargate 描述檔。

   ```
   aws eks list-fargate-profiles --cluster-name my-cluster
   ```

   d. 使用下列命令來刪除每個 Fargate 描述檔。刪除叢集中的所有 Fargate 描述檔。

   ```
   aws eks delete-fargate-profile --fargate-profile-name my-fargate-profile --
   cluster-name my-cluster
   ```

4. 刪除所有自我管理的 node AWS CloudFormation 堆疊。

   a. 使用下列命令列出您的可用 AWS CloudFormation 堆疊。在產生輸出中尋找節點範本名稱。

   ```
   aws cloudformation list-stacks --query "StackSummaries[].StackName"
   ```

   b. 使用以下命令刪除每個節點堆疊，以節點堆疊名稱取代 node-*stack*。刪除叢集中的所有自我管理節點堆疊。

   ```
   aws cloudformation delete-stack --stack-name node-stack
   ```

5. 使用以下命令，將 *my-cluster* 替換為您的叢集名稱以刪除叢集。

   ```
   aws eks delete-cluster --name my-cluster
   ```

6. （選用）刪除 VPC AWS CloudFormation 堆疊。

   a. 使用下列命令列出您的可用 AWS CloudFormation 堆疊。在產生輸出中尋找 VPC 範本名稱。

```
aws cloudformation list-stacks --query "StackSummaries[].StackName"
```

b. 使用以下命令，將 *my-vpc-stack* 替換為您的 VPC 堆疊名稱以刪除 VPC 堆疊。

```
aws cloudformation delete-stack --stack-name my-vpc-stack
```

保護 EKS 叢集免於意外刪除

不小心刪除 EKS 叢集可能會影響 Kubernetes 叢集操作。

您現在可以保護 EKS 叢集免於意外刪除。如果您在叢集上啟用刪除保護，您必須先停用刪除保護，才能刪除叢集。

刪除保護的目的是防止意外。您應該謹慎限制誰有權刪除叢集。

如果您嘗試刪除已啟用刪除保護的作用中叢集，您將會收到 `InvalidRequestException` 。

Important

如果您在叢集上啟用刪除保護，您必須同時擁有 UpdateClusterConfig 和 DeleteCluster IAM 許可，才能先移除刪除保護，最後刪除叢集。

Note

如果叢集狀態正在建立、失敗或刪除，即使已啟用刪除保護，您也可以刪除叢集。

啟用現有叢集的刪除保護

您只能在處於作用中狀態的叢集上執行此操作。

```
aws eks update-cluster-config --deletion-protection
```

停用現有叢集的刪除保護

```
aws eks update-cluster-config --no-deletion-protection
```

叢集 API 伺服器端點

本主題可協助您為 Amazon EKS 叢集的 Kubernetes API 伺服器端點和限制啟用私有存取，或完全停用網際網路的公有存取。

建立新的叢集時，Amazon EKS 會為您用來與叢集通訊的受管 Kubernetes API 伺服器建立端點 (使用 Kubernetes 管理工具，例如 `kubectl`)。根據預設，此 API 伺服器端點對網際網路是公有的，並且會使用 AWS Identity and Access Management (IAM) 和原生 Kubernetes 角色型存取控制 (RBAC) 的組合來保護對 API 伺服器的存取。此端點稱為叢集公有端點。還有叢集私有端點。如需叢集私有端點的詳細資訊，請參閱下一節the section called “叢集私有端點”。

IPv6 叢集端點格式

EKS 會以下列格式為 2024 年 10 月之後建立的新IPv6叢集建立唯一的雙堆疊端點。IPv6 叢集是您在叢集的 IP 系列 (`ipFamily`) 設定IPv6中選取的叢集。

Example

AWS

EKS 叢集公有/私有端點： `eks-cluster.region.api.aws`

AWS GovCloud (US)

EKS 叢集公有/私有端點： `eks-cluster.region.api.aws`

Amazon Web Services in China

EKS 叢集公有/私有端點： `eks-cluster.region.api.amazonwebservices.com.cn`

> **Note**
>
> 雙堆疊叢集端點於 2024 年 10 月推出。如需IPv6叢集的詳細資訊，請參閱 the section called “IPv6”。在 2024 年 10 月之前建立的叢集，請改用下列端點格式。

IPv4 叢集端點格式

EKS 會針對在叢集的 IP 系列 (ipFamily) 設定IPv4中選取的每個叢集，以下列格式建立唯一的端點：

Example

AWS

EKS 叢集公有/私有端點 `eks-cluster.region.eks.amazonaws.com`

AWS GovCloud (US)

EKS 叢集公有/私有端點 `eks-cluster.region.eks.amazonaws.com`

Amazon Web Services in China

EKS 叢集公有/私有端點 `eks-cluster.region.amazonwebservices.com.cn`

> **Note**
>
> 在 2024 年 10 月之前，IPv6叢集也會使用此端點格式。對於這些叢集，公有端點和私有端點都只有從此端點解析IPv4的地址。

叢集私有端點

您可啟用 Kubernetes API 伺服器的私有存取，讓節點和 API 伺服器間的所有通訊都不會離開 VPC。您可以限制可以從網際網路存取 API 伺服器的 IP 地址，或完全停用對 API 伺服器的網際網路存取。

> **Note**
>
> 由於此端點適用於 Kubernetes API 伺服器，而不是與 AWS API 通訊的傳統 AWS PrivateLink 端點，因此不會在 Amazon VPC 主控台中顯示為端點。

當您為叢集啟用端點私有存取時，Amazon EKS 會代表您建立 Route 53 私有託管區域，並將其與您叢集的 VPC 建立關聯。此私有託管區域由 Amazon EKS 管理，不會出現在您帳戶的 Route 53 資源中。若要讓私有託管區域正確將流量路由到 API 伺服器，您的 VPC 必須將 `enableDnsHostnames` 和 `enableDnsSupport` 設為 `true`，且 VPC 設定的 DHCP 選項必須在網域名稱伺服器清單中包含 `AmazonProvidedDNS`。如需詳細資訊，請參閱《Amazon VPC 使用者指南》中的更新 VPC 的 DNS 支援。

建立新的叢集時，可定義您 API 伺服器端點的存取要求，您也可隨時更新叢集的 API 伺服器端點存取。

修改叢集端點存取

使用本節所述程序來修改現有叢集的端點存取。下表說明支援的 API 伺服器端點存取組合及其相關的行為。

端點公有存取	端點私有存取	Behavior (行為)
已啟用	已停用	• 這是新 Amazon EKS 叢集的預設行為。 • 源自叢集 VPC （例如控制平面通訊的節點） 的 Kubernetes API 請求會離開 VPC，但不會離開 Amazon 的網路。 • 您的叢集 API 伺服器可從網際網路上存取。您可以選擇性地限制可存取公有端點的 CIDR 區塊。如果您限制對特定 CIDR 區塊的存取，建議您也啟用私有端點，或確保您指定的 CIDR 區塊包含節點和 Fargate Pod （如果您使用它們） 從中存取公有端點的地址。
已啟用	已啟用	• 叢集 VPC 內的 Kubernetes API 請求 （例如控制平面通訊的節點） 會使用私有 VPC 端點。 • 您的叢集 API 伺服器可從網際網路上存取。您可以選擇性地限制可存取公有端點的 CIDR 區塊。 • 如果您搭配 Amazon EKS 叢集使用混合節點，不建議同時啟用公有和私有叢集端點存取。由於您的混合節點是在 VPC 外部執行，因此它們會將叢集端點解析為公有 IP 地址。對於具有混合節點的叢集，建議使用公有或私有叢集端點存取。
已停用	已啟用	• 到叢集 API 伺服器的所有流量都必須來自叢集的 VPC 或連線的網路。 • 您的 API 伺服器無法從網際網路上公有存取。任何 `kubectl` 命令都必須來自 VPC 或連接的網路內。如需連接選項，請參閱 the section called “存取僅限私有 API 伺服器”。 • 叢集的 API 伺服器端點由公有 DNS 伺服器解析為來自 VPC 的私有 IP 地址。在過去，端點只能從 VPC 內解析。

端點公有存取	端點私有存取	Behavior (行為)
		如果您的端點未針對現有叢集解析為 VPC 內的私有 IP 地址，您可以： • 啟用公有存取，然後再次停用它。您只需對叢集執行一次，從此以後，端點就會解析為私有 IP 地址。 • 更新您的叢集。

公有端點 (**IPv6** 叢集） 中的 CIDR 區塊

您可以將 IPv6和 IPv4 CIDR 區塊新增至 IPv6叢集的公有端點，因為公有端點是雙堆疊。這僅適用於新叢集，並將 ipFamily設定為IPv6您在 2024 年 10 月或之後建立的 。您可以透過新的端點網域名稱 來識別這些叢集api.aws。

公有端點 (**IPv4** 叢集） 中的 CIDR 區塊

您可以將 IPv4 CIDR 區塊新增至 IPv4叢集的公有端點。您無法將 IPv6 CIDR 區塊新增至IPv4叢集的公有端點。如果您嘗試，EKS 會傳回下列錯誤訊息： The following CIDRs are invalid in publicAccessCidrs

公有端點中的 CIDR 區塊 (**IPv6** 叢集在 2024 年 10 月之前建立）

您可以將 IPv4 CIDR 區塊新增至您在 2024 年 10 月之前建立之舊IPv6叢集的公有端點。您可以依eks.amazonaws.com端點識別這些叢集。您無法將 IPv6 CIDR 區塊新增至您在 2024 年 10 月之前建立的這些舊IPv6叢集的公有端點。如果您嘗試，EKS 會傳回下列錯誤訊息： The following CIDRs are invalid in publicAccessCidrs

存取僅限私有 API 伺服器

如果您已停用叢集 Kubernetes API 伺服器端點的公有存取，您只能從 VPC 或連線網路中存取 API 伺服器。以下是存取 Kubernetes API 伺服器端點的幾種可能方法：

連線的網路

使用 AWS 傳輸閘道或其他連線選項，將您的網路連線到 VPC，然後使用連線網路中的電腦。您必須確認 Amazon EKS 控制平面安全群組包含的規則允許連接埠 443 上來自連接網路的傳入流量。

Amazon EC2 堡壘主機

您可以在叢集 VPC 中的公有子網路中啟動 Amazon EC2 執行個體，然後透過 SSH 登入該執行個體以執行`kubectl`命令。如需詳細資訊，請參閱 AWS上的 Linux 堡壘主機。您必須確認 Amazon EKS 控制平面安全群組包含的規則允許連接埠 443 上來自堡壘主機的傳入流量。如需詳細資訊，請參閱the section called “安全群組要求”。

當您`kubectl`為堡壘主機設定 時，請務必使用已映射至叢集 RBAC 組態的 AWS 登入資料，或在移除端點公開存取之前，將堡壘將使用的 IAM 主體新增至 RBAC 組態。如需詳細資訊，請參閱the section called “Kubernetes API 存取”及the section called “未經授權或存取遭拒 (`kubectl`)”。

AWS Cloud9 IDE

AWS Cloud9 是以雲端為基礎的整合式開發環境 (IDE)，可讓您使用瀏覽器撰寫、執行和偵錯程式碼。您可以在叢集的 VPC 中建立 AWS Cloud9 IDE，並使用 IDE 與您的叢集通訊。如需詳細資訊，請參閱在 AWS Cloud9 中建立環境。您必須確認 Amazon EKS 控制平面安全群組包含的規則允許連接埠 443 上來自 IDE 安全群組的傳入流量。如需詳細資訊，請參閱the section called “安全群組要求”。

當您`kubectl`為 AWS Cloud9 IDE 設定 時，請務必使用已映射至叢集 RBAC 組態的 AWS 登入資料，或在移除端點公開存取之前，將 IDE 將使用的 IAM 主體新增至 RBAC 組態。如需詳細資訊，請參閱the section called “Kubernetes API 存取”及the section called “未經授權或存取遭拒 (`kubectl`)”。

在 GitHub 上編輯此頁面

設定叢集 API 伺服器端點的網路存取

您可以在下列各節中使用 AWS Management Console 或 CLI AWS 修改叢集 API 伺服器端點存取。

設定端點存取 - AWS 主控台

1. 開啟 Amazon EKS 主控台。
2. 選擇叢集名稱以顯示您叢集的資訊。
3. 選擇聯網索引標籤，然後選擇管理端點存取。
4. 針對私有存取，選擇是否啟用或停用叢集 Kubernetes API 伺服器端點的私有存取。如果您啟用私有存取，則來自叢集 VPC 內的 Kubernetes API 請求會使用私有 VPC 端點。您必須啟用私有存取，才能停用公有存取。

5. 針對公有存取，選擇是否啟用或停用叢集 Kubernetes API 伺服器端點的公有存取。如果您停用公有存取，叢集的 Kubernetes API 伺服器只能接收來自叢集 VPC 內的請求。
6. （選用）如果您已啟用公有存取，您可以指定哪些來自網際網路的地址可以與公有端點通訊。選取 Advanced Settings (進階設定)。輸入 CIDR 區塊，例如 *203.0.113.5/32*。區塊不能包含預留地址。您可以透過選取 Add source (新增來源) 來輸入其他區塊。您可以指定的 CIDR 區塊有數量上限。如需詳細資訊，請參閱the section called “Service Quotas”。如果您未指定區塊，則公有 API 伺服器端點會收到來自雙堆疊IPv6叢集 IPv4(0.0.0.0/0) 和額外 IPv6(::/0) 之所有 IP 地址的請求。如果您使用 CIDR 區塊限制對公有端點的存取，我們建議您也啟用私有端點存取，以便節點和 Fargate Pods（如果您使用它們）可以與叢集通訊。若不啟用私有端點，您的公有存取端點 CIDR 來源必須包含來自 VPC 的輸出來源。例如，如果您的私有子網路中有節點，透過 NAT 閘道與網際網路通訊，則您必須將 NAT 閘道的對外 IP 地址新增至公有端點上的白名單 CIDR 區塊。
7. 選擇 Update (更新) 以完成操作。

設定端點存取 - AWS CLI

使用 CLI AWS 版本 1.27.160 或更新版本完成下列步驟。您可以使用 aws --version 來檢查您的目前版本。若要安裝或升級 AWS CLI，請參閱安裝 AWS CLI。

1. 使用以下 CLI AWS 命令更新您的叢集 API 伺服器端點存取權。替換叢集名稱和所需的端點存取值。如果您設定了 endpointPublicAccess=true，則可以 (選擇性地) 為 publicAccessCidrs 輸入單一 CIDR 區塊或以逗號分隔的 CIDR 區塊清單。區塊不能包含預留地址。如果您指定 CIDR 區塊，則公有 API 伺服器端點只會接收來自所列出區塊的要求。您可以指定的 CIDR 區塊有數量上限。如需詳細資訊，請參閱the section called “Service Quotas”。如果您使用 CIDR 區塊限制對公有端點的存取，則建議您也啟用私有端點存取，以便節點和 Fargate Pods (如果您使用它們) 可以與叢集通訊。若不啟用私有端點，您的公有存取端點 CIDR 來源必須包含來自 VPC 的輸出來源。例如，如果您的私有子網路中有節點，透過 NAT 閘道與網際網路通訊，則您必須將 NAT 閘道的對外 IP 地址新增至公有端點上的白名單 CIDR 區塊。如果您未指定 CIDR 區塊，則公有 API 伺服器端點會收到來自所有 (0.0.0.0/0) IP 地址的請求，以及雙堆疊IPv6叢集的其他 IPv6(::/0) 請求。

> **Note**
>
> 下列命令會啟用 API 伺服器端點的私有存取以及來自單一 IP 地址的公有存取。使用單一 CIDR 區塊或您想要限制網路存取的 CIDR 區塊清單取代 *203.0.113.5/32*。

```
aws eks update-cluster-config \
    --region region-code \
    --name my-cluster \
    --resources-vpc-config
 endpointPublicAccess=true,publicAccessCidrs="203.0.113.5/32",endpointPrivateAccess=true
```

範例輸出如下。

```
{
    "update": {
        "id": "e6f0905f-a5d4-4a2a-8c49-EXAMPLE00000",
        "status": "InProgress",
        "type": "EndpointAccessUpdate",
        "params": [
            {
                "type": "EndpointPublicAccess",
                "value": "true"
            },
            {
                "type": "EndpointPrivateAccess",
                "value": "true"
            },
            {
                "type": "publicAccessCidrs",
                "value": "[\"203.0.113.5/32\"]"
            }
        ],
        "createdAt": 1576874258.137,
        "errors": []
    }
}
```

2. 使用叢集名稱和前述命令傳回的更新 ID，藉由以下命令監控端點存取的更新狀態，當狀態顯示為 Successful，您的更新就完成了。

```
aws eks describe-update \
    --region region-code \
    --name my-cluster \
    --update-id e6f0905f-a5d4-4a2a-8c49-EXAMPLE00000
```

範例輸出如下。

```
{
    "update": {
        "id": "e6f0905f-a5d4-4a2a-8c49-EXAMPLE00000",
        "status": "Successful",
        "type": "EndpointAccessUpdate",
        "params": [
            {
                "type": "EndpointPublicAccess",
                "value": "true"
            },
            {
                "type": "EndpointPrivateAccess",
                "value": "true"
            },
            {
                "type": "publicAccessCidrs",
                "value": "[\"203.0.113.5/32\"]"
            }
        ],
        "createdAt": 1576874258.137,
        "errors": []
    }
}
```

在 GitHub 上編輯此頁面

在 EKS 叢集上部署 Windows 節點

了解如何啟用和管理 Amazon EKS 叢集的 Windows 支援，以搭配 Linux 容器執行 Windows 容器。

考量事項

部署 Windows 工作節點之前，請注意下列考量事項。

- EKS Auto Mode 不支援 Windows 節點
- 您可以透過 `HostProcess` Pod 在 Windows 節點上使用主機聯網。如需詳細資訊，請參閱 Kubernetes 文件中的建立 Windows HostProcessPod。

- Amazon EKS 叢集必須包含一或多個 Linux 或 Fargate 節點，才能執行僅在 Linux 上執行的核心系統 Pod，例如 CoreDNS。
- kubelet 和 kube-proxy事件日誌會重新導向至EKS Windows事件日誌，並設定為 200 MB 限制。
- 您無法對 Linux 和 Windows 節點使用相同的 IAM 角色。
- 您無法使用指派安全群組給在 Windows 節點上執行 Pod 的個別 Pod。
- 您無法搭配 Windows 節點使用自訂聯網。
- 您無法IPv6搭配 Windows 節點使用 。
- Windows 節點支援每個節點一個彈性網路介面。根據預設，每個 Windows 節點可執行的 Pod 數量等於節點執行個體類型每個彈性網路界面可用的 IP 地址數量減去一。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的每個網路介面 IP 地址。
- 在 Amazon EKS 叢集中，具有負載平衡器的單一服務最多可支援 1024 個後端 Pod。每個 Pod 都有自己的唯一 IP 地址。從作業系統組建 開始的 Windows Server 更新之後，上述 64 個 Pod 的限制不再適用。 17763.2746
- Fargate 上的 Amazon EKS Pod 不支援 Windows 容器。
- 您無法將 Amazon EKS 混合節點與 Windows 搭配使用，做為主機的作業系統。
- 您無法從 Pod vpc-resource-controller 擷取日誌。您先前在將控制器部署到資料平面時可以這麼做。
- 在將 IPv4 地址指派給新的 Pod 之前，有一段冷卻期間。這可以防止流量因 kube-proxy 規則過時而流動到具有相同 IPv4 地址的較舊 Pod。
- 在 GitHub 上管理控制器的來源。若要對控制器做出貢獻或提出問題，請造訪 GitHub 上的專案。
- 為 Windows 受管節點群組指定自訂 AMI ID 時，請將 eks:kube-proxy-windows新增至您的 AWS IAM Authenticator 組態映射。如需詳細資訊，請參閱the section called “指定 AMI ID 時的限制和條件”。
- 如果保留可用的 IPv4 地址對您的子網路至關重要，請參閱 EKS 最佳實務指南 - Windows 網路 IP 地址管理以取得指引。
- EKS 存取項目的考量事項
  - 用於 Windows 節點的存取項目需要 的類型EC2_WINDOWS。如需詳細資訊，請參閱the section called “建立存取項目”。

    若要建立 Windows 節點的存取項目：

```
aws eks create-access-entry --cluster-name my-cluster --principal-arn arn:aws:
 iam::111122223333:role/<role-name> --type EC2_Windows
```

先決條件

- 現有的叢集。
- 您的叢集必須至少有一個 （我們建議至少兩個） Linux 節點或 Fargate Pod 來執行 CoreDNS。如果您啟用舊版 Windows 支援，則必須使用 Linux 節點 （無法使用 Fargate Pod) 來執行 CoreDNS。
- 現有的 Amazon EKS 叢集 IAM 角色。

啟用 Windows 支援

1. 如果您的叢集中沒有 Amazon Linux 節點，並針對 Pod 使用安全群組，請跳到下一個步驟。否則，請確認 AmazonEKSVPCResourceController 受管政策會連接至叢集角色。使用叢集角色名稱取代 *eksClusterRole*。

   ```
   aws iam list-attached-role-policies --role-name eksClusterRole
   ```

   範例輸出如下。

   ```
   {
       "AttachedPolicies": [
           {
               "PolicyName": "AmazonEKSClusterPolicy",
               "PolicyArn": "arn:aws: iam::aws:policy/AmazonEKSClusterPolicy"
           },
           {
               "PolicyName": "AmazonEKSVPCResourceController",
               "PolicyArn": "arn:aws: iam::aws:policy/AmazonEKSVPCResourceController"
           }
       ]
   }
   ```

   如果已連接政策 (如上一個輸出所示)，請略過下一個步驟。

2. 將 AmazonEKSVPCResourceController 受管政策連接至您的 Amazon EKS 叢集 IAM 角色。使用叢集角色名稱取代 *eksClusterRole*。

```
aws iam attach-role-policy \
  --role-name eksClusterRole \
  --policy-arn arn:aws: iam::aws:policy/AmazonEKSVPCResourceController
```

3. 更新 VPC CNI ConfigMap 以啟用 Windows IPAM。請注意，如果您的叢集使用 Helm Chart 或 Amazon EKS 附加元件安裝 VPC CNI，則可能無法直接修改 ConfigMap。如需設定 Amazon EKS 附加元件的資訊，請參閱 the section called “您可以自訂的欄位”。

   a. 使用下列內容，建立名為 *vpc-resource-controller-configmap.yaml* 的檔案。

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: amazon-vpc-cni
  namespace: kube-system
data:
  enable-windows-ipam: "true"
```

   b. 將 ConfigMap 套用至您的叢集。

```
kubectl apply -f vpc-resource-controller-configmap.yaml
```

4. 如果您的叢集已設定身分驗證模式來啟用 aws-auth configmap：

   - 確認您的 aws-authConfigMap包含 Windows 節點執行個體角色的映射，以包含 RBAC eks:kube-proxy-windows 許可群組。您可以透過執行以下命令來驗證。

```
kubectl get configmap aws-auth -n kube-system -o yaml
```

     範例輸出如下。

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: aws-auth
  namespace: kube-system
data:
  mapRoles: |
    - groups:
      - system:bootstrappers
      - system:nodes
```

```
      - eks:kube-proxy-windows # This group is required for Windows DNS resolution
 to work
      rolearn: arn:aws: iam::111122223333:role/eksNodeRole
      username: system:node:{{EC2PrivateDNSName}}
[...]
```

您應該會在群組下看到 eks:kube-proxy-windows 列出。如果未指定群組，您需要更新 ConfigMap或建立它以包含必要的群組。如需 aws-auth ConfigMap 的詳細資訊，請參閱 the section called “將 aws-authConfigMap 套用至您的叢集”。

5. 如果您的叢集已將身分驗證模式設定為停用 aws-auth configmap，則可以使用 EKS 存取項目。建立新的節點角色以搭配 Windows 執行個體使用，EKS 會自動建立 類型的存取項目EC2_WINDOWS。

部署 Windows Pod

當您將 Pod 部署到叢集時，您需要指定它們在執行混合節點類型時使用的作業系統。

對於 Linux Pod，請在資訊清單中使用下列節點選取器文字。

```
nodeSelector:
        kubernetes.io/os: linux
        kubernetes.io/arch: amd64
```

對於 Windows Pod，請在資訊清單中使用下列節點選取器文字。

```
nodeSelector:
        kubernetes.io/os: windows
        kubernetes.io/arch: amd64
```

您可以部署範例應用程式來查看正在使用的節點選擇器。

在 Windows 節點上支援更高的 Pod 密度

在 Amazon EKS 中，每個 Pod 都會從您的 VPC 配置一個IPv4地址。因此，您可以部署到節點的 Pod 數量受到可用 IP 地址的限制，即使有足夠的資源可在節點上執行更多 Pod。由於 Windows 節點僅支援一個彈性網路介面，因此依預設，Windows 節點上可用 IP 地址的數量上限為：

```
Number of private IPv4 addresses for each interface on the node - 1
```

一個 IP 地址用作網路介面的主要 IP 地址，因此無法配置給 Pod。

您可以啟用 IP 字首委派，在 Windows 節點上啟用更高的 Pod 密度。此功能可讓您將 /28 IPv4 字首指派給主要網路介面，而不是指派次要 IPv4 位址。而指派 IP 字首會將節點上可用 IPv4 地址的數量上限增加到：

```
(Number of private IPv4 addresses assigned to the interface attached to the node - 1) *
 16
```

有了這個大量的可用 IP 地址，可用的 IP 地址不應限制您擴展節點上 Pod 數量的能力。如需詳細資訊，請參閱the section called “增加 IP 地址”。

停用 Windows 支援

1. 如果您的叢集包含 Amazon Linux 節點，而且您使用 Pod 的安全群組搭配這些節點，請略過此步驟。

   從叢集角色中移除 `AmazonVPCResourceController` 受管 IAM 政策。將 *`eksClusterRole`* 取代為您的叢集角色名稱。

   ```
   aws iam detach-role-policy \
       --role-name eksClusterRole \
       --policy-arn arn:aws: iam::aws:policy/AmazonEKSVPCResourceController
   ```

2. 停用 `amazon-vpc-cni` ConfigMap 中的 Windows IPAM。

   ```
   kubectl patch configmap/amazon-vpc-cni \
                       -n kube-system \
                       --type merge \
                       -p '{"data":{"enable-windows-ipam":"false"}}'
   ```

部署網際網路存取受限的私有叢集

本主題說明如何部署部署在 AWS 雲端上，但沒有傳出網際網路存取權的 Amazon EKS 叢集。如果您在 AWS Outposts 上有本機叢集，請參閱 the section called “節點”，而非本主題。

如果您不熟悉 Amazon EKS 網路，請參閱為 Amazon EKS 工作者節點取消對叢集網路的迷宮化。如果您的叢集沒有傳出網際網路存取，則必須符合下列要求：

- 您的叢集必須從 VPC 中的容器登錄檔提取映像。您可以在 VPC 中建立 Amazon Elastic Container Registry，並將容器映像複製到其中，以供節點提取。如需詳細資訊，請參閱the section called “將映像複製到儲存庫”。
- 您的叢集必須啟用端點私有存取。這對節點向叢集端點註冊而言是必要的。端點公有存取權限並非必要。如需詳細資訊，請參閱the section called “叢集端點存取”。
- 自我管理的 Linux 和 Windows 節點在啟動之前，必須包含下列引導引數。這些引數會繞過 Amazon EKS 自我檢查，不需要從 VPC 內存取 Amazon EKS API。

  a. 使用以下命令判斷叢集端點的值。使用您叢集的名稱取代 *my-cluster*。

  ```
  aws eks describe-cluster --name my-cluster --query cluster.endpoint --output text
  ```

  範例輸出如下。

  ```
  https://EXAMPLE108C897D9B2F1B21D5EXAMPLE.sk1.region-code.eks.amazonaws.com
  ```

  b. 使用下列命令判斷叢集憑證授權單位的值。使用您叢集的名稱取代 *my-cluster*。

  ```
  aws eks describe-cluster --name my-cluster --query cluster.certificateAuthority --
  output text
  ```

  傳回的輸出是長字串。

  c. 將下列命令中的 *cluster-endpoint* 和 *certificate-authority* 取代為先前命令輸出中傳回的值。如需有關在啟動自我管理節點時指定引導參數的詳細資訊，請參閱 the section called “Amazon Linux” 和 the section called “Windows”。

  - 對於 Linux 節點：

  ```
  --apiserver-endpoint cluster-endpoint --b64-cluster-ca certificate-authority
  ```

  如需其他引數，請參閱 GitHub 上的 Bootstrap 指令碼。

  - 對於 Windows 節點：

  > **Note**
  >
  > 如果您使用的是自訂服務 CIDR，則需要使用 `-ServiceCIDR` 參數來指定它。否則，叢集中 Pod 的 DNS 解析將會失敗。

```
-APIServerEndpoint cluster-endpoint -Base64ClusterCA certificate-authority
```

如需其他引數，請參閱 the section called “引導指令碼組態參數”。

- 叢集的 aws-auth ConfigMap 必須在 VPC 內建立。若要進一步了解如何建立項目並將項目新增至 aws-auth ConfigMap，請在終端機中輸入 eksctl create iamidentitymapping --help。如果您的伺服器上ConfigMap不存在 ， eksctl會在您使用 命令新增身分映射時建立它。
- 為服務帳戶設定 IAM 角色的 Pod 會從 AWS 安全字符服務 (AWS STS) API 呼叫取得憑證。如果沒有傳出網際網路存取，您必須在 VPC 中建立和使用 AWS STS VPC 端點。大多數 AWS v1 SDKs 預設使用全域 AWS STS 端點 (sts.amazonaws.com)，這不使用 AWS STS VPC 端點。若要使用 AWS STS VPC 端點，您可能需要設定 SDK 以使用區域 AWS STS 端點 (sts.*region-code*.amazonaws.com)。如需詳細資訊，請參閱the section called “STS 端點”。
- 您叢集的 VPC 子網路必須具有 VPC 介面端點，以供您的 Pod 需要存取的任何 AWS 服務使用。如需詳細資訊，請參閱使用介面 VPC 端點存取 AWS 服務。下表列出了一些常用的服務和端點。如需端點的完整清單，請參閱《 AWSAWS PrivateLink 指南》中的與PrivateLink 整合的 服務。 AWS PrivateLink

  我們建議您為 VPC 端點啟用私有 DNS 名稱，這樣工作負載就可以繼續使用公有 AWS 服務端點，而不會發生問題。

服務	端點
Amazon EC2	com.amazonaws.*region-code* .ec2
Amazon Elastic Container Registry (用於提取容器映像)	com.amazonaws.*region-code* .ecr.api、com.amazonaws.*region-code* .ecr.dkr 和 com.amazonaws.*region-code* .s3
Application Load Balancer 與 Network Load Balancer	com.amazonaws.*region-code* .elasticloadbalancing
AWS X-Ray	com.amazonaws.*region-code* .xray
Amazon CloudWatch Logs	com.amazonaws.*region-code* .logs

服務	端點
AWS Security Token Service （使用服務帳戶的 IAM 角色時需要）	com.amazonaws.*region-code* .sts
Amazon EKS 驗證 （使用 Pod 身分關聯時需要）	com.amazonaws.*region-code* .eks-auth
Amazon EKS	com.amazonaws.*region-code* .eks

- 任何自我管理節點都必須部署至具有您所需 VPC 介面端點的子網路。如果您建立受管節點群組，VPC 介面端點安全群組必須允許子網路的 CIDR，或者您必須將建立的節點安全群組新增至 VPC 介面端點安全群組。
- 如果您的 Pod 使用 Amazon EFS 磁碟區，則在部署使用 Amazon EFS 存放彈性檔案系統之前，必須變更驅動程式的 kustomization.yaml 檔案，以將容器映像設定為使用與 Amazon EKS 叢集相同的 AWS 區域。
- Route53 不支援 AWS PrivateLink。您無法從私有 Amazon EKS 叢集管理 Route53 DNS 記錄。這會影響 Kubernetes external-dns。
- 如果您使用 EKS Optimized AMI，您應該在上表中啟用ec2端點。或者，您可以手動設定節點 DNS 名稱。最佳化 AMI 使用 EC2 APIs 自動設定節點 DNS 名稱。
- 您可以使用AWS Load Balancer控制器，將 AWS Application Load Balancer (ALB) 和 Network Load Balancer 部署到您的私有叢集。部署時，您應使用命令列旗標將 `enable-shield`、`enable-waf` 和 `enable-wafv2` 設定為 false。不支援使用來自輸入物件的主機名稱進行憑證探索。這是因為控制器需要連線到沒有 VPC 介面端點的 AWS Certificate Manager。

  控制器支援具有 IP 目標的 Network Load Balancer，這些目標與 Fargate 一起使用。如需詳細資訊，請參閱 the section called “應用程式負載平衡” 和 the section called “建立 Network Load Balancer”。
- 支援 Cluster Autoscaler。部署 Cluster Autoscaler Pod 時，請確定命令列包含 `--aws-use-static-instance-list=true`。如需詳細資訊，請參閱 GitHub 上的使用靜態執行個體清單。工作者節點 VPC 也必須包含 AWS STS VPC 端點和自動擴展 VPC 端點。
- 有些容器軟體產品使用 API 呼叫來存取 AWS Marketplace Metering Service 來監控用量。私有叢集不允許這些呼叫，因此您無法在私有叢集中使用這些容器類型。

使用 Karpenter 和 Cluster Autoscaler 擴展叢集運算

自動擴展功能可自動將您的資源向內外擴展，滿足不斷變化的需求。這是 Kubernetes 的主要功能，否則需要大量人力資源才能手動執行。

EKS 自動模式

Amazon EKS Auto Mode 會自動擴展叢集運算資源。如果 Pod 無法容納現有節點，EKS Auto Mode 會建立新的節點。EKS Auto Mode 也會合併工作負載並刪除節點。EKS Auto Mode 以 Karpenter 為基礎。

如需詳細資訊，請參閱：

- EKS 自動模式
- the section called “建立節點集區”
- the section called “部署膨脹工作負載”

其他解決方案

Amazon EKS 支援兩種額外的自動擴展產品：

Karpenter

Karpenter 是一款靈活的高效能 Kubernetes 叢集自動擴展程式，可協助提升應用程式的可用性和叢集效率。Karpenter 會啟動適當大小的運算資源 （例如 Amazon EC2 執行個體），以回應一分鐘內變更的應用程式負載。透過將 Kubernetes 與 整合 AWS，Karpenter 可以佈建完全符合工作負載需求的just-in-time運算資源。Karpenter 會根據叢集工作負載的特定需求，自動佈建新的運算資源。其中包括運算、儲存、加速和排程需求。Amazon EKS 支援使用 Karpenter 的叢集，然而 Karpenter 可與所有符合標準的 Kubernetes 叢集搭配使用。如需詳細資訊，請參閱 Karpenter 文件。

> **Important**
>
> Karpenter 是開放原始碼軟體， AWS 客戶負責在其 Kubernetes 叢集中安裝、設定和管理。當 Karpenter 使用 Amazon EKS 叢集中的相容版本執行未修改時， AWS 會提供技術支援。客戶升級 Karpenter 控制器或其執行所在 Kubernetes 叢集時，必須維持其可用性和安全性以及適當的測試程序，就像任何其他客戶受管軟體一樣。Karpenter 沒有 AWS 服務水準協議 (SLA)，客戶負責確保 Karpenter 啟動的 EC2 執行個體符合其業務需求。

Cluster Autoscaler

當 Pod 故障或重新排程到其他節點時，Kubernetes Cluster Autoscaler 會自動調整叢集中的節點數量。Cluster Autoscaler 使用 Auto Scaling 群組。如需詳細資訊，請參閱 Cluster Autoscaler on AWS。

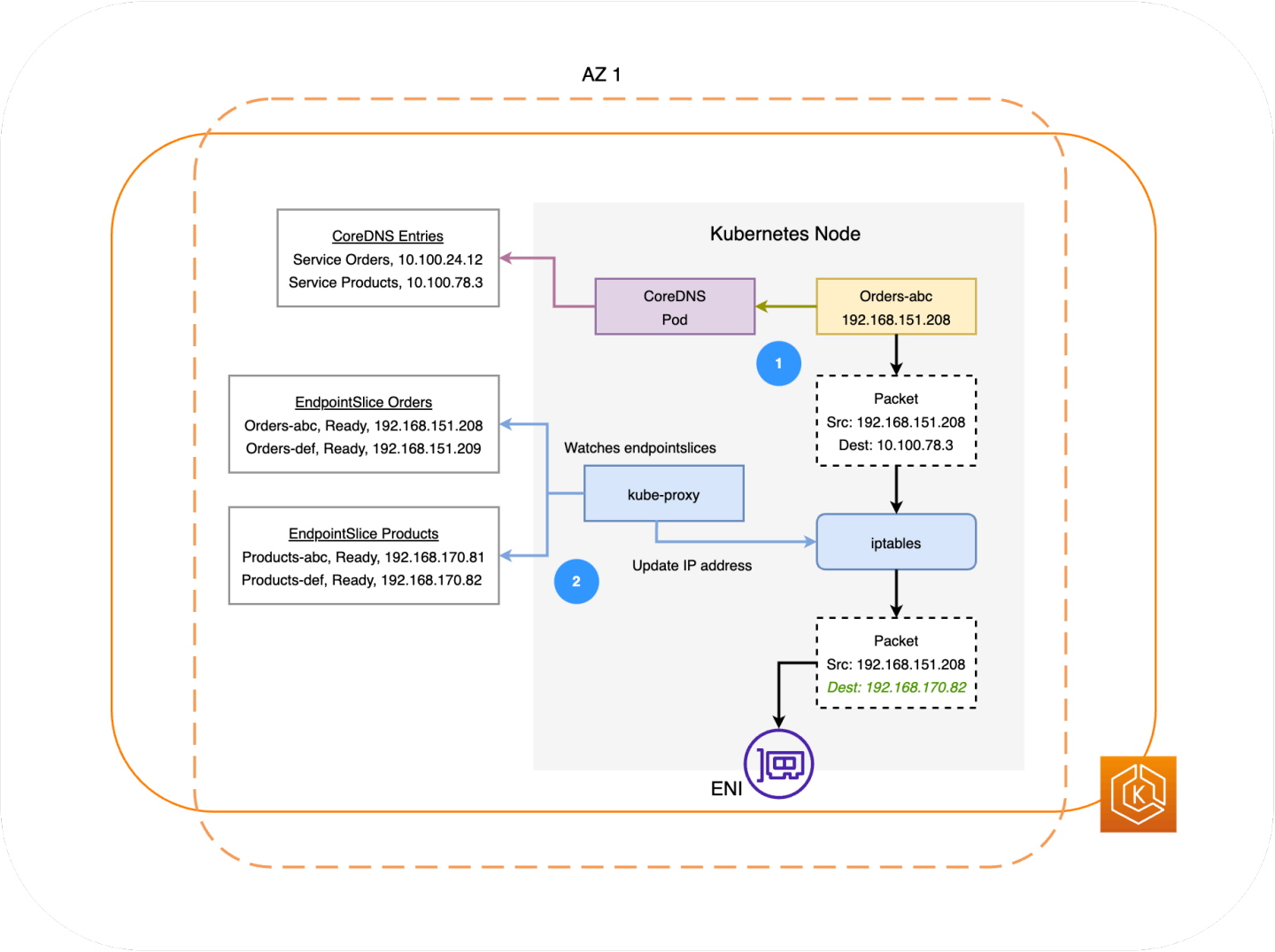
了解 Amazon EKS 中的 Amazon Application Recovery Controller (ARC) 區域轉移

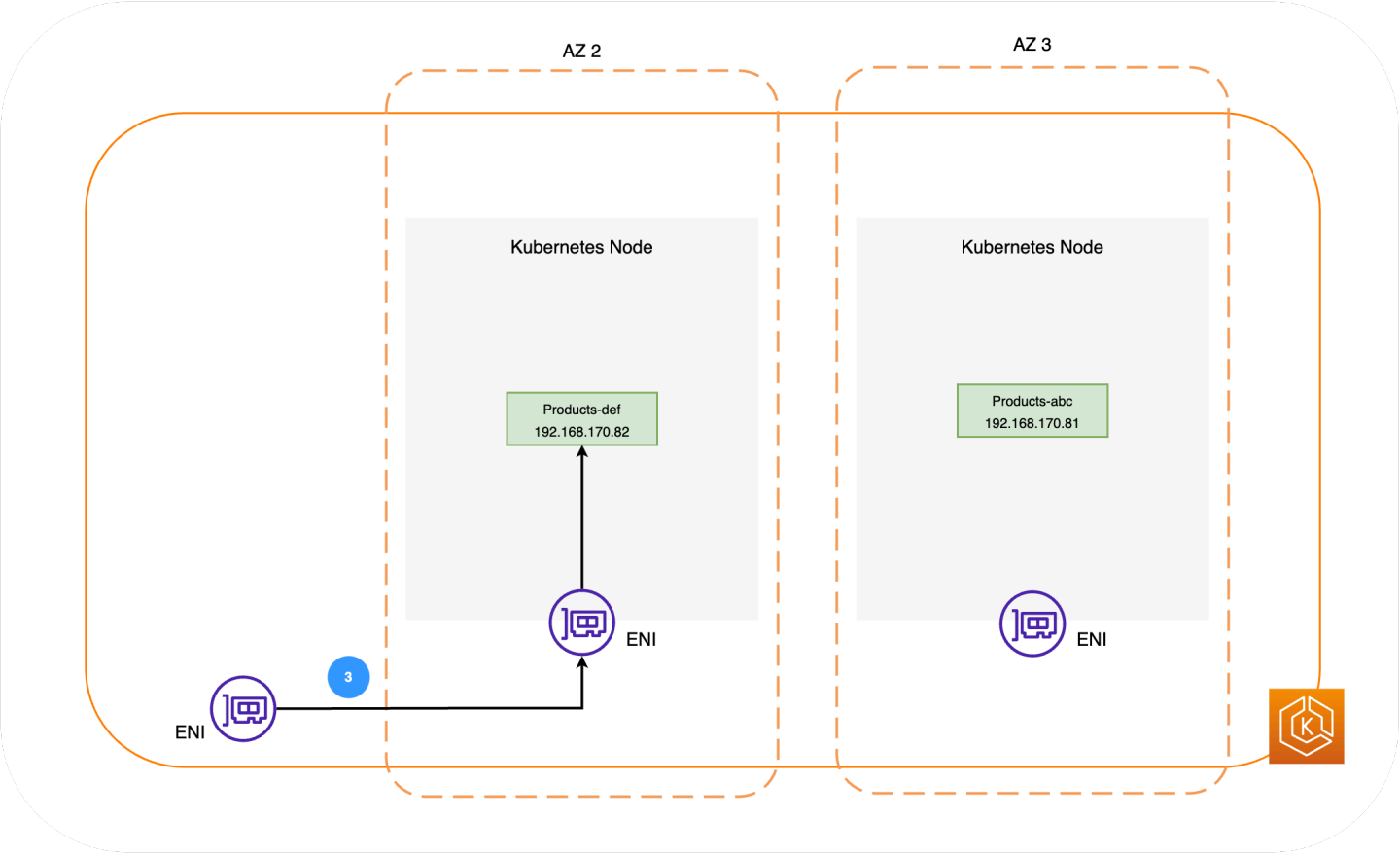
Kubernetes 具有原生功能，可讓您的應用程式對運作狀態降低或可用區域 (AZ) 受損等事件更具彈性。在 Amazon EKS 叢集中執行工作負載時，您可以使用 Amazon Application Recovery Controller (ARC) 的區域轉移或區域自動轉移，進一步改善應用程式環境的容錯能力和應用程式復原。ARC 區域轉移旨在作為暫時措施，可讓您將資源的流量從受損的可用區域移出，直到區域轉移過期或您將其取消。您可以視需要延長區域轉移。

您可以為 EKS 叢集啟動區域轉移，也可以透過啟用區域自動轉移 AWS 來允許 為您執行。此轉移會更新叢集中east-to-west網路流量的流程，只考慮在運作狀態良好的 AZs 中的工作者節點上執行之 Pod 的網路端點。此外，處理 EKS 叢集中應用程式傳入流量的任何 ALB 或 NLB 都會自動將流量路由到運作狀態良好的 AZs 中的目標。對於尋求最高可用性目標的客戶，如果 AZ 受損，則必須能夠將所有流量從受損的 AZ 轉向，直到復原為止。為此，您也可以啟用具有 ARC 區域轉移的 ALB 或 NLB。

了解 Pod 之間的東西網路流量流程

下圖說明兩個範例工作負載：訂單和產品。此範例的目的是顯示不同 AZs 中的工作負載和 Pod 如何通訊。


AZ 1
CoreDNS Entries
Service Orders, 10.100.24.12
Service Products, 10.100.78.3
Kubernetes Node
CoreDNS
Pod
Orders-abc
192.168.151.208
1
Packet
Src: 192.168.151.208
Dest: 10.100.78.3
EndpointSlice Orders
Orders-abc, Ready, 192.168.151.208
Orders-def, Ready, 192.168.151.209
Watches endpointslices
kube-proxy
iptables
EndpointSlice Products
Products-abc, Ready, 192.168.170.81
Products-def, Ready, 192.168.170.82
Update IP address
2
Packet
Src: 192.168.151.208
Dest: 192.168.170.82
ENI



1. 若要讓訂單與產品通訊，必須先解析目的地服務的 DNS 名稱。訂單將與 CoreDNS 通訊，以擷取該服務的虛擬 IP 地址（叢集 IP)。一旦 Orders 解析產品服務名稱，就會將流量傳送至該目標 IP。
2. kube-proxy 會在叢集中的每個節點上執行，並持續監看 EndpointSlices for Services。建立服務時，EndpointSlice 控制器會在背景中建立和管理 EndpointSlice。每個 EndpointSlice 都有端點清單或資料表，其中包含一部分的 Pod 地址及其執行所在的節點。kube-proxy 會在節點`iptables`上使用 為這些 Pod 端點設定路由規則。kube-proxy 也負責基本形式的負載平衡，方法是將目的地為服務叢集 IP 的流量重新導向至 ，而不是直接傳送至 Pod 的 IP 地址。kube-proxy 會在傳出連線上重寫目的地 IP 來執行此操作。
3. 然後，網路封包會透過個別節點上的 ENIs 傳送至 AZ 2 中的產品 Pod（如上圖所述）。

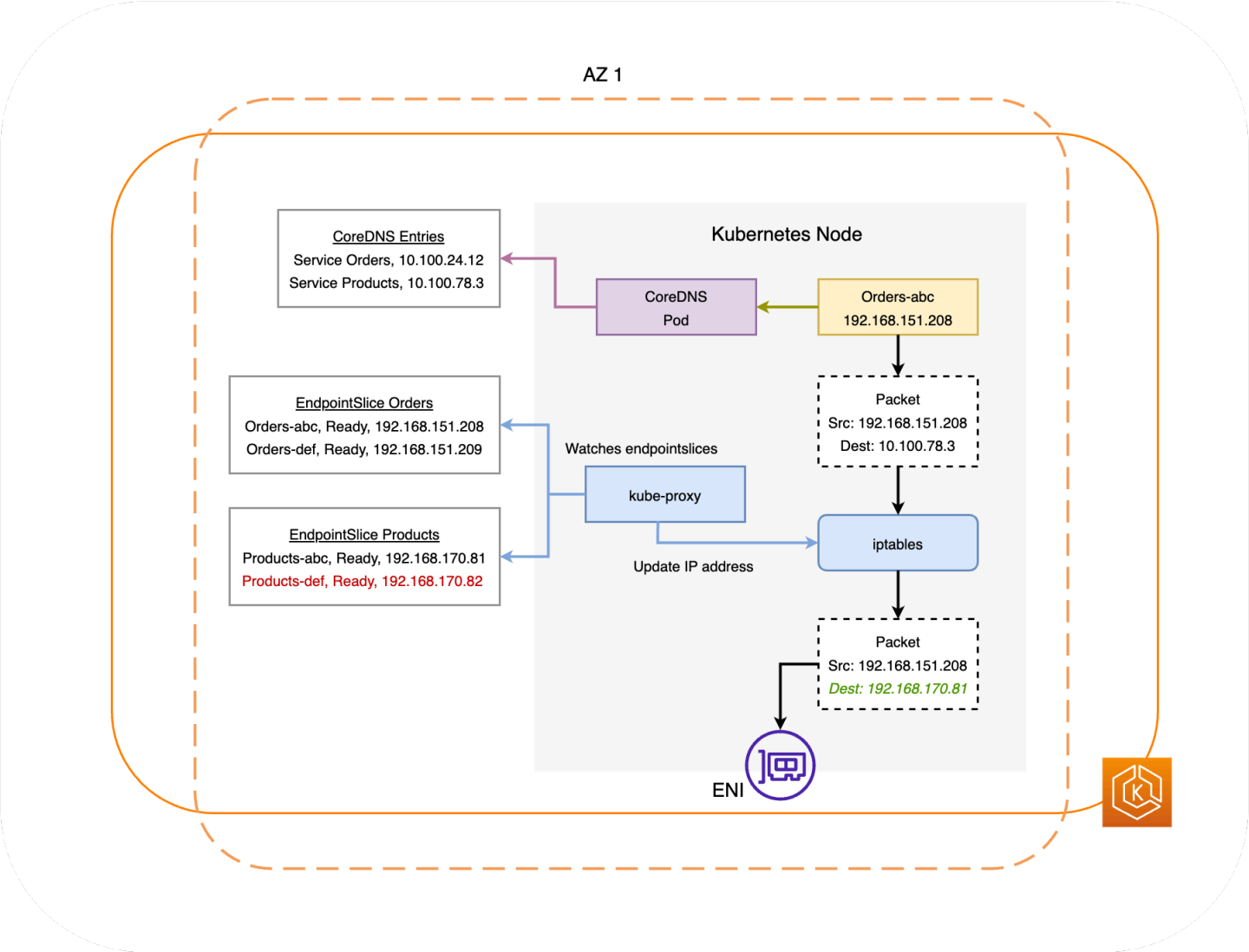
了解 Amazon EKS 中的 ARC 區域轉移

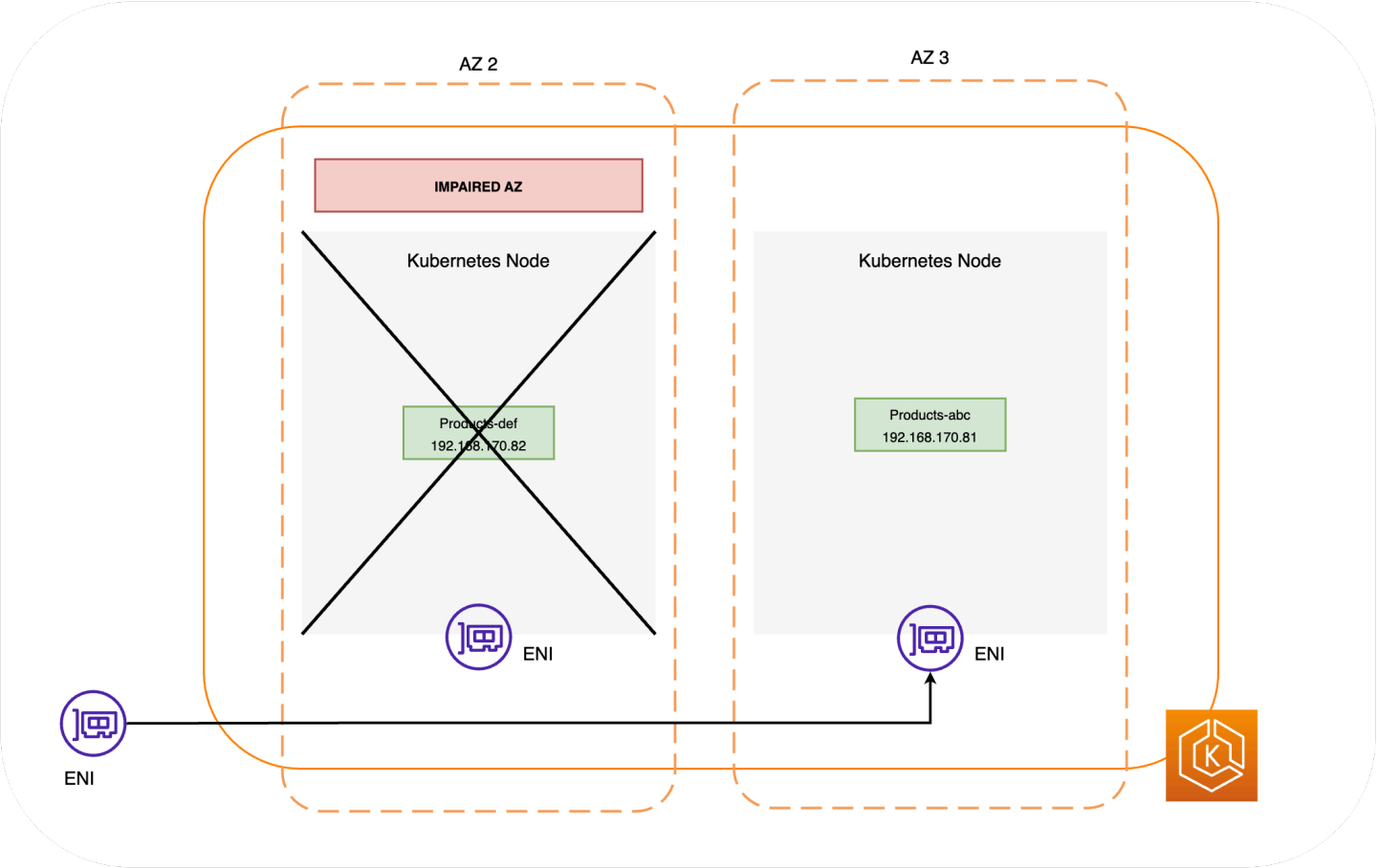
如果您的環境中有 AZ 受損，您可以為 EKS 叢集環境啟動區域轉移。或者，您可以允許 使用區域自動轉移為您 AWS 管理。使用區域自動轉移時， AWS 會監控整體 AZ 運作狀態，並透過自動將流量從叢集環境中受損的 AZ 轉移來回應潛在的 AZ 損害。

一旦 Amazon EKS 叢集已啟用 ARC 的區域轉移，您就可以使用 ARC AWS 主控台、CLI 或區域轉移和區域自動轉移 APIs 來觸發區域轉移或啟用區域自動轉移。在 EKS 區域轉移期間，會自動發生下列狀況：

- 受影響 AZ 中的所有節點都會進行封鎖。這將防止 Kubernetes 排程器將新的 Pod 排程到運作狀態不佳的 AZ 中的節點。
- 如果您使用的是受管節點群組，則可用區域重新平衡將暫停，而且您的 Auto Scaling 群組 (ASG) 將更新，以確保新的 EKS 資料平面節點僅在運作狀態良好的AZs啟動。
- 運作狀態不佳的 AZ 中的節點不會終止，也不會從這些節點移出 Pod。這是為了確保當區域轉移過期或取消時，您的流量可以安全地傳回至仍有完整容量的 AZ
- EndpointSlice 控制器會在受損的 AZ 中找到所有 Pod 端點，並從相關的 EndpointSlices 中移除它們。這將確保只有運作狀態良好的 AZs 中的 Pod 端點才能接收網路流量。當區域轉移取消或過期時，EndpointSlice 控制器會更新 EndpointSlices，以將端點包含在還原的 AZ 中。

下圖說明 EKS 區域轉移如何確保叢集環境中只有運作狀態良好的 Pod 端點成為目標的高層級流程。


AZ 1
Kubernetes Node
CoreDNS Entries
Service Orders, 10.100.24.12
Service Products, 10.100.78.3
CoreDNS
Pod
Orders-abc
192.168.151.208
Packet
Src: 192.168.151.208
Dest: 10.100.78.3
EndpointSlice Orders
Orders-abc, Ready, 192.168.151.208
Orders-def, Ready, 192.168.151.209
Watches endpointslices
kube-proxy
EndpointSlice Products
Products-abc, Ready, 192.168.170.81
Products-def, Ready, 192.168.170.82
Update IP address
iptables
Packet
Src: 192.168.151.208
Dest: 192.168.170.81
ENI



EKS 區域轉移要求

若要讓區域轉移在 EKS 中成功運作，您需要事先設定叢集環境以應對 AZ 受損。以下是您必須遵循的步驟清單。

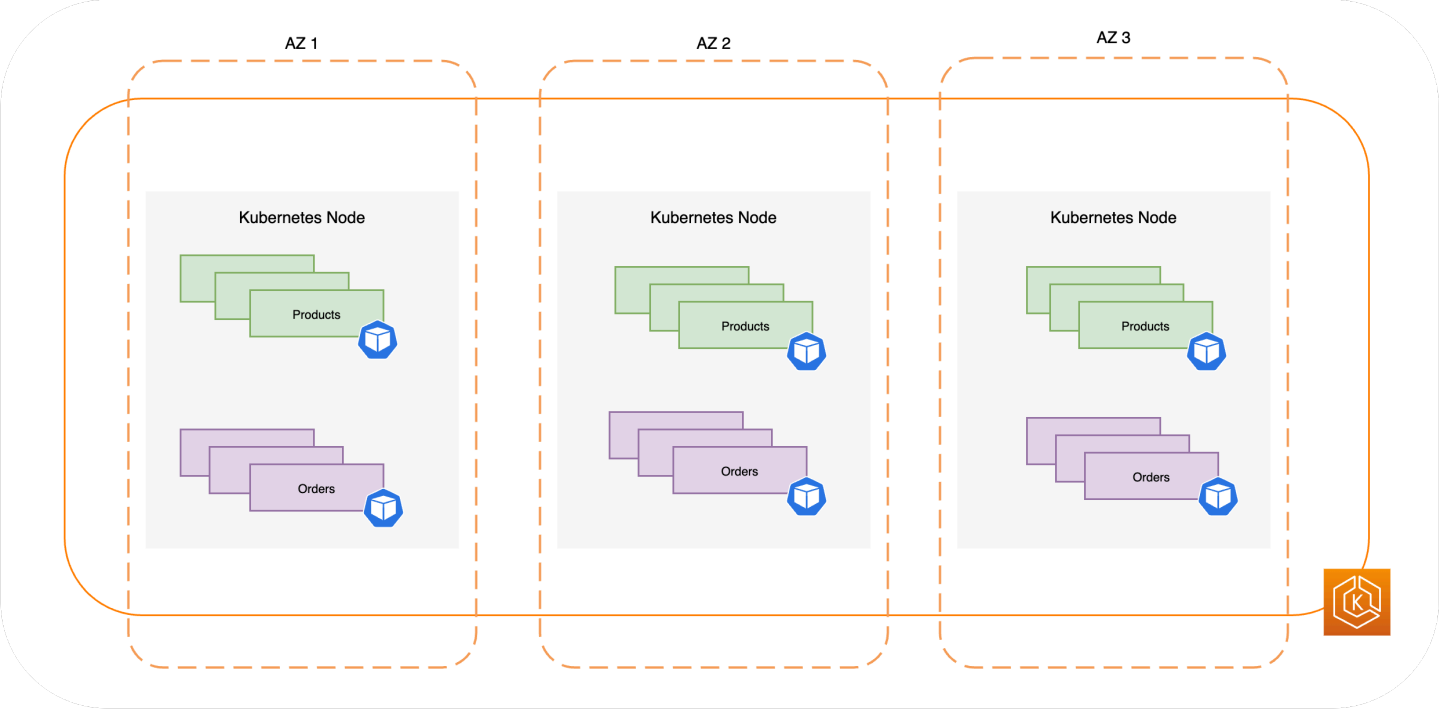
- 跨多個AZs佈建叢集的工作者節點
- 佈建足夠的運算容量，以承受移除單一可用區域
- 在每個 AZ 中預先擴展您的 Pod （包括 CoreDNS)
- 將多個 Pod 複本分散到所有AZs，以確保從單一可用區域轉移將讓您擁有足夠的容量
- 將相互依賴或相關的 Pod 共同放置在相同的 AZ 中
- 透過手動啟動區域轉移，測試您的叢集環境是否可如預期使用一個較少的可用區域。或者，您可以啟用區域自動轉移，並在自動轉移實務執行時回覆。區域轉移在 EKS 中運作時不需要，但強烈建議這麼做。

在多個AZs佈建 EKS 工作者節點

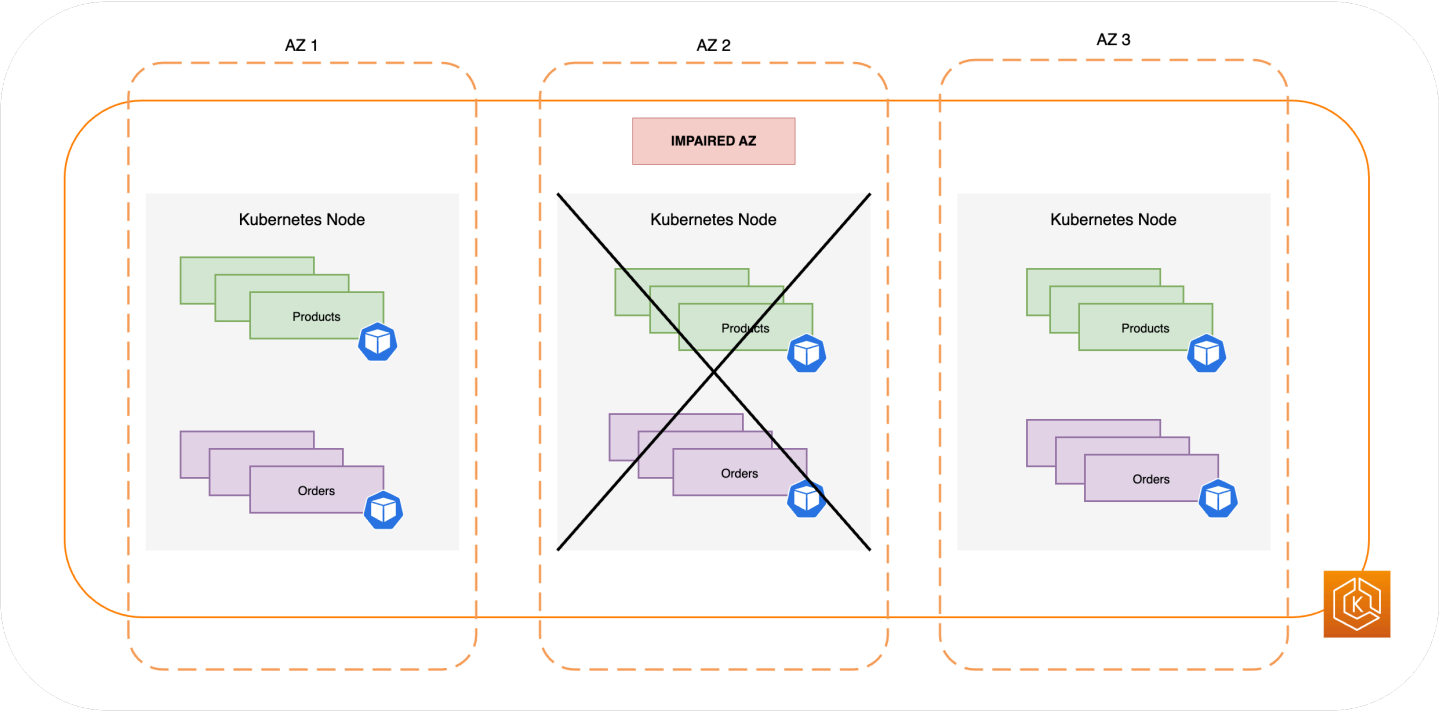
AWS 區域具有多個不同的位置，其實體資料中心稱為可用區域 (AZs)。AZs旨在彼此實體隔離，以避免可能影響整個區域的同時影響。佈建 EKS 叢集時，您應該將工作者節點部署到區域中的多個 AZs。這將使您的叢集環境對單一可用區受損更具彈性，並允許您維護在其他AZs區中執行之應用程式的高可用性 (HA)。當您從受影響的可用區域開始區域轉移時，EKS 環境的叢集內網路會自動更新為僅使用運作狀態良好的可用AZs，同時為您的叢集維持高可用性狀態。

確保您擁有 EKS 環境的此類多可用區設定，可增強系統的整體可靠性。不過，多可用區域環境在傳輸和處理應用程式資料的方式上扮演重要角色，這反而會影響您環境的網路費用。特別是，頻繁的輸出跨區域流量（分散在AZs之間的流量）可能會對您的網路相關成本產生重大影響。您可以套用不同的策略來控制 EKS 叢集中 Pod 之間的跨區域流量，並降低相關聯的成本。如需如何在執行高可用性 EKS 環境時最佳化網路成本的詳細資訊，請參閱此最佳實務指南。

下圖說明具有 3 個運作狀態良好的 AZs 的高可用性 EKS 環境。



下圖說明具有 3 個AZs EKS 環境如何對可用區域受損具有彈性，並由於其他 2 個運作狀態良好的AZs而保持高可用性。



佈建足夠的運算容量，可承受移除單一可用區域

若要最佳化 EKS 資料平面中運算基礎設施的資源使用率和成本，最佳實務是使運算容量符合工作負載需求。不過，如果您的所有工作者節點都已滿容量，這可讓您依賴將新的工作者節點新增至 EKS 資料平面，然後才能排程新的 Pod。執行關鍵工作負載時，通常最佳實務是在線上使用備援容量執行，以處理負載突然增加、節點運作狀態問題等事件。如果您計劃使用區域轉移，則計劃移除整個 AZ 容量，因此您需要調整備援運算容量，以便即使 AZ 離線也能處理負載。

擴展運算時，將新節點新增至 EKS 資料平面的程序需要一些時間，這可能會影響應用程式的即時效能和可用性，尤其是在區域受損的情況下。您的 EKS 環境應具有彈性，以吸收遺失可用區負載，以避免最終使用者或用戶端體驗降級。這表示在需要新 Pod 的時間，以及實際排程在工作者節點的時間之間，盡可能減少或消除任何延遲。

此外，如果發生區域受損，您應該降低潛在運算容量限制的風險，以防止在運作狀態良好的 AZs 中將新所需的節點新增至 EKS 資料平面。

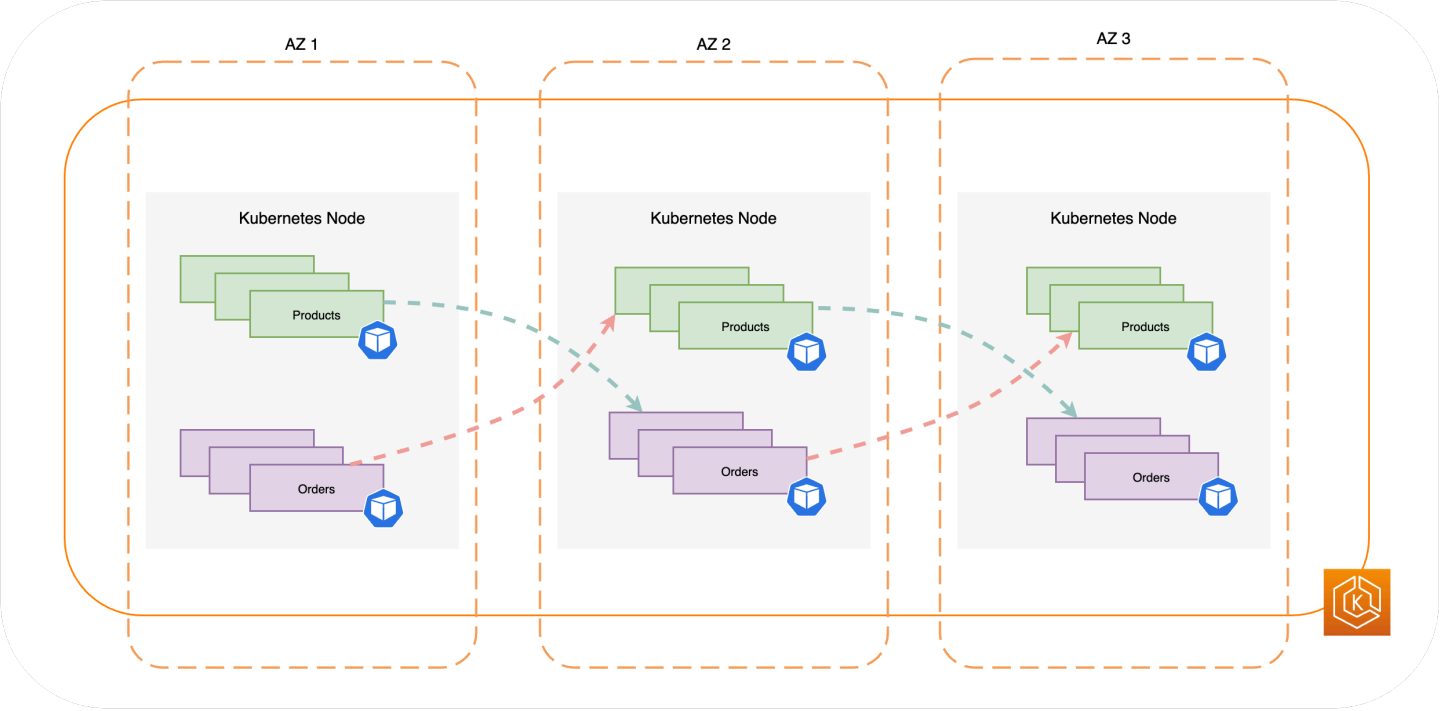
若要達成此目的，您應該在每個AZs的某些工作者節點中過度佈建運算容量，以便 Kubernetes 排程器具有可用於新 Pod 置放的預先存在容量，特別是當您的環境中有一個較少的可用區域時。

在 AZs 中執行並分散多個 Pod 複本

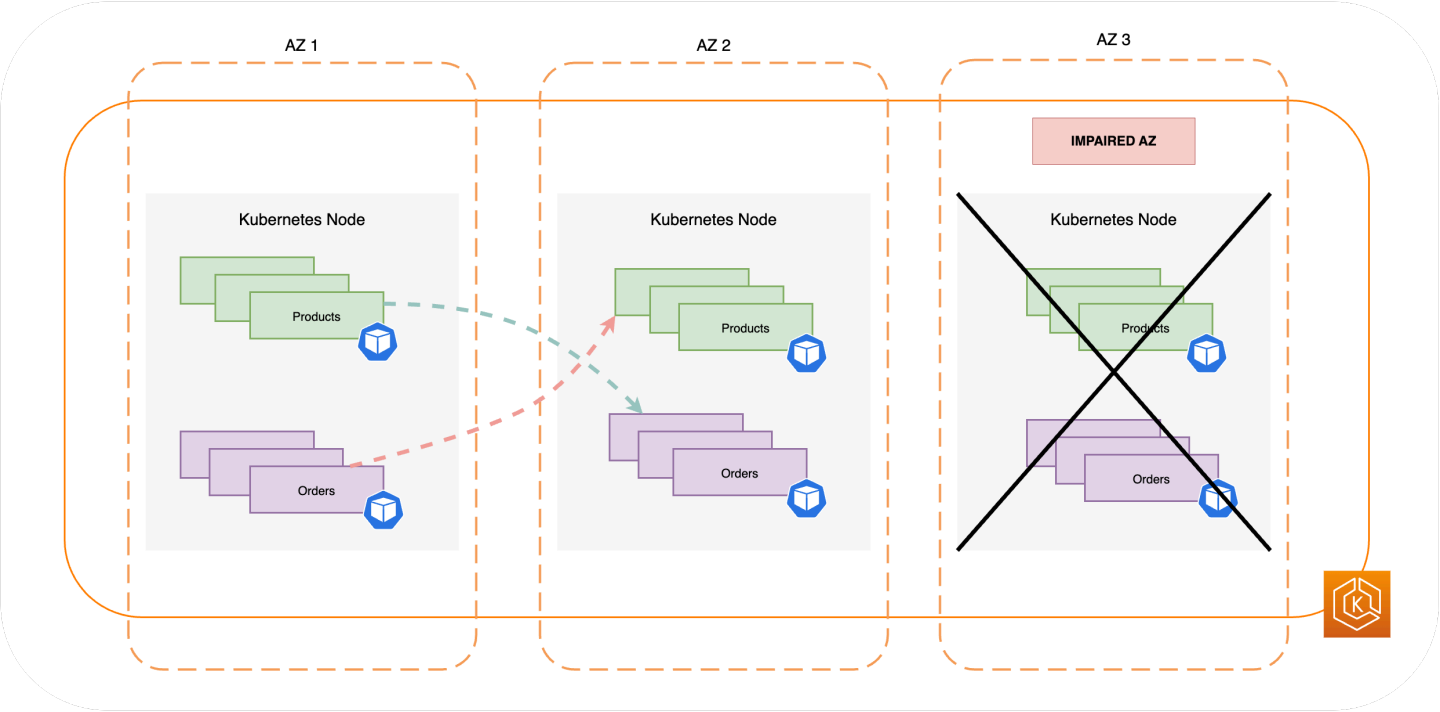
Kubernetes 可讓您透過執行單一應用程式的多個執行個體 (Pod 複本） 來預先擴展工作負載。為應用程式執行多個 Pod 複本可消除單一故障點，並透過減少單一複本上的資源負擔來提高其整體效能。

不過，若要同時擁有高可用性和更佳的應用程式容錯能力，您應該在不同的故障網域（也稱為拓撲網域）中執行並分散應用程式的多個複本，在此情況下為 AZs。透過拓撲分散限制，您可以將應用程式設定為具有預先存在的靜態穩定性，以便在 AZ 受損的情況下，您會在運作狀態良好的 AZs 中擁有足夠的複本，以立即處理他們可能遇到的任何額外尖峰或激增流量。

下圖說明當所有 AZs都正常運作時，具有east-to-west流量流的 EKS 環境。



下圖說明單一可用區故障時具有east-to-west流量流的 EKS 環境，而且您啟動區域轉移。



以下程式碼片段是如何使用此 Kubernetes 功能設定工作負載的範例。

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: orders
spec:
  replicas: 9
  selector:
    matchLabels:
      app:orders
  template:
    metadata:
      labels:
        app: orders
        tier: backend
    spec:
      topologySpreadConstraints:
      - maxSkew: 1
        topologyKey: "topology.kubernetes.io/zone"
        whenUnsatisfiable: ScheduleAnyway
        labelSelector:
          matchLabels:
            app: orders
```

最重要的是，您應該執行 DNS 伺服器軟體 (CoreDNS/kube-dns) 的多個複本，並在尚未預設設定時套用類似的拓撲分散限制。這有助於確保您在運作狀態良好的AZs有足夠的 DNS Pod，以便在單一可用區域受損時，繼續處理叢集中其他通訊 Pod 的服務探索請求。CoreDNS EKS 附加元件具有預設設定，可在多個可用AZs有節點時，將 CoreDNS Pod 分散到叢集的可用區域。您也可以將這些預設設定取代為您自己的自訂組態。

使用 Helm 安裝 CoreDNS 時，您可以更新 values.yaml 檔案中`replicaCount`的 ，以確保您在每個可用區域中有足夠數量的複本。此外，為了確保這些複本分散到叢集環境中的不同AZs，您應該更新相同 value.yaml 檔案中的 `topologySpreadConstraints` 屬性。下面的程式碼片段示範如何為此設定 CoreDNS。

CoreDNS Helm values.yaml

```
replicaCount: 6
topologySpreadConstraints:
  - maxSkew: 1
    topologyKey: topology.kubernetes.io/zone
    whenUnsatisfiable: ScheduleAnyway
    labelSelector:
      matchLabels:
        k8s-app: kube-dns
```

如果 AZ 受損，您可以使用 CoreDNS 的自動擴展系統來吸收 CoreDNS Pod 上增加的負載。您需要的 DNS 執行個體數目取決於叢集中執行的工作負載數目。CoreDNS 是 CPU 繫結，允許它使用 Horizontal Pod Autoscaler (HPA) 根據 CPU 進行擴展。以下是您可以修改以滿足您的需求的範例。

```
apiVersion: autoscaling/v1
kind: HorizontalPodAutoscaler
metadata:
  name: coredns
  namespace: default
spec:
  maxReplicas: 20
  minReplicas: 2
  scaleTargetRef:
    apiVersion: apps/v1
    kind: Deployment
    name: coredns
  targetCPUUtilizationPercentage: 50
```

或者，EKS 可以在 CoreDNS 的 EKS 附加元件版本中管理 CoreDNS 部署的自動擴展。此 CoreDNS 自動擴展器會持續監控叢集狀態，包括節點和 CPU 核心的數量。根據該資訊，控制器將動態調整 EKS 叢集中 CoreDNS 部署的複本數量。

若要在 CoreDNS EKS 附加元件中啟用自動擴展組態，您應該新增下列選用組態設定：

```
{
  "autoScaling": {
    "enabled": true
  }
}
```

您也可以使用 NodeLocal DNS 或叢集比例自動擴展器來擴展 CoreDNS。您可以在此處進一步閱讀如何水平擴展 CoreDNS。

在相同 AZ 中共置相互依存 Pod

在大多數情況下，您可能正在執行不同的工作負載，這些工作負載必須互相通訊，才能成功執行end-to-end程序。如果不同的應用程式分散在不同 AZs 中，但未共置在相同的 AZ 中，則單一 AZ 受損可能會影響基礎end-to-end程序。例如，如果應用程式 A 在 AZ 1 和 AZ 2 中有多個複本，但應用程式 B 在 AZ 3 中有其所有複本，則 AZ 3 的遺失將影響這兩個工作負載 (應用程式 A 和 B) 之間的任何end-to-end程序。將拓撲分散限制與 Pod 親和性結合，可以透過將 Pod 分散到所有AZs，以及設定特定 Pod 之間的關係，以確保它們共置在一起，來增強應用程式的彈性。

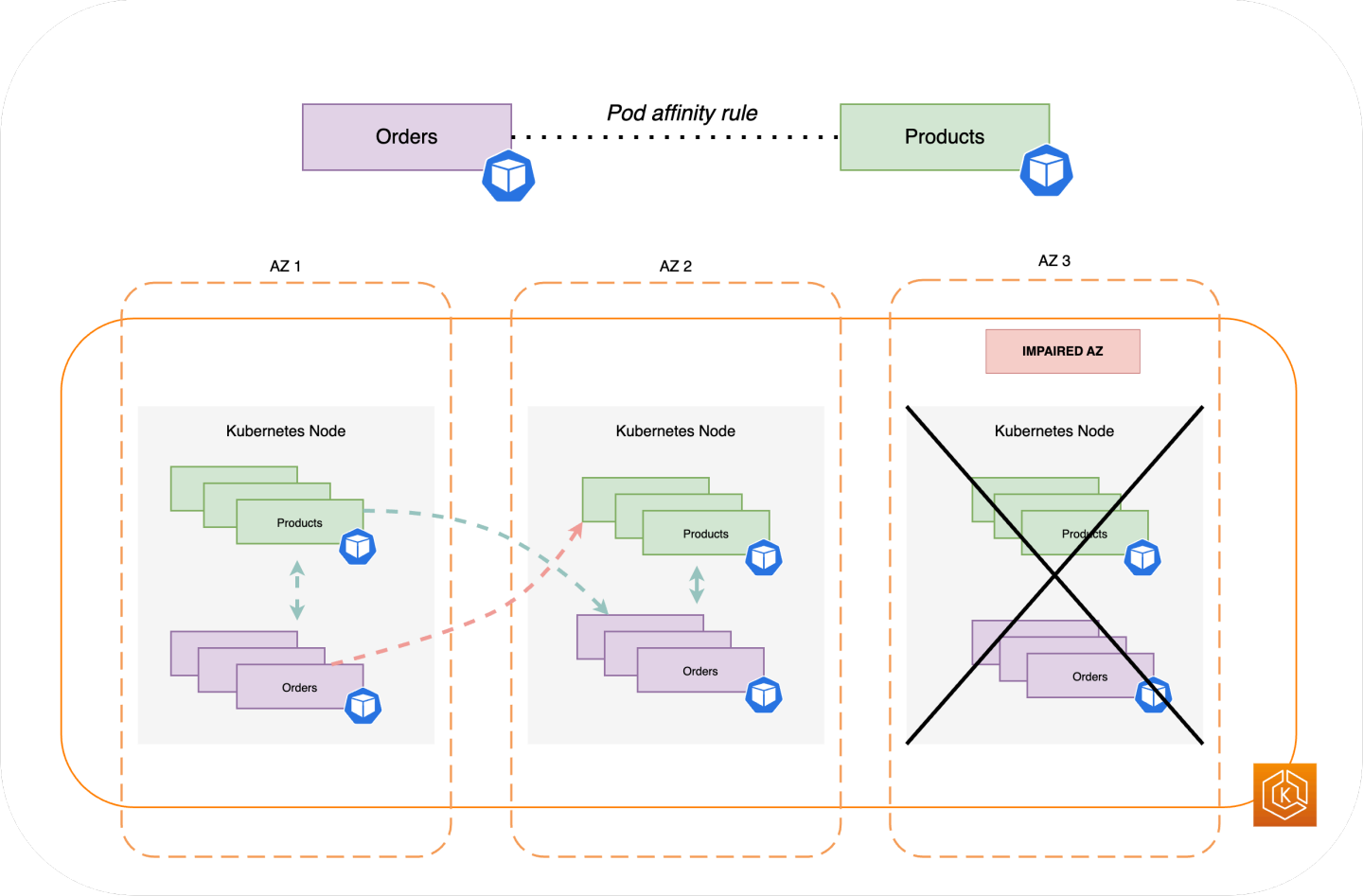
使用 Pod 親和性規則，您可以定義工作負載之間的關係，以影響 Kubernetes 排程器的行為，使其將 Pod 共置在相同的工作者節點或相同的可用區域中。您也可以設定這些排程限制的嚴格程度。

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: products
  namespace: ecommerce
  labels:
    app.kubernetes.io/version: "0.1.6"

    spec:
      serviceAccountName: graphql-service-account
      affinity:
        podAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
          - labelSelector:
```

```
            matchExpressions:
            - key: app
              operator: In
              values:
              - orders
          topologyKey: "kubernetes.io/hostname"
```

下圖說明已使用 Pod 親和性規則在相同節點上共置的 Pod。



測試您的叢集環境是否可以處理 AZ 的遺失

完成上述要求後，下一個重要步驟是測試您是否有足夠的運算和工作負載容量來處理 AZ 遺失。您可以透過手動觸發 EKS 中的區域轉移來執行此操作。或者，您可以啟用區域自動轉移，並設定實務執行，以測試您的應用程式在叢集環境中的可用區域是否如預期運作。

常見問答集

為什麼我應該使用此功能？

透過在 EKS 叢集中使用 ARC 區域轉移或區域自動轉移，您可以藉由自動化快速復原程序，將叢集內網路流量從受損的可用區域轉移出來，以更好地維持 Kubernetes 應用程式可用性。使用 ARC，您可以避免長時間且複雜的步驟，這通常會導致在受損的 AZ 事件期間延長復原期間。

此功能如何與其他 AWS 服務搭配使用？

EKS 與 ARC 整合，提供主要界面供您完成復原操作 AWS。為了確保叢集內流量適當路由離開受損的可用區域，會修改在 Kubernetes 資料平面中執行之 Pod 的網路端點清單。如果您使用 AWS 負載平衡器將外部流量路由到叢集，則已經可以向 ARC 註冊負載平衡器，並在它們上觸發區域轉移，以防止流量流入降級的區域。此功能也會與 EKS 受管節點群組 (MNG) 建立的 Amazon EC2 Auto Scaling 群組 (ASG) 互動。為了防止受損的 AZ 用於新的 Kubernetes Pod 或節點啟動，EKS 會從 ASG 中移除受損的 AZ。

此功能與預設 Kubernetes 保護有何不同？

此功能可搭配多種 Kubernetes 原生內建保護，協助客戶保持彈性。您可以設定 Pod 整備和活體探查，以決定 Pod 何時應接收流量。當這些探查失敗時，Kubernetes 會將這些 Pod 移除為服務的目標，且流量不會再傳送至 Pod。雖然這很實用，但客戶設定這些運作狀態檢查並不容易，因此當區域降級時，保證它們會失敗。ARC 區域轉移功能為您提供額外的安全網，當 Kubernetes 的原生保護尚未足夠時，可協助它們完全隔離降級的可用區域。它也提供簡單的方法來測試架構的操作準備程度和彈性。

可以代表我 AWS 觸發區域轉移嗎？

是，如果您想要完全自動化的方式來使用 ARC 區域轉移，您可以啟用 ARC 區域自動轉移。透過區域自動轉移，您可以依賴 AWS 來監控 EKS 叢集的AZs運作狀態，並在偵測到可用區域受損時自動觸發轉移。

如果我使用此功能，而且我的工作者節點和工作負載未預先擴展，會發生什麼情況？

如果您未預先調整規模，並依賴於在區域轉移期間佈建其他節點或 Pod，則會有延遲復原的風險。將新節點新增至 Kubernetes 資料平面的程序需要一些時間，這可能會影響應用程式的即時效能和可用性，尤其是在區域受損的情況下。此外，如果區域受損，您可能會遇到潛在的運算容量限制，這可能會阻止將新所需的節點新增至運作狀態良好的AZs。

如果您的工作負載未預先擴展，並分散到叢集中的所有AZs，則區域受損可能會影響僅在受影響可用區域中的工作者節點上執行的應用程式可用性。為了降低應用程式完全可用性中斷的風險，如果工作負載在運作狀態不佳的可用區域中擁有其所有端點，EKS 會無法安全地將流量傳送至受損區域中的 Pod 端點。不過，強烈建議您預先擴展應用程式，並將其分散到所有AZs，以便在發生區域問題時維持可用性。

如果我執行具狀態的應用程式，會發生什麼情況？

如果您執行具狀態的應用程式，則需要根據使用案例和架構來評估其容錯能力。如果您有作用中/待命架構或模式，則可能會有作用中位於受損 AZ 中的執行個體。在應用程式層級，如果未啟用待命，您可能會遇到應用程式的問題。當新的 Kubernetes Pod 在運作狀態良好的可用AZs啟動時，您可能會遇到問題，因為它們無法連接至繫結至受損可用區域的持久性磁碟區。

此功能是否適用於 Karpenter？

Karpenter 支援目前不適用於 EKS 中的 ARC 區域轉移和區域自動轉移。如果 AZ 受損，您可以透過移除運作狀態不佳的 AZ 來調整相關的 Karpenter NodePool 組態，以便僅在運作狀態良好的 AZs 中啟動新的工作者節點。

此功能是否適用於 EKS Fargate？

此功能不適用於 EKS Fargate。根據預設，當 EKS Fargate 辨識區域運作狀態事件時，Pod 會偏好在其他 AZs 中執行。

EKS 受管 Kubernetes 控制平面是否會受到影響？

否，根據預設，Amazon EKS 會在多個AZs執行和擴展 Kubernetes 控制平面，以確保高可用性。ARC 區域轉移和區域自動轉移只會在 Kubernetes 資料平面上運作。

此新功能是否有任何相關成本？

您可以在 EKS 叢集中使用 ARC 區域轉移和區域自動轉移，無需額外費用。不過，您將繼續支付佈建執行個體的費用，強烈建議您在使用此功能之前預先擴展 Kubernetes 資料平面。您應該考慮成本和應用程式可用性之間的正確平衡。

其他資源

- 區域轉移的運作方式
- ARC 中區域轉移的最佳實務
- 區域轉移和區域自動轉移支援的資源和案例
- 在 Amazon EKS 上操作彈性工作負載
- 以 Pod 優先順序和過度佈建來消除 Kubernetes 節點擴展延遲
- 擴展高 DNS 流量的 CoreDNS Pod

啟用 EKS 區域轉移以避免可用區域受損

Amazon Application Recovery Controller (ARC) 可協助您管理和協調跨可用區域 (AZs) 的應用程式復原，並與許多 服務搭配使用，包括 Amazon EKS。透過 ARC 區域轉移的 EKS 支援，您可以將叢集內網路流量從受損的 AZ 轉移。您也可以授權 AWS 監控 AZs 的運作狀態，並代表您暫時將網路流量移離運作狀態不佳的 AZ。

如何使用 EKS 區域轉移：

1. 使用 Amazon Application Recovery Controller (ARC) 啟用 EKS 叢集。這會在叢集層級使用 Amazon EKS AWS 主控台、CLI、CloudFormation 或 eksctl 完成。
2. 啟用後，您可以使用 ARC 主控台、CLI 或區域轉移和區域自動轉移 APIs AWS 來管理區域轉移或區域自動轉移。

請注意，在向 ARC 註冊 EKS 叢集之後，您仍然需要設定 ARC。例如，您可以使用 ARC 主控台來設定區域 Autoshift。

如需有關 EKS 區域轉移如何運作，以及如何設計工作負載來處理可用區域受損的詳細資訊，請參閱 the section called “了解區域轉移”。

考量事項

- EKS Auto Mode 不支援 Amazon Application Recovery Controller、Zonal Shift 和 Zonal Autoshift。
- 我們建議在區域轉移操作之間等待至少 60 秒，以確保正確處理每個請求。

  嘗試快速連續執行區域轉移時（相隔 60 秒內），Amazon EKS 服務可能無法正確處理所有輪班請求。這是由於更新叢集區域狀態的目前輪詢機制所致。如果您需要執行多個區域轉移，請確保系統在操作之間有足夠的時間來處理每個變更。

什麼是 Amazon 應用程式復原控制器？

Amazon Application Recovery Controller (ARC) 可協助您為執行中的應用程式做好準備並完成更快的復原 AWS。區域轉移可讓您從可用區域 (AZ) 受損快速復原，方法是暫時將支援資源的流量從可用區域移出至 AWS 區域中運作狀態良好的AZs。

進一步了解 Amazon Application Recovery Controller (ARC)

什麼是區域轉移？

區域轉移是 ARC 的一項功能，可讓您將 EKS 叢集或 Elastic Load Balancer 等資源的流量移離 AWS 區域中的可用區域，以快速緩解問題並快速復原應用程式。例如，您可以選擇轉移流量，因為部署不良會導致延遲問題，或因為可用區域受損。區域轉移不需要進階組態步驟。

進一步了解 ARC 區域轉移

什麼是區域自動轉移？

區域自動轉移是 ARC 中的一項功能，您可以授權 代表您 AWS 將流量從可用區域轉移到 AWS 區域中運作狀態良好的AZs。當內部遙測顯示 區域中的一個可用區域發生可能影響客戶的損害時， 會 AWS 啟動自動轉移。內部遙測包含來自多個來源的指標，包括 AWS 網路，以及 Amazon EC2 和 Elastic Load Balancing 服務。

AWS 當指標顯示不再存在問題或潛在問題時， 會結束自動轉移。

進一步了解 ARC Zonal Autoshift

EKS 在自動轉移期間會做什麼？

EKS 會更新聯網組態，以避免將流量導向受損AZs。此外，如果您使用受管節點群組，EKS 只會在區域轉移期間在運作狀態良好的 AZs 中啟動新節點。當輪班到期或取消時，將會還原聯網組態，以包含先前偵測到運作狀態不佳的 AZ。

進一步了解 EKS 區域轉移。

向 Amazon Application Recovery Controller (ARC) 註冊 EKS 叢集AWS （主控台）

1. 尋找您要向 ARC 註冊的 EKS 叢集名稱和區域。
2. 導覽至該區域的 EKS 主控台，然後選取您的叢集。
3. 在叢集資訊頁面上，選取概觀索引標籤。
4. 在區域轉移標題下，選取管理按鈕。
5. 選取啟用或停用 EKS 區域轉移。

現在您的 EKS 叢集已向 ARC 註冊。

如果您想要 AWS 偵測並避免可用區域受損，則需要設定 ARC Zonal Autoshift。例如，您可以在 ARC 主控台中執行此操作。

後續步驟

- 了解如何啟用區域自動轉移。
- 了解如何手動啟動區域轉移。

了解存取控制如何在 Amazon EKS 中運作

了解如何管理 Amazon EKS 叢集的存取權。使用 Amazon EKS 需要了解 Kubernetes 和 AWS Identity and Access Management (AWS IAM) 如何處理存取控制。

本節包括：

the section called “Kubernetes API 存取” — 了解如何讓應用程式或使用者驗證 Kubernetes API。您可以使用存取項目、aws-auth ConfigMap 或外部 OIDC 供應商。

the section called “存取叢集資源” — 了解如何設定 AWS Management Console 以與您的 Amazon EKS 叢集通訊。使用 主控台檢視叢集中的 Kubernetes 資源，例如命名空間、節點和 Pod。

the section called “使用 kubectl 存取叢集” — 了解如何設定 kubectl 以與您的 Amazon EKS 叢集通訊。使用 AWS CLI 建立 kubeconfig 檔案。

the section called “工作負載存取 AWS ” — 了解如何將 Kubernetes 服務帳戶與 IAM AWS 角色建立關聯。您可以使用 Pod Identity 或 IAM Roles for Service Accounts (IRSA)。

常見任務

- 授予開發人員對 Kubernetes API 的存取權。在 中檢視 Kubernetes 資源 AWS Management Console。
  - 解決方案：使用存取項目將 Kubernetes RBAC 許可與 IAM AWS 使用者或角色建立關聯。
- 設定 kubectl 以使用 AWS 登入資料與 Amazon EKS 叢集通訊。
  - 解決方案：使用 AWS CLI 建立 kubeconfig 檔案。
- 使用 Ping Identity 等外部身分提供者，對 Kubernetes API 的使用者進行身分驗證。
  - 解決方案：連結外部 OIDC 供應商。
- 授予 Kubernetes 叢集上的工作負載呼叫 AWS APIs的能力。
  - 解決方案：使用 Pod Identity 將 AWS IAM 角色與 Kubernetes 服務帳戶建立關聯。

背景介紹

- 了解 Kubernetes 服務帳戶的運作方式。
- 檢閱 Kubernetes 角色型存取控制 (RBAC) 模型

- 如需管理 AWS 資源存取權的詳細資訊，請參閱 AWS IAM 使用者指南。或者，您也可以使用 IAM 在 AWS 上進行免費入門訓練。

EKS Auto 模式的考量事項

EKS Auto Mode 與 EKS Pod Identity 和 EKS EKS 存取項目整合。

- EKS Auto Mode 使用存取項目來授予 EKS 控制平面 Kubernetes 許可。例如，存取政策可讓 EKS Auto Mode 讀取網路端點和服務的相關資訊。
  - 您無法在 EKS Auto Mode 叢集上停用存取項目。
  - 您可以選擇性地啟用 `aws-auth ConfigMap`。
  - EKS Auto Mode 的存取項目會自動設定。您可以檢視這些存取項目，但無法修改它們。
  - 如果您使用 NodeClass 建立自訂節點 IAM 角色，則需要使用 AmazonEKSAutoNodePolicy 存取政策來建立角色的存取項目。
- 如果您想要授予 AWS 服務的工作負載許可，請使用 EKS Pod Identity。
  - 您不需要在 EKS Auto Mode 叢集上安裝 Pod Identity 代理程式。

授予 IAM 使用者和角色對 Kubernetes APIs存取權

您的叢集具有 Kubernetes API 端點。Kubectl 使用此 API。您可以使用兩種類型的身分來驗證此 API：

- AWS Identity and Access Management (IAM) 主體 （角色或使用者） – 此類型需要對 IAM 進行身分驗證。使用者可以 AWS 使用透過身分來源提供的登入資料，以 IAM 使用者身分或使用聯合身分登入 。如果管理員先前已設定使用 IAM 角色的聯合身分，則使用者只能夠以聯合身分登入。當使用者使用聯合 AWS 身分存取 時，他們間接擔任角色。當使用者使用此類身分時，您：
  - 可以為其指派 Kubernetes 許可，以便他們可以在您的叢集上使用 Kubernetes 物件。如需如何將許可指派給 IAM 主體以使其能夠存取叢集上 Kubernetes 物件的詳細資訊，請參閱 the section called “授予許可”。
  - 可以指派 IAM 許可給他們，讓他們可以使用 Amazon EKS API、 AWS CLI AWS Management Console、 AWS CloudFormation 或 使用您的 Amazon EKS 叢集及其資源`eksctl`。如需詳細資訊，請參閱《服務授權參考》中的 Amazon Elastic Kubernetes Service 定義的動作一節。
  - 節點透過承擔 IAM 角色來加入叢集。IAM AWS Authenticator for Kubernetes 在 Amazon EKS 控制平面上執行，可讓您使用 IAM 主體存取叢集。

- 您自己的 OpenID Connect (OIDC) 提供者中的使用者 – 此類型需要向您的 OIDC 提供者進行身分驗證。如需使用 Amazon EKS 叢集設定您自己的 OIDC 提供者的詳細資訊，請參閱 the section called "連結 OIDC 供應商"。當使用者使用此類身分時，您：
  - 可以為其指派 Kubernetes 許可，以便他們可以在您的叢集上使用 Kubernetes 物件。
  - 無法為他們指派 IAM 許可，讓他們可以使用 Amazon EKS API、 AWS CLI AWS Management Console、 AWS CloudFormation 或 來使用您的 Amazon EKS 叢集及其資源`eksctl`。

您可以在叢集中使用這兩種類型的身分。IAM 身分驗證方法無法停用。OIDC 身分驗證方法為選用。

將 IAM 身分與 Kubernetes 許可建立關聯

AWS 適用於 Kubernetes 的 IAM Authenticator 安裝在叢集的控制平面上。它可讓 Identity AWS and Access Management (IAM) 主體 （ 角色和使用者 ） 存取叢集上的 Kubernetes 資源。您可以使用下列其中一種方法，允許 IAM 主體存取叢集上的 Kubernetes 物件：

- 建立存取項目 – 如果您的叢集是等於或晚於叢集 Kubernetes 版本先決條件區段中列出的平台版本，建議您使用此選項。

  使用存取項目從叢集外部管理 IAM 主體的 Kubernetes 許可。您可以使用 EKS API、 AWS 命令列界面、 AWS SDKs、 AWS CloudFormation 和 新增和管理對叢集的存取 AWS Management Console。這意味著您可以使用建立叢集時使用的工具來管理使用者。

  若要開始使用，請遵循 eks/latest/userguide/setting-up-access-entries.html【Change authentication mode to use access entry，type="documentation"】，然後遷移現有的 aws-auth ConfigMap 項目來存取項目。
- 將項目新增至 **`aws-authConfigMap`**：如果您叢集的平台版本早於先決條件區段中列出的版本，則必須使用此選項。如果您叢集的平台版本等於或晚於叢集 Kubernetes 版本先決條件區段中列出的平台版本，且您已將項目新增至 `ConfigMap`，則建議您將這些項目遷移至存取項目。不過，您無法遷移 Amazon EKS 新增至 的項目`ConfigMap`，例如與受管節點群組或 Fargate 設定檔搭配使用的 IAM 角色項目。如需詳細資訊，請參閱the section called "Kubernetes API 存取"。
  - 如果必須使用 `aws-auth ConfigMap` 選項，則可以使用 `eksctl create iamidentitymapping` 命令將項目新增至 `ConfigMap`。如需詳細資訊，請參閱 `eksctl` 文件中的 Manage IAM users and roles 一節。

設定叢集身分驗證模式

每個叢集都有自己的身分驗證模式。身分驗證模式會決定您可以使用哪些方法來允許 IAM 主體存取叢集上的 Kubernetes 物件。身分驗證模式有 3 種。

> **Important**
>
> 啟用存取項目方法後，就無法停用。
> 如果在叢集建立期間未啟用 ConfigMap方法，則稍後無法啟用。在引入存取項目之前建立的所有叢集都已啟用 ConfigMap方法。
> 如果您搭配叢集使用混合節點，則必須使用 API或 API_AND_CONFIG_MAP 叢集身分驗證模式。

叢集aws-authConfigMap內的

這是 Amazon EKS 叢集的原始身分驗證模式。建立叢集的 IAM 主體是可以使用 kubectl 存取叢集的初始使用者。初始使用者必須將其他使用者新增至 aws-auth ConfigMap 的清單中，並為這些使用者指派相應許可。這些其他使用者無法管理或移除初始使用者，因為 中沒有ConfigMap要管理的項目。

ConfigMap和 存取項目

使用這種身分驗證模式時，您可以使用這兩種方法將 IAM 主體新增至叢集。請注意，每個方法都會存放個別的項目；例如，如果您從 AWS CLI 新增存取項目，aws-authConfigMap則 不會更新。

僅存取項目

透過此身分驗證模式，您可以使用 EKS API、 AWS 命令列界面、 AWS SDKs、 AWS CloudFormation 和 AWS Management Console 來管理 IAM 主體對叢集的存取。

每個存取項目都有一個類型，您可以使用存取範圍的組合將主體限制為特定命名空間，並使用存取政策來設定預先設定的可重複使用許可政策。或者，您可以使用 STANDARD 類型和 Kubernetes RBAC 群組來指派自訂許可。

身分驗證方式	方法
僅 ConfigMap （CONFIG_MAP ）	aws-auth ConfigMap

身分驗證方式	方法
EKS API 和 `ConfigMap` (`API_AND_CONFIG_MAP`)	EKS API、 AWS 命令列界面、 AWS SDKs、AWS CloudFormation 和 AWS Management Console 中的存取項目 `aws-auth ConfigMap`
僅 EKS API (`API`)	EKS API、 AWS 命令列界面、 AWS SDKs、AWS CloudFormation 和 中的存取項目 AWS Management Console

> **Note**
>
> Amazon EKS Auto Mode 需要存取項目。

授予 IAM 使用者使用 EKS 存取項目存取 Kubernetes

本節旨在向您展示如何使用存取項目和政策，管理 Amazon Elastic Kubernetes Service (EKS) 中 Kubernetes 叢集的 IAM 主體存取。您可以找到變更身分驗證模式、從舊版 `aws-auth` ConfigMap 項目遷移、建立、更新和刪除存取項目、將政策與項目建立關聯、檢閱預先定義的政策許可，以及安全存取管理的金鑰先決條件和考量等詳細資訊。

概觀

EKS 存取項目是授予使用者存取 Kubernetes API 的最佳方式。例如，您可以使用存取項目來授予開發人員使用 kubectl 的存取權。基本上，EKS 存取項目會將一組 Kubernetes 許可與 IAM 身分建立關聯，例如 IAM 角色。例如，開發人員可能會擔任 IAM 角色，並使用該角色向 EKS 叢集進行身分驗證。

功能

- 集中式身分驗證和授權：直接透過 Amazon EKS APIs 控制對 Kubernetes 叢集的存取，無需在 AWS 和 Kubernetes APIs 之間切換以取得使用者許可。
- 精細許可管理：使用存取項目和政策來定義 IAM AWS 主體的精細許可，包括修改或撤銷建立者的叢集管理員存取權。
- IaC 工具整合：支援基礎設施做為程式碼工具，例如 AWS CloudFormation、Terraform 和 AWS CDK，以在叢集建立期間定義存取組態。

- 錯誤組態復原：允許透過 Amazon EKS API 還原叢集存取，而不需要直接存取 Kubernetes API。
- 減少額外負荷和增強安全性：集中操作以降低額外負荷，同時利用 CloudTrail 稽核記錄和多重要素驗證等 AWS IAM 功能。

如何連接許可

您可以透過兩種方式連接 Kubernetes 存取項目的許可：

- 使用存取政策。存取政策是由 維護的預先定義 Kubernetes 許可範本 AWS。如需詳細資訊，請參閱the section called “檢閱存取政策”。
- 參考 Kubernetes 群組。如果您將 IAM Identity 與 Kubernetes 群組建立關聯，您可以建立授予群組許可的 Kubernetes 資源。如需詳細資訊，請參閱 Kubernetes 文件中的使用 RBAC 授權。

考量事項

在現有叢集上啟用 EKS 存取項目時，請記住下列事項：

- 傳統叢集行為：對於在引入存取項目之前建立的叢集（具有早於平台版本需求中指定之初始平台版本的叢集），EKS 會自動建立反映預先存在許可的存取項目。此項目包含最初建立叢集的 IAM 身分，以及在叢集建立期間授予該身分的管理許可。
- 處理舊版 **`aws-auth`** ConfigMap：如果您的叢集依賴舊版 `aws-auth` ConfigMap 進行存取管理，則只會在啟用存取項目時自動建立原始叢集建立者的存取項目。新增至 ConfigMap 的其他角色或許可（例如，開發人員或服務的自訂 IAM 角色）不會自動遷移。若要解決此問題，請手動建立對應的存取項目。

開始使用

1. 決定您要使用的 IAM Identity and Access 政策。
   - the section called “檢閱存取政策”
2. 在您的叢集上啟用 EKS 存取項目。確認您有支援的平台版本。
   - the section called “身分驗證方式”
3. 建立將 IAM Identity 與 Kubernetes 許可建立關聯的存取項目。
   - the section called “建立存取項目”
4. 使用 IAM 身分驗證叢集。
   - the section called “設定 AWS CLI”

- the section called “設定 kubectl 和 eksctl”

將存取政策與存取項目建立關聯

您可以向類型為 STANDARD 的存取項目指派一個或多個存取政策。Amazon EKS 會自動向其他類型的存取項目授予在叢集中正常運作所需的許可。Amazon EKS 存取政策包含 Kubernetes 許可，而非 IAM 許可。將存取政策與存取項目建立關聯之前，請確定您已熟悉每個存取政策中包含的 Kubernetes 許可。如需詳細資訊，請參閱the section called “檢閱存取政策”。如果沒有任何存取政策符合您的需求，則請勿將存取政策與存取項目建立關聯。反之，請為存取項目指定一或多個群組名稱，並建立和管理 Kubernetes 角色型存取控制物件。如需詳細資訊，請參閱the section called “建立存取項目”。

- 現有的存取項目。若要建立服務角色，請參閱the section called “建立存取項目”。
- Identity AWS and Access Management 角色或具有下列許可的使用者：`ListAccessEntries`、`DescribeAccessEntry`、`UpdateAccessEntryListAccessPolicies`、`AssociateAccessPolicy`和 `DisassociateAccessPolicy`。如需詳細資訊，請參閱《服務授權參考》中的 Amazon Elastic Kubernetes Service 定義的動作。

在將存取政策與存取項目關聯之前，請考量以下要求：

- 每個存取項目可以關聯多個存取政策，但只能將每個政策與一個存取項目關聯一次。如果您關聯多個存取政策，則存取項目的 IAM 主體具有所有包含在所有關聯存取政策中的所有許可。
- 您可以指定一或多個 Kubernetes 命名空間的名稱，將存取政策範圍限定在叢集上的所有資源。您可以將萬用字元用於命名空間名稱。例如，如果要將存取政策的適用範圍限定為以 `dev-` 開頭的所有命名空間，則可以指定 `dev-*` 作為命名空間名稱。確認叢集上存在相應命名空間，並且拼字與叢集上的實際命名空間名稱相符。Amazon EKS 不會確認叢集上命名空間的拼字或存在。
- 將存取政策關聯到存取項目後，您可以變更存取政策的存取範圍。如果您已將存取政策範圍限定為 Kubernetes 命名空間，您可以視需要新增和移除關聯的命名空間。
- 如果您將存取政策關聯到也指定了群組名稱的存取項目，則相應 IAM 主體擁有所有關聯存取政策中的所有許可。它還具有任何 Kubernetes 中指定的所有許可`Role`或`ClusterRole`物件，`Role`以及指定群組名稱的`RoleBinding`物件。
- 如果您執行 `kubectl auth can-i --list`命令，則不會看到任何由存取政策指派的 Kubernetes 許可，該政策與您執行命令時使用的 IAM 主體的存取項目相關聯。命令只會顯示您在 Kubernetes 中授予的 Kubernetes 許可，`Role`或您已繫結至您為存取項目指定的群組名稱或使用者名稱的`ClusterRole`物件。

- 如果您在與叢集上的 Kubernetes 物件互動時模擬 Kubernetes 使用者或群組，例如搭配 `--as username` 或 使用 kubectl命令`--as-group group-name` ，則表示您強制使用 Kubernetes RBAC 授權。因此，相應 IAM 主體沒有與相應存取項目關聯的任何存取政策指派的許可。IAM 主體模擬的使用者或群組唯一擁有的 Kubernetes 許可，是您在 Kubernetes 中授予的 Kubernetes 許可，`Role`或是您已繫結至群組名稱或使用者名稱的`ClusterRole`物件。若要讓您的 IAM 主體在相關聯的存取政策中擁有許可，請勿模擬 Kubernetes 使用者或群組。IAM 委託人仍會擁有您在 Kubernetes 中授予的任何許可，`Role`或您已繫結至您為存取項目指定的群組名稱或使用者名稱的`ClusterRole`物件。如需詳細資訊，請參閱 Kubernetes 文件中的使用者模擬。

您可以使用 AWS Management Console 或 AWS CLI 將存取政策與存取項目建立關聯。

AWS Management Console

1. 開啟 Amazon EKS 主控台。
2. 選擇要將存取政策關聯到的存取項目所在的叢集的名稱。
3. 選擇存取索引標籤。
4. 如果存取項目的類型為標準，則可以將 Amazon EKS存取政策與之關聯或取消關聯。如果您的存取項目類型是標準以外的任何項目，則無法使用此選項。
5. 選擇關聯存取政策。
6. 對於政策名稱，選取具有您希望相應 IAM 主體擁有的許可的政策。若要查看每個政策包含的許可，請參閱 the section called “檢閱存取政策”。
7. 對於存取範圍，選擇存取範圍。如果您選擇叢集，則存取政策中的許可會授予所有 Kubernetes 命名空間中資源的 IAM 主體。如果您選擇 Kubernetes 命名空間，則可以選擇新增命名空間。在出現的命名空間欄位中，您可以輸入叢集上 Kubernetes 命名空間的名稱。如果您希望相應 IAM 主體擁有跨多個命名空間的許可，則可以輸入多個命名空間。
8. 選擇新增存取政策。

AWS CLI

1. 在您的裝置或 AWS CloudShell 上安裝和設定的 AWS 命令列界面 (AWS CLI) 版本 `1.27.160` `2.12.3`或更新版本。若要檢查您目前的版本，請使用 `aws --version | cut -d / -f2 | cut -d ' ' -f1`。適用於 macOS 的 `yum`、 `apt-get`或 Homebrew 等套件管理員通常是最新版本 CLI AWS 後面的數個版本。若要安裝最新版本，請參閱《 AWS 命令列界面使用者指南》中的使用 aws 設定安裝 和 快速組態。 https://docs.aws.amazon.com/cli/latest/userguide/cli-configure-quickstart.html#cli-configure-quickstart-config在 AWS CloudShell 中安裝的 AWS CLI 版本也可能是

最新版本後面的幾個版本。若要更新它，請參閱《CloudShell AWS 使用者指南》中的將 CLI 安裝到您的主目錄。 AWS CloudShell

2. 檢視可用的存取政策。

```
aws eks list-access-policies --output table
```

範例輸出如下。

```
-------------------------------------------------------------------------------------------------
|                                          ListAccessPolicies
                  |
+-----------------------------------------------------------------------------------------------
+
||                                            accessPolicies
                 ||
|+-------------------------------------------------------------------
+-------------------------------+|
||                                arn                                 |
 name              ||
|+-------------------------------------------------------------------
+-------------------------------+|
||  {arn-aws}eks::aws:cluster-access-policy/AmazonEKSAdminPolicy        |
 AmazonEKSAdminPolicy         ||
||  {arn-aws}eks::aws:cluster-access-policy/AmazonEKSClusterAdminPolicy |
 AmazonEKSClusterAdminPolicy  ||
||  {arn-aws}eks::aws:cluster-access-policy/AmazonEKSEditPolicy         |
 AmazonEKSEditPolicy          ||
||  {arn-aws}eks::aws:cluster-access-policy/AmazonEKSViewPolicy         |
 AmazonEKSViewPolicy          ||
|+-------------------------------------------------------------------
+-------------------------------+|
```

若要查看每個政策包含的許可，請參閱 the section called “檢閱存取政策”。

3. 檢視現有的存取項目。使用您叢集的名稱取代 *my-cluster*。

```
aws eks list-access-entries --cluster-name my-cluster
```

範例輸出如下。

```
{
```

```
    "accessEntries": [
        "arn:aws: iam::111122223333:role/my-role",
        "arn:aws: iam::111122223333:user/my-user"
    ]
}
```

4. 將存取政策與存取項目關聯。以下範例將 AmazonEKSViewPolicy 存取政策與一個存取項目關聯。每當 *my-role* IAM 角色嘗試存取叢集上的 Kubernetes 物件時，Amazon EKS 將授權該角色使用政策中的許可，僅存取 *my-namespace1* 和 *my-namespace2* Kubernetes 命名空間中的 Kubernetes 物件。將 *my-cluster* 取代為您的叢集名稱、將 *111122223333* 取代為您的 AWS 帳戶 ID，並將 *my-role* 取代為您希望 Amazon EKS 授權存取 Kubernetes 叢集物件的 IAM 角色名稱。

```
aws eks associate-access-policy --cluster-name my-cluster --principal-arn arn:aws:
 iam::111122223333:role/my-role \
    --access-scope type=namespace,namespaces=my-namespace1,my-namespace2 --policy-arn
 arn:aws: eks::aws:cluster-access-policy/AmazonEKSViewPolicy
```

如果您希望相應 IAM 主體擁有適用於整個叢集的許可，請用 `type=cluster` 取代 `type=namespace,namespaces=`*my-namespace1*`,`*my-namespace2* 。如果要將多個存取政策關聯到該存取項目，請多次執行該命令，並每次使用不同的存取政策。每個關聯的存取政策都有自身的適用範圍。

> **Note**
>
> 如果您稍後想要變更關聯的存取政策的適用範圍，請再次執行先前的命令並指定新的適用範圍。例如，如果您想要移除 *my-namespace2*，則`type=namespace,namespaces=`*my-namespace1* 只會使用 再次執行命令。如果您想要將範圍從 變更為 `namespace` `cluster`，則可以再次使用 執行命令`type=cluster`，移除 `type=namespace,namespaces=`*my-namespace1*`,`*my-namespace2* 。

5. 確定哪些存取政策與某個存取項目關聯。

```
aws eks list-associated-access-policies --cluster-name my-cluster --principal-arn
 arn:aws: iam::111122223333:role/my-role
```

範例輸出如下。

```
{
```

```
    "clusterName": "my-cluster",
    "principalArn": "arn:aws: iam::111122223333",
    "associatedAccessPolicies": [
        {
            "policyArn": "arn:aws: eks::aws:cluster-access-policy/
AmazonEKSViewPolicy",
            "accessScope": {
                "type": "cluster",
                "namespaces": []
            },
            "associatedAt": "2023-04-17T15:25:21.675000-04:00",
            "modifiedAt": "2023-04-17T15:25:21.675000-04:00"
        },
        {
            "policyArn": "arn:aws: eks::aws:cluster-access-policy/
AmazonEKSAdminPolicy",
            "accessScope": {
                "type": "namespace",
                "namespaces": [
                    "my-namespace1",
                    "my-namespace2"
                ]
            },
            "associatedAt": "2023-04-17T15:02:06.511000-04:00",
            "modifiedAt": "2023-04-17T15:02:06.511000-04:00"
        }
    ]
}
```

在上述範例中，此存取項目的 IAM 主體具有叢集上所有命名空間的檢視許可，以及兩個 Kubernetes 命名空間的管理員許可。

6. 將存取政策與存取項目取消關聯。在此範例中，AmazonEKSAdminPolicy 政策與存取項目取消關聯。但是，相應 IAM 主體保留 AmazonEKSViewPolicy 存取政策中對 *my-namespace1* 和 *my-namespace2* 命名空間中物件的許可，因為該存取政策並未與該存取項目取消關聯。

```
aws eks disassociate-access-policy --cluster-name my-cluster --principal-arn arn:aws:
 iam::111122223333:role/my-role \
    --policy-arn arn:aws: eks::aws:cluster-access-policy/AmazonEKSAdminPolicy
```

若要列出可用的存取政策，請參閱 the section called “檢閱存取政策”。

遷移現有 aws-auth ConfigMap 項目至存取項目

如果您已將項目新增至叢集aws-authConfigMap上的 ，建議您為 aws-auth 中的現有項目建立存取項目ConfigMap。建立存取項目後，您可以從 ConfigMap 中移除對應項目。您無法將存取政策與 aws-auth 中的項目建立關聯ConfigMap。如果您想要將存取政策與 IAM 主體相關聯，則請建立存取項目。

> **Important**
>
> - 當叢集處於API_AND_CONFIGMAP身分驗證模式，且在 aws-authConfigMap和存取項目中都有相同 IAM 角色的映射時，該角色將使用存取項目的映射進行身分驗證。存取項目優先於相同 IAM 主體ConfigMap的項目。
> - 在將 Amazon EKS 為受管節點群組或 Fargate 設定檔建立的現有aws-authConfigMap項目移除至叢集之前，請仔細檢查 Amazon EKS 叢集中是否存在這些特定資源的正確存取項目。如果您移除 Amazon EKS 在 中建立ConfigMap但沒有同等存取項目的項目，您的叢集將無法正常運作。

先決條件

- 熟悉存取項目和存取政策。如需詳細資訊，請參閱the section called “授予許可”及the section called “關聯存取政策”。
- 平台版本等於或高於 the section called “授予許可”主題先決條件中所列版本的現有叢集。
- 裝置或 AWS CloudShell 上安裝的eksctl命令列工具版本 0.210.0或更新版本。如需有關安裝或更新 eksctl 的指示，請參閱 eksctl 文件中的安裝一節。
- 修改kube-system命名空間aws-authConfigMap中 的 Kubernetes 許可。
- Identity AWS and Access Management 角色或具有下列許可的使用者： CreateAccessEntry和 ListAccessEntries。如需詳細資訊，請參閱《服務授權參考》中的 Amazon Elastic Kubernetes Service 定義的動作一節。

eksctl

1. 檢視 aws-auth ConfigMap 中的現有項目。使用您叢集的名稱取代 *my-cluster*。

```
eksctl get iamidentitymapping --cluster my-cluster
```

範例輸出如下。

```
ARN
           USERNAME                                      GROUPS
                        ACCOUNT
arn:aws: iam::111122223333:role/EKS-my-cluster-Admins
            Admins                                        system:masters
arn:aws: iam::111122223333:role/EKS-my-cluster-my-namespace-Viewers
            my-namespace-Viewers                          Viewers
arn:aws: iam::111122223333:role/EKS-my-cluster-self-managed-ng-1
            system:node:{{EC2PrivateDNSName}}       system:bootstrappers,system:nodes
arn:aws: iam::111122223333:user/my-user
            my-user
arn:aws: iam::111122223333:role/EKS-my-cluster-fargateprofile1
                              system:node:{{SessionName}}
 system:bootstrappers,system:nodes,system:node-proxier
arn:aws: iam::111122223333:role/EKS-my-cluster-managed-ng
            system:node:{{EC2PrivateDNSName}}       system:bootstrappers,system:nodes
```

2. the section called “建立存取項目” 對於您在上一個輸出中傳回的任何ConfigMap項目。建立存取項目時，請確保為 ARN、USERNAME、GROUPS 和 ACCOUNT 指定與輸出中傳回的值相同的值。在範例輸出中，您將為最後兩個項目之外的所有項目建立存取項目，因為這些項目是由 Amazon EKS 為 Fargate 描述檔和受管節點群組建立的。
3. 從 ConfigMap 中刪除您建立的任何存取項目對應的項目。如果您未從 刪除項目ConfigMap，IAM 主體 ARN 存取項目的設定會覆寫ConfigMap項目。將 *111122223333* 取代為 AWS 您的帳戶 ID，並將 *EKS-my-cluster-my-namespace-Viewers* 取代為 中項目中角色的名稱ConfigMap。如果您要移除的項目適用於 IAM 使用者，而非 IAM 角色，請將 role 取代為 user，並將 *EKS-my-cluster-my-namespace-Viewers* 取代為 使用者名稱。

```
eksctl delete iamidentitymapping --arn arn:aws: iam::111122223333:role/EKS-my-
cluster-my-namespace-Viewers --cluster my-cluster
```

檢閱存取政策許可

存取政策rules包括包含 Kubernetes verbs（許可）和 的 resources。存取政策不包含 IAM 許可或資源。與 Kubernetes Role和 ClusterRole 物件類似，存取政策僅包含 allow rules。您無法修改存取政策的內容。您無法建立自己的存取政策。如果存取政策中的許可不符合您的需求，請建立 Kubernetes RBAC 物件並指定存取項目的群組名稱。如需詳細資訊，請參閱the section called “建立存

取項目”。存取政策中包含的許可類似於面向 Kubernetes 使用者的叢集角色中的許可。如需詳細資訊，請參閱 Kubernetes 文件中的面向使用者的角色。

列出所有政策

使用此頁面列出的任一存取政策，或使用 CLI AWS 擷取所有可用存取政策的清單：

```
aws eks list-access-policies
```

預期的輸出看起來應該像這樣（為了簡潔起見而簡化）：

```
{
    "accessPolicies": [
        {
            "name": "AmazonAIOpsAssistantPolicy",
            "arn": "arn:aws:eks::aws:cluster-access-policy/AmazonAIOpsAssistantPolicy"
        },
        {
            "name": "AmazonARCRegionSwitchScalingPolicy",
            "arn": "arn:aws:eks::aws:cluster-access-policy/
AmazonARCRegionSwitchScalingPolicy"
        },
        {
            "name": "AmazonEKSAdminPolicy",
            "arn": "arn:aws:eks::aws:cluster-access-policy/AmazonEKSAdminPolicy"
        },
        {
            "name": "AmazonEKSAdminViewPolicy",
            "arn": "arn:aws:eks::aws:cluster-access-policy/AmazonEKSAdminViewPolicy"
        },
        {
            "name": "AmazonEKSAutoNodePolicy",
            "arn": "arn:aws:eks::aws:cluster-access-policy/AmazonEKSAutoNodePolicy"
        }
        // Additional policies omitted
    ]
}
```

AmazonEKSAdminPolicy

此存取政策包含向相應 IAM 主體授予對資源的大部分許可的許可。與存取項目相關聯時，其存取範圍通常是一或多個 Kubernetes 命名空間。如果您希望相應 IAM 主體對叢集上的所有資源具有管理員存取權，請將 the section called “AmazonEKSClusterAdminPolicy” 存取政策與相應存取項目關聯。

ARN – arn:aws: eks::aws:cluster-access-policy/AmazonEKSAdminPolicy

Kubernetes API 群組	Kubernetes 資源	Kubernetes 動詞（許可）
apps	daemonsets , deployments , deployments/rollback , deployments/scale , replicasets , replicasets/scale, statefulsets , statefulsets/scale	create, delete, deletecollection , patch, update
apps	controllerrevisions , daemonsets , daemonsets/status , deployments , deployments/scale , deployments/status , replicasets , replicasets/scale , replicasets/status , statefulsets , statefulsets/scale , statefulsets/status	get, list, watch
authorization.k8s.io	localsubjectaccessreviews	create
autoscaling	horizontalpodautoscalers	create, delete, deletecollection , patch, update
autoscaling	horizontalpodautoscalers , horizontalpodautoscalers/status	get, list, watch
batch	cronjobs, jobs	create, delete, deletecollection , patch, update

Kubernetes API 群組	Kubernetes 資源	Kubernetes 動詞（許可）
batch	cronjobs, cronjobs/status , jobs, jobs/status	get, list, watch
discovery.k8s.io	endpointslices	get, list, watch
extensions	daemonsets , deployments , deployments/rollback , deployments/scale , ingresses , networkpolicies , replicasets , replicasets/scale , replicationcontrollers/scale	create, delete, deletecollection , patch, update
extensions	daemonsets , daemonsets/status , deployments , deployments/scale , deployments/status , ingresses , ingresses/status , networkpolicies , replicasets , replicasets/scale , replicasets/status , replicationcontrollers/scale	get, list, watch
networking.k8s.io	ingresses , ingresses/status , networkpolicies	get, list, watch
networking.k8s.io	ingresses , networkpolicies	create, delete, deletecollection , patch, update
policy	poddisruptionbudgets	create, delete, deletecollection , patch, update

Kubernetes API 群組	Kubernetes 資源	Kubernetes 動詞（許可）
policy	poddisruptionbudgets , poddisruptionbudgets/status	get, list, watch
rbac.authorization.k8s.io	rolebindings , roles	create, delete, deletecollection , get, list, patch, update, watch
	configmaps , endpoints , persistentvolumeclaims , persistentvolumeclaims/status , pods, replicationcontrollers , replicationcontrollers/scale , serviceaccounts , services, services/status	get,list, watch
	pods/attach , pods/exec , pods/portforward , pods/proxy , secrets, services/proxy	get, list, watch
	configmaps , events, persistentvolumeclaims , replicationcontrollers , replicationcontrollers/scale , secrets, serviceaccounts , services, services/proxy	create, delete, deletecollection , patch, update

Kubernetes API 群組	Kubernetes 資源	Kubernetes 動詞（許可）
	pods, pods/attach , pods/exec , pods/port forward , pods/proxy	create, delete, deletecollection , patch, update
	serviceaccounts	impersonate
	bindings, events, limitranges , namespaces/status , pods/log, pods/status , replicationcontrollers/status , resourcequotas , resourcequotas/status	get, list, watch
	namespaces	get,list, watch

AmazonEKSClusterAdminPolicy

此存取政策包含授予相應 IAM 主體對叢集的管理員存取權的許可。與存取項目相關聯時，其存取範圍通常是叢集，而不是 Kubernetes 命名空間。如果您希望限制相應 IAM 主體的管理範圍，請考慮將 the section called “AmazonEKSAdminPolicy” 存取政策與相應存取項目關聯。

ARN – arn:aws: eks::aws:cluster-access-policy/AmazonEKSClusterAdminPolicy

Kubernetes API 群組	Kubernetes nonResourceURLs	Kubernetes 資源	Kubernetes 動詞（許可）
*		*	*
	*		*

AmazonEKSAdminViewPolicy

此存取政策包含許可，授予 IAM 主體列出/檢視叢集中所有資源的存取權。請注意，這包含 Kubernetes 秘密。

ARN – arn:aws: eks::aws:cluster-access-policy/AmazonEKSAdminViewPolicy

Kubernetes API 群組	Kubernetes 資源	Kubernetes 動詞（許可）
*	*	get, list, watch

AmazonEKSEditPolicy

此存取政策包含允許 IAM 主體編輯大多數 Kubernetes 資源的許可。

ARN – arn:aws: eks::aws:cluster-access-policy/AmazonEKSEditPolicy

Kubernetes API 群組	Kubernetes 資源	Kubernetes 動詞（許可）
apps	daemonsets , deployments , deployments/rollback , deployments/scale , replicasets , replicasets/scale, statefulsets , statefulsets/scale	create, delete, deletecollection , patch, update
apps	controllerrevisions , daemonsets , daemonsets/status , deployments , deployments/scale , deployments/status , replicasets , replicasets/scale , replicasets/status , statefulsets , statefulsets/scale , statefulsets/status	get, list, watch
autoscaling	horizontalpodautoscalers , horizonta	get, list, watch

Kubernetes API 群組	Kubernetes 資源	Kubernetes 動詞（許可）
	lpodautoscalers/status	
autoscaling	horizontalpodautoscalers	create, delete, deletecollection , patch, update
batch	cronjobs, jobs	create, delete, deletecollection , patch, update
batch	cronjobs, cronjobs/status , jobs, jobs/status	get, list, watch
discovery.k8s.io	endpointslices	get, list, watch
extensions	daemonsets , deployments , deployments/rollback , deployments/scale , ingresses , networkpolicies , replicasets , replicasets/scale , replicationcontrollers/scale	create, delete, deletecollection , patch, update
extensions	daemonsets , daemonsets/status , deployments , deployments/scale , deployments/status , ingresses , ingresses/status , networkpolicies , replicasets , replicasets/scale , replicasets/status , replicationcontrollers/scale	get, list, watch

Kubernetes API 群組	Kubernetes 資源	Kubernetes 動詞（許可）
networking.k8s.io	ingresses , networkpolicies	create, delete, deletecollection , patch, update
networking.k8s.io	ingresses , ingresses/status , networkpolicies	get, list, watch
policy	poddisruptionbudgets	create, delete, deletecollection , patch, update
policy	poddisruptionbudgets , poddisruptionbudgets/status	get, list, watch
	namespaces	get, list, watch
	pods/attach , pods/exec , pods/portforward , pods/proxy , secrets, services/proxy	get, list, watch
	serviceaccounts	impersonate
	pods, pods/attach , pods/exec , pods/portforward , pods/proxy	create, delete, deletecollection , patch, update
	configmaps , events, persistentvolumeclaims , replicationcontrollers , replicationcontrollers/scale , secrets, serviceaccounts , services, services/proxy	create, delete, deletecollection , patch, update

Kubernetes API 群組	Kubernetes 資源	Kubernetes 動詞（許可）
	configmaps , endpoints , persistentvolumeclaims , persistentvolumeclaims/status , pods, replicationcontrollers , replicationcontrollers/scale , serviceaccounts , services, services/status	get, list, watch
	bindings, events, limitranges , namespaces/status , pods/log, pods/status , replicationcontrollers/status , resourcequotas , resourcequotas/status	get, list, watch

AmazonEKSViewPolicy

此存取政策包含允許 IAM 主體檢視大多數 Kubernetes 資源的許可。

ARN – arn:aws: eks::aws:cluster-access-policy/AmazonEKSViewPolicy

Kubernetes API 群組	Kubernetes 資源	Kubernetes 動詞（許可）
apps	controllerrevisions , daemonsets , daemonsets/status , deployments , deployments/scale , deployments/status , replicasets , replicasets/scale , replicase	get, list, watch

Kubernetes API 群組	Kubernetes 資源	Kubernetes 動詞（許可）
	ts/status ,statefuls ets ,statefulsets/ scale ,statefulsets/ status	
autoscaling	horizontalpodautos calers ,horizonta lpodautoscalers/st atus	get, list, watch
batch	cronjobs, cronjobs/ status ,jobs, jobs/stat us	get, list, watch
discovery.k8s.io	endpointslices	get, list, watch
extensions	daemonsets ,daemonset s/status ,deploymen ts ,deployments/scale , deployments/status , ingresses ,ingresses /status ,networkpo licies ,replicasets , replicasets/scale , replicasets/status ,replicationcontrol lers/scale	get, list, watch
networking.k8s.io	ingresses ,ingresses /status ,networkpo licies	get, list, watch
policy	poddisruptionbudgets , poddisruptionbudge ts/status	get, list, watch

Kubernetes API 群組	Kubernetes 資源	Kubernetes 動詞（許可）
	configmaps ,endpoints ,persistentvolumeclaims ,persistentvolumeclaims/status ,pods, replicationcontrollers ,replicationcontrollers/scale ,serviceaccounts ,services, services/status	get, list, watch
	bindings、events、limitranges 、namespaces/status、pods/log、pods/status 、replicationcontrollers/status 、resourcequotas 、、resourcequotas/status	get, list, watch
	namespaces	get, list, watch

AmazonEKSAutoNodePolicy

ARN – arn:aws: eks::aws:cluster-access-policy/AmazonEKSAutoNodePolicy

此政策包含下列許可，允許 Amazon EKS 元件完成下列任務：

- kube-proxy – 監控網路端點和服務，並管理相關事件。這可啟用整個叢集的網路代理功能。
- ipamd – 管理 AWS VPC 網路資源和容器網路介面 (CNI)。這可讓 IP 地址管理協助程式處理 Pod 網路。
- coredns – 存取服務探索資源，例如端點和服務。這會在叢集內啟用 DNS 解析。
- ebs-csi-driver – 使用 Amazon EBS 磁碟區的儲存相關資源。這允許動態佈建和連接持久性磁碟區。

- neuron – 監控 AWS Neuron 裝置的節點和 Pod。這可讓您管理 AWS Inferentia 和 Trainium 加速器。
- node-monitoring-agent – 存取節點診斷和事件。這可啟用叢集運作狀態監控和診斷收集。

每個元件使用專用服務帳戶，且僅限於其特定函數所需的許可。

如果您在 NodeClass 中手動指定節點 IAM 角色，則需要建立將新節點 IAM 角色與此存取政策建立關聯的存取項目。

AmazonEKSBlockStoragePolicy

> **Note**
>
> 此政策僅針對 AWS 服務連結角色指定，無法與客戶受管角色搭配使用。

ARN – arn:aws: eks::aws:cluster-access-policy/AmazonEKSBlockStoragePolicy

此政策包含允許 Amazon EKS 管理儲存操作領導者選擇和協調資源的許可：

- coordination.k8s.io – 為領導者選擇建立和管理租用物件。這可讓 EKS 儲存元件透過領導者選擇機制協調其跨叢集的活動。

此政策的範圍限定為 EKS 儲存元件使用的特定租用資源，以防止對叢集中其他協調資源的衝突存取。

啟用自動模式時，Amazon EKS 會自動為叢集 IAM 角色建立具有此存取政策的存取項目，以確保區塊儲存功能具備正常運作所需的許可。

AmazonEKSLoadBalancingPolicy

ARN – arn:aws: eks::aws:cluster-access-policy/AmazonEKSLoadBalancingPolicy

此政策包含允許 Amazon EKS 管理領導者選擇資源以進行負載平衡的許可：

- coordination.k8s.io – 為領導者選擇建立和管理租用物件。這可讓 EKS 負載平衡元件透過選擇領導者來協調多個複本的活動。

此政策的範圍專門用於負載平衡租用資源，以確保適當的協調，同時防止存取叢集中的其他租用資源。

啟用自動模式時，Amazon EKS 會自動為叢集 IAM 角色建立具有此存取政策的存取項目，以確保網路功能具備正常運作所需的許可。

AmazonEKSNetworkingPolicy

ARN – `arn:aws: eks::aws:cluster-access-policy/AmazonEKSNetworkingPolicy`

此政策包含允許 Amazon EKS 管理聯網領導者選擇資源的許可：

- `coordination.k8s.io` – 為領導者選擇建立和管理租用物件。這可讓 EKS 網路元件透過選擇領導者來協調 IP 地址分配活動。

政策的範圍特別限定於聯網租用資源，以確保適當的協調，同時防止存取叢集中的其他租用資源。

啟用自動模式時，Amazon EKS 會自動為叢集 IAM 角色建立具有此存取政策的存取項目，以確保網路功能具備正常運作所需的許可。

AmazonEKSComputePolicy

> **Note**
>
> 此政策僅針對 AWS 服務連結角色指定，無法與客戶受管角色搭配使用。

ARN – `arn:aws: eks::aws:cluster-access-policy/AmazonEKSComputePolicy`

此政策包含允許 Amazon EKS 管理運算操作領導者選擇資源的許可：

- `coordination.k8s.io` – 為領導者選擇建立和管理租用物件。這可讓 EKS 運算元件透過選擇領導者來協調節點擴展活動。

此政策的範圍專門用於運算管理租用資源，同時允許叢集中所有租用資源的基本讀取存取 (`get`、`watch`)。

啟用自動模式時，Amazon EKS 會自動為叢集 IAM 角色建立具有此存取政策的存取項目，以確保網路功能具備正常運作所需的許可。

AmazonEKSBlockStorageClusterPolicy

> **Note**
>
> 此政策僅針對 AWS 服務連結角色指定，無法與客戶受管角色搭配使用。

ARN – `arn:aws: eks::aws:cluster-access-policy/AmazonEKSBlockStorageClusterPolicy`

此政策會授予 Amazon EKS Auto Mode 區塊儲存功能所需的許可。它可有效管理 Amazon EKS 叢集內的區塊儲存資源。此政策包含下列許可：

CSI 驅動程式管理：

- 建立、讀取、更新和刪除 CSI 驅動程式，特別是區塊儲存。

磁碟區管理：

- 列出、監看、建立、更新、修補和刪除持久性磁碟區。
- 列出、監看和更新持久性磁碟區宣告。
- 修補持久性磁碟區宣告狀態。

節點和 Pod 互動：

- 讀取節點和 Pod 資訊。
- 管理與儲存操作相關的事件。

儲存類別和屬性：

- 讀取儲存類別和 CSI 節點。
- 讀取磁碟區屬性類別。

磁碟區附件：

- 列出、監看和修改磁碟區附件及其狀態。

快照操作：

- 管理磁碟區快照、快照內容和快照類別。
- 處理磁碟區群組快照和相關資源的操作。

此政策旨在支援在自動模式下執行的 Amazon EKS 叢集內的全面區塊儲存管理。它結合了各種操作的許可，包括區塊儲存磁碟區的佈建、連接、調整大小和快照。

啟用自動模式時，Amazon EKS 會自動為叢集 IAM 角色使用此存取政策建立存取項目，以確保區塊儲存功能具備正常運作所需的許可。

AmazonEKSComputeClusterPolicy

> **Note**
>
> 此政策僅針對 AWS 服務連結角色指定，無法與客戶受管角色搭配使用。

ARN – `arn:aws: eks::aws:cluster-access-policy/AmazonEKSComputeClusterPolicy`

此政策會授予 Amazon EKS Auto Mode 運算管理功能所需的許可。它可有效協調和擴展 Amazon EKS 叢集中的運算資源。此政策包含下列許可：

節點管理：

- 建立、讀取、更新、刪除和管理 NodePools 和 NodeClaims 的狀態。
- 管理 NodeClasses，包括建立、修改和刪除。

排程和資源管理：

- 對 Pod、節點、持久性磁碟區、持久性磁碟區宣告、複寫控制器和命名空間的讀取存取權。
- 存取儲存類別、CSI 節點和磁碟區附件。
- 列出並監看部署、協助程式集、複本集和具狀態集。
- 讀取 Pod 中斷預算。

事件處理：

- 建立、讀取和管理叢集事件。

節點取消佈建和 Pod 移除：

- 更新、修補和刪除節點。
- 視需要建立 Pod 移出和刪除 Pod。

自訂資源定義 (CRD) 管理：

- 建立新的 CRDs。
- 管理與節點管理相關的特定 CRDs(NodeClasses、NodePools、NodeClaims 和 NodeDiagnostics)。

此政策旨在支援在自動模式下執行的 Amazon EKS 叢集內進行全面的運算管理。它結合了各種操作的許可，包括節點佈建、排程、擴展和資源最佳化。

啟用自動模式時，Amazon EKS 會自動為叢集 IAM 角色建立具有此存取政策的存取項目，確保具備必要的許可，讓運算管理功能正常運作。

AmazonEKSLoadBalancingClusterPolicy

ARN – `arn:aws: eks::aws:cluster-access-policy/AmazonEKSLoadBalancingClusterPolicy`

此政策會授予 Amazon EKS Auto Mode 負載平衡功能所需的許可。它可有效管理和組態 Amazon EKS 叢集內的負載平衡資源。此政策包含下列許可：

事件和資源管理：

- 建立和修補事件。
- Pod、節點、端點和命名空間的讀取存取權。
- 更新 Pod 狀態。

服務和輸入管理：

- 完整管理 服務及其狀態。
- 全面控制輸入及其狀態。
- 端點配量和傳入類別的讀取存取權。

目標群組繫結：

- 建立和修改目標群組繫結及其狀態。
- 對輸入類別參數的讀取存取權。

自訂資源定義 (CRD) 管理：

- 建立和讀取所有 CRDs。
- targetgroupbindings.eks.amazonaws.com 和 ingressclassparams.eks.amazonaws.com CRDs的特定管理。

Webhook 組態：

- 建立和讀取變動和驗證 Webhook 組態。
- 管理 eks-load-balancing-webhook 組態。

此政策旨在支援在自動模式下執行的 Amazon EKS 叢集內進行全面的負載平衡管理。它結合了各種操作的許可，包括服務公開、輸入路由，以及與 AWS 負載平衡服務的整合。

啟用自動模式時，Amazon EKS 會自動為叢集 IAM 角色建立具有此存取政策的存取項目，以確保負載平衡功能具有正常運作所需的許可。

AmazonEKSNetworkingClusterPolicy

ARN – `arn:aws: eks::aws:cluster-access-policy/AmazonEKSNetworkingClusterPolicy`

AmazonEKSNetworkingClusterPolicy

此政策會授予 Amazon EKS Auto Mode 聯網功能所需的許可。它可有效管理和組態 Amazon EKS 叢集內的聯網資源。此政策包含下列許可：

節點和 Pod 管理：

- 對 NodeClasses 及其狀態的讀取存取權。
- 對 NodeClaims 及其狀態的讀取存取權。
- Pod 的讀取存取權。

CNI 節點管理：

- CNINodes 及其狀態的許可，包括建立、讀取、更新、刪除和修補。

自訂資源定義 (CRD) 管理：

- 建立和讀取所有 CRDs。
- cninodes.eks.amazonaws.com CRD 的特定管理（更新、修補、刪除）。

事件管理：

- 建立和修補事件。

此政策旨在支援在自動模式下執行的 Amazon EKS 叢集內進行全面的聯網管理。它結合了各種操作的許可，包括節點聯網組態、CNI（容器網路界面）管理，以及相關的自訂資源處理。

此政策允許聯網元件與節點相關資源互動、管理 CNI 特定的節點組態，以及處理叢集中聯網操作至關重要的自訂資源。

啟用自動模式時，Amazon EKS 會自動建立具有此存取政策的存取項目給叢集 IAM 角色，確保具備必要的許可，讓聯網功能正常運作。

AmazonEKSHybridPolicy

> **Note**
>
> 此政策僅針對 AWS 服務連結角色指定，無法與客戶受管角色搭配使用。

此存取政策包含授予 EKS 存取叢集節點的許可。與存取項目相關聯時，其存取範圍通常是叢集，而不是 Kubernetes 命名空間。Amazon EKS 混合節點使用此政策。

ARN – `arn:aws: eks::aws:cluster-access-policy/AmazonEKSHybridPolicy`

Kubernetes API 群組	Kubernetes nonResourceURLs	Kubernetes 資源	Kubernetes 動詞（許可）
*		nodes	list

AmazonEKSClusterInsightsPolicy

> **Note**
>
> 此政策僅針對 AWS 服務連結角色指定，無法與客戶受管角色搭配使用。

ARN – `arn:aws: eks::aws:cluster-access-policy/AmazonEKSClusterInsightsPolicy`

此政策會授予 Amazon EKS Cluster Insights 功能的唯讀許可。此政策包含下列許可：

節點存取： - 列出和檢視叢集節點 - 讀取節點狀態資訊

DaemonSet 存取：- 對 kube-proxy 組態的讀取存取

此政策由 Cluster Insights 的 EKS 服務自動管理。如需詳細資訊，請參閱the section called “叢集洞察”。

存取政策更新

檢視有關存取政策更新 (自引進以來) 的詳細資訊。如需此頁面變更的自動提醒，請訂閱 中的 RSS 摘要文件歷史紀錄。

變更	描述	日期
新增 EKS Cluster Insights 的政策	發佈 `AmazonEKSClusterInsightsPolicy`	2024 年 12 月 2 日
新增 Amazon EKS 混合的政策	發佈 `AmazonEKSHybridPolicy`	2024 年 12 月 2 日
新增 Amazon EKS Auto 模式的政策	這些存取政策提供叢集 IAM 角色和節點 IAM 角色呼叫 Kubernetes APIs許可。 AWS 使用這些許可來自動化儲存、運算和聯網資源的例行任務。	2024 年 12 月 2 日
加 `AmazonEKSAdminViewPolicy`	新增新政策以擴展檢視存取，包括 Secrets 等資源。	2024 年 4 月 23 日

變更	描述	日期
引進存取政策。	Amazon EKS 引進了存取政策。	2023 年 5 月 29 日

變更身分驗證模式以使用存取項目

若要開始使用存取項目，您必須將叢集的身分驗證模式變更為 API_AND_CONFIG_MAP 或 API 模式。這會新增用於存取項目的 API。

AWS 主控台

1. 開啟 Amazon EKS 主控台。
2. 選擇要在其中建立存取項目的叢集名稱。
3. 選擇存取索引標籤。
4. 身分驗證模式會顯示叢集目前的身分驗證模式。如果模式顯示 EKS API，您可以新增存取項目，也可以略過剩餘的步驟。
5. 選擇管理存取。
6. 針對叢集身分驗證模式，選取具有 EKS API 的模式。請注意，您無法將身分驗證模式變更回移除 EKS API 和存取項目的模式。
7. 選擇儲存變更。Amazon EKS 開始更新叢集、叢集的狀態變更為更新，且變更會記錄在更新歷史記錄索引標籤中。
8. 等待叢集的狀態回到作用中。當叢集處於作用中狀態時，您可以遵循 中的步驟the section called “建立存取項目”，為 IAM 主體新增對叢集的存取。

AWS CLI

1. 安裝 AWS CLI，如 AWS 命令列界面使用者指南中的安裝中所述。
2. 執行下列命令。使用您叢集的名稱取代 *my-cluster*。如果您想永久停用 ConfigMap 方法，則請用 API 取代 API_AND_CONFIG_MAP。

   Amazon EKS 開始更新叢集、叢集的狀態變更為 UPDATING，且變更會記錄在 aws eks list-updates 中。

```
aws eks update-cluster-config --name my-cluster --access-config
 authenticationMode=API_AND_CONFIG_MAP
```

3. 等待叢集的狀態回到作用中。當叢集處於作用中狀態時，您可以遵循 中的步驟the section called “建立存取項目”，為 IAM 主體新增對叢集的存取。

所需的平台版本

若要使用存取項目，叢集的平台版本必須與下表列出的版本相同或更新，或 Kubernetes 版本必須高於表格中列出的版本。如果您的 Kubernetes 版本未列出，則所有平台版本都支援存取項目。

Kubernetes 版本	平台版本
1.30	eks.2
1.29	eks.1
1.28	eks.6
1.27	eks.10
1.26	eks.11

如需詳細資訊，請參閱 eks/latest/userguide/platform-versions.html【platform-versions，type="documentation"】。

建立存取項目

在建立存取項目之前，請考慮以下事項：

- 正確設定身分驗證模式。請參閱 the section called “身分驗證方式”。
- 存取項目包含一個 (且僅一個) 現有 IAM 主體的 Amazon Resource Name (ARN)。IAM 主體不能包含在多個存取項目中。對於您指定的 ARN 的其他考量：
  - IAM 最佳實務建議使用具有短期憑證的 IAM 角色存取叢集，而不是使用具有長期憑證的 IAM 使用者進行存取。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 。

- 如果 ARN 用於 IAM 角色，則它可以包含路徑。aws-auth ConfigMap 項目中的 ARNs 不能包含路徑。例如， ARN 可以是 arn:aws: iam::<111122223333>:role/<development/apps/my-role> 或 arn:aws: iam::<111122223333>:role/<my-role>。
- 如果存取項目的類型是 以外的任何項目 STANDARD（請參閱有關 類型的下一個考量），則 ARN 必須位於叢集所在的相同 AWS 帳戶中。如果類型為 STANDARD，則 ARN 可以是與您叢集所在的帳戶相同或不同的 AWS 帳戶。
- 建立存取項目後，您無法變更 IAM 主體。
- 如果您使用此 ARN 刪除 IAM 主體，則不會自動刪除存取項目。建議您刪除帶有您刪除之 IAM 主體的 ARN 的存取項目。如果您未刪除存取項目，而且曾經重新建立 IAM 主體，即使該主體具有相同的 ARN，則存取項目將無法運作。這是因為即使重新建立的 IAM 主體的 ARN 相同，對於重新建立的 IAM 主體， roleID或 userID（您可以使用 aws sts get-caller-identity AWS CLI 命令查看） 與原始 IAM 主體的 ARN 不同。即使您沒有看到 IAM 主體的 roleID或 userID 存取項目，Amazon EKS 仍會將它與存取項目一起存放。

- 每個存取項目都有一個類型。存取項目的類型取決於與其相關聯的資源類型，並且不會定義許可。如果您未指定類型，Amazon EKS 會自動將類型設定為 STANDARD
  - EC2_LINUX - 用於與 Linux 或 Bottlerocket 自我管理節點搭配使用的 IAM 角色
  - EC2_WINDOWS - 用於與 Windows 自我管理節點搭配使用的 IAM 角色
  - FARGATE_LINUX - 用於搭配 AWS Fargate (Fargate) 使用的 IAM 角色
  - HYBRID_LINUX - 用於與混合節點搭配使用的 IAM 角色
  - STANDARD - 若未指定，則為預設類型
  - EC2 - 適用於 EKS Auto Mode 自訂節點類別。如需詳細資訊，請參閱the section called “建立節點類別存取項目”。
  - 您無法在建立存取項目後變更類型。
- 不需要為用於受管節點群組或 Fargate 設定檔的 IAM 角色建立存取項目。EKS 將建立存取項目 （如果啟用），或更新身分驗證組態映射 （如果存取項目不可用）
- 如果某個存取項目的類型為 STANDARD，則可以為其指定一個使用者名稱。如果您未指定使用者名稱的值，Amazon EKS 會根據存取項目的類型，以及您指定的 IAM 主體是 IAM 角色或 IAM 使用者，為您設定下列其中一個值。除非您有指定自己的使用者名稱的特定原因，否則建議您不要指定一個，並讓 Amazon EKS 為您自動產生。如果您自己指定使用者名稱：
  - 它不能以 system:、eks:、amazon:、 aws:或 開頭iam:。
  - 如果使用者名稱適用於 IAM 角色，建議您將 {{SessionName}}或 {{SessionNameRaw}} 新增至使用者名稱的結尾。如果您將 {{SessionName}}或 {{SessionNameRaw}} 新增至使用者名稱，使用者名稱必須在 {{SessionName}} 之前包含冒號。擔任此角色時，擔任角色時指定的

AWS STS 工作階段名稱名稱會自動傳遞至叢集，並會出現在 CloudTrail 日誌中。例如，您無法擁有 的使用者名稱john{{SessionName}}。使用者名稱必須為 :john{{SessionName}} 或 jo:hn{{SessionName}}。冒號只需位於 {{SessionName}} 之前。以下資料表中 Amazon EKS 產生的使用者名稱包含 ARN。由於 ARN 包含冒號，因此它符合此項要求。如果您未將 包含在使用者名稱{{SessionName}}中，則不需要冒號。請注意，在{{SessionName}}特殊字元 "@" 中，工作階段名稱會取代為 "-"。 會將所有特殊字元{{SessionNameRaw}}保留在工作階段名稱中。

IAM 主體類型	Type	Amazon EKS 自動設定的使用者名稱值
使用者	STANDARD	使用者的 ARN。範例：arn:aws: iam::<111122223333>:user/<my-user>
角色	STANDARD	擔任角色時的 STS ARN。Amazon EKS 將 {{SessionName}} 附加到角色。 範例： arn:aws: sts::<111122223333>:assumed-role/<my-role>/{{SessionName}} 如果您指定的角色的 ARN 包含路徑，則 Amazon EKS 會將其從產生的使用者名稱中移除。
角色	EC2_LINUX 或 EC2_Windows	system:node:{{EC2PrivateDNSName}}
角色	FARGATE_LINUX	system:node:{{SessionName}}

IAM 主體類型	Type	Amazon EKS 自動設定的使用者名稱值
角色	HYBRID_LINUX	system:node:{{SessionName}}

建立存取項目後，您可以對使用者名稱進行變更。

- 如果存取項目的類型是 STANDARD，而且您想要使用 Kubernetes RBAC 授權，您可以將一或多個群組名稱新增至存取項目。建立存取項目後，您可以新增和移除群組名稱。若要讓 IAM 主體能夠存取叢集上的 Kubernetes 物件，您必須建立和管理 Kubernetes 角色型授權 (RBAC) 物件。在叢集上建立 Kubernetes RoleBinding或 ClusterRoleBinding 物件，將群組名稱指定為 subject的 kind: Group。Kubernetes 授權 IAM 主體存取您在 Kubernetes 中指定的任何叢集物件，Role或您在繫結的 中也指定的ClusterRole物件roleRef。如果您指定群組名稱，建議您熟悉 Kubernetes 角色型授權 (RBAC) 物件。如需詳細資訊，請參閱 Kubernetes 文件中的使用 RBAC 授權。

  > **Important**
  >
  > Amazon EKS 不會確認叢集上存在的任何 Kubernetes RBAC 物件是否包含您指定的任何群組名稱。例如，如果您為目前不存在的群組建立存取項目，EKS 將建立群組，而不是傳回錯誤。

  除了授權 IAM 主體存取叢集上的 Kubernetes 物件之外，您也可以將 Amazon EKS 存取政策與存取項目建立關聯。Amazon EKS 授權 IAM 主體使用存取政策中的許可來存取叢集上的 Kubernetes 物件。您可以將存取政策的許可範圍限定為您指定的 Kubernetes 命名空間。使用存取政策不需要您管理 Kubernetes RBAC 物件。如需詳細資訊，請參閱the section called “關聯存取政策”。

- 如果您建立類型為 EC2_LINUX 或 EC2_Windows 的存取項目，則建立該存取項目的 IAM 主體必須具有 iam:PassRole 許可。如需詳細資訊，請參閱《IAM 使用者指南》中的授予使用者將角色傳遞至 AWS 服務的許可。
- 與標準 IAM 行為類似，存取項目的建立和更新採用最終一致模式，在初始 API 呼叫成功返回後可能需要幾秒鐘才能生效。您設計的應用程式必須能夠處理這些可能的延遲問題。我們建議您不要在應用程式的關鍵、高可用性程式碼路徑中包含存取項目建立或更新。而應在不常運作的、單獨的初始化或設定常式中進行 更改。另外，在生產工作流程套用這些變更之前，請務必確認變更已傳播完畢。

- 存取項目不支援服務連結角色。您無法建立主體 ARN 為服務連結角色的存取項目。您可以依其 ARN 識別服務連結角色，其格式為 `arn:aws: iam::*:role/aws-service-role/*`。

您可以使用 AWS Management Console 或 CLI AWS 建立存取項目。

AWS Management Console

1. 開啟 Amazon EKS 主控台。
2. 選擇要在其中建立存取項目的叢集名稱。
3. 選擇存取索引標籤。
4. 選擇建立存取項目。
5. 對於 IAM 主體，選取現有 IAM 角色或使用者。IAM 最佳實務建議使用具有短期憑證的 IAM 角色存取叢集，而不是使用具有長期憑證的 IAM 使用者進行存取。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 。
6. 對於類型，如果存取項目針對用於自我管理 Amazon EC2 節點的節點角色，請選取 EC2 Linux 或 EC2 Windows。否則，請接受預設值 (標準)。
7. 如果您選擇的類型是標準並且您想要指定使用者名稱，則請輸入使用者名稱。
8. 如果您選擇的類型是標準類型，而且您想要對 IAM 委託人使用 Kubernetes RBAC 授權，請指定群組的一或多個名稱。如果您不指定任何群組名稱，並想要使用 Amazon EKS 授權，您可以在後續步驟中或在建立存取項目之後建立存取政策的關聯。
9. (選用) 您可以使用標籤為存取項目指派標籤。例如，為了更輕鬆地找到具有相同標籤的所有資源而指定標籤。

10選擇下一步。

11在新增存取政策頁面上，如果您選擇的類型是標準類型，而且您希望 Amazon EKS 授權 IAM 主體擁有叢集上 Kubernetes 物件的許可，請完成以下步驟。否則請選擇 Next (下一步)。/

   a. 對於政策名稱，請選擇存取政策。您無法檢視存取政策的許可，但它們包含與 Kubernetes 使用者面向`ClusterRole`物件類似的許可。如需詳細資訊，請參閱 Kubernetes 文件中的面向使用者的角色。

   b. 請選擇下列其中一個選項：

      - 叢集 – 如果您希望 Amazon EKS 授權 IAM 主體在叢集上所有 Kubernetes 物件的存取政策中擁有許可，請選擇此選項。
      - Kubernetes 命名空間 – 如果您希望 Amazon EKS 授權 IAM 主體在存取政策中擁有叢集上特定 Kubernetes 命名空間中所有 Kubernetes 物件的許可，請選擇此選項。在命名空間中，輸入叢

集上 Kubernetes 命名空間的名稱。如果要新增其他命名空間，則請選擇新增命名空間並輸入命名空間名稱。

c. 如果要新增其他政策，則請選擇新增政策。您可以對每個政策設定不同的範圍，但每個政策只能新增一次。

d. 選擇下一步。

12檢查存取項目的組態。如果有任何內容看起來不正確，請選擇上一步以返回上一步並修正錯誤。如果組態正確，請選擇建立。

AWS CLI

1. 安裝 AWS CLI，如 AWS 命令列界面使用者指南中的安裝中所述。
2. 若要建立存取項目 您可以使用下列任一範例來建立存取項目：

- 為自我管理的 Amazon EC2 Linux 節點群組建立存取項目。將 *my-cluster* 取代為您的叢集名稱、將 *111122223333* 取代為您的 AWS 帳戶 ID，並將 *EKS-my-cluster-self-managed-ng-1* 取代為您的節點 IAM 角色名稱。如果您的節點群組是 Windows 節點群組，請以 取代 *EC2_LINUX*EC2_Windows。

```
aws eks create-access-entry --cluster-name my-cluster --principal-arn arn:aws:
 iam::111122223333:role/EKS-my-cluster-self-managed-ng-1 --type EC2_LINUX
```

當您指定 以外的類型時，無法使用 --kubernetes-groups選項STANDARD。您無法將存取政策與此存取項目建立關聯，因為其類型是 以外的值STANDARD。

- 建立存取項目，允許 IAM 角色不用於您希望 Kubernetes 授權存取叢集的 Amazon EC2 自我管理節點群組。將 *my-cluster* 取代為您的叢集名稱、將 *111122223333* 取代為您的 AWS 帳戶 ID，並將 *my-role* 取代為您的 IAM 角色名稱。以您在叢集上的 Kubernetes RoleBinding或ClusterRoleBinding物件中指定的群組名稱取代*###*。

```
aws eks create-access-entry --cluster-name my-cluster --principal-arn arn:aws:
 iam::111122223333:role/my-role --type STANDARD --user Viewers --kubernetes-groups
 Viewers
```

- 建立一個允許 IAM 使用者向叢集進行身分驗證的存取項目。在此提供此範例只是為了說明這種可能性。IAM 最佳實務建議使用具有短期憑證的 IAM 角色存取叢集，而不是使用具有長期憑證的 IAM 使用者進行存取。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 。

```
aws eks create-access-entry --cluster-name my-cluster --principal-arn arn:aws:
 iam::111122223333:user/my-user --type STANDARD --username my-user
```

如果您希望此使用者比 Kubernetes API 探索角色中的許可擁有更多的叢集存取權，則需要將存取政策與存取項目建立關聯，因為未使用 `--kubernetes-groups`選項。如需詳細資訊，請參閱 Kubernetes 文件中的 the section called “關聯存取政策”和 API 探索角色。

更新存取項目

您可以使用 AWS Management Console 或 CLI AWS 更新存取項目。

AWS Management Console

1. 開啟 Amazon EKS 主控台。
2. 選擇要在其中建立存取項目的叢集名稱。
3. 選擇存取索引標籤。
4. 選擇要更新的存取項目。
5. 選擇編輯。
6. 對於使用者名稱，您可以變更現有值。
7. 對於群組，您可以移除現有群組名稱或新增群組名稱。如果下列群組名稱存在，請勿將其移除：System：nodes 或 system：bootstrappers。移除這些群組可能會導致叢集無法正常運作。如果您未指定任何群組名稱，並想要使用 Amazon EKS 授權，請在後續步驟中建立存取政策的關聯。
8. 您可以使用標籤為存取項目指派標籤。例如，為了更輕鬆地找到具有相同標籤的所有資源而指定標籤。您也可以移除現有的標籤。
9. 選擇 Save changes (儲存變更)。
10. 如果您想要將存取政策關聯到項目，請參閱 the section called “關聯存取政策”。

AWS CLI

1. 安裝 AWS CLI，如《 AWS 命令列界面使用者指南》中的安裝所述。
2. 若要更新存取項目，請將 *my-cluster* 取代為您的叢集名稱、將 *111122223333* 取代為您的 AWS 帳戶 ID，並將 *EKS-my-cluster-my-namespace-Viewers* 取代為 IAM 角色的名稱。

```
aws eks update-access-entry --cluster-name my-cluster --principal-arn arn:aws:
 iam::111122223333:role/EKS-my-cluster-my-namespace-Viewers --kubernetes-groups
 Viewers
```

如果存取項目的類型是 以外的值，則無法使用 --kubernetes-groups選項STANDARD。您也無法將存取政策與 以外的類型建立關聯STANDARD。

刪除存取項目

如果您發現某個存取項目被錯誤刪除，則您可以隨時重新建立。如果您要刪除的存取項目與任何存取政策相關聯，則會自動刪除關聯。刪除存取項目之前，您不需要取消存取政策與存取項目的關聯。

您可以使用 AWS Management Console 或 CLI AWS 刪除存取項目。

AWS Management Console

1. 開啟 Amazon EKS 主控台。
2. 選擇要從中刪除存取項目之叢集的名稱。
3. 選擇存取索引標籤。
4. 在存取項目清單中，選擇要刪除的存取項目。
5. 選擇 刪除 。
6. 在確認對話方塊中，選擇 Delete (刪除)。

AWS CLI

1. 安裝 AWS CLI，如 AWS 命令列界面使用者指南中的安裝中所述。
2. 若要刪除存取項目 將 *my-cluster* 取代為您的叢集名稱、將 *111122223333* 取代為您的 AWS 帳戶 ID，並將 *my-role* 取代為您不再想要存取叢集的 IAM 角色名稱。

```
aws eks delete-access-entry --cluster-name my-cluster --principal-arn arn:aws:
 iam::111122223333:role/my-role
```

使用 ConfigMap 授予 IAM 使用者對 Kubernetes 的存取權

> **Important**
>
> aws-auth ConfigMap 已棄用。如需管理 Kubernetes APIs 存取的建議方法，請參閱 the section called “授予許可”。

IAM AWS Authenticator for Kubernetes 在 Amazon EKS 控制平面上執行，可讓您使用 IAM 主體存取叢集。驗證器會從 aws-auth ConfigMap 獲得其組態資訊。如需所有 aws-auth ConfigMap 設定的詳細資訊，請參閱 GitHub 上的完整組態格式。

將 IAM 主體新增至 Amazon EKS 叢集

當您建立 Amazon EKS 叢集時，會自動在 Amazon EKS 控制平面的叢集角色型存取控制 (RBAC) 組態中授予建立叢集的 IAM 主體system:masters許可。此主體不會出現在任何可見的組態中，因此請務必追蹤最初建立叢集的主體。若要授予其他 IAM 主體與叢集互動的能力，請在 Kubernetes aws-auth ConfigMap中編輯 ，並使用group您在 中指定的 clusterrolebinding名稱建立 Kubernetes rolebinding或 aws-auth ConfigMap。

> **Note**
>
> 如需 Kubernetes 角色型存取控制 (RBAC) 組態的詳細資訊，請參閱 Kubernetes 文件中的使用 RBAC 授權。

1. 判斷 kubectl 正使用哪些憑證來存取叢集。您可以在電腦上透過一下命令查看 kubectl 正在使用的憑證。如果您不使用預設路徑，請將 *~/.kube/config* 取代為 kubeconfig 檔案的路徑。

   ```
   cat ~/.kube/config
   ```

   範例輸出如下。

   ```
   [...]
   contexts:
   - context:
       cluster: my-cluster.region-code.eksctl.io
       user: admin@my-cluster.region-code.eksctl.io
   ```

```
  name: admin@my-cluster.region-code.eksctl.io
current-context: admin@my-cluster.region-code.eksctl.io
[...]
```

在先前的範例輸出中，名為 *admin* 之使用者的登入資料會設定為名為 *my-cluster* 的叢集。若該叢集是由這名使用者建立，則其已擁有您叢集的存取權。如果不是建立叢集的使用者，則需要完成剩餘的步驟，才能為其他 IAM 主體啟用叢集存取。IAM 最佳實務建議您將許可授予角色而非使用者。您可以使用以下命令查看目前哪些其他主體可以存取您的叢集：

```
kubectl describe -n kube-system configmap/aws-auth
```

範例輸出如下。

```
Name:         aws-auth
Namespace:    kube-system
Labels:       <none>
Annotations:  <none>

Data
====
mapRoles:
----
- groups:
  - system:bootstrappers
  - system:nodes
  rolearn: arn:aws: iam::111122223333:role/my-node-role
  username: system:node:{{EC2PrivateDNSName}}


BinaryData
====

Events:  <none>
```

上一個範例是預設的 aws-auth ConfigMap。只有節點執行個體角色有權存取叢集。

2. 請確定您擁有現有的 Kubernetes roles和 rolebindings或 clusterrolebindings，clusterroles而且您可以將 IAM 主體映射至其中。如需有關這些資源的詳細資訊，請參閱 Kubernetes 文件中的使用 RBAC 授權。

   a. 檢視您現有的 Kubernetes roles 或 clusterroles。Roles 的作用域為 namespace，但 clusterroles 的作用域限定為叢集。

```
kubectl get roles -A
```

```
kubectl get clusterroles
```

b. 檢視先前輸出中傳回的任何 role 或 clusterrole 的詳細資訊，並確認其具有您希望 IAM 主體在叢集中擁有的許可 (rules)。

將 *role-name* 取代為上一個命令的輸出中傳回的 role 名稱。將 *kube-system* 取代為 role 的命名空間。

```
kubectl describe role role-name -n kube-system
```

將 *cluster-role-name* 取代為上一個命令的輸出中傳回的 clusterrole 名稱。

```
kubectl describe clusterrole cluster-role-name
```

c. 檢視您現有的 Kubernetes rolebindings 或 clusterrolebindings。Rolebindings 的作用域為 namespace，但 clusterrolebindings 的作用域限定為叢集。

```
kubectl get rolebindings -A
```

```
kubectl get clusterrolebindings
```

d. 檢視任何 rolebinding 或 clusterrolebinding 的詳細資訊，並確認其具有上一步中列為 roleRef 的 role 或 clusterrole，以及為 subjects 列出的群組名稱。

將 *role-binding-name* 取代為上一個命令的輸出中傳回的 rolebinding 名稱。將 *kube-system* 取代為 rolebinding 的 namespace。

```
kubectl describe rolebinding role-binding-name -n kube-system
```

範例輸出如下。

```
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: eks-console-dashboard-restricted-access-role-binding
  namespace: default
```

```
subjects:
- kind: Group
  name: eks-console-dashboard-restricted-access-group
  apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: Role
  name: eks-console-dashboard-restricted-access-role
  apiGroup: rbac.authorization.k8s.io
```

將 *cluster-role-binding-name* 取代為上一個命令的輸出中傳回的 clusterrolebinding 名稱。

```
kubectl describe clusterrolebinding cluster-role-binding-name
```

範例輸出如下。

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: eks-console-dashboard-full-access-binding
subjects:
- kind: Group
  name: eks-console-dashboard-full-access-group
  apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: ClusterRole
  name: eks-console-dashboard-full-access-clusterrole
  apiGroup: rbac.authorization.k8s.io
```

3. 編輯 aws-auth ConfigMap。您可以使用例如 eksctl 的工具來更新 ConfigMap，或者您可以透過編輯來手動更新它。

> **Important**
>
> 我們建議您使用 eksctl 或其他工具來編輯 ConfigMap。如需有關您可以使用的其他工具的資訊，請參閱《Amazon EKS 最佳實務指南》中的使用工具對 aws-authConfigMap 進行變更。格式錯誤的 aws-auth ConfigMap可能會導致您失去叢集存取權。

- 檢視使用 eksctl 編輯組態圖的步驟。

- 檢視手動編輯組態圖的步驟。

使用 Eksctl 編輯 Configmap

1. 您需要在裝置0.210.0或 AWS CloudShell 上安裝版本 或更新版本的eksctl命令列工具。如需有關安裝或更新 eksctl 的指示，請參閱 eksctl 文件中的安裝一節。
2. 在 ConfigMap 檢視目前的映射項目。使用您叢集的名稱取代 *my-cluster*。將 *region-code* 取代為您的叢集所在的 AWS 區域。

```
eksctl get iamidentitymapping --cluster my-cluster --region=region-code
```

範例輸出如下。

```
ARN
                USERNAME                                GROUPS
 ACCOUNT
arn:aws: iam::111122223333:role/eksctl-my-cluster-my-nodegroup-
NodeInstanceRole-1XLS7754U3ZPA    system:node:{{EC2PrivateDNSName}}
 system:bootstrappers,system:nodes
```

3. 為角色新增映射項目。將 *my-group* 取代為您的角色名稱。將 *eks-console-dashboard-full-access-group* 取代為 Kubernetes RoleBinding或 ClusterRoleBinding 物件中指定的群組名稱。使用您的帳戶 ID 取代 *111122223333*。您可以將 *admin* 取代為您選擇的任何名稱。

```
eksctl create iamidentitymapping --cluster my-cluster --region=region-code \
    --arn arn:aws: iam::111122223333:role/my-role --username admin --group eks-
console-dashboard-full-access-group \
    --no-duplicate-arns
```

> **Important**
>
> 角色 ARN 不能包含路徑，例如 role/my-team/developers/my-role。ARN 的格式必須是 arn:aws: iam::*111122223333*:role/*my-role* 。在此範例中，需移除 my-team/developers/。

範例輸出如下。

```
[...]
2022-05-09 14:51:20 [#]  adding identity "{arn-aws}iam::111122223333:role/my-role" to
 auth ConfigMap
```

4. 為使用者新增映射項目。IAM 最佳實務建議您將許可授予角色而非使用者。將 *my-user* 取代為您的使用者名稱。將 *eks-console-dashboard-restricted-access-group* 取代為 Kubernetes RoleBinding或 ClusterRoleBinding 物件中指定的群組名稱。使用您的帳戶 ID 取代 *111122223333*。您可以將 *my-user* 取代為您選擇的任何名稱。

```
eksctl create iamidentitymapping --cluster my-cluster --region=region-code \
    --arn arn:aws: iam::111122223333:user/my-user --username my-user --group eks-
console-dashboard-restricted-access-group \
    --no-duplicate-arns
```

範例輸出如下。

```
[...]
2022-05-09 14:53:48 [#]  adding identity "arn:aws: iam::111122223333:user/my-user" to
 auth ConfigMap
```

5. 再次檢視 ConfigMap 中的映射項目。

```
eksctl get iamidentitymapping --cluster my-cluster --region=region-code
```

範例輸出如下。

```
ARN
           USERNAME                                GROUPS
      ACCOUNT
arn:aws: iam::111122223333:role/eksctl-my-cluster-my-nodegroup-
NodeInstanceRole-1XLS7754U3ZPA    system:node:{{EC2PrivateDNSName}}
 system:bootstrappers,system:nodes
arn:aws: iam::111122223333:role/admin
               my-role                                   eks-console-dashboard-full-
access-group
arn:aws: iam::111122223333:user/my-user
            my-user                                 eks-console-dashboard-restricted-
access-group
```

手動編輯 Configmap

1. 開啟 ConfigMap 進行編輯。

```
kubectl edit -n kube-system configmap/aws-auth
```

> **Note**
>
> 如果您收到錯誤，指出 "Error from server (NotFound): configmaps "aws-auth" not found"，請使用「將 aws-auth ConfigMap 套用至叢集中的程序來套用庫存 ConfigMap。

2. 將您的 IAM 主體新增至 ConfigMap。IAM 群組不是 IAM 主體，因此無法新增至 ConfigMap。
   - 新增 IAM 角色 (例如：為聯合身分使用者)：在 data 下位於 ConfigMap 中的 mapRoles 區段新增角色詳細資訊。若此區段在檔案不存在，則將其新增。每個項目支援以下參數：
     - rolearn：要新增的 IAM 角色之 ARN。此值不能包含路徑。例如，您無法指定 ARN，例如 arn:aws: iam::*111122223333*:role/my-team/developers/*role-name* 。ARN 需要改為 arn:aws: iam::*111122223333*:role/*role-name* 。
     - username：Kubernetes 內映射至 IAM 角色的使用者名稱。
     - 群組：要將角色映射到的 Kubernetes 群組的群組或清單。群組可以是預設群組，也可以是 clusterrolebinding 或 rolebinding 中指定的群組。如需詳細資訊，請參閱 Kubernetes 文件中的預設角色和角色繫結。
   - 若要新增 IAM 使用者：IAM 最佳實務建議您將許可授予角色而非使用者。將使用者詳細資訊新增至 ConfigMap 的 mapUsers 區段 (在 data 下方)。若此區段在檔案不存在，則將其新增。每個項目支援以下參數：
     - userarn：要新增之 IAM 使用者的 ARN。
     - username：Kubernetes 內應設至 IAM 使用者的使用者名稱。
     - 群組：要將使用者映射到的 Kubernetes 群組的群組或清單。群組可以是預設群組，也可以是 clusterrolebinding 或 rolebinding 中指定的群組。如需詳細資訊，請參閱 Kubernetes 文件中的預設角色和角色繫結。
3. 例如，下列的 YAML 區塊包含：
   - mapRoles 區段，將 IAM 節點執行個體映射至 Kubernetes 群組，讓節點可以將自己註冊到叢集，以及 my-console-viewer-role IAM 角色，映射至 Kubernetes 群組，可以檢視所有叢集的所有 Kubernetes 資源。如需 my-console-viewer-role IAM 角色所需的 IAM 和 Kubernetes 群組許可的清單，請參閱 the section called “所需的許可”。

- 將 admin IAM 使用者從預設 AWS 帳戶映射至 system:masters Kubernetes 群組，以及將my-user使用者從映射至可檢視特定命名空間 Kubernetes 資源之 Kubernetes 群組的不同 AWS 帳戶映射的mapUsers區段。如需 my-user IAM 使用者所需的 IAM 和 Kubernetes 群組許可的清單，請參閱 the section called “所需的許可”。

  視需要新增或移除行，並將所有###取代為您自己的值。

```
# Please edit the object below. Lines beginning with a '#' will be ignored,
# and an empty file will abort the edit. If an error occurs while saving this file
 will be
# reopened with the relevant failures.
#
apiVersion: v1
data:
  mapRoles: |
    - groups:
      - system:bootstrappers
      - system:nodes
      rolearn: arn:aws: iam::111122223333:role/my-role
      username: system:node:{{EC2PrivateDNSName}}
    - groups:
      - eks-console-dashboard-full-access-group
      rolearn: arn:aws: iam::111122223333:role/my-console-viewer-role
      username: my-console-viewer-role
  mapUsers: |
    - groups:
      - system:masters
      userarn: arn:aws: iam::111122223333:user/admin
      username: admin
    - groups:
      - eks-console-dashboard-restricted-access-group
      userarn: arn:aws: iam::444455556666:user/my-user
      username: my-user
```

4. 儲存檔案並結束您的文字編輯器。

將 aws-authConfigMap 套用至您的叢集

在建立受管節點群組或以 eksctl 建立節點群組時，便會自動建立並套用 aws-auth ConfigMap 至您的叢集。最初建立的目的是要允許將節點加入您的叢集，但您也可以使用此 ConfigMap 來新增角色

型存取控制 (RBAC) 以存取 IAM 主體。如果您已啟動自我管理節點，且尚未aws-authConfigMap將套用至叢集，您可以使用下列程序執行此操作。

1. 檢查您是否已套用 aws-auth ConfigMap。

```
kubectl describe configmap -n kube-system aws-auth
```

如果您收到錯誤，指出 "Error from server (NotFound): configmaps "aws-auth" not found"，請繼續執行下列步驟以套用庫存 ConfigMap。

2. 下載、編輯和套用 AWS 驗證器組態映射。

a. 下載組態對應。

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/cloudformation/2020-10-29/
aws-auth-cm.yaml
```

b. 在 aws-auth-cm.yaml 檔案中，將 rolearn 設定至與您節點關聯的 IAM 角色之 Amazon Resource Name (ARN)。您可以使用文字編輯器執行此操作，或取代 *my-node-instance-role* 並執行下列命令：

```
sed -i.bak -e 's|<ARN of instance role (not instance profile)>|my-node-instance-
role|' aws-auth-cm.yaml
```

請勿修改此檔案中的任何其他行。

> **Important**
>
> 角色 ARN 不能包含路徑，例如 role/my-team/developers/my-role。ARN 的格式必須是 arn:aws: iam::*111122223333*:role/*my-role* 。在此範例中，需移除 my-team/developers/。

您可以檢查節點群組的 AWS CloudFormation 堆疊輸出，並尋找下列值：

- InstanceRoleARN：適用於以 eksctl 建立的節點群組
- NodeInstanceRole – 適用於使用 中的 Amazon EKS vended AWS CloudFormation 範本建立的節點群組 AWS Management Console

c. 套用組態。此命令可能需要幾分鐘的時間來完成。

```
kubectl apply -f aws-auth-cm.yaml
```

> **Note**
>
> 如果您收到任何授權或資源類型錯誤，請參閱故障診斷主題中的the section called “未經授權或存取遭拒 (kubectl)”。

3. 查看節點的狀態，並等待他們到達 Ready 狀態。

```
kubectl get nodes --watch
```

輸入 Ctrl+C 傳回 Shell 提示。

使用外部 OIDC 供應商授予使用者對 Kubernetes 的存取權

Amazon EKS 支援使用 OpenID Connect (OIDC) 身分提供者作為向您的叢集對使用者進行身分驗證的方法。OIDC 身分提供者可以與 AWS Identity and Access Management (IAM) 搭配使用，或作為其替代方案。如需使用 IAM 的詳細資訊，請參閱 the section called “Kubernetes API 存取”。設定叢集身分驗證後，您可以建立 Kubernetes roles 和 clusterroles 指派許可給角色，然後使用 Kubernetes rolebindings 和 clusterrolebindings 將角色繫結至身分。如需詳細資訊，請參閱 Kubernetes 文件中的使用 RBAC 授權。

- 您可以將一個 OIDC 身分提供者與叢集建立關聯。
- Kubernetes 不提供 OIDC 身分提供者。您可以使用現有的公有 OIDC 身分提供者，也可以執行您自己的身分提供者。如需認證提供商的清單，請參閱 OpenID 網站上的 OpenID 認證。
- OIDC 身分提供者的發行者 URL 必須可公開存取，以便 Amazon EKS 可以探索簽署金鑰。Amazon EKS 不支援具有自我簽署憑證的 OIDC 身分提供者。
- 您無法停用叢集的 IAM 身分驗證，因為將節點加入叢集仍需要。
- Amazon EKS 叢集仍然必須由 AWS IAM 主體建立，而非 OIDC 身分提供者使用者。這是因為叢集建立者會與 Amazon EKS API 進行互動，而不是與 Kubernetes API 進行互動。
- 如果為控制平面開啟 CloudWatch 日誌，OIDC 身分提供者驗證的使用者會列在叢集的稽核日誌中。如需詳細資訊，請參閱the section called “啟用或停用控制平面日誌”。
- 您無法 AWS Management Console 透過 OIDC 供應商的帳戶登入 。您只能the section called “存取叢集資源”使用 AWS Identity and Access Management AWS Management Console 帳戶登入 。

關聯 OIDC 身分提供者

在可以將 OIDC 身分提供者與叢集建立關聯之前，您需要提供商的下列資訊：

發行者 URL

OIDC 身分提供者的 URL，可讓 API 伺服器探索用於驗證權杖的公有簽署金鑰。URL 必須以 開頭，`https://`且應該對應至提供者的 OIDC ID 字符中的`iss`宣告。根據 OIDC 標準，允許路徑元件，但不允許查詢參數。通常，URL 只包含一個主機名稱，如 https://server.example.org 或 https://example.com 。此 URL 應指向 `.well-known/openid-configuration` 以下的層級，必須可透過網際網路公開存取。

用戶端 ID（也稱為對象)

向 OIDC 身分提供者提出驗證請求的用戶端應用程式 ID。

您可以使用 `eksctl` 或 AWS Management Console關聯身分提供者。

使用 eksctl 關聯身分提供者

1. 使用下列內容建立名為 *`associate-identity-provider.yaml`* 的檔案。使用自己的取代*`###`*。`identityProviders` 區段中的值是從您的 OIDC 身分提供者取得的。值僅對 `identityProviders` 下的 `name`、`type`、`issuerUrl` 和 `clientId` 設定為必要項目。

```
---
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: my-cluster
  region: your-region-code

identityProviders:
  - name: my-provider
    type: oidc
    issuerUrl: https://example.com
    clientId: kubernetes
    usernameClaim: email
    usernamePrefix: my-username-prefix
    groupsClaim: my-claim
    groupsPrefix: my-groups-prefix
    requiredClaims:
```

```
    string: string
  tags:
    env: dev
```

> **Important**
>
> 請勿為 `system:` 或 指定 或該字串的任何部分`groupsPrefix``usernamePrefix`。

2. 建立供應商。

```
eksctl associate identityprovider -f associate-identity-provider.yaml
```

3. 若要使用 `kubectl` 搭配您的叢集和 OIDC 身分提供者，請參閱 Kubernetes 文件中的使用 kubectl。

使用 AWS 主控台關聯身分提供者

1. 開啟 Amazon EKS 主控台。
2. 選取您的叢集，然後選取存取索引標籤。
3. 在 OIDC 身分提供者區段中，選取* 關聯身分提供者*。
4. 在 Associate OIDC Identity Provider (關聯 OIDC 身分提供者) 頁面上，輸入或選取下列選項，然後選取 Associate (關聯)。
   - 對於 Name (名稱)，輸入提供商的唯一名稱。
   - 對於 Issuer URL (發行者 URL)，輸入您的提供商 URL。此 URL 必須可透過網際網路存取。
   - 針對用戶端 ID，輸入 OIDC 身分提供者的用戶端 ID （ 也稱為對象)。
   - 對於 Username claim (使用者名稱宣告)，輸入要用作使用者名稱的宣告。
   - 針對群組宣告，輸入宣告以用作使用者的群組。
   - (選用) 選取 Advanced options (進階選項)，輸入或選取下列資訊。
     - Username prefix (使用者名稱字首)：輸入要在使用者名稱宣告前面加上的字首。字首會加入使用者名稱宣告，以防止與現有名稱發生衝突。如果未提供值，且使用者名稱是非 `email` 的值，字首預設為 Issuer URL (發行者 URL) 的值。您可以使用值 `-` 停用所有字首。請勿指定 `system:` 或該字串的任何部分。
     - Groups prefix (群組字首)：輸入要在群組宣告前面加上的字首。字首會加入群組宣告，以防止與現有名稱發生衝突 (例如 `system: groups`)。例如，值 `oidc:` 會建立群組名稱，例如 `oidc:engineering` 和 `oidc:infra`。請勿指定 `system:` 或該字串的任何部分。

- Required claims (必要宣告)：選取 Add claim (新增宣告)，然後在用戶端 ID 字符中輸入一或多個描述必要宣告的鍵值對。配對描述 ID 字符中所需的宣告。如果設定，則會驗證每個宣告是否存在於具有相符值的 ID 字符中。
  a. 若要使用 kubectl 搭配您的叢集和 OIDC 身分提供者，請參閱 Kubernetes 文件中的使用 kubectl。

範例 IAM 政策

如果您想要防止 OIDC 身分提供者與叢集產生關聯，請建立下列 IAM 政策並將其與 Amazon EKS 管理員的 IAM 帳戶關聯。如需詳細資訊，請參閱《IAM 使用者指南》中的建立 IAM 政策和新增 IAM 身分許可和服務授權參考中的動作。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "denyOIDC",
            "Effect": "Deny",
            "Action": [
                "eks:AssociateIdentityProviderConfig"
            ],
            "Resource": "arn:aws: eks:us-west-2.amazonaws.com:111122223333:cluster/*"

        },
        {
            "Sid": "eksAdmin",
            "Effect": "Allow",
            "Action": [
                "eks:*"
            ],
            "Resource": "*"
        }
    ]
}
```

下列範例政策允許 OIDC 身分提供者關聯，如果 clientID 為 kubernetes 以及 issuerUrl 為 https://cognito-idp.us-west-2amazonaws.com/* 。

```
{
    "Version": "2012-10-17",
```

```
    "Statement": [
        {
            "Sid": "AllowCognitoOnly",
            "Effect": "Deny",
            "Action": "eks:AssociateIdentityProviderConfig",
            "Resource": "arn:aws: eks:us-west-2:111122223333:cluster/my-instance",
            "Condition": {
                "StringNotLikeIfExists": {
                    "eks:issuerUrl": "https://cognito-idp.us-west-2.amazonaws.com/*"
                }
            }
        },
        {
            "Sid": "DenyOtherClients",
            "Effect": "Deny",
            "Action": "eks:AssociateIdentityProviderConfig",
            "Resource": "arn:aws: eks:us-west-2:111122223333:cluster/my-instance",
            "Condition": {
                "StringNotEquals": {
                    "eks:clientId": "kubernetes"
                }
            }
        },
        {
            "Sid": "AllowOthers",
            "Effect": "Allow",
            "Action": "eks:*",
            "Resource": "*"
        }
    ]
}
```

取消 OIDC 身分提供者與叢集的關聯

如果您取消 OIDC 身分提供者與叢集的關聯，則提供商中包含的使用者將無法再存取叢集。不過，您仍然可以透過 IAM 主體存取叢集。

1. 開啟 Amazon EKS 主控台。
2. 在 OIDC 身分提供者區段中，選取 Disassociate (解除關聯)，輸入身分提供者名稱，然後選取 Disassociate。

在 中檢視 Kubernetes 資源 AWS Management Console

您可以檢視部署至使用 AWS Management Console的叢集的 Kubernetes 資源。您無法使用 CLI AWS 或 eksctl 檢視 Kubernetes 資源。若要使用命令列工具檢視 Kubernetes 資源，請使用 kubectl。

> **Note**
>
> 若要在 的運算索引標籤上檢視資源索引標籤和節點區段 AWS Management Console，您使用的 IAM 主體必須具有特定的 IAM 和 Kubernetes 許可。如需詳細資訊，請參閱the section called “所需的許可”。

1. 開啟 Amazon EKS 主控台。
2. 在 Clusters (叢集) 清單中，選取包含您要檢視的 Kubernetes 資源的叢集。
3. 選取 Resources (資源) 標籤。
4. 選取您要檢視資源的 Resource Type (資源類型) 群組，例如 Workloads (工作負載)。您可以看到該群組中的資源類型清單。
5. 選取資源類型，例如 Workloads (工作負載) 群組中的 Deployments (部署)。您可以看到資源類型的說明、一個 Kubernetes 文件連結以獲取有關資源類型的詳細資訊，以及部署在叢集上的該類型的資源列表。如果清單是空的，則不會將該類型的資源部署至您的叢集。
6. 請選取資源以檢視其詳細資訊。請試試看下列範例：
   - 選取 Workloads (工作負載) 群組，選取 Deployments (部署) 資源類型，然後選取 coredns 資源。當您選取資源時，依預設，您位於 Structured view (結構式檢視)。針對某些資源類型，您會在 Structured view (結構式檢視) 看到 Pods 區段。本節列出工作負載管理的 Pod。您可以選取列出的任何 Pod，以檢視 Pod 的相關資訊。並非所有資源類型都顯示於 Structured View (結構式檢視)。如果選擇資源頁面右上角的 Raw view (Raw 檢視)，您可以看到來自資源 Kubernetes API 的完整 JSON 回應。
   - 選取 Cluster (叢集) 群組，然後選取 Nodes (節點) 資源類型。您將看到叢集中所有節點的清單。節點可以是任何 Amazon EKS 節點類型。這個列表與您選取叢集的 Compute (運算) 標籤時，在 Nodes (節點)看到的列表為同一列表。從清單中選取節點資源。在 Structured view (結構式檢視) 中，您還會看到 Pods 區段。本節顯示節點上執行的所有 Pod。

所需的許可

若要檢視 中運算索引標籤上的資源索引標籤和節點區段 AWS Management Console，您使用的 IAM 主體必須具有特定的最低 IAM 和 Kubernetes 許可。完成以下步驟，將所需的許可指派給您的 IAM 主體。

1. 請確定 `eks:AccessKubernetesApi`和其他檢視 Kubernetes 資源的必要 IAM 許可已指派給您正在使用的 IAM 主體。如需如何編輯 IAM 主體許可的詳細資訊，請參閱《IAM 使用者指南》中的控制對主體的存取。如需如何編輯角色許可的詳細資訊，請參閱《IAM 使用者指南》中的變更角色許可 (主控台)。

   下列範例政策包含委託人檢視您帳戶中所有叢集 Kubernetes 資源的必要許可。將 *111122223333* 取代為 AWS 您的帳戶 ID。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "eks:ListFargateProfiles",
                "eks:DescribeNodegroup",
                "eks:ListNodegroups",
                "eks:ListUpdates",
                "eks:AccessKubernetesApi",
                "eks:ListAddons",
                "eks:DescribeCluster",
                "eks:DescribeAddonVersions",
                "eks:ListClusters",
                "eks:ListIdentityProviderConfigs",
                "iam:ListRoles"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "ssm:GetParameter",
            "Resource": "arn:aws: ssm:*:111122223333:parameter/*"
        }
    ]
}
```

若要檢視已連線叢集中的節點，Amazon EKS connector IAM 角色應能模擬叢集中的主體。這可讓 Amazon EKS 連接器將主體映射至 Kubernetes 使用者。

2. 建立繫結至 Kubernetes `role` 或 `clusterrole` 的 `rolebinding` 或`clusterrolebinding`，且該種 Kubernetes 角色具有檢視 Kubernetes 資源的必要許可。若要進一步了解 Kubernetes 角色和角色連結，請參閱 Kubernetes 文件中的 Using RBAC Authorization (使用 RBAC 授權)。
您可以將以下清單檔案之一套用至建立具備 Kubernetes 所需許可的 `role` 和 `rolebinding` 或 `clusterrole` 和 `clusterrolebinding` 的叢集：

   檢視所有命名空間中的 Kubernetes 資源

   - 檔案中的群組名稱為 `eks-console-dashboard-full-access-group`。使用下列命令套用清單檔案至您的叢集：

     ```
     kubectl apply -f https://s3.us-west-2.amazonaws.com/amazon-eks/docs/eks-console-full-access.yaml
     ```

   檢視特定命名空間中的 Kubernetes 資源

   - 此檔案中的命名空間為 `default`。檔案中的群組名稱為 `eks-console-dashboard-restricted-access-group`。使用下列命令套用清單檔案至您的叢集：

     ```
     kubectl apply -f https://s3.us-west-2.amazonaws.com/amazon-eks/docs/eks-console-restricted-access.yaml
     ```

     如果您需要變更 Kubernetes 群組名稱、命名空間、許可或檔案中的任何其他設定，請先下載該檔案並進行編輯，然後再將其套用到叢集：

     a. 執行下列其中一個命令，下載檔案：

     ```
     curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/docs/eks-console-full-access.yaml
     ```

     ```
     curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/docs/eks-console-restricted-access.yaml
     ```

     b. 視必要編輯檔案。

     c. 使用下列其中一個命令套用清單檔案至您的叢集：

     ```
     kubectl apply -f eks-console-full-access.yaml
     ```

```
kubectl apply -f eks-console-restricted-access.yaml
```

3. 將 IAM 主體映射至 中的 Kubernetes aws-auth 使用者或群組ConfigMap。您可以使用例如 eksctl 的工具來更新 ConfigMap，或者您可以透過編輯來手動更新它。

> **Important**
>
> 我們建議您使用 eksctl 或其他工具來編輯 ConfigMap。如需有關您可以使用的其他工具的資訊，請參閱《Amazon EKS 最佳實務指南》中的使用工具對 aws-authConfigMap 進行變更。格式錯誤的 aws-auth ConfigMap可能會導致您失去叢集存取權。

使用 eksctl 編輯

1. 您需要在裝置0.210.0或 AWS CloudShell 上安裝版本 或更新版本的eksctl命令列工具。如需有關安裝或更新 eksctl 的指示，請參閱 eksctl 文件中的安裝一節。
2. 在 ConfigMap 檢視目前的映射項目。使用您叢集的名稱取代 *my-cluster*。將 *region-code* 取代為您的叢集所在的 AWS 區域。

```
eksctl get iamidentitymapping --cluster my-cluster --region=region-code
```

範例輸出如下。

```
ARN
            USERNAME                                      GROUPS
 ACCOUNT
arn:aws: iam::111122223333:role/eksctl-my-cluster-my-nodegroup-
NodeInstanceRole-1XLS7754U3ZPA    system:node:{{EC2PrivateDNSName}}
 system:bootstrappers,system:nodes
```

3. 為角色新增映射項目。此範例假設您在第一個步驟中將 IAM 許可連接至名為 *my-console-viewer-role* 的角色。使用您的帳戶 ID 取代 *111122223333*。

```
eksctl create iamidentitymapping \
    --cluster my-cluster \
    --region=region-code \
    --arn arn:aws: iam::111122223333:role/my-console-viewer-role \
    --group eks-console-dashboard-full-access-group \
```

```
    --no-duplicate-arns
```

> **Important**
>
> 角色 ARN 不能包含路徑，例如 `role/my-team/developers/my-role`。ARN 的格式必須是 `arn:aws: iam::111122223333:role/my-role` 。在此範例中，需移除 `my-team/developers/`。

範例輸出如下。

```
[...]
2022-05-09 14:51:20 [#]  adding identity "arn:aws: iam::111122223333:role/my-console-viewer-role" to auth ConfigMap
```

4. 為使用者新增映射項目。IAM 最佳實務建議您將許可授予角色而非使用者。此範例假設您在第一個步驟中將 IAM 許可連接至名為 *my-user* 的使用者。使用您的帳戶 ID 取代 *111122223333*。

```
eksctl create iamidentitymapping \
    --cluster my-cluster \
    --region=region-code \
    --arn arn:aws: iam::111122223333:user/my-user \
    --group eks-console-dashboard-restricted-access-group \
    --no-duplicate-arns
```

範例輸出如下。

```
[...]
2022-05-09 14:53:48 [#]  adding identity "arn:aws: iam::111122223333:user/my-user" to auth ConfigMap
```

5. 再次檢視 ConfigMap 中的映射項目。

```
eksctl get iamidentitymapping --cluster my-cluster --region=region-code
```

範例輸出如下。

```
ARN
            USERNAME                                      GROUPS
       ACCOUNT
arn:aws: iam::111122223333:role/eksctl-my-cluster-my-nodegroup-
NodeInstanceRole-1XLS7754U3ZPA    system:node:{{EC2PrivateDNSName}}
 system:bootstrappers,system:nodes
arn:aws: iam::111122223333:role/my-console-viewer-role
                                                          eks-console-dashboard-full-
access-group
arn:aws: iam::111122223333:user/my-user
                                                       eks-console-dashboard-restricted-
access-group
```

手動編輯 ConfigMap

如需新增使用者或角色到 aws-auth ConfigMap 的詳細資訊，請參閱 the section called “將 IAM 主體新增至 Amazon EKS 叢集”。

1. 開啟 aws-auth ConfigMap 進行編輯。

```
kubectl edit -n kube-system configmap/aws-auth
```

2. 將映射新增至 aws-auth ConfigMap，但不要取代任何現有的映射。下列範例會新增 IAM 主體之間的映射，並在第一個步驟中新增許可，以及在上一個步驟中建立的 Kubernetes 群組：
   - *my-console-viewer-role* 角色和 eks-console-dashboard-full-access-group。
   - *my-user* 使用者和 eks-console-dashboard-restricted-access-group。

   這些範例假設您在第一個步驟中將 IAM 許可連接至名為 *my-console-viewer-role* 的角色，以及連接至名為 *my-user* 的使用者。將 *111122223333* 取代為 AWS 您的帳戶 ID。

```
apiVersion: v1
data:
mapRoles: |
  - groups:
    - eks-console-dashboard-full-access-group
    rolearn: arn:aws: iam::111122223333:role/my-console-viewer-role
    username: my-console-viewer-role
mapUsers: |
  - groups:
```

```
    - eks-console-dashboard-restricted-access-group
    userarn: arn:aws: iam::111122223333:user/my-user
    username: my-user
```

> **Important**
>
> 角色 ARN 不能包含路徑，例如 `role/my-team/developers/my-console-viewer-role`。ARN 的格式必須是 `arn:aws: iam::111122223333:role/my-console-viewer-role` 。在此範例中，需移除 `my-team/developers/`。

3. 儲存檔案並結束您的文字編輯器。

建立 kubeconfig 檔案，將 kubectl 連接至 EKS 叢集

在本主題中，您將為您的叢集建立 `kubeconfig` 檔案 (或更新現有的檔案)。

`kubectl` 命令列工具會使用 `kubeconfig` 檔案中的組態資訊，以與叢集的 API 伺服器進行通訊。如需詳細資訊，請參閱 Kubernetes 文件中的使用 kubeconfig 檔案組織叢集存取。

Amazon EKS 使用 `aws eks get-token` 命令搭配 `kubectl` 進行叢集身分驗證。根據預設，CLI AWS 會使用與下列命令傳回的相同登入資料：

```
aws sts get-caller-identity
```

- 現有 Amazon EKS 叢集。若要部署叢集，請參閱 開始使用。
- `kubectl` 命令列工具安裝在您的裝置或 AWS CloudShell 上。該版本可與叢集的 Kubernetes 版本相同，或最多可以比您叢集的 Kubernetes 版本早或晚一個次要版本。例如，如果您的叢集版本為 `1.29`，則可以搭配使用 `kubectl` `1.28`、`1.29` 或 `1.30` 版。若要安裝或升級 `kubectl`，請參閱 the section called “設定 kubectl 和 eksctl”。
- 在您的裝置或 AWS CloudShell 上安裝和設定的 AWS 命令列界面 (AWS CLI) 版本 `1.27.160` `2.12.3`或更新版本。若要檢查您目前的版本，請使用 `aws --version | cut -d / -f2 | cut -d ' ' -f1`。適用於 macOS 的 `yum`、 `apt-get`或 Homebrew 等套件管理員通常是最新版本 CLI AWS 後面的幾個版本。若要安裝最新版本，請參閱《 AWS 命令列界面使用者指南》中的使用 aws 設定安裝 和 Quick configuration。 https://docs.aws.amazon.com/cli/latest/userguide/cli-configure-quickstart.html#cli-configure-quickstart-config安裝在 AWS CloudShell 中的 AWS CLI 版本也可能是最新版本後面的幾個版本。若要更新它，請參閱 CloudShell AWS 使用者指南中的將 CLI 安裝到您的主目錄。 AWS CloudShell

- 具有對您指定的叢集使用 `eks:DescribeCluster` API 動作許多的 IAM 使用者或角色。如需詳細資訊，請參閱the section called “身分型政策”。如果您使用自己的 OpenID Connect 提供者身分來存取叢集，請參閱 Kubernetes 文件中的使用 kubectl 來建立或更新您的`kube config`檔案。

自動建立 kubeconfig 檔案

- 在您的裝置或 AWS CloudShell 上安裝和設定的 AWS 命令列界面 (AWS CLI) 版本 `1.27.160` `2.12.3`或更新版本。若要檢查您目前的版本，請使用 `aws --version | cut -d / -f2 | cut -d ' ' -f1`。適用於 macOS 的 `yum`、 `apt-get`或 Homebrew 等套件管理員通常是最新版本 CLI AWS 後面的幾個版本。若要安裝最新版本，請參閱《 AWS 命令列界面使用者指南》中的使用 aws 設定安裝 和 Quick configuration。 https://docs.aws.amazon.com/cli/latest/userguide/cli-configure-quickstart.html#cli-configure-quickstart-config安裝在 AWS CloudShell 中的 AWS CLI 版本也可能是最新版本後面的幾個版本。若要更新它，請參閱 CloudShell AWS 使用者指南中的將 CLI 安裝到您的主目錄。 AWS CloudShell
- 對您指定的叢集使用 `eks:DescribeCluster` API 動作的許可。如需詳細資訊，請參閱the section called “身分型政策”。
  1. 為您的叢集建立或更新 `kubeconfig` 檔案。將 *`region-code`* 取代為您的叢集所在的 AWS 區域，並將 *`my-cluster`* 取代為您的叢集名稱。

     ```
     aws eks update-kubeconfig --region region-code --name my-cluster
     ```

     根據預設，產生的組態檔案會在主目錄預設的 `kubeconfig` 路徑下 (`.kube`)，或與位在該處的現有 `config` 檔案合併。您可以使用 `--kubeconfig` 選項指定其他路徑。

     您可以使用 `--role-arn` 選項指定 IAM 角色 ARN，用於在您發出 `kubectl` 命令時進行身分驗證。否則，會使用預設 CLI 或 SDK 登入資料鏈中的 IAM 主體。 AWS 您可以執行 AWS `aws sts get-caller-identity`命令來檢視預設的 CLI 或 SDK 身分。

     如需所有可用的選項，請執行 `aws eks update-kubeconfig help`命令，或參閱 AWS CLI 命令參考中的 update-kubeconfig。
  2. 測試組態。

     ```
     kubectl get svc
     ```

     範例輸出如下。

```
NAME              TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
svc/kubernetes    ClusterIP   10.100.0.1   <none>        443/TCP   1m
```

如果您收到任何授權或資源類型錯誤，請參閱故障診斷主題中的the section called “未經授權或存取遭拒 (kubectl)”。

AWS 使用 Kubernetes 服務帳戶授予 Kubernetes 工作負載存取權

A Kubernetes service account provides an identity for processes that run in a Pod. For more information see 管理服務帳戶 in the Kubernetes documentation. If your Pod needs access to AWS services, you can map the service account to an AWS Identity and Access Management identity to grant that access. For more information, see the section called “具有 IRSA 的登入資料” or the section called “Pod 身分”.

服務帳戶字符

BoundServiceAccountTokenVolume 功能預設為在 Kubernetes 版本中啟用。此功能允許在 Kubernetes 上執行的工作負載請求對象、時間和金鑰綁定的 JSON Web 字符，從而提升服務帳戶字符的安全性。服務帳戶字符的過期時間為一小時。在舊版 Kubernetes 中，字符沒有過期。這表示仰賴這些字符的客戶端必須在一小時內重新整理字符。下列 Kubernetes 用戶端 SDKs 會在所需的時間範圍內自動重新整理權杖：

- Go 版本 `0.15.7` 和更新版本
- Python 版本 `12.0.0` 和更新版本
- Java 版本 `9.0.0` 和更新版本
- JavaScript 版本 `0.10.3` 和更新版本
- Ruby `master` 分支
- Haskell 版本 `0.3.0.0`
- C# 版本 `7.0.5`和更新版本

如果工作負載使用較早的客戶端版本，則必須進行更新。為了讓用戶端順利遷移至較新的限時服務帳戶字符，Kubernetes 會在預設的 1 小時內將延長的到期期間新增至服務帳戶字符。針對 Amazon EKS 叢集，延長的到期期間為 90 天。Amazon EKS 叢集的 Kubernetes API 伺服器會拒絕具有超過 90 天權杖的請求。我們建議您檢查應用程式及其相依性，以確保 Kubernetes 用戶端開發套件與先前列出的版本相同或為更新版本。

當 API 服務器收到使用超過一小時的字符的請求時，它會使用 `annotations.authentication.k8s.io/stale-token` 註釋 API 稽核日誌事件。註釋的值如下範例所示：

```
subject: system:serviceaccount:common:fluent-bit, seconds after warning threshold:
 4185802.
```

如果您的叢集啟用了控制平面記錄，則註釋位於稽核日誌中。您可以使用下列 CloudWatch Logs Insights 查詢來識別 Amazon EKS 叢集中使用過時字符的所有 Pod：

```
fields @timestamp
|filter @logStream like /kube-apiserver-audit/
|filter @message like /seconds after warning threshold/
|parse @message "subject: *, seconds after warning threshold:*\"" as subject,
 elapsedtime
```

`subject` 是指 Pod 使用的服務帳戶。`elapsedtime` 表示讀取最新字符後的經過時間（以秒為單位）。當 `elapsedtime` 超過 90 天 (7,776,000 秒)，對 API 伺服器的請求將遭到拒絕。您應該主動更新應用程式的 Kubernetes 用戶端 SDK，以使用先前列出的其中一個會自動重新整理字符的版本。如果使用的服務帳戶字符接近 90 天，而且您在字符過期之前沒有足夠的時間更新用戶端 SDK 版本，則可以終止現有的 Pod 並建立新的 Pod。這會導致重新擷取服務帳戶字符，進而給您額外的 90 天來更新的客戶端版本 SDK。

如果 Pod 是部署的一部分，建議終止 Pod 同時保持高可用性的方式是使用下列命令執行推展。使用您的部署名稱替換 *my-deployment* (我的部署)。

```
kubectl rollout restart deployment/my-deployment
```

叢集附加元件

下列叢集附加元件已更新為使用會自動重新擷取服務帳戶字符的 Kubernetes 用戶端 SDKs。建議您確保已在叢集上安裝列出的版本或更新版本。

- 適用於 Kubernetes 的 Amazon VPC CNI 外掛程式和指標協助程式外掛程式版本 `1.8.0` 和更新版本。若要檢查目前版本或更新版本，請參閱 the section called “Amazon VPC CNI” 和 cni-metrics-helper。
- CoreDNS 版本 `1.8.4`和更新版本。若要檢查目前版本或更新版本，請參閱 the section called “CoreDNS”。

- AWS Load Balancer控制器版本 2.0.0和更新版本。若要檢查目前版本或更新版本，請參閱 the section called “ AWS Load Balancer控制器”。
- 目前的 kube-proxy 版本。若要檢查目前版本或更新版本，請參閱 the section called “kube-proxy”。
- AWS 適用於 Fluent Bit 版本 2.25.0 或更新版本。若要更新您的目前版本，請參閱 GitHub 上的 Releases (版本)。
- Fluentd 映像 1.14.6-1.2 版或更新版本，以及適用於 Kubernetes 中繼資料的 Fluentd 篩選器外掛程式 2.11.1 版或更新版本。

將 AWS Identity and Access Management 許可授予 Amazon Elastic Kubernetes Service 叢集上的工作負載

Amazon EKS 提供兩種方式，將 AWS Identity and Access Management 許可授予在 Amazon EKS 叢集中執行的工作負載：服務帳戶的 IAM 角色，以及 EKS Pod 身分。

服務帳戶的 IAM 角色

服務帳戶的 IAM 角色 (IRSA) 會設定在 上執行的 Kubernetes 應用程式 AWS ，具有精細的 IAM 許可，以存取各種其他 AWS 資源，例如 Amazon S3 儲存貯體、Amazon DynamoDB 資料表等。您可以在同一個 Amazon EKS 叢集中同時執行多個應用程式，並確保每個應用程式只有其所需的最低許可集。IRSA 的建置旨在支援 支援的各種 Kubernetes 部署選項， AWS 例如 Amazon EKS、Amazon EKS Anywhere、Red Hat OpenShift Service on AWS，以及 Amazon EC2 執行個體上的自我管理 Kubernetes 叢集。因此，IRSA 是使用 IAM 等基礎 AWS 服務建置的，並且不直接依賴 Amazon EKS 服務和 EKS API。如需詳細資訊，請參閱the section called “具有 IRSA 的登入資料”。

EKS Pod 身分

EKS Pod Identity 為叢集管理員提供簡化的工作流程，以驗證應用程式存取各種其他 AWS 資源，例如 Amazon S3 儲存貯體、Amazon DynamoDB 資料表等。EKS Pod 身分識別僅適用於 EKS，因此可簡化叢集管理員設定 Kubernetes 應用程式以取得 IAM 許可的方式。您現在可以直接透過 AWS Management Console EKS API 和 AWS CLI 以較少的步驟輕鬆設定這些許可，而且在任何 Kubernetes 物件中，叢集內不需要採取任何動作。叢集管理員不需要在 EKS 和 IAM 服務之間切換，或使用特權 IAM 操作來設定應用程式所需的許可。IAM 角色現在可跨多個叢集使用，無需在建立新叢集時更新角色信任政策。EKS Pod 身分識別提供的 IAM 憑證包含角色工作階段標籤，並具有叢集名稱、命名空間、服務帳戶名稱等屬性。角色工作階段標籤可讓管理員根據相符的標籤，

允許存取 AWS 資源，以撰寫可在服務帳戶間運作的單一角色。如需詳細資訊，請參閱the section called “Pod 身分”。

比較 EKS Pod 身分識別和 IRSA

整體上來說，EKS Pod 身分識別和 IRSA 都可讓您將 IAM 許可授予在 Kubernetes 叢集上執行的應用程式。但是，它們在設定方式、支援的限制和啟用的功能完全不同。我們在下方比較兩個解決方案的一些關鍵面向。

> **Note**
>
> AWS 建議盡可能使用 EKS Pod 身分來授予 Pod AWS 資源的存取權。如需詳細資訊，請參閱the section called “Pod 身分”。

屬性	EKS Pod 身分識別	IRSA
角色擴充性	您必須設定每個角色一次，才能與新推出的 Amazon EKS 服務主體 `pods.eks.amazonaws.com` 建立信任。在此一次性步驟之後，您不需要在每次在新叢集中使用角色的信任政策時更新該角色的信任政策。	每次您想要在新叢集中使用角色時，都必須使用新的 EKS 叢集 OIDC 提供者端點更新 IAM 角色的信任政策。
叢集可擴展性	EKS Pod Identity 不需要使用者設定 IAM OIDC 供應商，因此不適用此限制。	每個 EKS 叢集都有與其相關聯的 OpenID Connect (OIDC) 發行者 URL。若要使用 IRSA，需要為 IAM 中的每個 EKS 叢集建立唯一的 OpenID Connect 提供者。IAM 每個 AWS 帳戶的預設全域限制為 100 個 OIDC 供應商。如果您計劃為每個具有 IRSA AWS 的帳戶擁有超過 100 個 EKS 叢

屬性	EKS Pod 身分識別	IRSA
		集，則您將達到 IAM OIDC 提供者限制。
角色可擴展性	EKS Pod Identity 不需要使用者在信任政策中定義 IAM 角色和服務帳戶之間的信任關係，因此不適用此限制。	在 IRSA 中，您可以在角色的信任政策中定義 IAM 角色和服務帳戶之間的信任關係。根據預設，信任政策大小的長度為 2048。這表示您通常可以在單一信任政策中定義 4 個信任關係。雖然您可以提高信任政策長度限制，但在單一信任政策中，您通常只能使用最多 8 個信任關係。
STS API Quota 用量	EKS Pod Identity 可簡化將 AWS 登入資料交付至 Pod 的程序，而且不需要您的程式碼直接使用 AWS Security Token Service (STS) 進行呼叫。EKS 服務會處理角色假設，並將登入資料交付至使用 Pod 中 AWS 開發套件撰寫的應用程式，而不需要您的 Pod 與 AWS STS 通訊或使用 STS API Quota。	在 IRSA 中，使用 Pod 中的 AWS SDK 撰寫的應用程式會使用權杖來呼叫 AWS Security Token Service (STS) 上的 `AssumeRoleWithWebIdentity` API。視軟體 AWS 開發套件上程式碼的邏輯而定，您的程式碼可能會對 AWS STS 進行不謹慎的呼叫，並接收限流錯誤。

屬性	EKS Pod 身分識別	IRSA
角色可重複使用性	AWS EKS Pod Identity 提供的 STS 臨時登入資料包含角色工作階段標籤，例如叢集名稱、命名空間、服務帳戶名稱。角色工作階段標籤可讓管理員撰寫單一 IAM 角色，這些角色可以與多個服務帳戶搭配使用，具有不同的有效許可，方法是允許根據附加到他們的標籤存取 AWS 資源。這也稱為屬性型存取控制 (ABAC)。如需詳細資訊，請參閱the section called “授予 Pods 存取權”。	AWS 不支援 STS 工作階段標籤。您可以在叢集之間重複使用角色，但是每個 Pod 都會收到該角色的所有許可。
支援的環境	EKS Pod 身分識別僅適用於 Amazon EKS。	IRSA 可用於 Amazon EKS、Amazon EKS Anywhere、Red Hat OpenShift Service on AWS，以及 Amazon EC2 執行個體上的自我管理 Kubernetes 叢集。
支援 EKS 版本	所有支援的 EKS 叢集版本。如需特定平台版本，請參閱 the section called “EKS Pod 身分識別叢集版本”。	所有支援的 EKS 叢集版本。

服務帳戶的 IAM 角色

Pod 容器中的應用程式可以使用 AWS SDK 或 AWS CLI，使用 AWS Identity and Access Management (IAM) 許可向 AWS 服務提出 API 請求。應用程式必須使用 AWS 登入資料簽署其 AWS API 請求。服務帳戶的 IAM 角色 (IRSA) 可讓您管理應用程式的登入資料，類似於 Amazon EC2 執行個體描述檔向 Amazon EC2 執行個體提供登入資料的方式。您可以將 IAM 角色與 Kubernetes 服務帳戶建立關聯，並將您的 Pod 設定為使用服務帳戶，而不是建立 AWS 登入資料並將其分發至容器或使

用 Amazon EC2 執行個體的 角色。您無法將 IAM 角色用於具有 Amazon EKS on AWS Outposts 本機叢集的服務帳戶。

服務帳戶的 IAM 角色提供下列優點：

- 最低權限 – 您可以將 IAM 許可範圍限定為服務帳戶，而且只有使用該服務帳戶的 Pod 才能存取這些許可。有此功能也就不需要第三方解決方案，例如 `kiam` 或 `kube2iam`。
- 登入資料隔離 – 當對 Amazon EC2 執行個體中繼資料服務 (IMDS) 的存取受到限制時，Pod 的容器只能擷取與容器使用之服務帳戶相關聯的 IAM 角色登入資料。容器永遠無法存取其他 Pod 中其他容器使用的登入資料。如果 IMDS 不受限制，Pod 的容器也可以存取 Amazon EKS 節點 IAM 角色，而且容器可能可以存取相同節點上其他 Pod 的 IAM 角色憑證。如需詳細資訊，請參閱限制存取指派給工作節點的執行個體設定檔。

> **Note**
>
> 使用 設定的 Pod `hostNetwork: true` 一律具有 IMDS 存取權，但啟用時， AWS SDKs和 CLI 將使用 IRSA 登入資料。

- 可稽核性 – 可透過 AWS CloudTrail 存取和事件記錄，以協助確保回溯性稽核。

> **Important**
>
> 容器不是安全界限，而且服務帳戶使用 IAM 角色並不會變更此限制。指派給相同節點的 Pod 將共用核心和可能的其他資源，具體取決於您的 Pod 組態。雖然在個別節點上執行的 Pod 會在運算層隔離，但有些節點應用程式在 Kubernetes API 中具有超出個別執行個體範圍的額外許可。一些範例包括 `kubelet`、`kube-proxy`、CSI 儲存驅動程式或您自己的 Kubernetes 應用程式。

完成下列程序，為服務帳戶啟用 IAM 角色：

1. 為您的叢集建立 IAM OIDC 提供者 – 您只為每個叢集完成此程序一次。

> **Note**
>
> 如果您啟用 EKS VPC 端點，則無法從該 VPC 內部存取 EKS OIDC 服務端點。因此，您的作業 (例如在 VPC 利用 `eksctl` 建立 OIDC 提供者) 將無法運作，並且當嘗試請求 `https://oidc.eks .region.amazonaws.com` 時，會導致逾時。以下是範例錯誤訊息：
>
> ```
> server cant find oidc.eks.region.amazonaws.com: NXDOMAIN
> ```
>
> 若要完成此步驟，您可以在 VPC 外部執行 命令，例如 in AWS CloudShell 或連接至網際網路的電腦。或者，您可以在 VPC 中建立分割期限條件式解析程式，例如 Route 53 Resolver，以針對 OIDC 發行者 URL 使用不同的解析程式，而不為其使用 VPC DNS。如需 CoreDNS 中條件式轉送的範例，請參閱 GitHub 上的 Amazon EKS 功能請求。

2. 將 IAM 角色指派給 Kubernetes 服務帳戶 – 為您希望應用程式擁有的每個唯一許可集完成此程序。
3. 將 Pod 設定為使用 Kubernetes 服務帳戶 – 為每個需要存取 AWS 服務的 Pod 完成此程序。
4. 搭配 AWS SDK 使用 IRSA – 確認工作負載使用支援版本的 AWS SDK，以及工作負載使用預設登入資料鏈。

IAM、Kubernetes 和 OpenID Connect (OIDC) 背景資訊

在 2014 年， AWS Identity and Access Management 新增了對使用 OpenID Connect (OIDC) 的聯合身分的支援。此功能可讓您向支援的身分提供者驗證 AWS API 呼叫，並接收有效的 OIDC JSON Web 字符 (JWT)。您可以將此字符傳遞至 AWS STS `AssumeRoleWithWebIdentity` API 操作，並接收 IAM 臨時角色登入資料。您可以使用這些登入資料與任何 AWS 服務互動，包括 Amazon S3 和 DynamoDB。

每個 JWT 權杖均由一個簽署金鑰對簽署。這些金鑰是由 Amazon EKS 管理的 OIDC 供應商提供服務，且私有金鑰每 7 天就會輪換一次。Amazon EKS 會保留公有金鑰，直到它們過期為止。如果您連接外部 OIDC 用戶端，請注意，您需要在公有金鑰過期之前重新整理簽署金鑰。了解如何擷取簽署金鑰來驗證 OIDC 權杖。

Kubernetes 長期以來使用服務帳戶作為自己的內部身分系統。Pod 可以使用只有 Kubernetes API 伺服器可以驗證的自動掛載字符 (非 OIDC JWT)，向 Kubernetes API 伺服器進行驗證。這些舊版服務帳戶字符不會過期，輪換簽署金鑰是一個困難的程序。在 Kubernetes 版本 中`1.12`，已新增新`ProjectedServiceAccountToken`功能的支援。此功能是 OIDC JSON Web 權杖，也包含服務帳戶身分，並支援可設定的對象。

Amazon EKS 會為每個叢集託管公有 OIDC 探索端點，其中包含 `ProjectedServiceAccountToken` JSON Web 字符的簽署金鑰，因此 IAM 等外部系統可以驗證並接受 Kubernetes 發行的 OIDC 字符。

為您的叢集建立 IAM OIDC 身分提供者

您的叢集具有與其相關聯的 OpenID Connect (OIDC) 發行者 URL。若要針對服務帳戶使用 AWS Identity and Access Management (IAM) 角色，叢集的 OIDC 發行者 URL 必須存在 IAM OIDC 供應商。

- 現有 Amazon EKS 叢集。若要部署叢集，請參閱 開始使用。
- 在您的裝置或 AWS CloudShell 上安裝和設定的 AWS 命令列界面 (AWS CLI) 版本 `1.27.160` `2.12.3`或更新版本。若要檢查您目前的版本，請使用 `aws --version | cut -d / -f2 | cut -d ' ' -f1`。適用於 macOS 的 `yum`、 `apt-get`或 Homebrew 等套件管理員通常是最新版本 CLI AWS 後面的幾個版本。若要安裝最新版本，請參閱《 AWS 命令列界面使用者指南》中的使用 aws 設定安裝 和快速組態。 https://docs.aws.amazon.com/cli/latest/userguide/cli-configure-quickstart.html#cli-configure-quickstart-config安裝在 AWS CloudShell 中的 AWS CLI 版本也可能是最新版本後面的幾個版本。若要更新它，請參閱《CloudShell AWS 使用者指南》中的將 CLI 安裝到您的主目錄。 AWS CloudShell
- `kubectl` 命令列工具安裝在您的裝置或 AWS CloudShell 上。版本可以與叢集的 Kubernetes 版本相同，也可以比叢集的 Kubernetes 版本更早或更晚一個次要版本。例如，如果您的叢集版本為 `1.29`，則可以搭配使用 `kubectl` `1.28`、`1.29` 或 `1.30` 版。若要安裝或升級 `kubectl`，請參閱 the section called “設定 kubectl 和 eksctl”。
- 包含叢集組態的現有 `kubectl config` 檔案。若要建立 `kubectl config` 檔案，請參閱the section called “使用 kubectl 存取叢集”。

您可以使用 `eksctl`或 為您的叢集建立 IAM OIDC 提供者 AWS Management Console。

建立 OIDC 提供者 (eksctl)

1. 裝置或 AWS CloudShell 上安裝的`eksctl`命令列工具版本 `0.210.0`或更新版本。如需有關安裝或更新 `eksctl` 的指示，請參閱 `eksctl` 文件中的安裝一節。
2. 判斷叢集的 OIDC 發行者 ID。

   擷取叢集的 OIDC 發行者 ID，並將其存放在變數中。使用您自己的值取代 `<my-cluster>`。

   ```
   cluster_name=<my-cluster>
   ```

```
oidc_id=$(aws eks describe-cluster --name $cluster_name --query
 "cluster.identity.oidc.issuer" --output text | cut -d '/' -f 5)
echo $oidc_id
```

3. 判斷具有叢集發行者 ID 的 IAM OIDC 供應商是否已在您的帳戶中。

```
aws iam list-open-id-connect-providers | grep $oidc_id | cut -d "/" -f4
```

如果傳回輸出，表示您已有叢集的 IAM OIDC 提供者，您可以略過下一個步驟。如果未傳回任何輸出，則您必須為叢集建立 IAM OIDC 提供商。

4. 使用下列命令為您的叢集建立 IAM OIDC 身分提供者。

```
eksctl utils associate-iam-oidc-provider --cluster $cluster_name --approve
```

> **Note**
>
> 如果您啟用 EKS VPC 端點，則無法從該 VPC 內部存取 EKS OIDC 服務端點。因此，在 VPC `eksctl`中使用 建立 OIDC 提供者等操作將無法運作，且會導致逾時。以下是範例錯誤訊息：

```
** server cant find oidc.eks.<region-code>.amazonaws.com: NXDOMAIN
```

若要完成此步驟，您可以在 VPC 外部執行 命令，例如 in AWS CloudShell 或連接至網際網路的電腦。或者，您可以在 VPC 中建立分割期限條件式解析程式，例如 Route 53 Resolver，以針對 OIDC 發行者 URL 使用不同的解析程式，而不為其使用 VPC DNS。如需 CoreDNS 中條件式轉送的範例，請參閱 GitHub 上的 Amazon EKS 功能請求。

建立 OIDC 供應商AWS （主控台）

1. 開啟 Amazon EKS 主控台。
2. 在左側窗格中，選取 Clusters (叢集)，然後在 Clusters (叢集) 頁面上選取您的叢集名稱。
3. 在 Overview (概觀) 標籤的 Details (詳細資訊) 區段中，記下 OpenID Connect provider URL (OpenID Connect 供應商 URL) 的值。
4. 前往 https://console.aws.amazon.com/iam/ 開啟 IAM 主控台。

5. 在左側導覽窗格中，選擇 Access management (存取管理) 下的 Identity Providers (身分提供者)。如果列出的提供商與您叢集的 URL 相符，則表示您已經擁有叢集提供商。如果未列出符合您叢集 URL 的提供者，則您必須建立一個。
6. 若要建立供應商，請選擇 Add provider (新增供應商)。
7. 針對提供者類型，選取 OpenID Connect。
8. 在提供者 URL 中，輸入叢集的 OIDC 提供者 URL。
9. 針對對象，輸入 `sts.amazonaws.com`。
10. (選用) 新增任何標籤，例如標籤，以識別適用於此提供者的叢集。
11選擇 Add provider (新增提供者)。

後續步驟：the section called “指派 IAM 角色”

將 IAM 角色指派給 Kubernetes 服務帳戶

本主題說明如何設定 Kubernetes 服務帳戶以擔任 AWS Identity and Access Management (IAM) 角色。然後，任何設定為使用服務帳戶的 Pod 都可以存取該角色有權存取的任何 AWS 服務。

先決條件

- 現有的叢集。如果您沒有，您可以遵循 中的其中一個指南來建立一個開始使用。
- 叢集的現有 IAM OpenID Connect (OIDC) 提供商。若要了解是否已經擁有，或是了解如何建立，請參閱the section called “IAM OIDC 供應商”。
- 裝置或 AWS CloudShell 上安裝和設定的 AWS 命令列界面 (AWS CLI) 版本 `1.27.160` `2.12.3`或更新版本。若要檢查您目前的版本，請使用 `aws --version | cut -d / -f2 | cut -d ' ' -f1`。適用於 macOS 的 `yum`、 `apt-get`或 Homebrew 等套件管理員通常是最新版本 CLI AWS 後面的幾個版本。若要安裝最新版本，請參閱《 AWS 命令列界面使用者指南》中的使用 aws 設定安裝 和快速組態。 https://docs.aws.amazon.com/cli/latest/userguide/cli-configure-quickstart.html#cli-configure-quickstart-config在 AWS CloudShell 中安裝的 AWS CLI 版本也可能是最新版本後面的幾個版本。若要更新它，請參閱《CloudShell AWS 使用者指南》中的將 CLI 安裝到您的主目錄。 AWS CloudShell
- `kubectl` 命令列工具安裝在您的裝置或 AWS CloudShell 上。版本可以與叢集的 Kubernetes 版本相同，也可以比叢集的 Kubernetes 版本更早或更晚一個次要版本。例如，如果您的叢集版本為 `1.29`，則可以搭配使用 `kubectl` `1.28`、`1.29` 或 `1.30` 版。若要安裝或升級 `kubectl`，請參閱 the section called “設定 kubectl 和 eksctl”。
- 包含叢集組態的現有 `kubectl config` 檔案。若要建立 `kubectl config` 檔案，請參閱the section called “使用 kubectl 存取叢集”。

步驟 1：建立 IAM 政策

如果您要將現有 IAM 政策與 IAM 角色建立關聯，請跳到下一步驟。

1. 建立 IAM 政策。您可以建立自己的政策，或複製已授予部分所需許可的 AWS 受管政策，並根據您的特定需求進行自訂。如需詳細資訊，請參閱「IAM 使用者指南」中的建立 IAM 政策。
2. 建立檔案，其中包含您希望 Pod 存取 AWS 之服務的許可。如需所有 AWS 服務的所有動作清單，請參閱服務授權參考。

   您可以執行以下命令來建立允許唯讀存取 Amazon S3 儲存貯體的範例政策檔案。您可以選擇性地在此儲存貯體中存放組態資訊或引導指令碼，而 Pod 中的容器可以從儲存貯體讀取檔案，並將其載入您的應用程式。如果您要建立此範例政策，請將以下內容複製到您的裝置。將 *my-pod-secrets-bucket* 取代為您的儲存貯體名稱並執行命令。

   ```
   cat >my-policy.json <<EOF
   {
       "Version": "2012-10-17",
       "Statement": [
           {
               "Effect": "Allow",
               "Action": "s3:GetObject",
               "Resource": "arn:aws: s3:::my-pod-secrets-bucket"
           }
       ]
   }
   EOF
   ```

3. 建立 IAM 政策。

   ```
   aws iam create-policy --policy-name my-policy --policy-document file://my-policy.json
   ```

步驟 2：建立和關聯 IAM 角色

建立 IAM 角色並將其與 Kubernetes 服務帳戶建立關聯。您可以使用 `eksctl`或 AWS CLI。

建立和關聯角色 (eksctl)

此`eksctl`命令會在指定的命名空間中建立 Kubernetes 服務帳戶、建立具有指定名稱的 IAM 角色（如果不存在）、將現有的 IAM 政策 ARN 連接到角色，並使用 IAM 角色 ARN 註釋服務帳戶。請務必將此

命令中的範例預留位置值取代為您的特定值。如需有關安裝或更新 eksctl 的指示，請參閱 eksctl 文件中的安裝一節。

```
eksctl create iamserviceaccount --name my-service-account --namespace default --cluster
 my-cluster --role-name my-role \
    --attach-policy-arn arn:aws: iam::111122223333:policy/my-policy --approve
```

> **Important**
>
> 如果角色或服務帳戶已經存在，上一個命令可能會失敗。您可以為 eksctl 提供在這些情況下的不同選項。如需詳細資訊，請執行 `eksctl create iamserviceaccount --help`。

建立角色並建立關聯 (AWS CLI)

如果您有想要擔任 IAM 角色的現有 Kubernetes 服務帳戶，則可以略過此步驟。

1. 建立 Kubernetes 服務帳戶。將以下內容複製到您的裝置。視需要將 *my-service-account* 取代為您要使用的名稱，將 *default* 取代為其他命名空間。如果您變更 *default*，命名空間必須已經存在。

   ```
   cat >my-service-account.yaml <<EOF
   apiVersion: v1
   kind: ServiceAccount
   metadata:
     name: my-service-account
     namespace: default
   EOF
   kubectl apply -f my-service-account.yaml
   ```

2. 使用下列命令將 AWS 您的帳戶 ID 設定為 環境變數。

   ```
   account_id=$(aws sts get-caller-identity --query "Account" --output text)
   ```

3. 使用下列命令，將叢集的 OIDC 身分提供者設定為 環境變數。使用您叢集的名稱取代 *my-cluster*。

   ```
   oidc_provider=$(aws eks describe-cluster --name my-cluster --region $AWS_REGION --
   query "cluster.identity.oidc.issuer" --output text | sed -e "s/^https:\/\///")
   ```

4. 設定命名空間和服務帳戶名稱的變數。將 *my-service-account* 取代為您想要擔任該角色的 Kubernetes 服務帳戶。將 *default* 取代為服務帳戶的命名空間。

```
export namespace=default
export service_account=my-service-account
```

5. 執行下列命令以建立 IAM 角色的信任政策檔案。如果您想要允許命名空間中的所有服務帳戶使用該角色，請將以下內容複製到您的裝置。將 *StringEquals* 取代為 StringLike，並將 *$service_account* 取代為 *。您可以在 StringEquals 和 StringLike 條件中新增多個項目，以允許多個服務帳戶或命名空間擔任角色。若要允許來自與叢集所在 AWS 帳戶不同的 帳戶的角色擔任該角色，請參閱 the section called “跨帳戶 IAM” 以取得詳細資訊。

```
cat >trust-relationship.json <<EOF
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Federated": "arn:aws: iam::$account_id:oidc-provider/$oidc_provider"
      },
      "Action": "sts:AssumeRoleWithWebIdentity",
      "Condition": {
        "StringEquals": {
          "$oidc_provider:aud": "sts.amazonaws.com",
          "$oidc_provider:sub": "system:serviceaccount:$namespace:$service_account"
        }
      }
    }
  ]
}
EOF
```

6. 建立角色。將 *my-role* 取代為您的 IAM 角色名稱，並將 *my-role-description* 取代為您的角色描述。

```
aws iam create-role --role-name my-role --assume-role-policy-document file://trust-
relationship.json --description "my-role-description"
```

7. 將 IAM 政策連接至您的角色。將 *my-role* 取代為您的 IAM 角色名稱，並將 *my-policy* 取代為您建立的現有政策名稱。

```
aws iam attach-role-policy --role-name my-role --policy-arn=arn:aws: iam::
$account_id:policy/my-policy
```

8. 使用您希望服務帳戶擔任之 IAM 角色的 Amazon Resource Name (ARN) 標註您的服務帳戶。使用現有 IAM 角色的名稱取代 *my-role*。假設您允許來自與叢集所在 AWS 帳戶不同的 帳戶的角色，以在上一個步驟中擔任該角色。然後，請務必從另一個 AWS 帳戶指定帳戶和角色。如需詳細資訊，請參閱the section called “跨帳戶 IAM”。

```
kubectl annotate serviceaccount -n $namespace $service_account eks.amazonaws.com/
role-arn=arn:aws: iam::$account_id:role/my-role
```

9. （選用）設定服務帳戶 AWS 的安全字符服務端點。 AWS 建議使用區域 AWS STS 端點而非全域端點。這樣可以減少延遲、提供內建備援，並增加工作階段字符的有效性。

步驟 3：確認組態

1. 確認 IAM 角色的信任政策已正確設定。

```
aws iam get-role --role-name my-role --query Role.AssumeRolePolicyDocument
```

範例輸出如下。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Federated": "arn:aws: iam::111122223333:oidc-provider/
oidc.eks.region-code.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE"
            },
            "Action": "sts:AssumeRoleWithWebIdentity",
            "Condition": {
                "StringEquals": {
                    "oidc.eks.region-code.amazonaws.com/id/
EXAMPLED539D4633E53DE1B71EXAMPLE:sub": "system:serviceaccount:default:my-service-
account",
                    "oidc.eks.region-code.amazonaws.com/id/
EXAMPLED539D4633E53DE1B71EXAMPLE:aud": "sts.amazonaws.com"
                }
```

```
            }
        }
    ]
}
```

2. 確認您在上一步連接至角色的政策已連接至該角色。

```
aws iam list-attached-role-policies --role-name my-role --query
 "AttachedPolicies[].PolicyArn" --output text
```

範例輸出如下。

```
 arn:aws: iam::111122223333:policy/my-policy
```

3. 設定變數以存放您要使用之政策的 Amazon Resource Name (ARN)。將 *my-policy* 取代為您要確認許可的政策名稱。

```
export policy_arn=arn:aws: iam::111122223333:policy/my-policy
```

4. 檢視預設政策版本。

```
aws iam get-policy --policy-arn $policy_arn
```

範例輸出如下。

```
{
    "Policy": {
        "PolicyName": "my-policy",
        "PolicyId": "EXAMPLEBIOWGLDEXAMPLE",
        "Arn": "arn:aws: iam::111122223333:policy/my-policy",
        "Path": "/",
        "DefaultVersionId": "v1",
        [...]
    }
}
```

5. 檢視政策內容，以確保政策包含 Pod 所需的所有許可。如有必要，請將下列命令中的 *1* 取代為先前輸出中傳回的版本。

```
aws iam get-policy-version --policy-arn $policy_arn --version-id v1
```

範例輸出如下。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "s3:GetObject",
            "Resource": "arn:aws: s3:::my-pod-secrets-bucket"
        }
    ]
}
```

如果您在上一步建立了範例政策，則輸出結果相同。如果您建立了不同的政策，則 *example* 內容有所不同。

6. 確認 Kubernetes 服務帳戶已標註 角色。

```
kubectl describe serviceaccount my-service-account -n default
```

範例輸出如下。

```
Name:                my-service-account
Namespace:           default
Annotations:         eks.amazonaws.com/role-arn: arn:aws: iam::111122223333:role/my-
role
Image pull secrets:  <none>
Mountable secrets:   my-service-account-token-qqjfl
Tokens:              my-service-account-token-qqjfl
[...]
```

後續步驟

- the section called “指派給 Pod”

設定 Pod 以使用 Kubernetes 服務帳戶

如果 Pod 需要存取 AWS 服務，則必須將其設定為使用 Kubernetes 服務帳戶。服務帳戶必須與具有存取 AWS 服務許可的 AWS Identity and Access Management (IAM) 角色相關聯。

- 現有的叢集。如果您沒有，您可以使用 中的其中一個指南建立一個開始使用。
- 叢集的現有 IAM OpenID Connect (OIDC) 提供商。若要了解是否已經擁有，或是了解如何建立，請參閱the section called “IAM OIDC 供應商”。
- 與 IAM 角色相關聯的現有 Kubernetes 服務帳戶。服務帳戶必須標註 IAM 角色的 Amazon Resource Name (ARN)。角色必須具有關聯的 IAM 政策，其中包含您希望 Pod 必須使用 AWS 服務的許可。如需有關服務帳戶和角色之建立和設定方式的詳細資訊，請參閱the section called “指派 IAM 角色”。
- 在您的裝置或 AWS CloudShell 上安裝和設定的 AWS 命令列界面 (AWS CLI) 版本 `1.27.160` `2.12.3`或更新版本。若要檢查您目前的版本，請使用 `aws --version | cut -d / -f2 | cut -d ' ' -f1`。適用於 macOS 的 `yum`、 `apt-get`或 Homebrew 等套件管理員通常是最新版本 CLI AWS 後面的數個版本。若要安裝最新版本，請參閱《 AWS 命令列界面使用者指南》中的使用 aws 設定安裝 和快速組態。 https://docs.aws.amazon.com/cli/latest/userguide/cli-configure-quickstart.html#cli-configure-quickstart-config在 AWS CloudShell 中安裝的 AWS CLI 版本也可能是最新版本後面的數個版本。若要更新它，請參閱《CloudShell AWS 使用者指南》中的將 CLI 安裝到您的主目錄。 AWS CloudShell
- `kubectl` 命令列工具安裝在您的裝置或 AWS CloudShell 上。該版本可以與叢集的 Kubernetes 版本相同，也可以比叢集的 Kubernetes 版本更早或更晚，最多一個次要版本。例如，如果您的叢集版本為 `1.29`，則可以搭配使用 `kubectl` `1.28`、`1.29` 或 `1.30` 版。若要安裝或升級 `kubectl`，請參閱 the section called “設定 kubectl 和 eksctl”。
- 包含叢集組態的現有 `kubectl config` 檔案。若要建立 `kubectl config` 檔案，請參閱the section called “使用 kubectl 存取叢集”。

  1. 使用下列命令來建立部署資訊清單，您可以部署 Pod 來確認組態。以您自己的值取代*###*。

```
cat >my-deployment.yaml <<EOF
apiVersion: apps/v1
kind: Deployment
metadata:
  name: my-app
spec:
  selector:
    matchLabels:
      app: my-app
  template:
    metadata:
      labels:
        app: my-app
    spec:
      serviceAccountName: my-service-account
      containers:
```

```
      - name: my-app
        image: public.ecr.aws/nginx/nginx:X.XX
EOF
```

2. 將清單檔案部署到叢集。

```
kubectl apply -f my-deployment.yaml
```

3. 確認 Pod 存在所需的環境變數。

   a. 檢視在上一個步驟中與部署一起部署的 Pod。

   ```
   kubectl get pods | grep my-app
   ```

   範例輸出如下。

   ```
   my-app-6f4dfff6cb-76cv9   1/1     Running   0          3m28s
   ```

   b. 檢視 Pod 使用的 IAM 角色 ARN。

   ```
   kubectl describe pod my-app-6f4dfff6cb-76cv9 | grep AWS_ROLE_ARN:
   ```

   範例輸出如下。

   ```
   AWS_ROLE_ARN: arn:aws: iam::111122223333:role/my-role
   ```

   角色 ARN 必須與您用來標註現有服務帳戶的角色 ARN 相符。如需有關標註服務帳戶的詳細資訊，請參閱the section called “指派 IAM 角色”。

   c. 確認 Pod 具有 Web 身分字符檔案掛載。

   ```
   kubectl describe pod my-app-6f4dfff6cb-76cv9 | grep AWS_WEB_IDENTITY_TOKEN_FILE:
   ```

   範例輸出如下。

   ```
   AWS_WEB_IDENTITY_TOKEN_FILE:  /var/run/secrets/eks.amazonaws.com/serviceaccount/
   token
   ```

代表 Pod kubelet請求和存放字符。根據預設，如果字符超過總存留時間的 80% 或 24 小時，kubelet 會重新整理字符。您可以使用 Pod 規格中的設定，修改預設服務帳戶以外的任何帳戶的過期期間。如需詳細資訊，請參閱 Kubernetes 文件中的服務帳戶字符磁碟區投影。

叢集上的 Amazon EKS Pod Identity Webhook 會監看使用具有下列註釋之服務帳戶的 Pod：

```
eks.amazonaws.com/role-arn: arn:aws: iam::111122223333:role/my-role
```

Webhook 會將先前的環境變數套用至這些 Pod。您的叢集不需要使用 Webhook 來設定環境變數和權杖檔案掛載。您可以手動將 Pod 設定為具有這些環境變數。支援的 AWS SDK 版本會先在登入資料鏈提供者中尋找這些環境變數。角色登入資料用於符合此條件的 Pod。

4. 確認您的 Pod 可以使用您在連接至角色的 IAM 政策中指派的許可來與 AWS 服務互動。

> **Note**
>
> 當 Pod 使用與服務帳戶相關聯的 IAM 角色 AWS 登入資料時，該 Pod 容器中的 AWS CLI 或其他 SDKs 會使用該角色提供的登入資料。如果您不限制存取提供給 Amazon EKS 節點 IAM 角色的登入資料，Pod 仍然可以存取這些登入資料。如需詳細資訊，請參閱限制存取指派給工作節點的執行個體設定檔。

如果您的 Pod 無法如預期與 服務互動，請完成下列步驟，以確認一切設定正確。

a. 確認您的 Pod 使用支援透過 OpenID Connect Web 身分字符檔案擔任 IAM 角色的 AWS SDK 版本。如需詳細資訊，請參閱the section called “支援的 SDK”。

b. 確認部署使用服務帳戶。

```
kubectl describe deployment my-app | grep "Service Account"
```

範例輸出如下。

```
Service Account:  my-service-account
```

c. 如果您的 Pod 仍然無法存取 服務，請檢閱將 IAM 角色指派給 Kubernetes 服務帳戶中所述的步驟，以確認您的角色和服務帳戶已正確設定。

設定服務帳戶 AWS 的安全字符服務端點

如果您使用具有服務帳戶 IAM 角色的 Kubernetes 服務帳戶，則如果您的叢集和平台版本與下表中列出的版本相同或更新版本，則可以設定服務帳戶所使用的 AWS Security Token Service 端點類型。如果您的 Kubernetes 或平台版本早於資料表中列出的版本，則您的服務帳戶只能使用全域端點。

Kubernetes 版本	平台版本	預設端點類型
1.31	eks.4	區域性
1.30	eks.2	區域性
1.29	eks.1	區域性
1.28	eks.1	區域性
1.27	eks.1	區域性
1.26	eks.1	區域性

AWS 建議使用區域 AWS STS 端點，而不是全域端點。這樣可以減少延遲、提供內建備援，並增加工作階段字符的有效性。 AWS Security Token Service 必須在執行 Pod 的區域中處於作用中 AWS 狀態。此外，如果 AWS 區域中的服務發生故障，您的應用程式必須具有不同 AWS 區域的內建備援。如需詳細資訊，請參閱《IAM 使用者指南AWS 》中的在 區域中管理 AWS STS。

- 現有的叢集。如果您沒有，您可以使用 中的其中一個指南建立一個開始使用。
- 叢集的現有 IAM OIDC 提供商。如需詳細資訊，請參閱the section called “IAM OIDC 供應商”。
- 現有的 Kubernetes 服務帳戶設定為與服務帳戶的 Amazon EKS IAM 功能搭配使用。

下列範例全部使用 Amazon VPC CNI 外掛程式所使用的 aws-node Kubernetes 服務帳戶。您可以將*###*取代為您自己的服務帳戶、Pod、命名空間和其他資源。

1. 選取使用您要變更端點之服務帳戶的 Pod。決定 Pod AWS 執行的區域。將 *aws-node-6mfgv* 取代為您的 Pod 名稱，並將 *kube-system* 取代為您的 Pod 命名空間。

   ```
   kubectl describe pod aws-node-6mfgv -n kube-system |grep Node:
   ```

   範例輸出如下。

```
ip-192-168-79-166.us-west-2/192.168.79.166
```

在先前的輸出中，Pod 正在 us-west-2 AWS Region 中的節點上執行。

2. 判斷 Pod 服務帳戶正在使用的端點類型。

```
kubectl describe pod aws-node-6mfgv -n kube-system |grep AWS_STS_REGIONAL_ENDPOINTS
```

範例輸出如下。

```
AWS_STS_REGIONAL_ENDPOINTS: regional
```

如果目前端點是全域範圍，則輸出中會傳回 `global`。如果未傳回任何輸出，則預設端點類型正在使用中且未被覆寫。

3. 如果您的叢集或平台版本與表中列出的版本相同或更高，則可以使用以下命令之一，將服務帳戶使用的端點類型從預設類型變更為其他類型。將 *aws-node* 取代為您服務帳戶的名稱，以及將 *kube-system* 取代為服務帳戶的命名空間。

- 如果您的預設或目前端點類型是全域範圍，且您想將其範圍變更為區域：

```
kubectl annotate serviceaccount -n kube-system aws-node eks.amazonaws.com/sts-regional-endpoints=true
```

如果您使用服務帳戶的 IAM 角色在 Pods 容器中執行的應用程式中產生預先簽章的 S3 URLs，則區域端點的 URL 格式與下列範例類似：

```
https://bucket.s3.us-west-2.amazonaws.com/path?...&X-Amz-Credential=your-access-key-id/date/us-west-2/s3/aws4_request&...
```

- 如果您的預設或目前端點類型是區域範圍，且您想將其範圍變更為全域：

```
kubectl annotate serviceaccount -n kube-system aws-node eks.amazonaws.com/sts-regional-endpoints=false
```

如果您的應用程式明確向 AWS STS 全域端點提出請求，而且您未覆寫在 Amazon EKS 叢集中使用區域端點的預設行為，則請求將會失敗並顯示錯誤。如需詳細資訊，請參閱the section called “Pod 容器會接收到下列錯誤：An error occurred (SignatureDoesNotMatch) when

calling the GetCallerIdentity operation: Credential should be scoped to a valid region”。

如果您使用服務帳戶的 IAM 角色在 Pods 容器中執行的應用程式中產生預先簽章的 S3 URLs，則全域端點的 URL 格式與下列範例類似：

```
https://bucket.s3.amazonaws.com/path?...&X-Amz-Credential=your-access-key-id/date/
us-west-2/s3/aws4_request&...
```

如果您的自動化預期特定格式的預先簽章 URL，或者您的應用程式或使用預先簽章 URLs下游相依性對目標 AWS 區域具有預期，則請進行必要的變更，以使用適當的 AWS STS 端點。

4. 刪除並重新建立與服務帳戶相關聯的任何現有 Pod，以套用登入資料環境變數。變動 Web 掛鉤不會套用至已在執行的 Pod。您可以將 *Pod*、*kube-system* 和 *-l k8s-app=aws-node* 取代為您設定註釋的 Pod 資訊。

```
kubectl delete Pods -n kube-system -l k8s-app=aws-node
```

5. 確認所有 Pod 已重新啟動。

```
kubectl get Pods -n kube-system -l k8s-app=aws-node
```

6. 檢視其中一個 Pod 的環境變數。確認 AWS_STS_REGIONAL_ENDPOINTS 值是您在上一個步驟中所設定的值。

```
kubectl describe pod aws-node-kzbtr -n kube-system |grep AWS_STS_REGIONAL_ENDPOINTS
```

範例輸出如下。

```
AWS_STS_REGIONAL_ENDPOINTS=regional
```

使用 IRSA 驗證至另一個帳戶

您可以透過從另一個帳戶的叢集建立身分提供者或使用鏈結AssumeRole操作來設定跨帳戶 IAM 許可。在下列範例中，帳戶 A 擁有支援服務帳戶 IAM 角色的 Amazon EKS 叢集。在該叢集上執行的 Pod 必須取得帳戶 B 的 IAM 許可。

Example 從另一個帳戶的叢集建立身分提供者

Example

在此範例中，帳戶 A 將為帳戶 B 提供其叢集的 OpenID Connect (OIDC) 發行者 URL。帳戶 B 遵循為叢集建立 IAM OIDC 提供者中的指示the section called “指派 IAM 角色”，並使用帳戶 A 叢集中的 OIDC 發行者 URL。然後，叢集管理員會註釋帳戶 A 叢集中的服務帳戶，以使用帳戶 B 中的角色 (*444455556666*)。

```
apiVersion: v1
kind: ServiceAccount
metadata:
  annotations:
    eks.amazonaws.com/role-arn: arn:aws: iam::444455556666:role/account-b-role
```

Example 使用鏈結的 **`AssumeRole`** 操作

Example

在此範例中，帳戶 B 會建立 IAM 政策，其具有授予帳戶 A 叢集中 Pod 的許可。帳戶 B (*444455556666*) 將該政策連接至 IAM 角色，該角色有信任關係將 `AssumeRole` 許可授予帳戶 A (*111122223333*)。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws: iam::111122223333:root"
      },
      "Action": "sts:AssumeRole",
      "Condition": {}
    }
  ]
}
```

帳戶 A 會建立具有信任政策的角色，該政策會從使用叢集的 OIDC 發行者地址建立的身分提供者取得憑證。

```
{
```

```
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Federated": "arn:aws: iam::111122223333:oidc-provider/oidc.eks.region-
code.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE"
      },
      "Action": "sts:AssumeRoleWithWebIdentity"
    }
  ]
}
```

帳戶 A 將政策連接到該角色，並給予以下許可來擔任帳戶 B 建立的角色。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "sts:AssumeRole",
            "Resource": "arn:aws: iam::444455556666:role/account-b-role"
        }
    ]
}
```

擔任帳戶 B 角色的 Pod 應用程式碼使用兩個設定檔： `account_b_role`和 `account_a_role`。`account_b_role` 描述檔使用 `account_a_role` 描述檔作為其來源。對於 AWS CLI，`~/.aws/config`檔案類似如下。

```
[profile account_b_role]
source_profile = account_a_role
role_arn=arn:aws: iam::444455556666:role/account-b-role

[profile account_a_role]
web_identity_token_file = /var/run/secrets/eks.amazonaws.com/serviceaccount/token
role_arn=arn:aws: iam::111122223333:role/account-a-role
```

若要指定其他 AWS SDKs的鏈結設定檔，請參閱您使用的 SDK 文件。如需詳細資訊，請參閱要建置的工具 AWS。

搭配 AWS SDK 使用 IRSA

使用憑證

若要針對服務帳戶 (IRSA) 使用來自 IAM 角色的登入資料，您的程式碼可以使用任何 AWS SDK 來透過 SDK 為 AWS 服務建立用戶端，並根據預設，開發套件會在位置鏈中搜尋要使用的 AWS Identity and Access Management 登入資料。如果您在建立用戶端或以其他方式初始化 SDK 時未指定登入資料提供者，則會使用服務帳戶登入資料的 IAM 角色。

這樣是有效的，因為服務帳戶的 IAM 角色已新增為預設憑證鏈中的一個步驟。如果您的工作負載目前使用憑證鏈中較早的憑證，那麼即使您為相同工作負載設定服務帳戶的 IAM 角色，系統仍會繼續使用這些憑證。

軟體開發套件會使用 `AssumeRoleWithWebIdentity`動作，自動交換服務帳戶 OIDC 字符，以取得來自 AWS Security Token Service 的臨時憑證。Amazon EKS 和此 SDK 動作會繼續輪換臨時憑證，方法是在臨時憑證到期前進行更新。

將 IAM 角色用於服務帳戶時，Pod 中的容器必須使用支援透過 OpenID Connect Web 身分字符檔案擔任 IAM 角色的 AWS SDK 版本。請確定您使用 SDK 的下列版本 或更新版本 AWS ：

- Java (第 2 版) – 2.10.11
- Java – 1.12.782
- AWS 適用於 Go 的 SDK v1 – 1.23.13
- AWS SDK for Go v2 – 所有版本支援
- Python (Boto3) – 1.9.220
- Python (botocore) – 1.12.200
- AWS CLI – 1.16.232
- 節點：2.525.0 和 3.27.0
- Ruby – 3.58.0
- C++ – 1.7.174
- .NET：3.3.659.1 – 您也必須包含 `AWSSDK.SecurityToken`。
- PHP – 3.110.7

許多熱門的 Kubernetes 附加元件，例如 Cluster Autoscaler、Route Internet traffic with AWS Load Balancer Controller 和 Amazon VPC CNI plugin for Kubernetes 支援服務帳戶的 IAM 角色。

為了確保您使用支援的開發套件，請遵循工具中您偏好的開發套件的安裝說明，在建置容器時建置 AWS。

考量事項

Java

使用 Java 時，您必須在 classpath 上包含 `sts`模組。如需詳細資訊，請參閱 Java SDK 文件中的 WebIdentityTokenFileCredentialsProvider。

擷取簽署金鑰以驗證 OIDC 權杖

Kubernetes 會`ProjectedServiceAccountToken`為每個 Kubernetes 服務帳戶發出 。此字符是 OIDC 字符，進一步是 JSON Web 字符 (JWT) 的類型。Amazon EKS 會為每個叢集託管公有 OIDC 端點，其中包含字符的簽署金鑰，以便外部系統可以驗證它。

若要驗證 `ProjectedServiceAccountToken`，您需要擷取 OIDC 公有簽署金鑰，也稱為 JSON Web 金鑰集 (JWKS)。在您的應用程式中使用這些金鑰來驗證字符。例如，您可以使用 PyJWT Python 程式庫來驗證使用這些金鑰的字符。如需 的詳細資訊`ProjectedServiceAccountToken`，請參閱 the section called “IAM、Kubernetes 和 OpenID Connect (OIDC) 背景資訊”。

先決條件

- 叢集的現有 AWS Identity and Access Management (IAM) OpenID Connect (OIDC) 提供者。若要判定您是否已經擁有一個，或是要建立一個，請參閱 the section called “IAM OIDC 供應商”。
- AWS CLI — 用於使用 AWS 服務的命令列工具，包括 Amazon EKS。如需詳細資訊，請參閱《 AWS 命令列界面使用者指南》中的安裝 。安裝 CLI AWS 之後，建議您也進行設定。如需詳細資訊，請參閱《 AWS 命令列界面使用者指南》中的具有 aws 設定的快速組態。

程序

1. 使用 CLI 擷取 Amazon EKS 叢集的 AWS OIDC URL。

```
$ aws eks describe-cluster --name my-cluster --query 'cluster.identity.oidc.issuer'
"https://oidc.eks.us-west-2.amazonaws.com/id/8EBDXXXX00BAE"
```

2. 使用 curl 或類似工具擷取公有簽署金鑰。結果是 JSON Web 金鑰集 (JWKS)。

> **Important**
>
> Amazon EKS 會調節對 OIDC 端點的呼叫。您應該快取公有簽署金鑰。請遵守回應中包含的`cache-control`標頭。

> **Important**
>
> Amazon EKS 每七天輪換 OIDC 簽署金鑰。

```
$ curl https://oidc.eks.us-west-2.amazonaws.com/id/8EBDXXXX00BAE/keys
{"keys":
[{"kty":"RSA","kid":"2284XXXX4a40","use":"sig","alg":"RS256","n":"wklbXXXXMVfQ","e":"AQAB"}]
```

了解 EKS Pod Identity 如何授予 Pod 對 AWS 服務的存取權

Pod 容器中的應用程式可以使用 AWS SDK 或 AWS CLI，使用 AWS Identity and Access Management (IAM) 許可向 AWS 服務提出 API 請求。應用程式必須使用 AWS 登入資料簽署其 AWS API 請求。

EKS Pod 身分識別提供管理應用程式憑證的功能，類似 Amazon EC2 執行個體設定檔將憑證提供給 Amazon EC2 執行個體的方式。您可以將 IAM 角色與 Kubernetes 服務帳戶建立關聯，並將 Pod 設定為使用服務帳戶，而不是建立 AWS 登入資料並將其分發至容器或使用 Amazon EC2 執行個體的 角色。

每個 EKS Pod 身分識別關聯會將角色對應至指定叢集內命名空間中的服務帳戶。如果您在多個叢集中具有相同的應用程式，則可以在每個叢集中建立相同的關聯，而不必修改角色的信任政策。

如果 Pod 使用具有關聯的服務帳戶，則 Amazon EKS 會在 Pod 的容器中設定環境變數。環境變數設定 AWS SDKs，包括 AWS CLI，以使用 EKS Pod Identity 憑證。

EKS Pod 身分識別的優勢

EKS Pod 身分識別提供下列優勢：

- 最低權限 – 您可以將 IAM 許可範圍限定為服務帳戶，而且只有使用該服務帳戶的 Pod 才能存取這些許可。有此功能也就不需要第三方解決方案，例如 `kiam` 或 `kube2iam`。
- 登入資料隔離 – 當對 Amazon EC2 執行個體中繼資料服務 (IMDS) 的存取受到限制時，Pod 的容器只能擷取與容器使用之服務帳戶相關聯的 IAM 角色登入資料。容器永遠無法存取其他 Pod 中其他容器使用的登入資料。如果 IMDS 未受到限制，Pod 的容器也可以存取 Amazon EKS 節點 IAM 角色，而且容器可能可以存取相同節點上其他 Pod 的 IAM 角色憑證。如需詳細資訊，請參閱限制存取指派給工作節點的執行個體設定檔。

> **Note**
>
> 使用 設定的 Pod `hostNetwork: true`一律具有 IMDS 存取權，但啟用時， AWS SDKs 和 CLI 將使用 Pod Identity 登入資料。

- 可稽核性 – 可透過 AWS CloudTrail 存取和事件記錄，以協助促進回溯性稽核。

> **Important**
>
> 容器不是安全界限，使用 Pod Identity 不會變更此限制。指派給相同節點的 Pod 將共用核心和可能的其他資源，具體取決於您的 Pod 組態。雖然在個別節點上執行的 Pod 會在運算層隔離，但有些節點應用程式在 Kubernetes API 中具有超出個別執行個體範圍的額外許可。一些範例包括 `kubelet`、`kube-proxy`、CSI 儲存驅動程式或您自己的 Kubernetes 應用程式。

EKS Pod Identity 比 更簡單the section called “具有 IRSA 的登入資料”，因為此方法不使用 OIDC 身分提供者。EKS Pod 身分識別具有下列增強功能：

- 獨立操作 – 在許多組織中，建立 OIDC 身分提供者是不同團隊的責任，而不是管理 Kubernetes 叢集。EKS Pod 身分識別具有完整的職責分離，其中 EKS Pod 身分識別關聯的所有組態都在 Amazon EKS 中完成，而 IAM 許可的所有組態則都在 IAM 中完成。
- 可重複使用性 – EKS Pod 身分識別會針對服務帳戶的 IAM 角色使用的每個叢集使用單一 IAM 主體，而不是使用不同的主體。您的 IAM 管理員會將下列主體新增至任何角色的信任政策，以便供 EKS Pod 身分識別使用。

```
            "Principal": {
                "Service": "pods.eks.amazonaws.com"
```

```
        }
```

- 可擴展性 - EKS Pod Identity 中的 EKS Auth 服務會擔任每組臨時登入資料，而不是您在每個 Pod 中執行的每個 AWS SDK。然後，在每個節點上執行的 Amazon EKS Pod 身分識別代理程式會向 SDK 發出憑證。因此，每個節點的負載會減少為一次，而且不會在每個 Pod 中重複。如需該程序的詳細資訊，請參閱 the section called “運作方式”。

如需比較兩種替代方案的詳細資訊，請參閱 the section called “工作負載存取 AWS ”。

設定 EKS Pod 身分識別的概觀

完成下列程序以開啟 EKS Pod 身分識別：

1. the section called “設定 代理程式”— 您只為每個叢集完成此程序一次。如果您的叢集上已啟用 EKS Auto Mode，則不需要完成此步驟。
2. the section called “指派 IAM 角色”— 針對您希望應用程式擁有的每個唯一許可集，完成此程序。
3. the section called “Pod 服務帳戶”— 為每個需要存取 AWS 服務的 Pod 完成此程序。
4. the section called “支援的 SDK”— 確認工作負載使用支援版本的 AWS SDK，以及工作負載使用預設登入資料鏈。

限制

- 每個叢集最多支援 5，000 個 EKS Pod 身分關聯，以將 IAM 角色映射至 Kubernetes 服務帳戶。

考量事項

- IAM 角色關聯：叢集中的每個 Kubernetes 服務帳戶都可以與叢集相同 AWS 帳戶中的一個 IAM 角色建立關聯。若要變更角色，請編輯 EKS Pod Identity 關聯。對於跨帳戶存取，請使用 IAM 角色將存取權委派給角色。若要進一步了解，請參閱《IAM 使用者指南》中的使用 IAM 角色在 AWS 帳戶之間委派存取權。
- EKS Pod Identity Agent：使用 EKS Pod Identity 需要 Pod Identity Agent。代理程式在叢集節點`DaemonSet`上做為 Kubernetes 執行，僅提供登入資料給相同節點上的 Pod。它使用節點的 `hostNetwork`、佔用連接埠`80`和連結本機地址 `2703` (`169.254.170.23`適用於 IPv4、`[fd00:ec2::23]`適用於 IPv6)。如果您的叢集中已停用 IPv6，請停用 Pod Identity Agent 的 IPv6。若要進一步了解，請參閱 EKS Pod Identity Agent 中的停用 IPv6。

- 最終一致性：EKS Pod 身分關聯最終一致，在 API 呼叫後可能延遲數秒。避免在關鍵、高可用性的程式碼路徑中建立或更新關聯。反之，請在個別、較不頻繁的初始化或設定常式中執行這些動作。若要進一步了解，請參閱《EKS 最佳實務指南》中的每個 Pod 的安全群組。
- Proxy 和安全群組考量：對於使用 Proxy 的 Pod，請將 `169.254.170.23`(IPv4) 和 `[fd00:ec2::23]`(IPv6) 新增至`no_proxy`/`NO_PROXY`環境變數，以防止對 EKS Pod Identity Agent 的請求失敗。如果搭配 AWS VPC CNI 使用 Pod 的安全群組，請將 `ENABLE_POD_ENI`旗標設為「true」，並將 `POD_SECURITY_GROUP_ENFORCING_MODE`旗標設為「standard」。若要進一步了解，請參閱將安全群組指派給個別 Pod。

EKS Pod 身分識別叢集版本

若要使用 EKS Pod 身分，叢集的平台版本必須與下表列出的版本相同或更新，或 Kubernetes 版本必須高於表格中列出的版本。若要尋找 Kubernetes 版本的 Amazon EKS Pod Identity Agent 建議版本，請參閱 the section called “驗證相容性”。

Kubernetes 版本	平台版本
未列出的 Kubernetes 版本	所有平台版本都支援
`1.28`	`eks.4`
`1.27`	`eks.8`
`1.26`	`eks.9`

EKS Pod 身分識別限制

EKS Pod 身分識別適用於下列情況：

- 上一個主題 the section called “EKS Pod 身分識別叢集版本” 中列出的 Amazon EKS 叢集版本。
- 叢集中為 Linux Amazon EC2 執行個體的工作節點。

EKS Pod 身分不適用於下列項目：

- AWS Outpost。
- Amazon EKS Anywhere。

- 您在 Amazon EC2 上建立和執行的 Kubernetes 叢集。EKS Pod 身分識別元件僅適用於 Amazon EKS。

您無法搭配以下項目使用 EKS Pod 身分：

- 在 Linux Amazon EC2 執行個體以外的任何位置執行的 Pod。不支援在 AWS Fargate (Fargate) 上執行的 Linux 和 Windows Pod。不支援在 Windows Amazon EC2 執行個體上執行的 Pod。

了解 EKS Pod Identity 的運作方式

Amazon EKS Pod 身分識別關聯提供管理應用程式憑證的功能，類似 Amazon EC2 執行個體設定檔將憑證提供給 Amazon EC2 執行個體的方式。

Amazon EKS Pod 身分識別透過其他 EKS 驗證 API 和在每個節點上執行的代理程式 Pod，為您的工作負載提供憑證。

在您的附加元件中，例如 Amazon EKS 附加元件和自我管理控制器、運算子和其他附加元件，作者需要更新其軟體，才能使用最新的 AWS SDKs。如需 EKS Pod 身分識別與 Amazon EKS 產生的附加元件之間的相容性清單，請參閱上節 the section called “EKS Pod 身分識別限制”。

在程式碼中使用 EKS Pod 身分識別

在程式碼中，您可以使用 AWS SDKs來存取 AWS 服務。您編寫程式碼來建立具有 SDK 之 AWS 服務的用戶端，並依預設在位置鏈中進行 SDK 搜尋，以供 AWS Identity and Access Management 憑證使用。找到有效的憑證後，系統就會停止搜尋。如需使用的預設位置的詳細資訊，請參閱《 AWS SDKs 和工具參考指南》中的登入資料提供者鏈結。

EKS Pod 身分識別已新增至容器憑證提供者，系統會在預設憑證鏈中的某個步驟搜尋此提供者。如果您的工作負載目前使用憑證鏈中較早的憑證，那麼即使您為相同工作負載設定 EKS Pod 身分識別關聯，系統仍會繼續使用這些憑證。如此一來，您就可以先建立關聯，然後再移除舊憑證，以安全地從其他類型的憑證遷移。

容器憑證提供者會從每個節點上執行的代理程式提供臨時憑證。在 Amazon EKS 中，代理程式為 Amazon EKS Pod 身分識別代理程式，而在 Amazon Elastic Container Service 上，代理程式則為 `amazon-ecs-agent`。SDK 使用環境變數來尋找要連線的代理程式。

相反地，服務帳戶的 IAM 角色提供 Web 身分字符， AWS 開發套件必須使用 與 AWS Security Token Service 交換`AssumeRoleWithWebIdentity`。

EKS Pod Identity Agent 如何與 Pod 搭配使用

1. 當 Amazon EKS 啟動使用具有 EKS Pod Identity 關聯的服務帳戶的新 Pod 時，叢集會將下列內容新增至 Pod 資訊清單：

```
    env:
    - name: AWS_CONTAINER_AUTHORIZATION_TOKEN_FILE
      value: "/var/run/secrets/pods.eks.amazonaws.com/serviceaccount/eks-pod-
identity-token"
    - name: AWS_CONTAINER_CREDENTIALS_FULL_URI
      value: "http://169.254.170.23/v1/credentials"
    volumeMounts:
    - mountPath: "/var/run/secrets/pods.eks.amazonaws.com/serviceaccount/"
      name: eks-pod-identity-token
  volumes:
  - name: eks-pod-identity-token
    projected:
      defaultMode: 420
      sources:
      - serviceAccountToken:
          audience: pods.eks.amazonaws.com
          expirationSeconds: 86400 # 24 hours
          path: eks-pod-identity-token
```

2. Kubernetes 會選取要在哪個節點上執行 Pod。然後，節點上的 Amazon EKS Pod 身分識別代理程式會使用 AssumeRoleForPodIdentity 動作以從 EKS 驗證 API 擷取臨時憑證。
3. EKS Pod Identity Agent 會將這些登入資料提供給您在容器內執行 AWS SDKs。
4. 您可以在應用程式中使用 SDK，無需指定憑證提供者來使用預設憑證鏈。或者，您可以指定容器憑證提供者。如需使用的預設位置的詳細資訊，請參閱《 AWS SDKs和工具參考指南》中的登入資料提供者鏈結。
5. SDK 會使用環境變數來連線至 EKS Pod 身分識別代理程式，並擷取憑證。

> **Note**
>
> 如果您的工作負載目前使用憑證鏈中較早的憑證，那麼即使您為相同工作負載設定 EKS Pod 身分識別關聯，系統仍會繼續使用這些憑證。

設定 Amazon EKS Pod Identity Agent

Amazon EKS Pod 身分識別關聯提供管理應用程式憑證的功能，類似 Amazon EC2 執行個體設定檔將憑證提供給 Amazon EC2 執行個體的方式。

Amazon EKS Pod 身分識別透過其他 EKS 驗證 API 和在每個節點上執行的代理程式 Pod，為您的工作負載提供憑證。

> **Tip**
>
> 您不需要在 EKS Auto Mode 叢集上安裝 EKS Pod Identity Agent。此功能內建於 EKS Auto 模式。

考量事項

- 根據預設，EKS Pod Identity Agent 會監聽 IPv4和 IPv6地址，讓 Pod 請求登入資料。代理程式使用 的迴路 (localhost) IP 地址169.254.170.23IPv4和 [fd00:ec2::23]的 localhost IP 地址IPv6。
- 如果您停用IPv6地址，或防止 localhost IPv6 IP 地址，則代理程式無法啟動。若要在無法使用 的節點上啟動代理程式IPv6，請依照中的步驟the section called “停用 IPv6”停用IPv6組態。

建立 Amazon EKS Pod 身分識別代理程式

代理程式先決條件

- 現有 Amazon EKS 叢集。若要部署叢集，請參閱 開始使用。叢集版本和平台版本必須與 EKS Pod Identity 叢集版本中列出的版本相同或更新版本。
- 節點角色具有代理程式在 EKS 驗證 API 中執行 AssumeRoleForPodIdentity 動作的許可。您可以使用 AWS 受管政策：AmazonEKSWorkerNodePolicy 或新增類似以下的自訂政策：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "eks-auth:AssumeRoleForPodIdentity"
            ],
```

```
            "Resource": "*"
        }
    ]
}
```

此動作可能會受到標籤的限制，以限制使用代理程式的 Pod 可以擔任哪些角色。

- 這些節點可以從 Amazon ECR 連上和下載映像。附加元件的容器映像位於檢視 Amazon EKS 附加元件的 Amazon 容器映像登錄檔中列出的登錄檔中。

  請注意，您可以在 的選用組態設定中變更映像位置並提供 `imagePullSecrets` EKS 附加元件 AWS Management Console，並在 CLI 的 AWS `--configuration-values`中提供。
- 這些節點可以連上 Amazon EKS 驗證 API。對於私有叢集，`eks-auth`端點 in AWS PrivateLink 是必要的。

使用 AWS 主控台設定代理程式

1. 開啟 Amazon EKS 主控台。
2. 在左側導覽窗格中，選取叢集，然後選取您要為其設定 EKS Pod 身分識別代理程式附加元件之叢集的名稱。
3. 選擇附加元件索引標籤。
4. 選擇取得更多附加元件。
5. 選取 EKS Pod 身分識別代理程式之附加元件方塊右上方的方塊，然後選擇下一步。
6. 在設定選取的附加元件設定頁面上，選取版本下拉式清單中的任何版本。
7. (選用) 展開選用組態設定以輸入其他組態。例如，您可以提供替代容器映像位置和 `ImagePullSecrets`。包含已接受金鑰的 JSON 結構描述會顯示在附加元件組態結構描述中。

   在組態值中輸入組態金鑰和值。
8. 選擇 Next (下一步)。
9. 確認 EKS Pod 身分識別代理程式 Pod 正在您的叢集上執行。

```
kubectl get pods -n kube-system | grep 'eks-pod-identity-agent'
```

範例輸出如下。

```
eks-pod-identity-agent-gmqp7                                          1/1     Running
   1 (24h ago)   24h
```

```
eks-pod-identity-agent-prnsh                                    1/1     Running
   1 (24h ago)   24h
```

您現在可以在叢集中使用 EKS Pod 身分識別關聯。如需詳細資訊，請參閱the section called “指派 IAM 角色”。

使用 CLI AWS 設定代理程式

1. 執行下列 AWS CLI 命令。使用您叢集的名稱取代 `my-cluster`。

```
aws eks create-addon --cluster-name my-cluster --addon-name eks-pod-identity-agent --
addon-version v1.0.0-eksbuild.1
```

> **Note**
>
> EKS Pod Identity Agent 不會`service-account-role-arn`針對服務帳戶的 IAM 角色使用 。您必須為 EKS Pod 身分識別代理程式提供節點角色的許可。

2. 確認 EKS Pod 身分識別代理程式 Pod 正在您的叢集上執行。

```
kubectl get pods -n kube-system | grep 'eks-pod-identity-agent'
```

範例輸出如下。

```
eks-pod-identity-agent-gmqp7                                    1/1     Running
   1 (24h ago)   24h
eks-pod-identity-agent-prnsh                                    1/1     Running
   1 (24h ago)   24h
```

您現在可以在叢集中使用 EKS Pod 身分識別關聯。如需詳細資訊，請參閱the section called “指派 IAM 角色”。

將 IAM 角色指派給 Kubernetes 服務帳戶

本主題說明如何設定 Kubernetes 服務帳戶，以使用 EKS Pod Identity 擔任 AWS Identity and Access Management (IAM) 角色。任何設定為使用 服務帳戶的 Pod 即可存取該角色具有存取許可的任何 AWS 服務。

若要建立 EKS Pod 身分關聯，只有一個步驟；您可以透過 AWS Management Console、 AWS CLI、AWS SDKs AWS CloudFormation 和其他工具在 EKS 中建立關聯。在任何 Kubernetes 物件中，叢集內沒有任何與關聯相關的資料或中繼資料，而且您不會將任何註釋新增至服務帳戶。

先決條件

- 現有的叢集。如果您沒有 ，您可以依照 中的其中一個指南建立開始使用。
- 建立關聯的 IAM 主體必須具有 `iam:PassRole`。
- 在您的裝置或 AWS CloudShell 上安裝和設定的最新版本 AWS CLI。您可以使用 `aws --version | cut -d / -f2 | cut -d ' ' -f1` 來檢查您的目前版本。適用於 macOS 的 `yum`、`apt-get`或 Homebrew 等套件管理員通常是最新版本 CLI AWS 後面的數個版本。若要安裝最新版本，請參閱《 AWS 命令列界面使用者指南》中的使用 aws 設定安裝 和快速組態。 https://docs.aws.amazon.com/cli/latest/userguide/cli-configure-quickstart.html#cli-configure-quickstart-config安裝在 AWS CloudShell 中的 AWS CLI 版本也可能是最新版本後面的數個版本。若要更新它，請參閱《 AWS CloudShell AWS 使用者指南》中的將 CLI 安裝到您的主目錄。
- `kubectl` 命令列工具安裝在您的裝置或 AWS CloudShell 上。該版本可以與叢集的 Kubernetes 版本相同，也可以比叢集的 Kubernetes 版本更早或更晚，最多一個次要版本。例如，如果您的叢集版本為 `1.29`，則可以搭配使用 `kubectl` `1.28`、`1.29` 或 `1.30` 版。若要安裝或升級 `kubectl`，請參閱 the section called “設定 kubectl 和 eksctl”。
- 包含叢集組態的現有 `kubectl config` 檔案。若要建立 `kubectl config` 檔案，請參閱the section called “使用 kubectl 存取叢集”。

建立 Pod Identity 關聯AWS （主控台）

1. 開啟 Amazon EKS 主控台。
2. 在左側導覽窗格中，選取叢集，然後選取您要為其設定 EKS Pod 身分識別代理程式附加元件之叢集的名稱。
3. 選擇存取索引標籤。
4. 在 Pod 身分識別關聯中，選擇建立。
5. 對於 IAM 角色，選取具有您希望工作負載擁有許可的 IAM 角色。

> **Note**
>
> 此清單僅包含具有下列信任政策的角色，允許 EKS Pod 身分識別使用這些角色。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowEksAuthToAssumeRoleForPodIdentity",
            "Effect": "Allow",
            "Principal": {
                "Service": "pods.eks.amazonaws.com"
            },
            "Action": [
                "sts:AssumeRole",
                "sts:TagSession"
            ]
        }
    ]
}
```

`sts:AssumeRole`— EKS Pod Identity 使用 `AssumeRole`來擔任 IAM 角色，然後再將暫時登入資料傳遞至您的 Pod。

`sts:TagSession`— EKS Pod Identity 使用 `TagSession` 在對 AWS STS 的請求中包含工作階段標籤。

您可以在信任政策中的條件索引鍵中使用這些標籤，以限制哪些服務帳戶、命名空間和叢集可以使用此角色。

如需 Amazon EKS 條件金鑰的清單，請參閱服務授權參考中的 Amazon Elastic Kubernetes Service 定義的條件。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 Kubernetes Service 定義的動作。

6. 針對 Kubernetes 命名空間，選取包含服務帳戶和工作負載的 Kubernetes 命名空間。或者，您可以依叢集中不存在的名稱指定命名空間。
7. 針對 Kubernetes 服務帳戶，選取要使用的 Kubernetes 服務帳戶。Kubernetes 工作負載的資訊清單必須指定此服務帳戶。或者，您可以依叢集中不存在的名稱指定服務帳戶。
8. (選用) 對於標籤，請選擇新增標籤，以在鍵值對中新增中繼資料。這些標籤會套用至關聯，可用於 IAM 政策。

   您可以多次重複此步驟以新增多個標籤。
9. 選擇 Create (建立)。

建立 Pod 身分關聯 (AWS CLI)

1. 如果您要將現有 IAM 政策與 IAM 角色建立關聯，請跳到下一步驟。

   建立 IAM 政策。您可以建立自己的政策，或複製已授予部分所需許可的 AWS 受管政策，並根據您的特定需求進行自訂。如需詳細資訊，請參閱「IAM 使用者指南」中的建立 IAM 政策。

   a. 建立檔案，其中包含您希望 Pod 存取 AWS 的服務許可。如需所有 AWS 服務的所有動作清單，請參閱服務授權參考。

      您可以執行以下命令來建立允許唯讀存取 Amazon S3 儲存貯體的範例政策檔案。您可以選擇性地將組態資訊或引導指令碼存放在此儲存貯體中，而 Pod 中的容器可以從儲存貯體讀取檔案，並將其載入您的應用程式。如果您要建立此範例政策，請將以下內容複製到您的裝置。將 *my-pod-secrets-bucket* 取代為您的儲存貯體名稱並執行命令。

      ```
      cat >my-policy.json <<EOF
      {
          "Version": "2012-10-17",
          "Statement": [
              {
                  "Effect": "Allow",
                  "Action": "s3:GetObject",
                  "Resource": "arn:aws: s3:::my-pod-secrets-bucket"
              }
          ]
      }
      EOF
      ```

   b. 建立 IAM 政策。

      ```
      aws iam create-policy --policy-name my-policy --policy-document file://my-
      policy.json
      ```

2. 建立 IAM 角色並將其與 Kubernetes 服務帳戶建立關聯。

   a. 如果您有想要擔任 IAM 角色的現有 Kubernetes 服務帳戶，則可以略過此步驟。

      建立 Kubernetes 服務帳戶。將以下內容複製到您的裝置。視需要將 *my-service-account* 取代為您要使用的名稱，將 *default* 取代為其他命名空間。如果您變更 *default*，命名空間必須已經存在。

      ```
      cat >my-service-account.yaml <<EOF
      apiVersion: v1
      ```

```
kind: ServiceAccount
metadata:
  name: my-service-account
  namespace: default
EOF
kubectl apply -f my-service-account.yaml
```

執行下列命令。

```
kubectl apply -f my-service-account.yaml
```

b. 執行下列命令以建立 IAM 角色的信任政策檔案。

```
cat >trust-relationship.json <<EOF
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowEksAuthToAssumeRoleForPodIdentity",
            "Effect": "Allow",
            "Principal": {
                "Service": "pods.eks.amazonaws.com"
            },
            "Action": [
                "sts:AssumeRole",
                "sts:TagSession"
            ]
        }
    ]
}
EOF
```

c. 建立角色。將 *my-role* 取代為您的 IAM 角色的名稱，並將 *my-role-description* 取代為您的角色描述。

```
aws iam create-role --role-name my-role --assume-role-policy-document file://
trust-relationship.json --description "my-role-description"
```

d. 將 IAM 政策連接至您的角色。將 *my-role* 取代為您的 IAM 角色名稱，將 *my-policy* 取代為您建立的現有政策名稱。

```
aws iam attach-role-policy --role-name my-role --policy-arn=arn:aws:
 iam::111122223333:policy/my-policy
```

> **Note**
>
> 與服務帳戶的 IAM 角色不同，EKS Pod Identity 不會在服務帳戶上使用註釋。

e. 執行下列命令以建立關聯。視需要使用叢集名稱取代 `my-cluster`，使用您要使用的名稱取代 *`my-service-account`*，使用其他命名空間取代 *`default`*。

```
aws eks create-pod-identity-association --cluster-name my-cluster --role-arn
 arn:aws: iam::111122223333:role/my-role --namespace default --service-account my-
service-account
```

範例輸出如下。

```
{
    "association": {
        "clusterName": "my-cluster",
        "namespace": "default",
        "serviceAccount": "my-service-account",
        "roleArn": "arn:aws: iam::111122223333:role/my-role",
        "associationArn": "arn:aws::111122223333:podidentityassociation/my-
cluster/a-abcdefghijklmnop1",
        "associationId": "a-abcdefghijklmnop1",
        "tags": {},
        "createdAt": 1700862734.922,
        "modifiedAt": 1700862734.922
    }
}
```

> **Note**
>
> 您可以依叢集中不存在的名稱指定命名空間和服務帳戶。您必須建立命名空間、服務帳戶以及使用服務帳戶進行 EKS Pod 身分識別關聯才能運作的工作負載。

確認組態

1. 確認 IAM 角色的信任政策已正確設定。

```
aws iam get-role --role-name my-role --query Role.AssumeRolePolicyDocument
```

範例輸出如下。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Allow EKS Auth service to assume this role for Pod Identities",
            "Effect": "Allow",
            "Principal": {
                "Service": "pods.eks.amazonaws.com"
            },
            "Action": [
                "sts:AssumeRole",
                "sts:TagSession"
            ]
        }
    ]
}
```

2. 確認您在上一步連接至角色的政策已連接至該角色。

```
aws iam list-attached-role-policies --role-name my-role --query
 'AttachedPolicies[].PolicyArn' --output text
```

範例輸出如下。

```
 arn:aws: iam::111122223333:policy/my-policy
```

3. 設定變數以存放您要使用之政策的 Amazon Resource Name (ARN)。將 *my-policy* 取代為您要確認許可的政策名稱。

```
export policy_arn=arn:aws: iam::111122223333:policy/my-policy
```

4. 檢視預設政策版本。

```
aws iam get-policy --policy-arn $policy_arn
```

範例輸出如下。

```
{
    "Policy": {
        "PolicyName": "my-policy",
        "PolicyId": "EXAMPLEBIOWGLDEXAMPLE",
        "Arn": "arn:aws: iam::111122223333:policy/my-policy",
        "Path": "/",
        "DefaultVersionId": "v1",
        [...]
    }
}
```

5. 檢視政策內容，以確保政策包含 Pod 所需的所有許可。如有必要，請將下列命令中的 *1* 取代為先前輸出中傳回的版本。

```
aws iam get-policy-version --policy-arn $policy_arn --version-id v1
```

範例輸出如下。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "s3:GetObject",
            "Resource": "arn:aws: s3:::my-pod-secrets-bucket"
        }
    ]
}
```

如果您在上一步建立了範例政策，則輸出結果相同。如果您建立了不同的政策，則 *example* 內容有所不同。

後續步驟

the section called “Pod 服務帳戶”

使用 EKS Pod 身分目標 IAM 角色存取 AWS 資源

在 Amazon Elastic Kubernetes Service (Amazon EKS) 上執行應用程式時，您可能需要存取存在於不同 AWS 帳戶中 AWS 的資源。本指南說明如何使用 EKS Pod Identity 設定跨帳戶存取，這可讓您的 Kubernetes Pod 使用目標角色存取其他 AWS 資源。

先決條件

開始之前，請確定您已完成下列步驟：

- 設定 Amazon EKS Pod Identity Agent
- 建立 EKS Pod 身分角色

運作方式

Pod Identity 可讓 EKS 叢集中的應用程式透過稱為角色鏈結的程序跨帳戶存取 AWS 資源。

建立 Pod Identity 關聯時，您可以提供兩個 IAM 角色：與 EKS 叢集位於相同帳戶中的 EKS Pod Identity 角色，以及包含您要存取 AWS 之資源的帳戶的目標 IAM 角色（例如 S3 儲存貯體或 RDS 資料庫）。由於 IAM PassRole 需求，EKS Pod 身分角色必須位於 EKS 叢集的帳戶中，而目標 IAM 角色可以位於任何 AWS 帳戶中。PassRole 可讓 AWS 實體將角色擔任委派給其他服務。EKS Pod Identity 使用 PassRole 將角色連線至 Kubernetes 服務帳戶，要求角色和傳遞該角色的身分都與 EKS 叢集位於相同的 AWS 帳戶中。當您的應用程式 Pod 需要存取 AWS 資源時，它會向 Pod Identity 請求登入資料。然後，Pod Identity 會自動依序執行兩個角色假設：先擔任 EKS Pod Identity 角色，然後使用這些登入資料來擔任目標 IAM 角色。此程序為您的 Pod 提供暫時登入資料，該登入資料具有目標角色中定義的許可，允許安全存取其他 AWS 帳戶中的資源。

快取考量

由於快取機制，現有 Pod Identity 關聯中 IAM 角色的更新可能不會在 EKS 叢集上執行的 Pod 中立即生效。Pod Identity Agent 會根據擷取憑證時的關聯組態快取 IAM 憑證。如果關聯只包含 EKS Pod 身分角色，而沒有目標 IAM 角色，則快取的憑證會持續 6 小時。如果關聯同時包含 EKS Pod 身分角色 ARN 和目標 IAM 角色，快取的憑證會持續 59 分鐘。修改現有的關聯，例如更新 EKS Pod 身分角色 ARN 或新增目標 IAM 角色，不會重設現有的快取。因此，在快取的登入資料重新整理之前，代理程式將無法辨識更新。若要更快套用變更，您可以重新建立現有的 Pod；否則，您將需要等待快取過期。

步驟 1：建立和關聯目標 IAM 角色

在此步驟中，您將透過建立和設定目標 IAM 角色來建立安全信任鏈。為了示範，我們將建立新的目標 IAM 角色，在兩個 AWS 帳戶之間建立信任鏈：EKS 叢集帳戶中的 EKS Pod 身分角色（例如

eks-pod-identity-primary-role) 取得在您的目標帳戶中擔任目標 IAM 角色 （ 例如 eks-pod-identity-aws-resources) 的許可，以允許存取 Amazon S3 儲存貯體等 AWS 資源。 AWS

建立目標 IAM 角色

1. 開啟 Amazon IAM 主控台。
2. 在頂端導覽列中，確認您已登入帳戶，其中包含目標 IAM 角色 AWS 的資源 （ 例如 S3 儲存貯體或 DynamoDB 資料表 ）。
3. 在左側導覽窗格中，選擇 Roles (角色)。
4. 選擇建立角色按鈕，然後在「可信任實體類型」下 AWS 建立帳戶。
5. 選擇另一個 AWS 帳戶，輸入 AWS 您的帳戶號碼 (EKS Pod 身分角色所在的帳戶 ），然後選擇下一步。
6. 新增您要與角色建立關聯的許可政策 （ 例如 AmazonS3FullAccess)，然後選擇下一步。
7. 輸入角色名稱，例如 MyCustomIAMTargetRole，然後選擇建立角色。

更新目標 IAM 角色信任政策

1. 建立角色後，您將返回角色清單。尋找並選取您在上一個步驟中建立的新角色 （ 例如 MyCustomIAMTargetRole)。
2. 選取信任關係標籤。
3. 按一下右側的編輯信任政策。
4. 在政策編輯器中，將預設 JSON 取代為您的信任政策。將角色名稱和 IAM 角色 ARN 111122223333中的預留位置值取代為託管 EKS 叢集 AWS 的帳戶 ID。您也可以選擇性地在角色信任政策中使用 PrincipalTags，僅授權來自指定叢集和命名空間的特定服務帳戶擔任您的目標角色。例如：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws: iam::111122223333:root"
      },
      "Action": [
        "sts:AssumeRole",
```

```
        "sts:TagSession"
      ],
      "Condition": {
        "StringEquals": {
          "aws:PrincipalTag/eks-cluster-arn": "arn:aws:eks:us-
east-1:111122223333:cluster/example-cluster",
          "aws:PrincipalTag/kubernetes-namespace": "ExampleNameSpace",
          "aws:PrincipalTag/kubernetes-service-account": "ExampleServiceAccountName"
        },
        "ArnEquals": {
          "aws:PrincipalARN": "arn:aws: iam::111122223333:role/eks-pod-identity-
primary-role"
        }
      }
    }
  ]
}
```

上述政策可讓eks-pod-identeity-primary-role AWS 帳戶 111122223333 中具有相關 EKS Pod Identity Session 標籤的角色擔任此角色。

如果您在 EKS Pod 身分中停用工作階段標籤，EKS Pod 身分也會在擔任目標角色時sts:ExternalId，使用 Pod 的叢集、命名空間和服務帳戶的相關資訊來設定 。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws: iam::111122223333:root"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "sts:ExternalId": "region/111122223333/cluster-name/namespace/service-
account-name"
        },
        "ArnEquals": {
          "aws:PrincipalARN": "arn:aws: iam::111122223333:role/eks-pod-identity-
primary-role"
        }
      }
```

```
    },
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws: iam::111122223333:root"
      },
      "Action": "sts:TagSession"
    }
  ]
}
```

上述政策有助於確保只有預期的叢集、命名空間和服務帳戶才能擔任目標角色。

更新 EKS Pod Identity 角色的許可政策

在此步驟中，您將新增目標 IAM 角色 ARN 作為資源，以更新與 Amazon EKS 叢集相關聯的 EKS Pod Identity 角色的許可政策。

1. 開啟 Amazon EKS 主控台。
2. 在左側導覽窗格中，選取叢集，然後選取 EKS 叢集的名稱。
3. 選擇存取索引標籤。
4. 在 Pod 身分關聯下，選取您的 EKS Pod 身分角色。
5. 選擇許可、新增許可，然後選擇建立內嵌政策。
6. 選擇右側的 JSON。
7. 在政策編輯器中，將預設 JSON 取代為您的許可政策。將角色名稱 和 IAM 角色 ARN 222233334444中的預留位置值取代為您的目標 IAM 角色。例如：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "sts:AssumeRole",
                "sts:TagSession"
            ],
            "Resource": "arn:aws: iam::222233334444:role/eks-pod-identity-aws-
resources"
        }
```

```
    ]
}
```

步驟 2：將目標 IAM 角色與 Kubernetes 服務帳戶建立關聯

在此步驟中，您將在目標 IAM 角色與 EKS 叢集中的 Kubernetes 服務帳戶之間建立關聯。

1. 開啟 Amazon EKS 主控台。
2. 在左側導覽窗格中，選取叢集，然後選取您要新增關聯的叢集名稱。
3. 選擇存取索引標籤。
4. 在 Pod 身分識別關聯中，選擇建立。
5. 為您的工作負載選擇 IAM 角色中的 EKS Pod 身分角色。
6. 在目標 IAM 角色中選擇將由 EKS Pod Identity 角色擔任的目標 IAM 角色。 https://docs.aws.amazon.com/eks/latest/userguide/pod-id-role.html
7. 在 Kubernetes 命名空間欄位中，輸入您要建立關聯的命名空間名稱（例如 `my-app-namespace`)。這會定義服務帳戶所在的位置。
8. 在 Kubernetes 服務帳戶欄位中，輸入將使用 IAM 憑證的服務帳戶名稱（例如 `my-service-account`)。這會將 IAM 角色連結至服務帳戶。
9. 選擇建立以建立關聯。

設定 Pod 以使用 AWS 服務帳戶存取 服務

如果 Pod 需要存取 AWS 服務，則必須將其設定為使用 Kubernetes 服務帳戶。服務帳戶必須與具有存取 AWS 服務許可的 AWS Identity and Access Management (IAM) 角色相關聯。

- 現有的叢集。如果您沒有，您可以使用 中的其中一個指南建立一個開始使用。
- 現有的 Kubernetes 服務帳戶和 EKS Pod Identity 關聯，將服務帳戶與 IAM 角色建立關聯。角色必須具有關聯的 IAM 政策，其中包含您希望 Pod 必須使用 AWS 服務的許可。如需有關服務帳戶和角色之建立和設定方式的詳細資訊，請參閱the section called “指派 IAM 角色”。
- 在您的裝置或 AWS CloudShell 上安裝和設定的最新版本 AWS CLI。您可以使用 `aws --version | cut -d / -f2 | cut -d ' ' -f1` 來檢查您的目前版本。適用於 macOS 的 `yum`、`apt-get`或 Homebrew 等套件管理員通常是最新版本 CLI AWS 後面的數個版本。若要安裝最新版本，請參閱《 AWS 命令列界面使用者指南》中的使用 aws 設定安裝 和快速組態。 https://docs.aws.amazon.com/cli/latest/userguide/cli-configure-quickstart.html#cli-configure-quickstart-config安裝在 AWS CloudShell 中的 AWS CLI 版本也可能是最新版本後面的數個版本。若要更新它，請參閱《 AWS CloudShell AWS 使用者指南》中的將 CLI 安裝到您的主目錄。

- kubectl 命令列工具安裝在您的裝置或 AWS CloudShell 上。該版本可以與叢集的 Kubernetes 版本相同，也可以比叢集的 Kubernetes 版本更早或更晚，最多一個次要版本。例如，如果您的叢集版本為 1.29，則可以搭配使用 kubectl 1.28、1.29 或 1.30 版。若要安裝或升級 kubectl，請參閱 the section called “設定 kubectl 和 eksctl”。
- 包含叢集組態的現有 kubectl config 檔案。若要建立 kubectl config 檔案，請參閱the section called “使用 kubectl 存取叢集”。

1. 使用下列命令來建立部署資訊清單，您可以部署 Pod 來確認組態。以您自己的值取代###。

```
cat >my-deployment.yaml <<EOF
apiVersion: apps/v1
kind: Deployment
metadata:
  name: my-app
spec:
  selector:
    matchLabels:
      app: my-app
  template:
    metadata:
      labels:
        app: my-app
    spec:
      serviceAccountName: my-service-account
      containers:
      - name: my-app
        image: public.ecr.aws/nginx/nginx:X.XX
EOF
```

2. 將清單檔案部署到叢集。

```
kubectl apply -f my-deployment.yaml
```

3. 確認 Pod 存在所需的環境變數。

   a. 檢視在上一個步驟中與部署一起部署的 Pod。

```
kubectl get pods | grep my-app
```

   範例輸出如下。

```
my-app-6f4dfff6cb-76cv9   1/1     Running   0          3m28s
```

b. 確認 Pod 具有服務帳戶字符檔案掛載。

```
kubectl describe pod my-app-6f4dfff6cb-76cv9 | grep
 AWS_CONTAINER_AUTHORIZATION_TOKEN_FILE:
```

範例輸出如下。

```
AWS_CONTAINER_AUTHORIZATION_TOKEN_FILE:  /var/run/secrets/
pods.eks.amazonaws.com/serviceaccount/eks-pod-identity-token
```

4. 確認您的 Pod 可以使用您在連接至角色的 IAM 政策中指派的許可來與 AWS 服務互動。

> **Note**
>
> 當 Pod 使用與服務帳戶相關聯的 IAM 角色 AWS 登入資料時，該 Pod 容器中的 AWS CLI 或其他 SDKs 會使用該角色提供的登入資料。如果您不限制存取提供給 Amazon EKS 節點 IAM 角色的登入資料，Pod 仍然可以存取這些登入資料。如需詳細資訊，請參閱限制存取指派給工作節點的執行個體設定檔。

如果您的 Pod 無法如預期與 服務互動，請完成下列步驟，以確認一切設定正確。

a. 確認您的 Pod 使用支援透過 EKS Pod Identity 關聯擔任 IAM 角色的 AWS SDK 版本。如需詳細資訊，請參閱the section called “支援的 SDK”。

b. 確認部署使用服務帳戶。

```
kubectl describe deployment my-app | grep "Service Account"
```

範例輸出如下。

```
Service Account:  my-service-account
```

根據標籤授予 Pod 對 AWS 資源的存取權

屬性型存取控制 (ABAC) 透過將屬性結合在一起的政策，將權限授予使用者。EKS Pod Identity 會使用叢集名稱、命名空間和服務帳戶名稱等屬性，將標籤連接至每個 Pod 的臨時憑證。這些角色工作階段標籤可讓管理員撰寫單一角色，以根據相符的標籤存取 AWS 資源，跨服務帳戶運作。透過新增對角色

工作階段標籤的支援，您可以在叢集和叢集內的工作負載之間強制執行更嚴格的安全界限，同時重複使用相同的 IAM 角色和 IAM 政策。

具有標籤的範例政策

以下是 IAM 政策範例，可在對應物件以 EKS 叢集名稱標記時授予`s3:GetObject`許可。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:ListBucket"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:GetObjectTagging"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "s3:ExistingObjectTag/eks-cluster-name": "${aws:PrincipalTag/eks-
cluster-name}"
                }
            }
        }
    ]
}
```

啟用或停用工作階段標籤

EKS Pod Identity 會在擔任角色時新增一組預先定義的工作階段標籤。這些工作階段標籤可讓管理員撰寫單一角色，允許根據相符標籤存取 AWS 資源，以跨資源運作。

啟用工作階段標籤

工作階段標籤會使用 EKS Pod Identity 自動啟用，您不需要採取任何動作。根據預設，EKS Pod Identity 會將一組預先定義的標籤連接至您的工作階段。若要在政策中參考這些標籤，請使用 語

法，${aws:PrincipalTag/後面接著 標籤索引鍵。例如 ${aws:PrincipalTag/kubernetes-namespace}。

- eks-cluster-arn
- eks-cluster-name
- kubernetes-namespace
- kubernetes-service-account
- kubernetes-pod-name
- kubernetes-pod-uid

停用工作階段標籤

AWS 會將內嵌工作階段政策、受管政策 ARNs 和工作階段標籤壓縮為具有個別限制的封裝二進位格式。如果您收到PackedPolicyTooLarge錯誤，指出封裝的二進位格式已超過大小限制，您可以停用 EKS Pod Identity 新增的工作階段標籤來嘗試縮減大小。若要停用這些工作階段標籤，請依照下列步驟執行：

1. 開啟 Amazon EKS 主控台。
2. 在左側導覽窗格中，選取叢集，然後選取您要修改的叢集名稱。
3. 選擇存取索引標籤。
4. 在 Pod Identity 關聯中，選擇您要在關聯 ID 中修改的關聯 ID，然後選擇編輯。
5. 在工作階段標籤下，選擇停用工作階段標籤。
6. 選擇儲存變更。

跨帳戶標籤

EKS Pod 身分識別新增的所有工作階段標籤都可轉移；標籤金鑰和值會傳遞至工作負載用於將角色切換至其他帳戶的任何 AssumeRole 動作。您可以在其他帳戶的政策中使用這些標籤，以限制跨帳戶情況中的存取。如需詳細資訊，請參閱《IAM 使用者指南》中的使用工作階段標籤鏈結角色。

自訂標籤

EKS Pod Identity 無法將其他自訂標籤新增至其執行AssumeRole的動作。不過，您套用至 IAM 角色的標籤一律可透過相同格式使用：${aws:PrincipalTag/後面接著 金鑰，例如 ${aws:PrincipalTag/MyCustomTag}。

> **Note**
>
> 在發生衝突的情況下，透過 `sts:AssumeRole` 請求新增至工作階段的標籤具有優先順序。例如，假設：
>
> - `my-cluster` 當 EKS 擔任客戶角色`eks-cluster-name`和
> - 您可以將`eks-cluster-name`標籤新增至值為 的 IAM 角色`my-own-cluster`。
>
> 在這種情況下，前者優先，標籤的值`eks-cluster-name`將為 `my-cluster`。

搭配 AWS SDK 使用 Pod 身分

使用 EKS Pod 身分識別憑證

若要使用來自 EKS Pod Identity 關聯的登入資料，您的程式碼可以使用任何 AWS SDK 來建立 AWS 服務的用戶端，搭配 SDK，而且預設 SDK 會在位置鏈中搜尋要使用的 AWS Identity and Access Management 登入資料。如果您在建立用戶端或初始化 SDK 時未指定登入資料提供者，將使用 EKS Pod Identity 登入資料。

這樣是有效的，因為 EKS Pod 身分識別已新增至容器憑證提供者，系統會在預設憑證鏈中的某個步驟搜尋此提供者。如果您的工作負載目前使用憑證鏈中較早的憑證，那麼即使您為相同工作負載設定 EKS Pod 身分識別關聯，系統仍會繼續使用這些憑證。

如需 EKS Pod 身分識別運作方式的詳細資訊，請參閱 the section called “運作方式”。

使用 了解 EKS Pod Identity 如何授予 Pod 對 AWS 服務的存取權時，Pod 中的容器必須使用支援從 EKS Pod Identity Agent 擔任 IAM 角色的 AWS SDK 版本。請確定您使用 SDK 的下列版本 或更新版本 AWS ：

- Java (版本 2) – 2.21.30
- Java – 1.12.746
- Go v1 – v1.47.11
- Go v2 – release-2023-11-14
- Python (Boto3) – 1.34.41
- Python (botocore) – 1.34.41
- AWS CLI – 1.30.0

AWS CLI – 2.15.0

- JavaScript v2 – 2.1550.0
- JavaScript v3 – v3.458.0
- Kotlin – 1.0.1 版
- Ruby – 3.188.0
- Rust – 版本 -2024-03-13
- C++ – 1.11.263
- .NET – 3.7.734.0
- PowerShell – 4.1.502
- PHP – 3.287.1

為確保您使用支援的 SDK，請遵循工具中您偏好的 SDK 的安裝說明，在建置容器時建置 AWS。

如需支援 EKS Pod Identity 的附加元件清單，請參閱 the section called “Pod Identity Support 參考”。

在 EKS Pod Identity Agent **IPv6**中停用

AWS Management Console

1. 若要在 EKS Pod Identity Agent IPv6中停用 ，請將下列組態新增至 EKS 附加元件的選用組態設定。
   a. 開啟 Amazon EKS 主控台。
   b. 在左側導覽窗格中，選取 Clusters (叢集)，然後選取您要為其設定附加元件之叢集的名稱。
   c. 選擇附加元件索引標籤。
   d. 選取 EKS Pod Identity Agent 附加元件方塊右上角的方塊，然後選擇編輯。
   e. 在設定 EKS Pod Identity Agent 頁面上：
      i. 選取您要使用的版本。我們建議您保留與上一個步驟相同的版本，並以不同的動作更新版本和組態。
      ii. 展開選用組態設定。
      iii. 在組態值"additionalArgs":中輸入具有金鑰的巢狀 JSON 物件的 JSON 金鑰"agent":和值。 產生的文字必須是有效的 JSON 物件。如果此金鑰和值是文字方塊中唯一的資料，請以大括號 { } 括住該金鑰和值。下列範例顯示已啟用網路政策：

```
{
```

```
    "agent": {
        "additionalArgs": {
            "-b": "169.254.170.23"
        }
    }
}
```

此組態會將IPv4地址設定為代理程式使用的唯一地址。

f. 若要透過取代 EKS Pod Identity Agent Pod 來套用新組態，請選擇儲存變更。

Amazon EKS 會使用 Kubernetes `DaemonSet` for EKS Pod Identity Agent 的推展，將變更套用至 EKS 附加元件。您可以在 和 的附加元件更新歷史記錄中追蹤推展的狀態 AWS Management Console `kubectl rollout status daemonset/eks-pod-identity-agent --namespace kube-system`。

`kubectl rollout` 具有下列命令：

```
$ kubectl rollout

history  -- View rollout history
pause    -- Mark the provided resource as paused
restart  -- Restart a resource
resume   -- Resume a paused resource
status   -- Show the status of the rollout
undo     -- Undo a previous rollout
```

如果推展時間過長，Amazon EKS 會復原推展，且具有附加元件更新類型且狀態為失敗的訊息會新增至附加元件的更新歷史記錄。若要調查任何問題，請從推展的歷史記錄開始，並在 EKS Pod Identity Agent Pod `kubectl logs`上執行，以查看 EKS Pod Identity Agent 的日誌。

2. 如果更新歷史記錄中的新項目狀態為成功，則推展已完成，且附加元件在所有 EKS Pod Identity Agent Pod 中使用新組態。

AWS CLI

1. 若要在 EKS Pod Identity Agent IPv6中停用 ，請將下列組態新增至 EKS 附加元件的組態值。

執行下列 AWS CLI 命令。將 `my-cluster` 取代為您的叢集名稱，並將 IAM 角色 ARN 取代為您正在使用的角色。

```
aws eks update-addon --cluster-name my-cluster --addon-name eks-pod-identity-agent \
    --resolve-conflicts PRESERVE --configuration-values '{"agent":{"additionalArgs":
 { "-b": "169.254.170.23"}}}'
```

此組態會將IPv4地址設定為代理程式使用的唯一地址。

Amazon EKS 使用 Kubernetes DaemonSet for EKS Pod Identity Agent 的推展，將變更套用至 EKS 附加元件。您可以在 和 的附加元件更新歷史記錄中追蹤推展的狀態 AWS Management Console `kubectl rollout status daemonset/eks-pod-identity-agent --namespace kube-system`。

`kubectl rollout` 具有下列命令：

```
kubectl rollout

history  -- View rollout history
pause    -- Mark the provided resource as paused
restart  -- Restart a resource
resume   -- Resume a paused resource
status   -- Show the status of the rollout
undo     -- Undo a previous rollout
```

如果推展時間過長，Amazon EKS 會復原推展，且具有附加元件更新類型且狀態為失敗的訊息會新增至附加元件的更新歷史記錄。若要調查任何問題，請從推展的歷史記錄開始，並在 EKS Pod Identity Agent Pod `kubectl logs`上執行，以查看 EKS Pod Identity Agent 的日誌。

使用 EKS Pod Identity 所需的信任政策建立 IAM 角色

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowEksAuthToAssumeRoleForPodIdentity",
            "Effect": "Allow",
            "Principal": {
                "Service": "pods.eks.amazonaws.com"
            },
            "Action": [
                "sts:AssumeRole",
```

```
                "sts:TagSession"
            ]
        }
    ]
}
```

sts:AssumeRole

EKS Pod 身分識別會先使用 AssumeRole 來擔任 IAM 角色，然後將臨時憑證傳至您的 Pod。

sts:TagSession

EKS Pod Identity 使用 TagSession 在 STS AWS 的請求中包含工作階段標籤。

您可以在信任政策中的條件索引鍵中使用這些標籤，以限制哪些服務帳戶、命名空間和叢集可以使用此角色。

如需 Amazon EKS 條件金鑰的清單，請參閱服務授權參考中的 Amazon Elastic Kubernetes Service 定義的條件。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 Kubernetes Service 定義的動作。

使用節點管理運算資源

Kubernetes 節點是執行容器化應用程式的機器。每個節點皆具有下列元件：

- 容器執行時間 – 負責執行容器的軟體。
- kubelet – 確保容器運作狀態良好，並在其相關聯的 Pod 中執行。
- kube-proxy – 維護允許與 Pod 通訊的網路規則。

如需詳細資訊，請參閱 Kubernetes 文件中的節點。

Amazon EKS 叢集可以在 EKS Auto Mode 受管節點、自我管理節點、Amazon EKS 受管節點群組、AWS Fargate 和 Amazon EKS 混合節點的任意組合上排程 Pod。若要進一步了解在叢集中部署的節點，請參閱 the section called “存取叢集資源”。

> **Note**
>
> 除了混合節點之外，節點必須與您建立叢集時選取的子網路位於相同的 VPC 中。不過，節點不必位於相同的子網路中。

比較運算選項

下表提供了幾個準則，以便在決定哪些選項最符合您的要求時進行評估。自我管理節點是支援所有列出的條件的另一個選項，但需要更多手動維護。如需詳細資訊，請參閱the section called “自我管理的節點”。

> **Note**
>
> Bottlerocket 與此表格中的一般資訊有一些特定的差異。如需詳細資訊，請參閱 GitHub 上的 Bottlerocket 文件。

條件	EKS 受管節點群組	EKS 自動模式	Amazon EKS 混合節點
可以部署到 AWS Outpost	否	否	否
可以部署到 AWS Local Zone	是	否	否
可以執行需要 Windows 的容器	是	否	否
可以執行需要 Linux 的容器	是	是	是
可以執行需要 Inferentia 晶片的工作負載	是：僅限 Amazon Linux 節點	是	否
可以執行需要 GPU 的工作負載	是：僅限 Amazon Linux 節點	是	是
可以執行需要 Arm 處理器的工作負載	是	是	是
可以執行 AWS Bottlerocket	是	是	否
Pod 與其他 Pod 共用 CPU、記憶體、儲存和網路資源。	是	是	是
必須部署和管理 Amazon EC2 執行個體	是	否 - 了解 EC2 受管執行個體	是 – 內部部署實體或虛擬機器是由您選擇的工具管理。
必須保護、維護和修補 Amazon EC2 執行個體的作業系統	是	否	是 – 在實體或虛擬機器上執行的作業系統是由您選擇的工具管理。

條件	EKS 受管節點群組	EKS 自動模式	Amazon EKS 混合節點
可以在部署節點時提供引導引數，例如額外的 kubelet 引數。	是 – 搭配自訂 AMI 使用 eksctl或 啟動範本。	否 - 使用 NodeClass 設定節點	是 - 您可以使用 nodeadm 自訂引導引數。請參閱 the section called “混合節點 nodeadm”。
可以從與指派給節點的 IP 地址不同的 CIDR 區塊，將 IP 地址指派給 Pod。	是：以自訂 AMI 使用啟動範本。如需詳細資訊，請參閱「the section called “啟動範本”」。	否	是 - 請參閱 the section called “設定 CNI”。
可以對節點執行 SSH	是	否 - 了解如何對節點進行故障診斷	是
可以將自己的自訂 AMI 部署至節點	是：使用啟動範本	否	是
可以將您自己的自訂 CNI 部署至節點	是：以自訂 AMI 使用啟動範本	否	是
必須自行更新節點 AMI	是 – 如果您部署了 Amazon EKS 最佳化 AMI，當有可用的更新時，您會在 Amazon EKS 主控台中收到通知。您只要在主控台中按一下即可執行更新。如果您部署了自訂 AMI，則在有可用的更新時，不會在 Amazon EKS 主控台中通知您。您必須自行執行更新。	否	是 - 在實體或虛擬機器上執行的作業系統是由您選擇的工具管理。請參閱 the section called “準備作業系統”。

條件	EKS 受管節點群組	EKS 自動模式	Amazon EKS 混合節點
必須自行更新節點 Kubernetes 版本	是 – 如果您部署了 Amazon EKS 最佳化 AMI，當有可用的更新時，您會在 Amazon EKS 主控台中收到通知。您只要在主控台中按一下即可執行更新。如果您部署了自訂 AMI，則在有可用的更新時，不會在 Amazon EKS 主控台中通知您。您必須自行執行更新。	否	是 - 您可以使用自己的工具或 來管理混合節點升級nodeadm。請參閱 the section called “升級混合節點”。
可以搭配 Pod 使用 Amazon EBS 儲存	是	是，作為整合功能。了解如何建立儲存體方案。	否
可以搭配 Pod 使用 Amazon EFS 儲存	是	是	否
可以使用 Amazon FSx for Lustre 儲存搭配 Pod	是	是	否
可以針對服務使用 Network Load Balancer	是	是	是 - 必須使用目標類型 ip。
Pod 可以在公有子網路中執行	是	是	否 - 在內部部署環境中執行的 Pod。
可以將不同的 VPC 安全群組指派給個別 Pod	是 – 僅限 Linux 節點	否	否

條件	EKS 受管節點群組	EKS 自動模式	Amazon EKS 混合節點
可以執行 Kubernetes DaemonSets	是	是	是
Pod 資訊清單中`HostNetwork` 的支援 `HostPort`和	是	是	是
AWS 區域可用性	所有 Amazon EKS 支援的區域	所有 Amazon EKS 支援的區域	除 AWS GovCloud (US) 區域和中國區域以外所有 Amazon EKS 支援的區域。
可以在 Amazon EC2 專用主機上執行容器	是	否	否
定價	執行多個 Pod 的 Amazon EC2 執行個體成本。如需詳細資訊，請參閱 Amazon EC2 定價。	在叢集中啟用 EKS Auto Mode 時，除了標準 EC2 執行個體費用之外，您還需要為使用 Auto Mode 運算功能啟動的執行個體支付個別費用。數量會隨啟動的執行個體類型和叢集所在的 AWS 區域而有所不同。如需詳細資訊，請參閱 Amazon EKS 定價。	混合節點 vCPU 每小時的成本。如需詳細資訊，請參閱 Amazon EKS 定價。

使用受管節點群組簡化節點生命週期

Amazon EKS 受管節點群組會自動化 Amazon EKS Kubernetes 叢集節點 (Amazon EC2 執行個體) 的佈建和生命週期管理。

使用 Amazon EKS 受管節點群組時，不需要個別佈建或註冊提供運算容量的 Amazon EC2 執行個體，來執行 Kubernetes 應用程式。您可以透過單一操作來建立、自動更新或終止叢集的節點。節點更新和終止會自動耗盡節點，以確保您的應用程式保持可用。

每個受管節點都會佈建為由 Amazon EKS 為您管理的 Amazon EC2 Auto Scaling 群組的一部分。包括執行個體和 Auto Scaling 群組在內的每個資源都會在您的 AWS 帳戶中執行。每個節點群組都可以跨您定義的多個可用區域執行。

受管節點群組也可以選擇性地利用節點自動修復，持續監控節點的運作狀態。它會自動回應偵測到的問題，並盡可能取代節點。這有助於叢集的整體可用性，只需最少的手動介入。如需詳細資訊，請參閱the section called “節點運作狀態”。

您可以使用 Amazon EKS 主控台、`eksctl`、 AWS CLI、 AWS API 或基礎設施做為程式碼工具，包括 AWS CloudFormation，將受管節點群組新增至新的或現有的叢集。Kubernetes Cluster Autoscaler 會自動標記作為受管節點群組一部分啟動的節點，以進行自動探索。您可以使用節點群組將 Kubernetes 標籤套用至節點，並隨時加以更新。

使用 Amazon EKS 受管節點群組無需額外費用，您只需支付您佈建 AWS 的資源。這些包括 Amazon EC2 執行個體、Amazon EBS 磁碟區、Amazon EKS 叢集時數，以及任何其他 AWS 基礎設施。沒有最低費用，也沒有前期承諾。

若要開始使用新的 Amazon EKS 叢集和受管節點群組，請參閱 the section called “建立叢集（主控台和 CLI)”。

若要將受管理的節點群組新增至現有叢集，請參閱 the section called “建立”。

受管節點群組概念

- Amazon EKS 受管節點群組會為您建立和管理 Amazon EC2 執行個體。
- 每個受管節點都會佈建為由 Amazon EKS 為您管理的 Amazon EC2 Auto Scaling 群組的一部分。此外，包括 Amazon EC2 執行個體和 Auto Scaling 群組在內的每個資源都會在您的 AWS 帳戶中執行。
- 受管節點群組的 Auto Scaling 群組會跨越您在建立群組時指定的每個子網路。
- Amazon EKS 會標記受管節點群組資源，以便將其設定為使用 Kubernetes Cluster Autoscaler。

> **Important**
>
> 如果您正在多個由 Amazon EBS 磁碟區支援的可用區域中執行具狀態的應用程式，並使用 Kubernetes Cluster Autoscaler，您應該設定多個節點群組，每個群組的範圍都限定在單一可用區域。另外，您應該啟用 `--balance-similar-node-groups` 功能。

- 部署受管節點時，您可以使用自訂啟動範本來獲得更高層級的靈活性和自訂能力。例如，可指定額外的 `kubelet` 引數並使用自訂 AMI。如需詳細資訊，請參閱the section called “啟動範本”。如果您在第一次建立受管節點群組時未使用自訂啟動範本，則會有自動產生的啟動範本。請勿手動修改此自動產生的範本，否則會發生錯誤。
- Amazon EKS 會遵循受管節點群組上 CVE 和安全修補程式的共同責任模型。受管節點執行 Amazon EKS 最佳化 AMI 時，Amazon EKS 負責在報告錯誤或問題時建置 AMI 的修補版本。我們可以發佈修正程式。不過，您必須負責將這些修補的 AMI 版本部署到受管節點群組。當受管節點執行自訂 AMI 時，當回報錯誤或問題，然後部署 AMI 時，您需負責建置 AMI 的修補版本。如需詳細資訊，請參閱the section called “更新”。
- Amazon EKS 受管節點群組可以在公有和私有子網路中啟動。如果在 2020 年 4 月 22 日或之後啟動公有子網路中的受管節點群組，則子網路必須將 `MapPublicIpOnLaunch` 設定為「true」，執行個體才能成功加入叢集。如果公有子網路是在 2020 年 3 月 26 日或之後使用 `eksctl` 或 Amazon EKS vended AWS CloudFormation 範本建立，則此設定已設為 true。如果公有子網路是在 2020 年 3 月 26 日之前所建立，則必須手動變更設定。如需詳細資訊，請參閱修改子網的公有 IPv4 定址屬性。
- 在私有子網路中部署受管節點群組時，您必須確保該群組可存取 Amazon ECR 以提取容器映像。您可以透過將 NAT 閘道連接到子網路的路由表，或新增下列 AWS PrivateLink VPC 端點來執行此操作：
  - Amazon ECR API 端點界面 – `com.amazonaws.region-code.ecr.api`
  - Amazon ECR Docker 登錄檔 API 端點界面 – `com.amazonaws.region-code.ecr.dkr`
  - Amazon S3 閘道端點 – `com.amazonaws.region-code.s3`

  如需了解其他常用的服務和端點，請參閱 the section called “私有叢集”。
- 受管節點群組無法部署在 AWS Outposts 或 AWS Wavelength 中。您可以在 AWS Local Zones 上建立受管節點群組。如需詳細資訊，請參閱the section called “ AWS 本機區域”。
- 您可以在單一叢集內建立多個受管節點群組。例如，您可以為某些工作負載建立一個具有標準 Amazon EKS 最佳化 Amazon Linux AMI 的節點群組，並為需要 GPU 支援的工作負載建立一個具有 GPU 變體的節點群組。

- 如果受管節點群組遇到 Amazon EC2 執行個體狀態檢查故障問題，Amazon EKS 會傳回錯誤代碼，以協助您診斷問題。如需詳細資訊，請參閱the section called “受管節點群組錯誤代碼”。
- Amazon EKS 會將 Kubernetes 標籤新增至受管節點群組執行個體。這些 Amazon EKS 提供的標籤前面會加上 `eks.amazonaws.com`。
- 在終止或更新期間，Amazon EKS 會使用 Kubernetes API 自動耗盡節點。
- 使用 終止節點`AZRebalance`或減少所需的節點計數時，不遵守 Pod 中斷預算。這些動作會嘗試在節點上移出 Pod。但是，如果超過 15 分鐘，無論節點上的所有 Pod 是否終止，節點都會終止。若要延長期間直到節點終止，請將 lifecycle hook 新增至 Auto Scaling 群組。如需詳細資訊，請參閱 Amazon EC2 Auto Scaling 使用者指南中的新增 lifecycle hook。
- 若要在收到 Spot 中斷通知或容量重新平衡通知後正確執行耗盡程序，必須將 `CapacityRebalance` 設定為 `true`。
- 更新受管節點群組會遵守您為 Pod 設定的 Pod 中斷預算。如需詳細資訊，請參閱the section called “更新行為詳細資訊”。
- 使用 Amazon EKS 受管節點群組不會產生額外成本。您只需為佈建 AWS 的資源付費。
- 如果要為節點加密 Amazon EBS 磁碟區，則可以使用啟動範本部署節點。若要在不使用啟動範本的情況下部署具有加密 Amazon EBS 磁碟區的受管節點，請加密帳戶中建立的所有新 Amazon EBS 磁碟區。如需詳細資訊，請參閱《Amazon EC2 使用者指南》中的預設加密。

受管節點群組容量類型

建立受管節點群組時，您可以選擇隨需或 Spot 容量類型。Amazon EKS 部署具有 Amazon EC2 Auto Scaling 群組的受管節點群組，該群組僅包含隨需執行個體或僅包含 Amazon EC2 Spot 執行個體。您可以將容錯應用程式的 Pod 排程至 Spot 受管節點群組，以及將容錯應用程式排程至單一 Kubernetes 叢集內的隨需節點群組。依預設，受管節點群組會部署隨需 Amazon EC2 執行個體。

On-Demand

透過隨需執行個體，您只需要按秒數支付運算容量開銷，無需簽訂長期合約。

預設情況下，如果未指定容量類型，則會使用隨需執行個體佈建受管節點群組。受管節點群組會代表您設定 Amazon EC2 Auto Scaling 群組，並套用下列設定：

- 佈建隨需容量的配置策略設定為 `prioritized`。在滿足隨需容量時，受管節點群組會 API 中傳遞執行個體類型的順序，決定先使用哪一種執行個體類型。例如，您可以按照下列順序指定三種執行個體類型：`c5.large`、`c4.large`，以及 `c3.large`。當您的隨需執行個體啟動時，受管節點群組會

從 c5.large 開始，然後是 c4.large，再來是 c3.large，以滿足隨需容量。如需詳細資訊，請參閱《Amazon EC2 Auto Scaling 使用者指南》中的 Amazon EC2 Auto Scaling 群組。

- Amazon EKS 會將下列 Kubernetes 標籤新增至指定容量類型之受管節點群組中的所有節點：eks.amazonaws.com/capacityType: ON_DEMAND。您可以使用此標籤，在隨需節點上排程可設定狀態或可容錯的應用程式。

Spot

Amazon EC2 Spot 執行個體是備用的 Amazon EC2 容量，可提供隨需價格的極低折扣。EC2 需要取回容量時，就可能會以兩分鐘中斷通知的方式中斷 Amazon EC2 Spot 執行個體。如需詳細資訊，請參閱《Amazon EC2 使用者指南》中的 Spot 執行個體。您可以使用 Amazon EC2 Spot 執行個體設定受管節點群組，以最佳化 Amazon EKS 叢集中執行之運算節點的成本。

若要在受管節點群組內使用 Spot 執行個體，請將容量類型設定為spot，以建立受管節點群組。受管節點群組代表您設定 Amazon EC2 Auto Scaling 群組，並套用下列 Spot 最佳實務：

- 為確保您的 Spot 節點已在最佳 Spot 容量集區中佈建，配置策略會設定為下列之一：
  - price-capacity-optimized (PCO) – 在具有 Kubernetes 版本1.28或更高版本的叢集中建立新的節點群組時，配置策略會設定為 price-capacity-optimized。不過，在 Amazon EKS 受管節點群組開始支援 PCO capacity-optimized之前，不會變更已使用 建立的節點群組的配置策略。
  - capacity-optimized (CO) – 在具有 Kubernetes 1.27版本或更低版本的叢集中建立新的節點群組時，配置策略會設為 capacity-optimized。

  若要增加可用於配置容量的 Spot 容量集區數量，請將受管節點群組設定為使用多個執行個體類型。
- 已啟用 Amazon EC2 Spot 容量重新平衡功能，以便 Amazon EKS 可以正常地消耗和重新平衡 Spot 節點，在 Spot 節點中斷風險提高時，將應用程式中斷情況降至最低。如需詳細資訊，請參閱《Amazon EC2 Auto Scaling 使用者指南》中的 Amazon EC2 Auto Scaling 容量重新平衡。
  - Spot 節點收到重新平衡建議時，Amazon EKS 會自動嘗試啟動新的替代 Spot 節點。
  - 如果 Spot 兩分鐘中斷通知在取代 Spot 節點處於 Ready 狀態前到達，則 Amazon EKS 會開始消耗收到重新平衡建議的 Spot 節點。Amazon EKS 將竭盡全力耗盡節點。因此，無法保證 Amazon EKS 會等待替代節點加入叢集，再耗盡現有的節點。
  - 啟動取代 Spot 節點，並且在 Kubernetes 處於 Ready 狀態時，Amazon EKS 會包圍隔離並消耗收到重新平衡建議的 Spot 節點。覆寫 Spot 節點可確保服務控制器不會傳送任何新請求至此 Spot 節點。它也會從狀況良好、作用中的 Spot 節點清單中移除。耗盡 Spot 節點可確保正常移出執行中的 Pod。

- Amazon EKS 會將下列 Kubernetes 標籤新增至指定容量類型之受管節點群組中的所有節點：`eks.amazonaws.com/capacityType: SPOT`。您可以使用此標籤在 Spot 節點上排程可容錯的應用程式。

決定是否要部署具有隨需或 Spot 容量的節點群組時，應考慮下列條件：

- Spot 執行個體適用於無狀態、容錯、靈活的應用程式。其中包括批次和機器學習訓練工作負載、大數據 ETL (例如 Apache Spark)、佇列處理應用程式，以及無狀態 API 端點。由於 Spot 是可能隨時間變更的備用 Amazon EC2 容量，因此建議您將 Spot 容量用於可接受中斷的工作負載。更具體地說，Spot 容量適用於可容忍無法使用所需容量期間的工作負載。
- 我們建議您將隨需模式用於不容錯的應用程式。這包括叢集管理工具 (例如監控和操作工具)、需要 `StatefulSets` 的部署，以及狀態應用程式 (例如資料庫)。
- 為了在使用 Spot 執行個體時最大化應用程式的可用性，建議您將 Spot 受管節點群組設定為使用多個執行個體類型。建議您在使用多個執行個體類型時，套用下列規則：
  - 在受管節點群組中，如果您使用的是 Cluster Autoscaler，我們建議您使用一組具有相同 vCPU 和記憶體資源數量的彈性執行個體類型。這是為了確保叢集中的節點如預期般擴展。例如，如果您需要四個 vCPU 和八個 GiB 記憶體，請使用 `c3.xlarge`、`c4.xlarge`、`c5.xlarge`、`c5d.xlarge`、`c5a.xlarge`、`c5n.xlarge` 或其他類似的執行個體類型。
  - 若要增強應用程式可用性，建議部署多個 Spot 受管節點群組。為此，每個群組應該使用具有相同 vCPU 和記憶體資源的彈性執行個體類型集。例如，如果您需要 4 個 vCPU 和 8 個 GiB 記憶體，建議您使用 `c3.xlarge`、`c4.xlarge`、`c5.xlarge`、`c5d.xlarge`、`c5a.xlarge`、`c5n.xlarge` 或其他類似的執行個體類型建立一個受管節點群組，以及使用 `m3.xlarge`、`m4.xlarge`、`m5.xlarge`、`m5d.xlarge`、`m5a.xlarge`、`m5n.xlarge` 或其他類似的執行個體類型建立另一個受管節點。
  - 使用自訂啟動範本的 Spot 容量類型部署節點群組時，請使用 API 來傳遞多個執行個體類型。請勿透過啟動範本傳遞單一執行個體類型。如需使用啟動範本部署節點群組的詳細資訊，請參閱 the section called “啟動範本”。

為您的叢集建立受管節點群組

本主題會描述如何啟動已向 Amazon EKS 叢集註冊的節點的 Amazon EKS 受管節點群組。節點加入叢集後，您就可以將 Kubernetes 應用程式部署至其中。

如果這是您第一次啟動 Amazon EKS 受管節點群組，我們建議您改為遵循 中的其中一個指南開始使用。這些指南提供使用節點建立 Amazon EKS 叢集的逐步解說。

> **Important**
>
> - Amazon EKS 節點是標準的 Amazon EC2 執行個體。會根據一般 Amazon EC2 價格向您收費。如需詳細資訊，請參閱 Amazon EC2 定價。
> - 您無法在已啟用 AWS Outposts 或 Wavelength AWS 的 AWS 區域中建立受管節點。您可改為建立自我管理的節點。如需詳細資訊，請參閱the section called “Amazon Linux”、the section called “Windows”及the section called “Bottlerocket”。您也可以在 Outpost 上建立自我管理的 Amazon Linux 節點群組。如需詳細資訊，請參閱the section called “節點”。
> - 如果您未為 Amazon EKS 最佳化 Linux 或 Bottlerocket 隨附的`bootstrap.sh`檔案指定 AMI ID，受管節點群組會強制執行 值的最大值`maxPods`。對於少於 30 個 vCPU 的執行個體，數量上限為 `110`。對於具有 30 個以上 vCPU 的執行個體，數量上限會跳至 `250`。這些數字是根據 Kubernetes 可擴展性閾值和內部 Amazon EKS 可擴展性團隊測試的建議設定。如需詳細資訊，請參閱 Amazon VPC CNI 外掛程式增加每節點的 Pod 限制部落格文章。

- 現有 Amazon EKS 叢集。若要部署叢集，請參閱 the section called “建立叢集”。
- 供節點使用的現有 IAM 角色。若要建立服務角色，請參閱the section called “節點 IAM 角色”。如果此角色沒有 VPC CNI 的任何政策，則 VPC CNI Pod 需要以下單獨的角色。
- （選用，但建議）適用於 Kubernetes 附加元件的 Amazon VPC CNI 外掛程式，其設定為具有必要 IAM 政策的自有 IAM 角色。如需詳細資訊，請參閱the section called “為 IRSA 設定”。
- 熟悉選擇最佳 Amazon EC2 節點執行個體類型中列出的考量事項。取決於您選擇的執行個體類型，您的叢集和 VPC 可能還有其他先決條件。
- 若要新增 Windows 受管節點群組，您必須先啟用叢集的 Windows 支援。如需詳細資訊，請參閱the section called “啟用 Windows 支援”。

您可以使用下列其中一項來建立受管節點群組：

- the section called “`eksctl`”
- the section called “AWS Management Console”

eksctl

使用 eksctl 建立受管節點群組

此程序需要 eksctl 版本 0.210.0 或更新版本。您可使用以下命令檢查您的版本：

```
eksctl version
```

如需有關安裝或更新 eksctl 的指示，請參閱 eksctl 文件中的 Installation 一節。

1. （選用）如果 AmazonEKS_CNI_Policy 受管 IAM 政策連接至您的 Amazon EKS 節點 IAM 角色，建議您改為將其指派給與 Kubernetes aws-node服務帳戶相關聯的 IAM 角色。如需詳細資訊，請參閱the section called “為 IRSA 設定”。
2. 使用或不使用自訂啟動範本來建立受管節點群組。手動指定啟動範本可讓您更靈活地自訂節點群組。例如，此方式可以允許部署自訂 AMI，或對 Amazon EKS 最佳化 AMI 中的 boostrap.sh 指令碼提供引數。如需每個可用選項和預設值的完整清單，請輸入以下命令。

   ```
   eksctl create nodegroup --help
   ```

   在下列命令中，將 *my-cluster* 取代為您的叢集名稱，並將 *my-mng* 取代為您的節點群組名稱。節點群組名稱不能超過 63 個字元。它必須以字母或數字開頭，但剩餘字元也可以包含連字符和底線。

   > **Important**
   >
   > 如果您在第一次建立受管節點群組時未使用自訂啟動範本，則之後不要為節點群組使用自訂啟動範本。如果您未指定自訂啟動範本，系統會自動產生啟動範本，我們不建議您手動修改。手動修改此自動產生的啟動範本可能會導致錯誤。

沒有啟動範本

eksctl 會在您的帳戶中建立預設 Amazon EC2 啟動範本，並使用根據您指定之選項建立的啟動範本來部署節點群組。指定 --node-type 的值之前，請參閱 the section called “Amazon EC2 執行個體類型”。

將 *ami-family* 取代為允許的關鍵字。如需詳細資訊，請參閱 eksctl 文件中的 Setting the node AMI Family (設定節點 AMI 系列)。將 *my-key* 取代為您的 Amazon EC2 金鑰對或公有金鑰的名稱。此金鑰會在節點啟動後用於將 SSH 套用至節點。

> **Note**
>
> 對於 Windows，此命令不會啟用 SSH。而是將 Amazon EC2 金鑰對與執行個體建立關聯，讓您可以 RDP 至該執行個體中。

如果您還沒有 Amazon EC2 金鑰對，您可以在 中建立一個金鑰對 AWS Management Console。如需 Linux 資訊，請參閱《Amazon EC2 使用者指南》中的 Amazon EC2 金鑰對和 Linux 執行個體。Amazon EC2 如需 Windows 資訊，請參閱《Amazon EC2 使用者指南》中的 Amazon EC2 金鑰對和 Windows 執行個體。 Amazon EC2

如果符合下列條件，我們建議封鎖 Pod 存取 IMDS：

- 您計劃將 IAM 角色指派給所有 Kubernetes 服務帳戶，讓 Pod 僅擁有所需的最低許可。
- 叢集中沒有任何 Pod 因其他原因需要存取 Amazon EC2 執行個體中繼資料服務 (IMDS)，例如擷取目前 AWS 區域。

如需詳細資訊，請參閱限制存取指派給工作節點的執行個體設定檔。

如果您想要封鎖對 IMDS 的 Pod 存取，請將 `--disable-pod-imds`選項新增至下列命令。

```
eksctl create nodegroup \
  --cluster my-cluster \
  --region region-code \
  --name my-mng \
  --node-ami-family ami-family \
  --node-type m5.large \
  --nodes 3 \
  --nodes-min 2 \
  --nodes-max 4 \
  --ssh-access \
  --ssh-public-key my-key
```

您的執行個體可以選擇性地將大量 IP 地址指派給 Pod、從與執行個體不同的 CIDR 區塊將 IP 地址指派給 Pod，以及部署到沒有網際網路存取的叢集。如需詳細資訊，請參閱 the section called “增加 IP 地址”、the section called “自訂聯網” 和 the section called “私有叢集” 以取得要新增至之前命令的其他選項。

受管節點群組會根據執行個體類型，計算和套用可在節點群組每個節點上執行的 Pod 數目上限的單一值。如果您建立具有不同執行個體類型的節點群組，所有執行個體類型計算的最小值會套用為節點群組

中每個執行個體類型可執行的 Pod 數量上限。受管節點群組會使用 Amazon EKS 建議的每個 Amazon EC2 執行個體類型最大 Pod 中參考的指令碼來計算值。

使用啟動範本

啟動範本必須已存在，且必須符合啟動範本組態基本中指定的要求。如果符合下列條件，我們建議封鎖 Pod 存取 IMDS：

- 您計劃將 IAM 角色指派給所有 Kubernetes 服務帳戶，讓 Pod 僅擁有所需的最低許可。
- 叢集中沒有任何 Pod 因其他原因需要存取 Amazon EC2 執行個體中繼資料服務 (IMDS)，例如擷取目前 AWS 區域。

如需詳細資訊，請參閱限制存取指派給工作節點的執行個體設定檔。

如果您想要封鎖對 IMDS 的 Pod 存取，請在啟動範本中指定必要的設定。

a. 將以下內容複製到您的裝置。取代 *example values*，然後執行修改後的命令來建立 `eks-nodegroup.yaml` 檔案。在不使用啟動範本的情況下部署時指定的數個設定會移至啟動範本。如果您未指定 `version`，則會使用範本的預設版本。

```
cat >eks-nodegroup.yaml <<EOF
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig
metadata:
  name: my-cluster
  region: region-code
managedNodeGroups:
- name: my-mng
  launchTemplate:
    id: lt-id
    version: "1"
EOF
```

如需 `eksctl` 組態檔案設定的完整清單，請參閱 `eksctl` 文件中的組態檔案結構描述。您的執行個體可以選擇性地將大量 IP 地址指派給 Pod、從與執行個體不同的 CIDR 區塊將 IP 地址指派給 Pod，以及部署到沒有傳出網際網路存取的叢集。如需詳細資訊，請參閱 the section called “增加 IP 地址”、 the section called “自訂聯網”和 ，the section called “私有叢集”以取得要新增至組態檔案的其他選項。

如果您未在啟動範本中指定 AMI ID，受管節點群組會根據執行個體類型，計算並套用可在節點群組每個節點上執行的 Pod 數目上限的單一值。如果您建立具有不同執行個體類型的節點群組，所有執行個體類型計算的最小值會套用為節點群組中每個執行個體類型可執行的 Pod 數量上限。受管節點群組會使用每個 Amazon EC2 執行個體類型的 Amazon EKS 建議 Pod 上限中參考的指令碼來計算值。

如果您在啟動範本中指定 AMI ID，請指定如果您使用自訂聯網或想要增加指派給執行個體的 IP 地址數量，可在節點群組的每個節點上執行的 Pod 數量上限。如需詳細資訊，請參閱the section called “Amazon EKS 為每種 Amazon EC2 執行個體類型建議 Pod 數量上限”。

b. 使用下列命令部署節點群組。

```
eksctl create nodegroup --config-file eks-nodegroup.yaml
```

AWS Management Console

使用 建立受管節點群組 AWS Management Console

1. 等待您的叢集狀態顯示為 ACTIVE。您無法為尚未 的叢集建立受管節點群組ACTIVE。
2. 開啟 Amazon EKS 主控台。
3. 選擇要在其中建立受管節點群組的叢集名稱。
4. 選取 Compute (運算) 標籤。
5. 選擇 Add node group (新增節點群組)。
6. 在 Configure node group (配置節點群組) 頁面上，相應地填寫參數，然後選擇 Next (下一步)。
   - Name (名稱) – 輸入受管節點群組的唯一名稱。節點群組名稱不能超過 63 個字元。它必須以字母或數字開頭，但剩餘字元也可以包含連字符和底線。
   - Node IAM role (節點 IAM 角色)：選擇要與節點群組搭配使用的節點執行個體角色。如需詳細資訊，請參閱the section called “節點 IAM 角色”。

> ⚠ Important
>
> - 您無法使用用來建立任何叢集的相同角色。
> - 我們建議使用任何自我管理節點群組目前未使用的角色。否則，您要計劃與新的自我管理節點組一起使用。如需詳細資訊，請參閱the section called “Delete”。

- Use launch template (使用啟動範本)：(選用) 選擇是否要使用現有的啟動範本。選取啟動範本名稱。然後，選取 Launch template version (啟動範本版本)。如果您未選取版本，Amazon EKS 會使用範本的預設版本。啟動範本可讓您自訂更多節點群組，例如允許您部署自訂 AMI、將相當多的 IP 地址指派給 Pod、從與執行個體不同的 CIDR 區塊將 IP 地址指派給 Pod，以及將節點部署至沒有傳出網際網路存取的叢集。如需詳細資訊，請參閱the section called “增加 IP 地址”、the section called “自訂聯網”及the section called “私有叢集”。

  啟動範本必須符合使用啟動範本自訂受管節點中的要求。如果您不使用自己的啟動範本，Amazon EKS API 會在您的帳戶中建立預設 Amazon EC2 啟動範本，並使用預設啟動範本部署節點群組。

  如果您為服務帳戶實作 IAM 角色，請直接將必要的許可指派給需要存取 AWS 服務的每個 Pod，而且叢集中沒有任何 Pod 需要存取 IMDS，例如擷取目前 AWS 區域，則您也可以停用在啟動範本中未使用主機聯網之 Pod 的 IMDS 存取權。如需詳細資訊，請參閱限制存取指派給工作節點的執行個體設定檔。
- Kubernetes labels (Kubernetes 標籤) – (選用) 您可以選擇將 Kubernetes 標籤套用至受管節點群組中的節點。
- Kubernetes 污點 – （選用） 您可以選擇將 Kubernetes 污點套用至受管節點群組中的節點。Effect (效果) 功能表中的可用選項是 **`NoSchedule`**、 **`NoExecute`** 及 **`PreferNoSchedule`**。如需詳細資訊，請參閱the section called “防止在特定節點上排程 Pod”。
- Tags (標籤) – (選用) 您可以選擇標記 Amazon EKS 受管節點群組。這些標籤不會傳播到節點群組中的其他資源，例如 Auto Scaling 群組或執行個體。如需詳細資訊，請參閱the section called “標記您的 資源”。

7. 在 Set compute and scaling configuration (設定運算和擴展組態) 頁面上，相應地填寫參數，然後選擇 Next (下一步)。
   - AMI 類型 – 選取 AMI 類型。如果您要部署 Arm 執行個體，請務必在部署之前檢閱 Amazon EKS 最佳化 Arm Amazon Linux AMIs 中的考量事項。

     如果您在上一頁指定了啟動範本，並在啟動範本中指定了 AMI，則無法選取值。系統會顯示範本中的值。範本中指定的 AMI 必須符合指定 AMI 的要求。
   - Capacity type (容量類型) – 選取容量類型。如需選擇容量類型的詳細資訊，請參閱 the section called “受管節點群組容量類型”。您無法在相同的節點群組中混合不同的容量類型。如果要同時使用這兩種容量類型，請建立個別的節點群組，每個群組都有自己的容量和執行個體類型。如需佈建和擴展 GPUs 加速工作者節點的相關資訊，請參閱預留受管節點群組的 GPU。

- Instance types (執行個體類型)：依預設會指定一或多個執行個體類型。若要移除預設執行個體類型，請選取執行個體類型的右側的 X。選擇要在受管節點群組中使用的執行個體類型。如需詳細資訊，請參閱the section called “Amazon EC2 執行個體類型”。

  主控台會顯示一組常用的執行個體類型。如果您需要使用未顯示的執行個體類型建立受管節點群組，請使用 `eksctl`、 CLI、 AWS CloudFormation AWS 或 SDK 來建立節點群組。如果您在上一頁指定了啟動範本，則無法選取值，因為必須在啟動範本中指定執行個體類型。系統會顯示啟動範本中的值。如果為 Capacity type (容量類型) 選取了 Spot，則建議您指定多個執行個體類型以增強可用性。

- 磁碟大小 – 輸入要用於節點根磁碟區的磁碟大小 （ 以 GiB 為單位 ）。

  如果您在上一頁指定了啟動範本，則無法選取值，因為它必須在啟動範本中指定。

- Desired size (所需大小) – 指定受管節點群組目前在啟動時應該維護的節點數量。

  > **Note**
  >
  > Amazon EKS 不會自動擴展您的節點群組。不過，您可以設定 Kubernetes Cluster Autoscaler 為您執行此操作。如需詳細資訊，請參閱 Cluster Autoscaler on AWS。

- Minimum size (大小下限) – 指定受管節點群組可縮減至的節點數量下限。
- Maximum size (大小上限)：指定受管節點群組可擴增至的節點數量上限。
- Node group update configuration (節點群組更新組態) – (選用) 您可以選取要平行更新的節點數量或百分比。這些節點在更新期間將無法使用。對於 Maximum unavailable (無法使用的上限)，選取下列其中一個選項，然後指定 Value (值)：
  - Number (數量)：選取並指定可平行更新節點群組的節點數量。
  - Percentage (百分比)：選取並指定可平行更新節點群組的節點百分比。如果您的節點群組中有大量節點，這會很有用。
- 節點自動修復組態 – （ 選用 ） 如果您啟用啟用節點自動修復核取方塊，Amazon EKS 會在偵測到問題時自動取代節點。如需詳細資訊，請參閱the section called “節點運作狀態”。

8. 在 Specify networking (指定聯網) 頁面上，據此填寫參數，然後選擇 Next (下一步)。
   - Subnets (子網路)：選擇要將受管節點啟動至其中的子網路。

> **Important**
>
> 如果您正在多個由 Amazon EBS 磁碟區支援的可用區域中執行具狀態應用程式，並使用 Kubernetes Cluster Autoscaler，您應該設定多個節點群組，每個群組的範圍都限定為單一可用區域。另外，您應該啟用 `--balance-similar-node-groups` 功能。

> **Important**
>
> - 如果選擇公有子網路，而且您的叢集只啟用了公有 API 伺服器端點，則子網路的必須將 `MapPublicIPOnLaunch` 設定為 `true`，讓執行個體成功加入叢集。如果子網路是在 2020 年 3 月 26 日當天或之後使用 `eksctl`或 Amazon EKS vended AWS CloudFormation 範本建立，則此設定已設定為 `true`。如果在 2020 年 3 月 26 日之前使用 `eksctl`或 AWS CloudFormation 範本建立子網路，則需要手動變更設定。如需詳細資訊，請參閱修改子網的公有 IPv4 定址屬性。
> - 如果您使用啟動範本並指定多個網路介面，即使 `MapPublicIpOnLaunch` 設為 ，Amazon EC2 也不會自動指派公有`IPv4`地址`true`。若要讓節點在此案例中加入叢集，您必須啟用叢集的私有 API 伺服器端點，或在透過 NAT Gateway 等替代方法提供傳出網際網路存取的私有子網路中啟動節點。如需詳細資訊，請參閱《Amazon EC2 使用者指南》中的 Amazon EC2 執行個體 IP 定址。 Amazon EC2

- 設定 SSH 存取節點 (選用)。啟用 SSH 可讓您連接到執行個體，並在發生問題時收集診斷資訊。強烈建議您在建立節點群組時啟用遠端存取。您無法在建立節點群組之後啟用遠端存取。

  如果您選擇使用啟動範本，則不會顯示此選項。若要啟用節點的遠端存取，請在啟動範本中指定金鑰對，並確定您在啟動範本中指定之安全群組中的節點已開啟適當的連接埠。如需詳細資訊，請參閱the section called “使用自訂安全群組”。

  > **Note**
  >
  > 對於 Windows，此命令不會啟用 SSH。而是將 Amazon EC2 金鑰對與執行個體建立關聯，讓您可以 RDP 至該執行個體中。

- 對於 SSH key pair (SSH 金鑰對) (選用)，選擇要使用的 Amazon EC2 SSH 金鑰。如需 Linux 資訊，請參閱《Amazon EC2 使用者指南》中的 Amazon EC2 金鑰對和 Linux 執行個體。 Amazon EC2 如需 Windows 資訊，請參閱《Amazon EC2 使用者指南》中的 Amazon EC2 金鑰對和

Windows 執行個體。 Amazon EC2 如果您選擇使用啟動範本，則無法選取。當使用 Bottlerocket AMI 為節點群組提供 Amazon EC2 SSH 金鑰時，也會啟用管理容器。如需詳細資訊，請參閱 GitHub 上的管理容器。

- 對於 Allow SSH remote access from (允許 SSH 遠端存取來自)，如果要限制對特定執行個體的存取，請選取與這些執行個體相關聯的安全群組。如果您未選取特定安全群組，則可從網際網路上的任何位置存取 SSH (`0.0.0.0/0`)。

9. 在 Review and create (檢閱並建立) 頁面上，檢閱您的受管節點群組組態，然後選擇 Create (建立)。

   如果節點無法加入叢集，請參閱疑難排解章節the section called “節點無法加入叢集”中的 。

10查看節點的狀態，並等待他們到達 `Ready` 狀態。

```
kubectl get nodes --watch
```

11. (僅限 GPU 節點) 如果您選擇 GPU 執行個體類型和 Amazon EKS 最佳化加速 AMI，則必須將 Kubernetes 的 NVIDIA 裝置外掛程式套用為叢集上的 DaemonSet。在執行下列命令之前，將 *vX.X.X* 取代為所需的 NVIDIA/k8s-device-plugin 版本。

```
kubectl apply -f https://raw.githubusercontent.com/NVIDIA/k8s-device-plugin/vX.X.X/
deployments/static/nvidia-device-plugin.yml
```

安裝 Kubernetes 附加元件

既然您有一個具有節點的正常運作 Amazon EKS 叢集，您就可以開始安裝 Kubernetes 附加元件，並將應用程式部署到您的叢集。以下文件主題可協助您擴展叢集的功能。

- 建立叢集的 IAM 主體是唯一可以使用 `kubectl`或 呼叫 Kubernetes API 伺服器的主體 AWS Management Console。如果要其他 IAM 主體能夠存取叢集，則需新增這些主體。如需詳細資訊，請參閱the section called “Kubernetes API 存取”及the section called “所需的許可”。
- 如果符合下列條件，我們建議封鎖 Pod 存取 IMDS：
  - 您計劃將 IAM 角色指派給所有 Kubernetes 服務帳戶，讓 Pod 僅擁有所需的最低許可。
  - 叢集中沒有任何 Pod 因其他原因需要存取 Amazon EC2 執行個體中繼資料服務 (IMDS)，例如擷取目前 AWS 區域。

  如需詳細資訊，請參閱限制存取指派給工作節點的執行個體設定檔。
- 設定 Kubernetes Cluster Autoscaler 以自動調整節點群組中的節點數量。

- 將範例應用程式部署至叢集。
- 使用管理叢集的重要工具來組織和監控叢集資源。

更新叢集的受管節點群組

當您啟動受管節點群組更新時，Amazon EKS 會自動為您更新節點，完成了解節點更新的每個階段中列出的步驟。如果您使用的是 Amazon EKS 最佳化 AMI，Amazon EKS 會自動將最新的安全修補程式和作業系統更新套用至節點，做為最新的 AMI 發行版本的一部分。

有幾種情況可讓您更新 Amazon EKS 受管節點群組的版本或組態：

- 您已更新 Amazon EKS 叢集的 Kubernetes 版本，並且想要更新您的節點，以使用相同的 Kubernetes 版本。
- 新的 AMI 發行版本可供受管節點群組使用。如需 AMI 版本的詳細資訊，請參閱下列各節：
  - the section called “取得版本資訊”
  - the section called “Bottlerocket”
  - the section called “取得版本資訊”
- 您想要調整受管節點群組中執行個體的下限、上限或所需計數。
- 您想要從受管節點群組中的執行個體新增或移除 Kubernetes 標籤。
- 您想要從受管節點群組新增或移除 AWS 標籤。
- 您需要部署具有組態變更的新版啟動範本，例如更新的自訂 AMI。
- 您已部署 版本 `1.9.0`或更新版本的 Amazon VPC CNI 附加元件、啟用用於字首委派的附加元件，並希望節點群組中的新 AWS Nitro System 執行個體支援大幅增加的 Pod 數量。如需詳細資訊，請參閱the section called “增加 IP 地址”。
- 您已為 Windows 節點啟用 IP 字首委派，並希望節點群組中的新 AWS Nitro System 執行個體支援大幅增加的 Pod 數量。如需詳細資訊，請參閱the section called “增加 IP 地址”。

如果受管節點群組的 Kubernetes 版本有較新的 AMI 發行版本，您可以更新節點群組的版本，以使用較新的 AMI 版本。同樣地，如果您的叢集執行比節點群組更新的 Kubernetes 版本，您可以更新節點群組以使用最新的 AMI 發行版本，以符合叢集的 Kubernetes 版本。

當受管節點群組中的節點因擴展操作或更新而終止時，會先耗盡該節點中的 Pod。如需詳細資訊，請參閱the section called “更新行為詳細資訊”。

更新節點群組版本

您可以使用下列其中一項來更新節點群組版本：

- the section called “eksctl”
- the section called “AWS Management Console”

您更新至 的版本不能大於控制平面的版本。

eksctl

使用 更新受管節點群組 **eksctl**

使用下列命令，將受管節點群組更新為目前部署在節點上相同 Kubernetes 版本的最新 AMI 版本。將每個*###*取代為您自己的值。

```
eksctl upgrade nodegroup \
  --name=node-group-name \
  --cluster=my-cluster \
  --region=region-code
```

> **Note**
>
> 如果您要將以啟動範本部署的節點群組升級至新的啟動範本版本，請將 `--launch-template-version` *`version-number`* 新增至上述命令。啟動範本必須符合使用啟動範本自訂受管節點中所述的要求。如果啟動範本包含自訂 AMI，則 AMI 必須符合指定 AMI 的要求。當您將節點群組升級至較新版本的啟動範本時，每個節點都會回收，以符合指定的啟動範本版本的新組態。

在沒有啟動範本的情況下，您無法將部署的節點群組直接升級至新的啟動範本版本。相反地，您必須使用啟動範本部署新的節點群組，才能將節點群組更新為新的啟動範本版本。

您可以將節點群組升級至與控制平面的 Kubernetes 版本相同的版本。例如，如果您有執行 Kubernetes 的叢集`1.33`，您可以使用下列命令將目前執行 Kubernetes `1.33`的節點升級至 `1.32` 版本。

```
eksctl upgrade nodegroup \
```

```
  --name=node-group-name \
  --cluster=my-cluster \
  --region=region-code \
  --kubernetes-version=1.33
```

AWS Management Console

使用 更新受管節點群組 AWS Management Console

1. 開啟 Amazon EKS 主控台。
2. 選擇包含要更新之節點群組的叢集。
3. 如果至少有一個節點群組有可用的更新，則頁面頂部會出現一個方塊，通知您可用的更新。如果您選取運算索引標籤，您會在節點群組資料表中的 AMI 發行版本欄中，看到具有可用更新的節點群組現在更新。若要更新節點群組，請選擇 Update now (立即更新)。

   您不會看到使用自訂 AMI 部署之節點群組的通知。如果節點是使用自訂 AMI 進行部署，則請完成以下步驟來部署新的已更新自訂 AMI。

   a. 建立 AMI 的新版本。

   b. 使用新的 AMI ID 建立新的啟動範本版本。

   c. 將節點升級至啟動範本的新版本。
4. 在 Update node group version (更新節點群組版本) 對話方塊中，啟用或停用以下選項：
   - Update node group version (更新節點群組版本)：如果您部署了自訂 AMI，或經 Amazon EKS 優化的 AMI 目前是叢集的最新版本，則無法使用此選項。
   - Change launch template version (變更啟動範本版本)：如果節點群組是在沒有自訂啟動範本的情況下進行部署，則無法使用此選項。您只能更新已使用自訂啟動範本部署之節點群組的啟動範本版本。選取節點群組要更新至的 Launch template version (啟動範本版本)。如果節點群組設定了自訂 AMI，則您選取的版本也必須指定 AMI。當您升級至較新版本的啟動範本時，每個節點都會回收，以符合指定之啟動範本版本的新組態。
5. 對於 Update strategy (更新策略)，選取下列其中一個選項：
   - 滾動更新 – 此選項遵守叢集的 Pod 中斷預算。如果 Pod 中斷預算問題導致 Amazon EKS 無法正常耗盡在此節點群組上執行的 Pod，則更新會失敗。
   - 強制更新 – 此選項不遵守 Pod 中斷預算。透過強制節點重新啟動，無論 Pod 中斷預算問題為何，都會進行更新。
6. 選擇 Update (更新)。

編輯節點群組組態

您可以修改受管節點群組中的部分組態。

1. 開啟 Amazon EKS 主控台。
2. 選擇包含要編輯之節點群組的叢集。
3. 選取 Compute (運算) 標籤。
4. 選取要編輯的節點群組，然後選擇 Edit (編輯)。
5. (選用) 在 Edit node group (編輯節點群組) 頁面上，執行以下作業：
   a. 編輯 Node group scaling configuration (節點群組擴展組態)。
      - Desired size (所需大小) – 指定受管節點群組目前應該維護的工作者節點數量。
      - Minimum size (大小下限) – 指定受管節點群組可縮減至的節點數量下限。
      - Maximum size (大小上限)：指定受管節點群組可擴增至的節點數量上限。如需節點群組中支援的節點數量上限，請參閱 the section called “Service Quotas”。
   b. （選用）將 Kubernetes 標籤新增至節點群組中的節點。這裡顯示的標籤只是您使用 Amazon EKS 所套用的標籤。節點上可能存在此處未顯示的其他標籤。
   c. （選用）新增或移除節點群組中節點的 Kubernetes 污點。已新增的污點可能會有 **`NoSchedule`**、**`NoExecute`** 或 **`PreferNoSchedule`** 效果。如需詳細資訊，請參閱the section called “防止在特定節點上排程 Pod”。
   d. (選用) 將 Tags (標籤) 新增至節點群組或從中移除標籤。這些標籤僅適用於 Amazon EKS 節點群組。它們不會傳播到其他資源，例如節點群組中的子網路或 Amazon EC2 執行個體。
   e. (選用) 編輯 Node Group update configuration (節點群組更新組態)。選取任何一個 Number (數字) 或 Percentage (百分比)。
      - Number (數量)：選取並指定可平行更新節點群組的節點數量。這些節點在更新期間將無法使用。
      - Percentage (百分比)：選取並指定可平行更新節點群組的節點百分比。這些節點在更新期間將無法使用。如果您的節點群組中有許多節點，這會很有用。
   f. 完成編輯後，請選擇儲存變更。

> **Important**
>
> 更新節點群組組態時，修改 `NodegroupScalingConfig` 不符合 Pod 中斷預算 (PDBs)。與更新節點群組程序（在升級階段消耗節點並遵循 PDBs) 不同，更新擴展組態會導致節點透過 Auto Scaling 群組 (ASG) 縮減呼叫立即終止。無論您要縮減的目標大小為何，都不會考慮

PDBs。這表示當您減少 Amazon EKS 受管節點群組`desiredSize`的 時，會在節點終止後立即移出 Pod，而不會接受任何 PDBs。

了解節點更新的每個階段

Amazon EKS 受管工作節點升級策略有四個不同的階段，如下節所述。

設定階段

設定階段具有下列步驟：

1. 它會為與您的節點群組相關聯的 Auto Scaling 群組建立新的 Amazon EC2 啟動範本版本。新的啟動範本版本使用目標 AMI 或自訂啟動範本版本進行更新。
2. 它會更新 Auto Scaling 群組，以使用最新的啟動範本版本。
3. 它使用節點群組的 `updateConfig` 屬性決定要平行升級的節點數量上限。不可用節點上限配額為 100。預設值為一個節點。如需詳細資訊，請參閱《Amazon EKS API 參考》中的 updateConfig 屬性。

擴充規模階段

升級受管節點群組中的節點時，已升級的節點會在與正在升級的節點相同的可用區域中啟動。為了保證此置放，我們使用 Amazon EC2 的可用區域重新平衡。如需詳細資訊，請參閱《Amazon EC2 Auto Scaling 使用者指南》中的可用區域容量重新平衡。為了符合此需求，我們最多可能會在您的受管節點群組中的每個可用區域啟動兩個執行個體。

擴充規模階段具有下列步驟：

1. 它會將 Auto Scaling 群組的大小上限和所需大小增加下列其中之一的較大者：
   - Auto Scaling 群組部署的可用區域數量最多兩倍。
   - 升級無法使用的上限。

     例如，如果節點群組有五個可用區域，而 `maxUnavailable` 為一個，則升級程序最多能啟動 10 個節點。不過，當 `maxUnavailable`為 20 （或任何高於 10 的項目） 時，程序會啟動 20 個新節點。
2. 擴展 Auto Scaling 群組後，會檢查使用最新配置的節點是否存在於節點群組之中。只有在符合下列條件時，此步驟才會成功：

- 在節點所在的每個可用區域中，至少會啟動一個新節點。
- 每個新節點都應該處於 Ready 狀態。
- 新節點應該有 Amazon EKS 套用標籤。

  以下是一般節點群組中工作節點上的 Amazon EKS 套用標籤：

  - eks.amazonaws.com/nodegroup-image=$amiName
  - eks.amazonaws.com/nodegroup=$nodeGroupName

  以下是自訂啟動範本或 AMI 節點群組中工作節點上的 Amazon EKS 套用標籤：

  - eks.amazonaws.com/nodegroup-image=$amiName
  - eks.amazonaws.com/nodegroup=$nodeGroupName
  - eks.amazonaws.com/sourceLaunchTemplateId=$launchTemplateId
  - eks.amazonaws.com/sourceLaunchTemplateVersion=$launchTemplateVersion

3. 它將節點標記為無法排程，以避免排程新的 Pod。它還會使用 標記節點node.kubernetes.io/exclude-from-external-load-balancers=true，以在終止節點之前從負載平衡器移除舊節點。

以下是在這個階段中會導致 NodeCreationFailure 錯誤的已知原因：

可用區域中的容量不足

可用區域可能沒有所請求執行個體類型的容量。建議在建立受管節點群組時設定多個執行個體類型。

您帳戶中的 EC2 執行個體限制

您可能需要增加帳戶可以使用 Service Quotas 同時執行的 Amazon EC2 執行個體的數量。如需詳細資訊，請參閱《Amazon Elastic Compute Cloud Linux 執行個體使用者指南》中的 EC2 Service Quotas。

自訂使用者資料

自訂使用者資料有時會中斷引導程序。這種情況可能會導致 kubelet 未在節點上啟動或節點上未取得預期的 Amazon EKS 標籤。如需詳細資訊，請參閱the section called “指定 AMI”。

讓節點運作狀態不佳或未就緒的任何變更

節點磁碟壓力、記憶體壓力和類似情況，可能導致節點無法進入 Ready 狀態。

每個節點在 15 分鐘內開機次數最多

如果任何節點需要超過 15 分鐘才能啟動並加入叢集，則會導致升級逾時。這是從需要新節點到加入叢集時，引導新節點的總執行時間。升級受管節點群組時，時間計數器會在 Auto Scaling 群組大小增加時立即啟動。

升級階段

升級階段的行為有兩種不同方式，取決於更新策略。更新策略有兩種：預設和最小。

我們建議在大多數情況下使用預設策略。它會在終止舊節點之前建立新節點，以便在升級階段維持可用容量。最基本的策略在您受限於資源或成本的情況下很有用，例如使用 GPUs 等硬體加速器。它會在建立新的節點之前終止舊節點，因此總容量永遠不會超過您設定的數量。

預設更新策略包含下列步驟：

1. 它會增加 Auto Scaling 群組中的節點數量（所需計數），導致節點群組建立其他節點。
2. 它隨機選取需要升級的節點，最大不超過為節點群組設定的無法使用上限。
3. 它會從節點耗盡 Pod。如果 Pod 未在 15 分鐘內離開節點，而且沒有強制旗標，則升級階段會失敗並顯示`PodEvictionFailure`錯誤。在此案例中，您可以將強制旗標套用至刪除 Pod 的`update-nodegroup-version`請求。
4. 它會在每個 Pod 被移出後封鎖節點，並等待 60 秒。這樣做是為了讓服務控制器不會傳送任何新請求至此節點，並從其作用中節點清單中移除此節點。
5. 它將終止請求傳送之包圍隔離節點的 Auto Scaling 群組。
6. 它重複步驟先前的升級步驟，直到節點群組中沒有使用舊版啟動範本部署的節點為止。

最低更新策略包含下列步驟：

1. 它會在開頭封鎖節點群組的所有節點，以便服務控制器不會將任何新請求傳送至這些節點。
2. 它隨機選取需要升級的節點，最大不超過為節點群組設定的無法使用上限。
3. 它會從選取的節點耗盡 Pod。如果 Pod 未在 15 分鐘內離開節點，而且沒有強制旗標，則升級階段會失敗並顯示`PodEvictionFailure`錯誤。在此案例中，您可以將強制旗標套用至刪除 Pod 的`update-nodegroup-version`請求。
4. 在每個 Pod 被移出並等待 60 秒後，它會傳送終止請求給所選節點的 Auto Scaling 群組。Auto Scaling 群組會建立新的節點（與所選節點的數量相同），以取代缺少的容量。
5. 它重複步驟先前的升級步驟，直到節點群組中沒有使用舊版啟動範本部署的節點為止。

PodEvictionFailure 升級階段期間的錯誤

以下是在這個階段中會導致 PodEvictionFailure 錯誤的已知原因：

積極 PDB

積極 PDB 是在 Pod 上定義，或者有多個 PDBs 指向相同的 Pod。

部署可容忍所有污點

一旦每個 Pod 被移出，節點預期會是空的，因為節點在先前的步驟中被污點化。不過，如果部署可容忍每個污點，則節點可能不是空的，導致 Pod 移出失敗。

縮減規模階段

縮減規模階段會將 Auto Scaling 群組的大小上限和所需大小遞減一，以便回到更新開始之前的值。

如果升級工作流程判斷 Cluster Autoscaler 在工作流程的縮減規模階段期間擴充節點群組，它會立即結束，而不會將節點群組恢復至原始大小。

使用啟動範本自訂受管節點

若要取得最高層級的自訂，您可以使用自己的啟動範本來部署受管節點。使用啟動範本可提供下列功能：

- 在部署節點時提供引導引數，例如額外的 kubelet 引數。
- 從與指派給節點的 IP 地址不同的 CIDR 區塊將 IP 地址指派給 Pod。
- 將自己的自訂 AMI 部署至節點。
- 將您自己的自訂 CNI 部署至節點。

如果您在第一次建立受管節點群組期間提供自己的啟動範本，稍後也會得到更出色的靈活性。只要使用自己的啟動範本部署受管節點群組，您就可以使用不同版本的相同啟動範本來反覆更新。將節點群組更新為不同版本的啟動範本時，群組中的所有節點都會回收，以符合指定啟動範本版本的新組態。

一律使用 Amazon EC2 Auto Scaling 群組所使用的啟動範本部署受管節點群組。當您不提供啟動範本時，Amazon EKS API 會在您的帳戶中自動建立預設值的啟動範本。不過，我們不建議您修改自動產生的啟動範本。此外，無法使用自訂啟動範本的現有節點群組無法直接更新。相反，您必須建立具有自訂啟動範本的新節點群組，才能執行此動作。

啟動範本組態基礎知識

您可以使用 、 AWS Management Console AWS CLI 或 AWS SDK 建立 Amazon EC2 Auto Scaling 啟動範本。如需詳細資訊，請參閱《Amazon EC2 Auto Scaling 使用者指南》中的建立 Auto Scaling 群組的啟動範本。啟動範本中的某些設定與用於受管理節點組態的設定類似。使用啟動範本部署或更新節點群組時，必須在節點群組組態或啟動範本其中一個位置中指定某些設定。請勿在這兩個位置指定設定。如果設定位於不應該的位置，則建立或更新節點群組等操作會失敗。

下表會列出啟動範本中禁止的設定。該表也會列出受管節點群組組態中需要的類似設定 (如果有)。列出的設定是顯示在主控台中的設定。它們在 CLI 和 SDK AWS 中可能有類似但不同的名稱。

啟動範本：禁止	Amazon EKS 節點群組組態
Network interfaces (網路介面) (Add network interface (新增網路介面)) 下的 Subnet (子網)	Specify networking (指定聯網) 頁面上 Node Group network configuration (節點群組網路組態) 下的 Subnets (子網)
Advanced details (進階詳細資訊) 下的 IAM instance profile (IAM 執行個體描述檔)	Configure Node Group (設定節點群組) 頁面上 Node Group configuration (節點群組組態) 下的 Node IAM Role (節點 IAM 角色)
Advanced details (進階詳細資訊) 下的 Shutdown behavior (關機行為) 和 Stop - Hibernate behavior (停用 – 休眠行為)。保留預設 請勿在啟動範本的啟動範本設定中包含這兩個設定。	無同等。Amazon EKS 必須控制執行個體生命週期，而非 Auto Scaling 群組。

下表會列出受管節點群組組態中禁止的設定。該表也會列出類似的設定 (如果有)，這些設定在啟動範本中是必要設定。列出的設定是顯示在主控台中的設定。它們在 CLI 和 SDK AWS 中可能會有類似的名稱。

Amazon EKS 節點群組組態設定：禁止	啟動範本
(僅當您在啟動模板中指定了自訂 AMI 時) 在 Set compute and scaling configuration (設定運算和擴展組態) 頁面上 Node Group compute configuration (節點群組運算組態) 下的 AMI type	Launch template contents (啟動範本內容) 下的 Application and OS Images (Amazon Machine Image) (應用程式和作業系統映像 (Amazon

Amazon EKS 節點群組組態設定：禁止	啟動範本

(AMI 類型) – 主控台顯示 Specified in launch template (在啟動範本中指定) 以及指定的 AMI ID。
 	
如果未在啟動範本中指定應用程式和作業系統映像 (Amazon Machine Image)，您可以在節點群組組態中選取 AMI。
	Machine Image))：如果您有下列任一要求，則必須指定 ID：
	
	• 使用自訂 AMI。如果您指定的 AMI 不符合指定 AMI 中列出的要求，節點群組部署將會失敗。
	• 想要提供使用者資料將引數提供給 `bootstrap.sh` 檔案，其中包含在 Amazon EKS 最佳化 AMI。您可以讓執行個體將大量 IP 地址指派給 Pod、從與執行個體不同的 CIDR 區塊將 IP 地址指派給 Pod，或在沒有傳出網際網路存取的情況下部署私有叢集。如需詳細資訊，請參閱下列主題：
	  • 使用字首將更多 IP 地址指派給 Amazon EKS 節點
	  • 使用自訂聯網在替代子網路中部署 Pod
	  • 部署網際網路存取受限的私有叢集
	  • 指定 AMI

Set compute and scaling configuration (設定運算和擴展組態) 頁面上 Node Group compute configuration (節點群組運算組態) 下的 Disk size (磁碟大小)：主控台顯示 Specified in launch template (在啟動範本中指定)。	Storage (Volumes) (儲存 (磁碟區)) (Add new volume (新增磁碟區)) 下的 Size (大小)。您必須在啟動範本中指定此選項。
Specify Networking (指定聯網) 頁面上 Node Group configuration (節點群組組態) 下的 SSH key pair (SSH 金鑰對) – 主控台會顯示在啟動範本中指定的金鑰，或顯示 Not specified in launch template (未在啟動範本中指定)。	Key pair (login) (金鑰對 (登入)) 下的 Key pair name (金鑰對名稱)。

Amazon EKS 節點群組組態設定：禁止	啟動範本
您無法指定使用啟動範本時允許遠端存取的來源安全群組。	針對執行個體的 Network settings (網路設定) 下的 Security groups (安全群組) 或 Network interfaces (網路介面) 下的 Security groups (安全群組) (Add network interface (新增網路介面))，但不能兩者同時選擇。如需詳細資訊，請參閱the section called “使用自訂安全群組”。

Note

- 如果使用啟動範本部署節點群組，請在啟動範本的 Launch template contents (啟動範本內容) 中指定零或一個 Instance type (執行個體類型)。您也可以在主控台上的 Set compute and scaling configuration (設定運算和擴展組態) 頁面中，為 Instance types (執行個體類型) 指定 0 到 20 個執行個體類型。您還可以使用其他使用 Amazon EKS API 的工具來執行這項操作。如果您在啟動範本中指定執行個體類型，並使用該啟動範本來部署節點群組，則無法在主控台中或使用其他使用 Amazon EKS API 的工具指定任何執行個體類型。如果您未在啟動範本、 主控台或使用其他使用 Amazon EKS API 的工具中指定執行個體類型，則會使用`t3.medium`執行個體類型。如果您的節點群組正在使用 Spot 容量類型，則建議使用主控台指定多個執行個體類型。如需詳細資訊，請參閱the section called “受管節點群組容量類型”。
- 如果您部署到節點群組的任何容器使用執行個體中繼資料服務版本 2，請確定在啟動範本中將 Metadata response hop limit (中繼資料回應躍點限制) 設定為 2。如需詳細資訊，請參閱《Amazon EC2 使用者指南》中的執行個體中繼資料與使用者資料。
- 啟動範本不支援允許彈性執行個體類型選擇`InstanceRequirements`的功能。

標記 Amazon EC2 執行個體

您可以使用啟動範本的 `TagSpecification` 參數，以指定要將哪些標籤套用至節點群組中的 Amazon EC2 執行個體。IAM 實體呼叫 `CreateNodegroup` 或 `UpdateNodegroupVersion` API 必須擁有 `ec2:RunInstances` 和 `ec2:CreateTags` 許可，而且標籤必須新增至啟動範本。

使用自訂安全群組

您可以使用啟動範本來指定自訂 Amazon EC2 安全群組以套用至節點群組中的執行個體。這可以是在執行個體層級安全群組參數中，也可以是網路介面組態參數的一部分。不過，您無法建立同時指定執行個體層級和網路介面安全群組的啟動範本。請考慮下列適用於搭配受管節點群組使用之自訂安全群組的條件：

- 使用 時 AWS Management Console，Amazon EKS 僅允許具有單一網路介面規格的啟動範本。
- 預設情況下，Amazon EKS 將叢集安全群組套用至節點群組中的執行個體，以促進節點與控制平面之間的通訊。如果您使用前述任一選項在啟動範本中指定自訂安全群組，Amazon EKS 不會新增叢集安全群組。因此，您必須確保安全群組的傳入和傳出規則可啟用與叢集端點的通訊。如果您的安全群組規則不正確，工作者節點無法加入叢集。如需安全群組規則的詳細資訊，請參閱 the section called “安全群組要求”。
- 如需 SSH 存取節點群組中的執行個體，請包含允許該存取的安全群組。

Amazon EC2 使用者資料

啟動範本包含自訂使用者資料的區段。您可以在此區段中指定節點群組的組態設定，無需手動建立個別自訂 AMI。如需有關 Bottlerocket 可用設定的詳細資訊，請參閱 GitHub 上的使用使用者資料。

您可以在啟動執行個體時使用 `cloud-init` 提供啟動範本中的 Amazon EC2 使用者資料。如需詳細資訊，請參閱 cloud-init 文件。您的使用者資料可用來執行一般組態操作。這包含下列操作：

- 包括使用者或群組
- 安裝套件

與受管節點群組搭配使用的啟動範本中的 Amazon EC2 使用者資料，必須為 Amazon Linux AMI 的 MIME 分段封存格式和 Bottlerocket AMI 的 TOML 格式。這是因為您的使用者資料與節點加入叢集所需的 Amazon EKS 使用者資料合併。請勿在您的使用者資料中指定任何啟動或修改 的命令`kubelet`。這會作為 Amazon EKS 合併使用者資料的一部分執行。某些 `kubelet` 參數 (例如在節點上設定標籤) 可以透過受管節點群組 API 直接設定。

> **Note**
>
> 如果有關進階 `kubelet` 自訂 (包括手動啟動或傳入自訂組態參數) 的詳細資訊，請參閱 the section called “指定 AMI”。如果在啟動範本中指定自訂 AMI ID，Amazon EKS 不會合併使用者資料。

下列詳細資訊提供有關使用者資料區段的詳細資訊。

Amazon Linux 2 使用者資料

您可以將多個使用者資料區塊組合在一起成為單一 MIME 分段檔案。例如，您可以將設定 Docker 常駐程式的雲端 Boothook 與安裝自訂套件的使用者資料 Shell 指令碼合併。MIME 分段檔案包含下列元件：

- 內容類型和部分邊界宣告：`Content-Type: multipart/mixed; boundary="==MYBOUNDARY=="`
- MIME 版本宣告：`MIME-Version: 1.0`
- 包含以下元件的一或多個使用者資料區塊：
  - 開啟邊界，表示使用者資料區塊的開始 – `--==MYBOUNDARY==`
  - 區塊的內容類型宣告：`Content-Type: text/cloud-config; charset="us-ascii"`。如需內容類型的詳細資訊，請參閱 cloud-init 文件。
  - 使用者資料的內容 (例如，Shell 命令或 `cloud-init` 指令的清單)。
  - 結束邊界，表示 MIME 分段檔案的結束：`--==MYBOUNDARY==--`

以下是您可以用來建立自己的 MIME 分段檔案的範例。

```
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary="==MYBOUNDARY=="

--==MYBOUNDARY==
Content-Type: text/x-shellscript; charset="us-ascii"

#!/bin/bash
echo "Running custom user data script"

--==MYBOUNDARY==--
```

Amazon Linux 2023 使用者資料

Amazon Linux 2023 (AL2023) 推出`nodeadm`使用 YAML 組態結構描述的新節點初始化程序。如果您使用自我管理節點群組或具有啟動範本的 AMI，您現在將需要在建立新的節點群組時明確提供額外的叢集中繼資料。最低必要參數的範例如下所示，其中`cidr`現在需要 `apiServerEndpoint`、`certificateAuthority`和服務：

```
---
apiVersion: node.eks.aws/v1alpha1
```

```
kind: NodeConfig
spec:
  cluster:
    name: my-cluster
    apiServerEndpoint: https://example.com
    certificateAuthority: Y2VydGlmaWNhdGVBdXRob3JpdHk=
    cidr: 10.100.0.0/16
```

您通常會在使用者資料中依原狀設定此組態，或內嵌在 MIME 分段文件中：

```
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary="BOUNDARY"

--BOUNDARY
Content-Type: application/node.eks.aws

---
apiVersion: node.eks.aws/v1alpha1
kind: NodeConfig spec: [...]

--BOUNDARY--
```

在 AL2 中，從 Amazon EKS `DescribeCluster` API 呼叫中發現了這些參數的中繼資料。透過 AL2023，此行為已變更，因為在大型節點擴展期間，額外的 API 呼叫風險調節。如果您使用沒有啟動範本的受管節點群組，或是使用 Karpenter，則此變更不會影響您。如需 `certificateAuthority`和服務的詳細資訊`cidr`，請參閱《Amazon EKS API 參考[DescribeCluster](#)》中的 。

以下是 AL2023 使用者資料的完整範例，結合用於自訂節點的 shell 指令碼 （例如安裝套件或預先快取容器映像） 與所需`nodeadm`組態。此範例顯示常見的自訂，包括：* 安裝其他系統套件 * 預先快取容器映像以改善 Pod 啟動時間 * 設定 HTTP 代理組態 * 設定節點標籤的`kubelet`旗標

```
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary="BOUNDARY"

--BOUNDARY
Content-Type: text/x-shellscript; charset="us-ascii"

#!/bin/bash
set -o errexit
set -o pipefail
set -o nounset
```

```
# Install additional packages
yum install -y htop jq iptables-services

# Pre-cache commonly used container images
nohup docker pull public.ecr.aws/eks-distro/kubernetes/pause:3.2 &

# Configure HTTP proxy if needed
cat > /etc/profile.d/http-proxy.sh << 'EOF'
export HTTP_PROXY="http://proxy.example.com:3128"
export HTTPS_PROXY="http://proxy.example.com:3128"
export NO_PROXY="localhost,127.0.0.1,169.254.169.254,.internal"
EOF

--BOUNDARY
Content-Type: application/node.eks.aws

apiVersion: node.eks.aws/v1alpha1
kind: NodeConfig
spec:
  cluster:
    name: my-cluster
    apiServerEndpoint: https://example.com
    certificateAuthority: Y2VydGlmaWNhdGVBdXRob3JpdHk=
    cidr: 10.100.0.0/16
  kubelet:
    config:
      clusterDNS:
      - 10.100.0.10
    flags:
    - --node-labels=app=my-app,environment=production

--BOUNDARY--
```

Bottlerocket 使用者資料

Bottlerocket 會以 TOML 格式建構使用者資料。您可以提供要與 Amazon EKS 提供的使用者資料合併的使用者資料。例如，您可以提供額外的 `kubelet` 設定。

```
[settings.kubernetes.system-reserved]
cpu = "10m"
memory = "100Mi"
ephemeral-storage= "1Gi"
```

如需有關受支援設定的詳細資訊，請參閱 Bottlerocket 文件。您可以在使用者資料中設定節點標籤和污點。不過，建議您改為在節點群組內設定。在執行這項操作時，Amazon EKS 會套用這些組態。

合併使用者資料時，不會保留格式，但內容會保持不變。您在使用者資料中提供的組態會覆寫 Amazon EKS 所設定的任何設定。因此，如果您設定 `settings.kubernetes.max-pods`或 `settings.kubernetes.cluster-dns-ip`，使用者資料中的這些值會套用至節點。

Amazon EKS 不支援所有有效的 TOML。以下是已知不支援格式的清單：

- 引號鍵內的引號：`'quoted "value"' = "value"`
- 值中的溢出引號：`str = "I'm a string. \"You can quote me\""`
- 混合浮點數和正整數：`numbers = [ 0.1, 0.2, 0.5, 1, 2, 5 ]`
- 陣列中的混合類型：`contributors = ["foo@example.com", { name = "Baz", email = "baz@example.com" }]`
- 帶引號鍵的括號標題：`[foo."bar.baz"]`

Windows 使用者資料

Windows 使用者資料使用 PowerShell 命令。建立受管節點群組時，您的自訂使用者資料會與 Amazon EKS 受管使用者資料結合。您的 PowerShell 命令會先出現，接著是受管使用者資料命令，全部都在一個`<powershell></powershell>`標籤內。

> **Important**
>
> 建立 Windows 節點群組時，Amazon EKS 會更新 `aws-authConfigMap`，以允許 Linux 型節點加入叢集。服務不會自動設定 Windows AMIs許可。如果您使用的是 Windows 節點，則需要透過存取項目 API 或`aws-authConfigMap`直接更新 來管理存取。如需詳細資訊，請參閱the section called “啟用 Windows 支援”。

> **Note**
>
> 在啟動範本中未指定 AMI ID 時，請勿使用使用者資料中的 Windows Amazon EKS 引導指令碼來設定 Amazon EKS。

範例使用者資料如下。

```
<powershell>
```

```
Write-Host "Running custom user data script"
</powershell>
```

指定 AMI

如果有下列其中一項要求，請在啟動範本 ImageId 欄位中指定 AMI ID。選取您對其他資訊的要求。

提供使用者資料，將引數傳遞至 Amazon EKS 最佳化 Linux/Bottlerocket AMI 隨附的**bootstrap.sh**檔案

引導是一個術語，用於描述新增在執行個體啟動時可以執行的命令。例如，引導允許使用額外的 kubelet 引數。您可以使用 eksctl 將引數傳遞給 bootstrap.sh 而不指定啟動範本。您也可以在啟動範本的使用者資料區段中指定資訊來執行此動作。

eksctl，無需指定啟動範本

使用下列內容建立名為 *my-nodegroup.yaml* 的檔案。將每個*###*取代為您自己的值。--apiserver-endpoint、--b64-cluster-ca，和 --dns-cluster-ip 引數為選用。但是，定義引述會允許 bootstrap.sh 指令碼避免進行 describeCluster 呼叫。這對於您經常向內擴展和向外擴展節點的私有叢集設定或叢集非常有用。如需bootstrap.sh指令碼的詳細資訊，請參閱 GitHub 上的 bootstrap.sh 檔案。

- 唯一的必要引數是叢集名稱 (*my-cluster*)。
- 若要擷取 的最佳化 AMI IDami-*1234567890abcdef0* ，請參閱下列章節：
  - 擷取建議的 Amazon Linux AMI IDs
  - 擷取建議的 Bottlerocket AMI IDs
  - 擷取建議的 Microsoft Windows AMI IDs
- 若要擷取叢集的 *certificate-authority*，請執行下列命令。

```
aws eks describe-cluster --query "cluster.certificateAuthority.data" --output text
 --name my-cluster --region region-code
```

- 若要擷取叢集的 *api-server-endpoint*，請執行下列命令。

```
aws eks describe-cluster --query "cluster.endpoint" --output text --name my-
cluster --region region-code
```

- --dns-cluster-ip 的值是結尾處為 .10 的服務 CIDR。若要擷取叢集的 *service-cidr*，請執行下列命令。例如，如果 的傳回值為 ipv4 10.100.0.0/16，則您的值為 *10.100.0.10*。

```
aws eks describe-cluster --query "cluster.kubernetesNetworkConfig.serviceIpv4Cidr"
 --output text --name my-cluster --region region-code
```

- 此範例使用其中包含 Amazon EKS 最佳化 AMI 的 bootstrap.sh 指令碼提供 kubelet 引數來設定自訂 max-pods 值。節點群組名稱不能超過 63 個字元。它必須以字母或數字開頭，但剩餘字元也可以包含連字符和底線。如需選取 *my-max-pods-value* 的說明，請參閱 the section called “Amazon EKS 為每種 Amazon EC2 執行個體類型建議 Pod 數量上限”。

```
---
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: my-cluster
  region: region-code

managedNodeGroups:
  - name: my-nodegroup
    ami: ami-1234567890abcdef0
    instanceType: m5.large
    privateNetworking: true
    disableIMDSv1: true
    labels: { x86-al2-specified-mng }
    overrideBootstrapCommand: |
      #!/bin/bash
      /etc/eks/bootstrap.sh my-cluster \
        --b64-cluster-ca certificate-authority \
        --apiserver-endpoint api-server-endpoint \
        --dns-cluster-ip service-cidr.10 \
        --kubelet-extra-args '--max-pods=my-max-pods-value' \
        --use-max-pods false
```

針對每一個可用的 eksctl config 檔案選項，請參閱 eksctl 文件中的 Config file schema (組態檔案結構描述)。此 eksctl 公用程式仍會為您建立啟動範本，並使用在 config 檔案提供的資料來填入其使用者資料。

使用下列命令來建立節點群組。

```
eksctl create nodegroup --config-file=my-nodegroup.yaml
```

啟動範本中的使用者資料

在啟動範本的使用者資料區段中指定下列資訊。將每個*###*取代為您自己的值。`--apiserver-endpoint`、`--b64-cluster-ca`，和 `--dns-cluster-ip` 引數為選用。但是，定義引述會允許 `bootstrap.sh` 指令碼避免進行 `describeCluster` 呼叫。這對於您經常向內擴展和向外擴展節點的私有叢集設定或叢集非常有用。如需`bootstrap.sh`指令碼的詳細資訊，請參閱 GitHub 上的 bootstrap.sh 檔案。

- 唯一的必要引數是叢集名稱 (*my-cluster*)。
- 若要擷取叢集的 *certificate-authority*，請執行下列命令。

```
aws eks describe-cluster --query "cluster.certificateAuthority.data" --output text
 --name my-cluster --region region-code
```

- 若要擷取叢集的 *api-server-endpoint*，請執行下列命令。

```
aws eks describe-cluster --query "cluster.endpoint" --output text --name my-
cluster --region region-code
```

- `--dns-cluster-ip` 的值是結尾處為 `.10` 的服務 CIDR。若要擷取叢集的 *service-cidr*，請執行下列命令。例如，如果 傳回的值為 `ipv4 10.100.0.0/16`，則您的值為 *10.100.0.10*。

```
aws eks describe-cluster --query "cluster.kubernetesNetworkConfig.serviceIpv4Cidr"
 --output text --name my-cluster --region region-code
```

- 此範例使用其中包含 Amazon EKS 最佳化 AMI 的 `bootstrap.sh` 指令碼提供 `kubelet` 引數來設定自訂 `max-pods` 值。如需選取 *my-max-pods-value* 的說明，請參閱 the section called “Amazon EKS 為每種 Amazon EC2 執行個體類型建議 Pod 數量上限”。

```
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary="==MYBOUNDARY=="

--==MYBOUNDARY==
Content-Type: text/x-shellscript; charset="us-ascii"

#!/bin/bash
set -ex
/etc/eks/bootstrap.sh my-cluster \
  --b64-cluster-ca certificate-authority \
  --apiserver-endpoint api-server-endpoint \
  --dns-cluster-ip service-cidr.10 \
  --kubelet-extra-args '--max-pods=my-max-pods-value' \
```

```
  --use-max-pods false

--===MYBOUNDARY==--
```

提供使用者資料，將引數傳遞至 Amazon EKS 最佳化 Windows AMI 隨附的**Start-EKSBootstrap.ps1**檔案

引導是一個術語，用於描述新增在執行個體啟動時可以執行的命令。您可以使用 `eksctl` 將引數傳遞給 `Start-EKSBootstrap.ps1` 而不指定啟動範本。您也可以在啟動範本的使用者資料區段中指定資訊來執行此動作。

如果您想要指定自訂 Windows AMI ID，請記住下列考量事項：

- 您必須使用啟動範本，並在使用者資料區段中提供所需引導命令。若要擷取所需的 Windows ID，您可以在建立具有最佳化 Windows AMIs節點中使用 資料表。
- 您需滿足幾個限制和條件。例如，您必須將 `eks:kube-proxy-windows`新增至 IAM Authenticator AWS 組態映射。如需詳細資訊，請參閱the section called “指定 AMI ID 時的限制和條件”。

在啟動範本的使用者資料區段中指定下列資訊。將每個*###*取代為您自己的值。`-APIServerEndpoint`、`-Base64ClusterCA`，和 `-DNSClusterIP` 引數為選用。但是，定義引述會允許 `Start-EKSBootstrap.ps1` 指令碼避免進行 `describeCluster` 呼叫。

- 唯一的必要引數是叢集名稱 (*my-cluster*)。
- 若要擷取叢集的 *certificate-authority*，請執行下列命令。

```
aws eks describe-cluster --query "cluster.certificateAuthority.data" --output text --
name my-cluster --region region-code
```

- 若要擷取叢集的 *api-server-endpoint*，請執行下列命令。

```
aws eks describe-cluster --query "cluster.endpoint" --output text --name my-cluster
 --region region-code
```

- `--dns-cluster-ip` 的值是結尾處為 `.10` 的服務 CIDR。若要擷取叢集的 *service-cidr*，請執行下列命令。例如，如果 的傳回值為 `ipv4 10.100.0.0/16`，則您的值為 *10.100.0.10*。

```
aws eks describe-cluster --query "cluster.kubernetesNetworkConfig.serviceIpv4Cidr" --
output text --name my-cluster --region region-code
```

- 如需其他引數，請參閱 the section called “引導指令碼組態參數”。

> **Note**
>
> 如果您使用的是自訂服務 CIDR，則需要使用 `-ServiceCIDR` 參數指定它。否則，叢集中 Pod 的 DNS 解析將會失敗。

```
<powershell>
[string]$EKSBootstrapScriptFile = "$env:ProgramFiles\Amazon\EKS\Start-EKSBootstrap.ps1"
& $EKSBootstrapScriptFile -EKSClusterName my-cluster `
  -Base64ClusterCA certificate-authority `
  -APIServerEndpoint api-server-endpoint `
  -DNSClusterIP service-cidr.10
</powershell>
```

由於特定安全、合規或內部政策要求，執行自訂 AMI

如需詳細資訊，請參閱《Amazon EC2 使用者指南》中的 Amazon Machine Images (AMI)。Amazon EKS AMI 建置規格包含資源和組態指令碼，用於建置以 Amazon Linux 為基礎的自訂 Amazon EKS AMI。如需詳細資訊，請參閱 GitHub 上的 Amazon EKS AMI 建置規範。若要建置使用其他作業系統安裝的自訂 AMI，則請參閱 GitHub 上的 Amazon EKS 樣本自訂 AMI。

您無法在與受管節點群組搭配使用的啟動範本中使用 AMI IDs 的動態參數參考。

> **Important**
>
> 指定 AMI 時，Amazon EKS 不會合併任何使用者資料。相反地，您負責提供節點加入叢集所需的`bootstrap`命令。如果節點無法加入叢集，則 Amazon EKS `CreateNodegroup` 和 `UpdateNodegroupVersion` 動作也會失敗。

指定 AMI ID 時的限制和條件

以下是使用受管節點群組指定 AMI ID 所涉及的限制和條件：

- 您必須建立新的節點群組，才能在指定啟動範本的 AMI ID 與不指定 AMI ID 之間切換。
- 有較新的 AMI 版本可用時，主控台不會通知您。若要將節點群組更新為較新的 AMI 版本，您需要使用更新的 AMI ID 來建立新版本的啟動範本。然後，您需要使用新的啟動範本版本更新節點群組。

- 如果您指定 AMI ID，則無法在 API 中設定下列欄位：
  - amiType
  - releaseVersion
  - version
- 如果您指定 AMI ID，則 API 的任何 taints 集合都會以非同步方式套用。若要在節點加入叢集之前套用污點，您必須使用 --register-with-taints 命令列旗標將污點傳遞給使用者資料的 kubelet。如需詳細資訊，請參閱 Kubernetes 文件中的 kubelet。
- 為 Windows 受管節點群組指定自訂 AMI ID 時，請將 eks:kube-proxy-windows新增至您的 AWS IAM Authenticator 組態映射。必須要有這項，DNS 才能正常運作。
  a. 開啟 AWS IAM Authenticator 組態映射以進行編輯。

```
kubectl edit -n kube-system cm aws-auth
```

  b. 將此項目新增至與 Windows 節點rolearn相關聯的每個 下的groups清單。您的組態映射看起來應該類似於 aws-auth-cm-windows.yaml。

```
- eks:kube-proxy-windows
```

  c. 儲存檔案並結束您的文字編輯器。
- 對於使用自訂啟動範本的任何 AMI，受管節點群組HttpPutResponseHopLimit的預設設定為 2。

從叢集刪除受管節點群組

本主題會描述可以如何刪除 Amazon EKS 受管節點群組。刪除受管節點群組時，Amazon EKS 會先將 Auto Scaling 群組的最小、最大和所需大小設定為零。這就會讓節點群組縮減規模。

在每個執行個體終止之前，Amazon EKS 會傳送訊號以從該節點耗盡 Pod。如果 Pod 在幾分鐘後仍未耗盡，Amazon EKS 可讓 Auto Scaling 繼續終止執行個體。在終止每個執行個體之後，便會刪除 Auto Scaling 群組。

> **Important**
>
> 如果您刪除使用節點 IAM 角色的受管節點群組，而叢集中的任何其他受管節點群組未使用該角色，則會從 aws-auth 中移除該角色ConfigMap。如果叢集中有任何自我管理節點群組使用相同的節點 IAM 角色，則自我管理節點會轉為 NotReady 狀態。此外，叢集操作也會中斷。若要為僅針對自我管理節點群組使用的角色新增映射，請參閱the section called “建立存取項目”，如果您叢集的平台版本至少為 版本，列於 授予 IAM 使用者使用 EKS 存取項目存取

Kubernetes 的先決條件區段中。如果您的平台版本早於存取項目所需的最低版本，您可以將項目新增至 aws-auth ConfigMap。如需詳細資訊，請在您的終端機中輸入 eksctl create iamidentitymapping --help。

您可以使用下列方式刪除受管節點群組：

- the section called “eksctl”
- the section called “AWS Management Console”
- the section called “AWS CLI”

eksctl

使用 刪除受管節點群組 **eksctl**

輸入以下命令。將每個###取代為您自己的值。

```
eksctl delete nodegroup \
  --cluster my-cluster \
  --name my-mng \
  --region region-code
```

如需更多選項，請參閱 eksctl 文件中的刪除和耗盡節點群組。

AWS Management Console

使用 刪除受管節點群組 AWS Management Console

1. 開啟 Amazon EKS 主控台。
2. 在叢集頁面上，選擇包含要刪除之節點群組的叢集。
3. 在所選叢集頁面上，選擇運算索引標籤。
4. 在 Node Groups (節點群組) 區段中，選擇要刪除的節點群組。然後選擇 Delete (刪除)。
5. 在刪除節點群組確認對話方塊中，輸入節點群組的名稱。然後選擇 Delete (刪除)。

AWS CLI

使用 CLI AWS 刪除受管節點群組

1. 輸入以下命令。將每個###取代為您自己的值。

```
aws eks delete-nodegroup \
  --cluster-name my-cluster \
  --nodegroup-name my-mng \
  --region region-code
```

2. 使用鍵盤上的方向鍵來捲動回應輸出。完成後，請按 q鍵。

如需更多選項，請參閱 CLI `delete-nodegroup` 命令參考中的 命令。 AWS

使用自我管理節點自行維護節點

叢集包含排程 Pod 的一或多個 Amazon EC2 節點。Amazon EKS 節點會在 AWS 您的帳戶中執行，並透過叢集 API 伺服器端點連線到叢集的控制平面。您需要根據 Amazon EC2 價格支付這些費用。如需詳細資訊，請參閱 Amazon EC2 定價。

一個叢集可以包含數個節點群組。每個節點群組包含部署在 Amazon EC2 Auto Scaling 群組中的一或多個節點。群組內節點的執行個體類型可能會有所不同，例如搭配 Karpenter 使用屬性型執行個體類型選取時。節點群組的所有執行個體都必須使用 Amazon EKS 節點 IAM 角色。

Amazon EKS 提供專門的Amazon 機器映像 (AMI)，稱為Amazon EKS 最佳化 AMI。AMI 已設定為搭配 Amazon EKS 一起使用。其元件包括 `containerd`、 `kubelet`和 AWS IAM Authenticator。AMIs 也包含專門的引導指令碼，允許它自動探索和連接到叢集的控制平面。

如果使用 CIDR 區塊限制對叢集公有端點的存取，則建議您也啟用私有端點存取。這樣可以讓節點與叢集通訊。若不啟用私有端點，您指定用於公有存取的 CIDR 區塊必須包含來自 VPC 的輸出來源。如需詳細資訊，請參閱the section called “叢集端點存取”。

若要將自我管理節點新增至 Amazon EKS 叢集，請參閱以下主題。如果您手動啟動自我管理節點，請將下列標籤新增至每個節點，同時確定 與您的叢集<cluster-name>相符。如需詳細資訊，請參閱新增和刪除個別資源的標籤。如果遵循以下指南中的步驟，系統會為您自動新增必要的標籤至節點。

金鑰	值
kubernetes.io/cluster/<cluster-name>	owned

> **Important**
>
> Amazon EC2 執行個體中繼資料服務 (IMDS) 中的標籤與 EKS 節點不相容。啟用執行個體中繼資料標籤時，會防止在標籤值中使用正斜線 ('/')。此限制可能會導致執行個體啟動失敗，尤其是在使用 Karpenter 或 Cluster Autoscaler 等節點管理工具時，因為這些服務依賴包含正斜線的標籤來提供適當的功能。

如需 Kubernetes 對節點普遍觀點的詳細資訊，請參閱 Kubernetes 文件中的節點。

主題

- 建立自我管理的 Amazon Linux 節點
- 建立自我管理的 Bottlerocket 節點
- 建立自我管理的 Microsoft Windows 節點
- 建立自我管理的 Ubuntu Linux 節點
- 更新叢集的自我管理節點

建立自我管理的 Amazon Linux 節點

本主題會說明如何啟動已向 Amazon EKS 叢集註冊的 Linux 節點的 Auto Scaling 群組。節點加入叢集後，您就可以將 Kubernetes 應用程式部署至其中。您也可以使用 `eksctl`或 啟動自我管理的 Amazon Linux 節點 AWS Management Console。如果您需要在 AWS Outposts 上啟動節點，請參閱 the section called “節點”。

- 現有 Amazon EKS 叢集。若要部署叢集，請參閱 the section called “建立叢集”。如果您在已啟用 AWS Outposts、 AWS Wavelength 或 AWS Local Zones AWS 的區域中有子網路，則這些子網路不得在您建立叢集時傳入。
- 供節點使用的現有 IAM 角色。若要建立服務角色，請參閱the section called “節點 IAM 角色”。如果此角色沒有 VPC CNI 的任何政策，則 VPC CNI Pod 需要以下單獨的角色。
- （選用，但建議）適用於 Kubernetes 附加元件的 Amazon VPC CNI 外掛程式，其設定為具有必要 IAM 政策的自有 IAM 角色。如需詳細資訊，請參閱the section called “為 IRSA 設定”。
- 熟悉選擇最佳 Amazon EC2 節點執行個體類型中列出的考量事項。取決於您選擇的執行個體類型，您的叢集和 VPC 可能還有其他先決條件。

您可以使用下列其中一項啟動自我管理的 Linux 節點：

- the section called “eksctl”
- the section called “AWS Management Console”

eksctl

使用 啟動自我管理的 Linux 節點 **eksctl**

1. 在您的裝置0.210.0或 AWS CloudShell 上安裝版本 或更新版本的eksctl命令列工具。如需有關安裝或更新 eksctl 的指示，請參閱 eksctl 文件中的安裝一節。
2. （選用）如果 AmazonEKS_CNI_Policy 受管 IAM 政策連接至您的 Amazon EKS 節點 IAM 角色，建議您改為將其指派給與 Kubernetes aws-node服務帳戶相關聯的 IAM 角色。如需詳細資訊，請參閱the section called “為 IRSA 設定”。
3. 以下命令會在現有的叢集建立節點群組。將 *al-nodes* 取代為您的節點群組名稱。節點群組名稱不能超過 63 個字元。它必須以字母或數字開頭，但剩餘字元也可以包含連字符和底線。使用您叢集的名稱取代 *my-cluster*。此名稱僅能使用英數字元 (區分大小寫) 和連字號。它必須以英數字元開頭，且長度不可超過 100 個字元。名稱在您要建立叢集 AWS 的區域和 AWS 帳戶中必須是唯一的。將剩餘的###取代為您自己的值。依預設，系統會使用與控制平面相同的 Kubernetes 版本來建立節點。

   在選擇 的值之前--node-type，請檢閱選擇最佳的 Amazon EC2 節點執行個體類型。

   將 *my-key* 取代為您的 Amazon EC2 金鑰對或公有金鑰的名稱。此金鑰會在節點啟動後用於將 SSH 套用至節點。如果您還沒有 Amazon EC2 金鑰對，您可以在 中建立一個金鑰對 AWS Management Console。如需詳細資訊，請參閱《Amazon EC2 使用者指南》中的 Amazon EC2 金鑰對。

   使用下列命令來建立您的節點群組。

   > **Important**
   >
   > 如果您想要將節點群組部署到 AWS Outposts、Wavelength 或 Local Zone 子網路，還有其他考量：
   >
   > - 在建立叢集時，不得傳入子網路。
   > - 您必須使用指定子網路和 volumeType: gp2 的組態檔案來建立節點群組。如需詳細資訊，請參閱 eksctl 文件中的從組態檔案建立節點群組和組態檔案結構描述。

```
eksctl create nodegroup \
  --cluster my-cluster \
  --name al-nodes \
  --node-type t3.medium \
  --nodes 3 \
  --nodes-min 1 \
  --nodes-max 4 \
  --ssh-access \
  --managed=false \
  --ssh-public-key my-key
```

若要部署節點群組：

- 可以將比預設組態更多的 IP 地址指派給 Pod，請參閱 the section called “增加 IP 地址”。
- 可以從與執行個體不同的 CIDR 區塊將IPv4地址指派給 Pod，請參閱 the section called “自訂聯網”。
- 可以將IPv6地址指派給 Pod 和服務，請參閱 the section called “IPv6”。
- 沒有傳出網際網路存取，請參閱 the section called “私有叢集”。

  如需所有可用選項和預設值的完整清單，請輸入以下命令。

  ```
  eksctl create nodegroup --help
  ```

  如果節點無法加入叢集，請參閱疑難排解章節the section called “節點無法加入叢集”中的 。

  範例輸出如下。建立節點時，會有數行輸出。輸出的最後幾行之一類似於以下的範例行。

  ```
  [#]  created 1 nodegroup(s) in cluster "my-cluster"
  ```

4. (選用) 部署範例應用程式以測試您的叢集和 Linux 節點。
5. 如果符合下列條件，我們建議封鎖 Pod 存取 IMDS：
   - 您計劃將 IAM 角色指派給所有 Kubernetes 服務帳戶，讓 Pod 僅擁有所需的最低許可。
   - 叢集中沒有任何 Pod 因其他原因需要存取 Amazon EC2 執行個體中繼資料服務 (IMDS)，例如擷取目前 AWS 區域。

   如需詳細資訊，請參閱限制存取指派給工作節點的執行個體設定檔。

AWS Management Console

步驟 1：使用 啟動自我管理的 Linux 節點 AWS Management Console

1. 下載最新版本的 AWS CloudFormation 範本。

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/cloudformation/2022-12-23/
amazon-eks-nodegroup.yaml
```

2. 等待您的叢集狀態顯示為 ACTIVE。如果在叢集處於作用中狀態之前啟動節點，則節點向叢集註冊將會失敗，導致必須再次啟動。
3. 開啟 AWS CloudFormation 主控台。
4. 選擇 Create stack (建立堆疊)，然後選取 With new resources (standard) (使用新資源 (標準))。
5. 針對 Specify template (指定範本)，選取 Upload a template file (上傳範本檔案)，然後選取 Choose file (選擇檔案)。
6. 選取您下載的 amazon-eks-nodegroup.yaml 檔案。
7. 選取下一步。
8. 在 Specify stack details (指定堆疊詳細資訊) 頁面上，據此填寫下列參數，然後選擇 Next (下一步)：
   - 堆疊名稱：為您的 AWS CloudFormation 堆疊選擇堆疊名稱。例如，您可以呼叫 *my-cluster-nodes*。此名稱僅能使用英數字元 (區分大小寫) 和連字號。它必須以英數字元開頭，且長度不可超過 100 個字元。在您建立叢集 AWS 的區域和 AWS 帳戶中，名稱必須是唯一的。
   - ClusterName：輸入建立 Amazon EKS 叢集時使用的名稱。此名稱必須等於叢集名稱，否則您的節點無法加入叢集。
   - ClusterControlPlaneSecurityGroup：從您在建立 VPC 時產生的 AWS CloudFormation 輸出中選擇 SecurityGroups 值。

     下列步驟顯示擷取適用群組的一種操作。

     a. 開啟 Amazon EKS 主控台。

     b. 選擇叢集的名稱。

     c. 選擇 Networking (網路) 索引標籤。

     d. 從 ClusterControlPlaneSecurityGroup 下拉式清單中選取時，請使用 Additional Security Group (其他安全群組) 值作為參考。

- NodeGroupName：輸入節點群組的名稱。此名稱稍後可用來識別為節點建立的 Auto Scaling 節點群組。節點群組名稱不能超過 63 個字元。它必須以字母或數字開頭，但剩餘字元也可以包含連字符和底線。
- NodeAutoScalingGroupMinSize：輸入節點 Auto Scaling 群組可以縮減的最低節點數。
- NodeAutoScalingGroupDesiredCapacity：堆疊建立時，輸入欲擴展的所需節點數量。
- NodeAutoScalingGroupMaxSize：輸入節點 Auto Scaling 群組可以擴增的最大節點數。
- NodeInstanceType：選擇節點的執行個體類型。如需詳細資訊，請參閱the section called “Amazon EC2 執行個體類型”。
- NodeImageIdSSMParam：預先填入變數 Kubernetes 版本的最新 Amazon EKS 最佳化 AMI 的 Amazon EC2 Systems Manager 參數。若要使用 Amazon EKS 支援的不同 Kubernetes 次要版本，請以不同的 eks/latest/userguide/kubernetes-versions.html【支援的版本，type="documentation"】 取代 *1.XX#*建議指定與叢集相同的 Kubernetes 版本。

  您也可以將 *amazon-linux-2* 取代為不同的 AMI 類型。如需詳細資訊，請參閱the section called “取得最新的 IDs”。

  > **Note**
  >
  > Amazon EKS 節點 AMIs 是以 Amazon Linux 為基礎。您可以透過 Amazon Linux 安全中心或是訂閱關聯的 RSS 摘要，追蹤 Amazon Linux 2 的安全或隱私權事件。安全與隱私權事件包含問題的概觀、哪些套件受到影響，以及如何更新您的執行個體以修正問題。

- NodeImageId：（選用）如果您使用自己的自訂 AMI（而非 Amazon EKS 最佳化 AMI)，請輸入您 AWS 區域的節點 AMI ID。如果您在此指定值，則會覆寫 NodeImageIdSSMParam 欄位中的任何值。
- NodeVolumeSize：為您的節點指定根磁碟區大小 (以 GiB 為單位)。
- NodeVolumeType：為您的節點指定根磁碟區類型。
- KeyName：輸入 Amazon EC2 SSH 金鑰對的名稱，您可以在節點啟動後使用該金鑰對來透過 SSH 連接至節點。如果您還沒有 Amazon EC2 金鑰對，您可以在 中建立一個金鑰對 AWS Management Console。如需詳細資訊，請參閱《Amazon EC2 使用者指南》中的 Amazon EC2 金鑰對。

  > **Note**
  >
  > 如果您未在此處提供金鑰對， AWS CloudFormation 堆疊建立會失敗。

- BootstrapArguments：指定要傳遞至節點引導指令碼的任何選用引數，例如額外的 `kubelet` 引數。如需詳細資訊，請檢視 GitHub 上的 bootstrap script usage information (啟動程序指令碼使用資訊)。

  若要部署節點群組：

  - 可以將比預設組態更多的 IP 地址指派給 Pod，請參閱 the section called “增加 IP 地址”。
  - 可以從與執行個體不同的 CIDR 區塊將IPv4地址指派給 Pod，請參閱 the section called “自訂聯網”。
  - 可以將IPv6地址指派給 Pod 和服務，請參閱 the section called “IPv6”。
  - 沒有傳出網際網路存取，請參閱 the section called “私有叢集”。
- DisableIMDSv1：預設情況下，每個節點都支援執行個體中繼資料服務版本 1 (IMDSv1) 和 IMDSv2。您可以停用 IMDSv1。若要防止節點群組中的未來節點和 Pod 使用 MDSv1，請將 DisableIMDSv1 設為 true。如需 IMDS 的詳細資訊，請參閱設定執行個體中繼資料服務。如需在節點上限制存取的詳細資訊，請參閱限制存取指派給工作節點的執行個體設定檔。
- VpcId：輸入您建立的 VPC ID。
- 子網路：選擇您為 VPC 建立的子網路。如果您使用為 Amazon EKS 叢集建立 Amazon VPC 中所述的步驟建立 VPC，請僅指定要啟動節點之 VPC 內的私有子網路。您可以看到哪些子網是私有子網，方法是從叢集的 Networking (聯網) 標籤打開每一個子網連結。

  > **Important**
  >
  > - 如有任何子網路是公有子網路，則必須啟用自動公有 IP 地址指派設定。如果未針對公有子網路啟用 設定，則您部署至該公有子網路的任何節點都不會指派公有 IP 地址，也無法與叢集或其他 AWS 服務通訊。如果子網路是在 2020 年 3 月 26 日之前使用任一 Amazon EKS AWS CloudFormation VPC 範本或使用 部署，`eksctl`則會停用公有子網路的自動公有 IP 地址指派。如需如何啟用子網路的公有 IP 地址指派的相關資訊，請參閱修改您子網路的公有 IPv4 定址屬性。如果節點部署到私有子網路，則可以透過 NAT 閘道與叢集和其他 AWS 服務通訊。
  > - 如果子網路無法存取網際網路，請確定您知道部署具有有限網際網路存取的私有叢集中的考量事項和額外步驟。
  > - 如果您選取 AWS Outpost、Wavelength 或 Local Zone 子網路，則建立叢集時不得傳入子網路。

9. 請在 Configure stack options (設定堆疊選項) 頁面上選取所需的選項，然後選擇 Next (下一頁)。

10選取我確認 AWS CloudFormation 可能會建立 IAM 資源的左側核取方塊，然後選擇建立堆疊。

11.當當堆疊已完成建立時，從主控台將其選取，然後選擇 Outputs (輸出)。

12.為已建立的節點群組記錄 NodeInstanceRole。當您設定 Amazon EKS 節點時會需要此值。

步驟 2：啟用節點以加入您的叢集

> **Note**
>
> 如果您在沒有傳出網際網路存取權的私有 VPC 內啟動節點，則必須啟用節點才能從 VPC 內加入叢集。

1. 檢查以瞭解是否有 aws-auth ConfigMap。

```
kubectl describe configmap -n kube-system aws-auth
```

2. 如果您看到 aws-auth ConfigMap，請視需要更新之。
   a. 開啟 ConfigMap 進行編輯。

```
kubectl edit -n kube-system configmap/aws-auth
```

   b. 視需要新增 mapRoles 個項目。將 rolearn 值設定為您在先前程序中記錄的 NodeInstanceRole 值。

```
[...]
data:
  mapRoles: |
    - rolearn: <ARN of instance role (not instance profile)>
      username: system:node:{{EC2PrivateDNSName}}
      groups:
        - system:bootstrappers
        - system:nodes
[...]
```

   c. 儲存檔案並結束您的文字編輯器。

3. 如果您收到一個錯誤，說明 "Error from server (NotFound): configmaps "aws-auth" not found，那麼請套用股票 ConfigMap。
   a. 下載組態對應。

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/cloudformation/2020-10-29/
aws-auth-cm.yaml
```

b. 在 aws-auth-cm.yaml 檔案中，將 rolearn 值設定為您在先前程序中紀錄的 NodeInstanceRole 值。您可以使用文字編輯器執行此操作，或取代 *my-node-instance-role* 並執行下列命令：

```
sed -i.bak -e 's|<ARN of instance role (not instance profile)>|my-node-instance-
role|' aws-auth-cm.yaml
```

c. 套用組態。此命令可能需要幾分鐘的時間來完成。

```
kubectl apply -f aws-auth-cm.yaml
```

4. 查看節點的狀態，並等待他們到達 Ready 狀態。

```
kubectl get nodes --watch
```

輸入 Ctrl+C 傳回 Shell 提示。

> **Note**
>
> 如果您收到任何授權或資源類型錯誤，請參閱故障診斷主題中的the section called “未經授權或存取遭拒 (kubectl)”。

如果節點無法加入叢集，請參閱疑難排解章節the section called “節點無法加入叢集”中的 。

5. （僅限 GPU 節點）如果您選擇 GPU 執行個體類型和 Amazon EKS 最佳化加速 AMI，則必須將 Kubernetes 的 NVIDIA 裝置外掛程式套用為叢集上的 DaemonSet。在執行下列命令之前，將 *vX.X.X* 取代為所需的 NVIDIA/k8s-device-plugin 版本。

```
kubectl apply -f https://raw.githubusercontent.com/NVIDIA/k8s-device-plugin/vX.X.X/
deployments/static/nvidia-device-plugin.yml
```

步驟 3：其他動作

1. (選用) 部署範例應用程式以測試您的叢集和 Linux 節點。

2. （選用）如果 AmazonEKS_CNI_Policy 受管 IAM 政策（如果您有 IPv4叢集）或 *AmazonEKS_CNI_IPv6_Policy*（如果您有 IPv6叢集，則為您自己建立的）已連接至 Amazon EKS 節點 IAM 角色，建議您改為將其指派給與 Kubernetes aws-node服務帳戶相關聯的 IAM 角色。如需詳細資訊，請參閱the section called “為 IRSA 設定”。
3. 如果符合下列條件，我們建議封鎖 Pod 存取 IMDS：
   - 您計劃將 IAM 角色指派給所有 Kubernetes 服務帳戶，讓 Pod 僅擁有所需的最低許可。
   - 叢集中沒有任何 Pod 因其他原因需要存取 Amazon EC2 執行個體中繼資料服務 (IMDS)，例如擷取目前 AWS 區域。

   如需詳細資訊，請參閱限制存取指派給工作節點的執行個體設定檔。

建立自我管理的 Bottlerocket 節點

> **Note**
>
> 受管節點群組可能會為您的使用案例提供一些優勢。如需詳細資訊，請參閱the section called “受管節點群組”。

本主題說明如何啟動向 Amazon EKS 叢集註冊的 Bottlerocket 節點 Auto Scaling 群組。Bottlerocket 是以 Linux 為基礎的開放原始碼作業系統 AWS ，可用於在虛擬機器或裸機主機上執行容器。節點加入叢集後，您就可以將 Kubernetes 應用程式部署至其中。如需 Bottlerocket 的詳細資訊，請參閱 GitHub 上的搭配 Amazon EKS 使用 Bottlerocket AMI 和 `eksctl` 文件上的自訂 AMI 支援。

如需就地升級的資訊，請參閱 GitHub 上的 Bottlerocket 更新運算子。

> **Important**
>
> - Amazon EKS 節點為標準 Amazon EC2 執行個體，會根據一般 Amazon EC2 執行個體價格向您收取這些節點的費用。如需詳細資訊，請參閱 Amazon EC2 定價。
> - 您可以在 AWS Outposts 上的 Amazon EKS 擴充叢集中啟動 Bottlerocket 節點，但無法在 AWS Outposts 的本機叢集中啟動它們。如需詳細資訊，請參閱 AWS Outposts 上的 Amazon EKS。
> - 您可以使用 x86或 Arm 處理器部署至 Amazon EC2 執行個體。不過，您無法部署到具有 Inferentia 晶片的執行個體。

- Bottlerocket 與 AWS CloudFormation 相容。不過，沒有官方 CloudFormation 範本可以複製以部署 Amazon EKS 的 Bottlerocket 節點。
- Bottlerocket 映像不會隨附 SSH 伺服器或 Shell。您可以使用頻外存取方法來允許 SSH 啟用管理員容器，並傳遞一些引導組態步驟與使用者資料。如需詳細資訊，請參閱 GitHub 上 bottlerocket README.md 中的這些部分：
  - 探勘
  - 管理員容器
  - Kubernetes 設定

此程序需要 eksctl 版本 0.210.0 或更新版本。您可使用以下命令檢查您的版本：

```
eksctl version
```

如需如何安裝或升級 的指示eksctl，請參閱 eksctl 文件中的安裝。注意：此程序僅適用於使用 建立的叢集eksctl。

1. 將以下內容複製到您的裝置。使用您叢集的名稱取代 *my-cluster*。此名稱僅能使用英數字元 (區分大小寫) 和連字號。它必須以英數字元開頭，且長度不可超過 100 個字元。在您建立叢集 AWS 的區域和 AWS 帳戶中，名稱必須是唯一的。以節點群組的名稱取代 *ng-bottlerocket*。節點群組名稱不能超過 63 個字元。它必須以字母或數字開頭，但剩餘字元也可以包含連字符和底線。若要在 Arm 執行個體上部署，請將 *m5.large* 取代為 Arm 執行個體類型。將 *my-ec2-keypair-name* 取代為 Amazon EC2 SSH 金鑰對的名稱，您可以在啟動後使用 SSH 連線至節點。如果您還沒有 Amazon EC2 金鑰對，您可以在 中建立一個金鑰對 AWS Management Console。如需詳細資訊，請參閱《Amazon EC2 使用者指南》中的 Amazon EC2 金鑰對。將所有剩餘的###取代為您自己的值。進行替換後，請執行修改後的命令來建立 bottlerocket.yaml 檔案。

   如果指定 Arm Amazon EC2 執行個體類型，請在部署之前檢閱 Amazon EKS 最佳化 Arm Amazon Linux AMIs中的考量事項。如需了解如何使用自訂 AMI 部署，請參閱 GitHub 上的建置 Bottlerocket 和 eksctl 文件中的自訂 AMI 支援。若要部署受管節點群組，請使用啟動範本部署自訂 AMI。如需詳細資訊，請參閱the section called “啟動範本”。

   > **Important**
   >
   > 若要將節點群組部署至 AWS Outposts、 AWS Wavelength 或 AWS Local Zone 子網路，請勿在建立叢集時傳遞 AWS Outposts、 AWS Wavelength 或 AWS Local Zone 子網路。您必

須在下列範例中指定子網路。如需詳細資訊，請參閱 eksctl 文件中的從組態檔案建立節點群組和組態檔案結構描述。將 *region-code* 取代為您的叢集所在的 AWS 區域。

```
cat >bottlerocket.yaml <<EOF
---
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: my-cluster
  region: region-code
  version: '1.33'

iam:
  withOIDC: true

nodeGroups:
  - name: ng-bottlerocket
    instanceType: m5.large
    desiredCapacity: 3
    amiFamily: Bottlerocket
    ami: auto-ssm
    iam:
       attachPolicyARNs:
          - arn:aws: iam::aws:policy/AmazonEKSWorkerNodePolicy
          - arn:aws: iam::aws:policy/AmazonEC2ContainerRegistryReadOnly
          - arn:aws: iam::aws:policy/AmazonSSMManagedInstanceCore
          - arn:aws: iam::aws:policy/AmazonEKS_CNI_Policy
    ssh:
        allow: true
        publicKeyName: my-ec2-keypair-name
EOF
```

2. 使用下列命令部署節點。

```
eksctl create nodegroup --config-file=bottlerocket.yaml
```

範例輸出如下。

建立節點時，會有數行輸出。輸出的最後幾行之一類似於以下的範例行。

```
[#]  created 1 nodegroup(s) in cluster "my-cluster"
```

3. (選用) 使用 Amazon EBS CSI 外掛程式在 Bottlerocket 節點上建立 Kubernetes 持續性磁碟區。預設 Amazon EBS 驅動程式倚賴 Bottlerocket 未包含的檔案系統工具。如需使用驅動程式建立儲存類別的詳細資訊，請參閱 the section called “Amazon EBS”。
4. (選用) 依預設 kube-proxy 會將 nf_conntrack_max 核心參數設定為預設值，此值可能與開機時 Bottlerocket 原先設定的不同。若要保留 Bottlerocket 的預設設定，請使用下列命令編輯kube-proxy組態。

   ```
   kubectl edit -n kube-system daemonset kube-proxy
   ```

   新增 --conntrack-max-per-core 和 --conntrack-min 到 kube-proxy 引數，這些引數位於以下範例中。0 設定意味著沒有改變。

   ```
         containers:
         - command:
           - kube-proxy
           - --v=2
           - --config=/var/lib/kube-proxy-config/config
           - --conntrack-max-per-core=0
           - --conntrack-min=0
   ```
5. (選用) 部署範例應用程式來測試您的 Bottlerocket 節點。
6. 如果符合下列條件，我們建議封鎖 Pod 存取 IMDS：
   - 您計劃將 IAM 角色指派給所有 Kubernetes 服務帳戶，讓 Pod 僅擁有所需的最低許可。
   - 叢集中沒有任何 Pod 因其他原因需要存取 Amazon EC2 執行個體中繼資料服務 (IMDS)，例如擷取目前 AWS 區域。

   如需詳細資訊，請參閱限制存取指派給工作節點的執行個體設定檔。

建立自我管理的 Microsoft Windows 節點

本主題說明如何啟動已向 Amazon EKS 叢集註冊的 Windows 節點的 Auto Scaling 群組。節點加入叢集後，您就可以將 Kubernetes 應用程式部署至其中。

Important

- Amazon EKS 節點為標準 Amazon EC2 執行個體，會根據一般 Amazon EC2 執行個體價格向您收取這些節點的費用。如需詳細資訊，請參閱 Amazon EC2 定價。
- 您可以在 AWS Outposts 上的 Amazon EKS 延伸叢集中啟動 Windows 節點，但無法在 AWS Outposts 的本機叢集中啟動它們。如需詳細資訊，請參閱 AWS Outposts 上的 Amazon EKS。

為您的叢集啟用 Windows 支援。我們建議您在啟動 Windows 節點群組之前，先檢閱重要的考量。如需詳細資訊，請參閱the section called “啟用 Windows 支援”。

您可以使用下列其中一項啟動自我管理 Windows 節點：

- the section called “`eksctl`”
- the section called “AWS Management Console”

`eksctl`

使用 啟動自我管理的 Windows 節點 **`eksctl`**

此程序需要您已安裝 `eksctl`，且您的 `eksctl` 版本至少是 `0.210.0`。您可使用以下命令檢查您的版本。

```
eksctl version
```

如需有關安裝或更新 `eksctl` 的指示，請參閱 `eksctl` 文件中的 Installation 一節。

Note

此程序只適用於使用 `eksctl` 所建立的叢集。

1. （選用）如果 AmazonEKS_CNI_Policy 受管 IAM 政策（如果您有 IPv4叢集）或 *AmazonEKS_CNI_IPv6_Policy*（如果您有 IPv6叢集，則為您自己建立的）已連接至 Amazon EKS 節點 IAM 角色，建議您改為將其指派給與 Kubernetes aws-node服務帳戶相關聯的 IAM 角色。如需詳細資訊，請參閱the section called “為 IRSA 設定”。

2. 此程序假設您有現有的叢集。如果您還沒有要新增 Windows 節點群組的 Amazon EKS 叢集和 Amazon Linux 節點群組，建議您遵循 the section called “建立叢集 (eksctl)”。本指南提供如何使用 Amazon Linux 節點建立 Amazon EKS 叢集的完整逐步解說。

   使用下列命令來建立您的節點群組。將 *region-code* 取代為您的叢集所在的 AWS 區域。使用您的叢集名稱取代 *my-cluster*。此名稱僅能使用英數字元 (區分大小寫) 和連字號。它必須以英數字元開頭，且長度不可超過 100 個字元。名稱在您要建立叢集 AWS 的區域和 AWS 帳戶中必須是唯一的。將 *ng-windows* 取代為節點群組的名稱。節點群組名稱不能超過 63 個字元。它必須以字母或數字開頭，但剩餘字元也可以包含連字符和底線。您可以使用 取代 *2019* 2022以使用 Windows Server 2022。將其餘*###*取代為您自己的值。

   > **Important**
   >
   > 若要將節點群組部署至 AWS Outposts、 AWS Wavelength 或 AWS Local Zone 子網路，請勿在建立叢集時傳遞 AWS Outposts、Wavelength 或 Local Zone 子網路。使用組態檔案建立節點群組，指定 AWS Outposts、Wavelength 或 Local Zone 子網路。如需詳細資訊，請參閱 `eksctl` 文件中的從組態檔案建立節點群組和組態檔案結構描述。

   ```
   eksctl create nodegroup \
       --region region-code \
       --cluster my-cluster \
       --name ng-windows \
       --node-type t2.large \
       --nodes 3 \
       --nodes-min 1 \
       --nodes-max 4 \
       --managed=false \
       --node-ami-family WindowsServer2019FullContainer
   ```

   > **Note**
   >
   > - 如果節點無法加入叢集，請參閱故障診斷指南中的 the section called “節點無法加入叢集”。
   > - 若要查看 `eksctl` 命令可用的選項，請輸入下列命令。
   >
   >   ```
   >   eksctl command -help
   >   ```

範例輸出如下。建立節點時，會有數行輸出。輸出的最後幾行之一類似於以下的範例行。

```
[#]  created 1 nodegroup(s) in cluster "my-cluster"
```

3. (選用) 部署範例應用程式以測試您的叢集和 Windows 節點。
4. 如果符合下列條件，我們建議封鎖 Pod 存取 IMDS：
   - 您計劃將 IAM 角色指派給所有 Kubernetes 服務帳戶，讓 Pod 僅擁有所需的最低許可。
   - 叢集中沒有任何 Pod 因其他原因需要存取 Amazon EC2 執行個體中繼資料服務 (IMDS)，例如擷取目前 AWS 區域。

   如需詳細資訊，請參閱限制存取指派給工作節點的執行個體設定檔。

AWS Management Console

先決條件

- 現有 Amazon EKS 叢集和 Linux 節點群組。如果您沒有這些資源，建議您使用 中的其中一個指南來建立這些資源開始使用。這些指南說明如何使用 Linux 節點建立 Amazon EKS 叢集。
- 符合 Amazon EKS 叢集要求的現有 VPC 和安全群組。如需詳細資訊，請參閱the section called “VPC 和子網要求”及the section called “安全群組要求”。中的指南開始使用會建立符合要求的 VPC。或者，您也可以遵循為您的 Amazon EKS 叢集建立 Amazon VPC 來手動建立。
- 現有的 Amazon EKS 叢集，其使用符合 Amazon EKS 叢集要求的 VPC 和安全群組。如需詳細資訊，請參閱the section called “建立叢集”。如果您在已啟用 AWS AWS Outposts、 AWS Wavelength 或 AWS Local Zones 的區域中有子網路，則這些子網路不得在您建立叢集時傳入。

步驟 1：使用 啟動自我管理 Windows 節點 AWS Management Console

1. 等待您的叢集狀態顯示為 `ACTIVE`。如果在叢集處於作用中狀態之前啟動節點，則節點向叢集註冊將會失敗，導致您必須再次啟動。
2. 開啟 AWS CloudFormation 主控台
3. 選擇建立堆疊。
4. 對於 Specify template (指定範本)，選取 Amazon S3 URL。
5. 複製以下 URL 並將其貼到 Amazon S3 URL 中。

```
https://s3.us-west-2.amazonaws.com/amazon-eks/cloudformation/2023-02-09/amazon-eks-windows-nodegroup.yaml
```

6. 選取 Next (下一步) 兩次。
7. 在 Quick create stack (快速建立堆疊) 頁面上，視需要輸入下列參數：
   - 堆疊名稱：為您的 AWS CloudFormation 堆疊選擇堆疊名稱。例如，您可以稱它為 `my-cluster-nodes`。
   - ClusterName：輸入建立 Amazon EKS 叢集時使用的名稱。

     > **Important**
     >
     > 此名稱必須與您在步驟 1：建立 Amazon EKS 叢集中使用的名稱完全相符。否則，您的節點無法加入叢集。

   - ClusterControlPlaneSecurityGroup：從您在建立 VPC 時產生的 AWS CloudFormation 輸出中選擇安全群組。下列步驟顯示擷取適用群組的一種方法。
     a. 開啟 Amazon EKS 主控台。
     b. 選擇叢集的名稱。
     c. 選擇 Networking (網路) 索引標籤。
     d. 從 ClusterControlPlaneSecurityGroup 下拉式清單中選取時，請使用 Additional Security Group (其他安全群組) 值作為參考。
   - NodeGroupName：輸入節點群組的名稱。此名稱稍後可用來識別為節點建立的 Auto Scaling 節點群組。節點群組名稱不能超過 63 個字元。它必須以字母或數字開頭，但剩餘字元也可以包含連字符和底線。
   - NodeAutoScalingGroupMinSize：輸入節點 Auto Scaling 群組可以縮減的最低節點數。
   - NodeAutoScalingGroupDesiredCapacity：堆疊建立時，輸入欲擴展的所需節點數量。
   - NodeAutoScalingGroupMaxSize：輸入節點 Auto Scaling 群組可以擴增的最大節點數。
   - NodeInstanceType：選擇節點的執行個體類型。如需詳細資訊，請參閱the section called “Amazon EC2 執行個體類型”。

> **Note**
>
> 最新版本的 Kubernetes 專用 Amazon VPC CNI 外掛程式支援的執行個體類型會在 GitHub 上的 vpc_ip_resource_limit.go 中列出。您可能需要更新 CNI 版本，才能使用最新支援的執行個體類型。如需詳細資訊，請參閱the section called “Amazon VPC CNI”。

- NodeImageIdSSMParam：預先填入目前所建議 Amazon EKS 最佳化 Windows Core AMI ID 的 Amazon EC2 Systems Manager 參數。若要使用完整版本的 Windows，請將 *Core* 取代為 Full。
- NodeImageId：（選用）如果您使用自己的自訂 AMI（而非 Amazon EKS 最佳化 AMI)，請輸入您 AWS 區域的節點 AMI ID。如果您在此欄位指定值，則會覆寫 NodeImageIdSSMParam 欄位中的任何值。
- NodeVolumeSize：為您的節點指定根磁碟區大小 (以 GiB 為單位)。
- KeyName：輸入 Amazon EC2 SSH 金鑰對的名稱，您可以在節點啟動後使用該金鑰對來透過 SSH 連接至節點。如果您還沒有 Amazon EC2 金鑰對，您可以在 中建立一個金鑰對 AWS Management Console。如需詳細資訊，請參閱《Amazon EC2 使用者指南》中的 Amazon EC2 金鑰對。

> **Note**
>
> 如果您未在此處提供金鑰對，則無法建立 AWS CloudFormation 堆疊。

- BootstrapArguments：指定要傳遞至節點引導指令碼的任何選用引數，例如使用 -KubeletExtraArgs 的額外 kubelet 引數。
- DisableIMDSv1：預設情況下，每個節點都支援執行個體中繼資料服務版本 1 (IMDSv1) 和 IMDSv2。您可以停用 IMDSv1。若要防止節點群組中的未來節點和 Pod 使用 MDSv1，請將 DisableIMDSv1 設為 true。如需 IMDS 的詳細資訊，請參閱設定執行個體中繼資料服務。
- VpcId：選取您建立的 VPC ID。
- NodeSecurityGroups：選取在建立 VPC 時為您的 Linux 節點群組建立的安全群組。如果 Linux 節點連接了一個以上的安全群組，請指定所有的群組。例如，如果 Linux 節點群組是使用 eksctl 建立。
- Subnets (子網路)：選擇您建立的子網路。如果您使用為 Amazon EKS 叢集建立 Amazon VPC 中的步驟建立 VPC，則請僅指定要啟動節點之 VPC 內的私有子網路。

> **Important**
>
> - 如有任何子網路是公有子網路，則必須啟用自動公有 IP 地址指派設定。如果未針對公有子網路啟用 設定，則您部署至該公有子網路的任何節點都不會指派公有 IP 地址，也無法與叢集或其他 AWS 服務通訊。如果子網路是在 2020 年 3 月 26 日之前使用任一 Amazon EKS AWS CloudFormation VPC 範本或使用 部署，eksctl則公有子網路的自動公有 IP 地址指派會停用。如需如何啟用子網路的公有 IP 地址指派的相關資訊，請參閱修改您子網路的公有 IPv4 定址屬性。如果節點部署到私有子網路，則可以透過 NAT 閘道與叢集和其他 AWS 服務通訊。
> - 如果子網路沒有網際網路存取，請確定您知道部署具有有限網際網路存取的私有叢集中的考量事項和額外步驟。
> - 如果您選取 AWS Outpost、Wavelength 或 Local Zone 子網路，則建立叢集時不得傳入子網路。

8. 確認堆疊可建立 IAM 資源，然後選擇 Create stack (建立堆疊)。
9. 當當堆疊已完成建立時，從主控台將其選取，然後選擇 Outputs (輸出)。
10. 為已建立的節點群組記錄 NodeInstanceRole。當您設定 Amazon EKS Windows 節點時會需要此值。

步驟 2：啟用節點以加入您的叢集

1. 檢查以瞭解是否有 aws-auth ConfigMap。

```
kubectl describe configmap -n kube-system aws-auth
```

2. 如果您看到 aws-auth ConfigMap，請視需要更新之。
   a. 開啟 ConfigMap 進行編輯。

```
kubectl edit -n kube-system configmap/aws-auth
```

   b. 視需要新增 mapRoles 個項目。將 rolearn 值設定為您在先前程序中記錄的 NodeInstanceRole 值。

```
[...]
data:
  mapRoles: |
- rolearn: <ARN of linux instance role (not instance profile)>
```

```
      username: system:node:{{EC2PrivateDNSName}}
      groups:
        - system:bootstrappers
        - system:nodes
    - rolearn: <ARN of windows instance role (not instance profile)>
      username: system:node:{{EC2PrivateDNSName}}
      groups:
        - system:bootstrappers
        - system:nodes
        - eks:kube-proxy-windows
[...]
```

c. 儲存檔案並結束您的文字編輯器。

3. 如果您收到一個錯誤，說明 "Error from server (NotFound): configmaps "aws-auth" not found，那麼請套用股票 ConfigMap。

a. 下載組態對應。

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/cloudformation/2020-10-29/
aws-auth-cm-windows.yaml
```

b. 在 aws-auth-cm-windows.yaml 檔案中，將 rolearn 值設定為您在先前程序中記錄的適用 NodeInstanceRole 值。您可以使用文字編輯器執行此操作，或取代###並執行下列命令：

```
sed -i.bak -e 's|<ARN of linux instance role (not instance profile)>|my-node-
linux-instance-role|' \
    -e 's|<ARN of windows instance role (not instance profile)>|my-node-windows-
instance-role|' aws-auth-cm-windows.yaml
```

> **Important**
>
> - 請勿修改此檔案中的任何其他行。
> - 請勿對 Windows 和 Linux 節點使用相同的 IAM 角色。

c. 套用組態。此命令可能需要幾分鐘的時間來完成。

```
kubectl apply -f aws-auth-cm-windows.yaml
```

4. 查看節點的狀態，並等待他們到達 Ready 狀態。

```
kubectl get nodes --watch
```

輸入 Ctrl+C 傳回 Shell 提示。

> **Note**
>
> 如果您收到任何授權或資源類型錯誤，請參閱故障診斷主題中的the section called “未經授權或存取遭拒 (kubectl)”。

如果節點無法加入叢集，請參閱疑難排解章節the section called “節點無法加入叢集”中的 。

步驟 3：其他動作

1. (選用) 部署範例應用程式以測試您的叢集和 Windows 節點。
2. （選用） 如果 AmazonEKS_CNI_Policy 受管 IAM 政策 （如果您有 IPv4叢集） 或 *AmazonEKS_CNI_IPv6_Policy* （如果您有 IPv6叢集，則為您自己建立的） 已連接至 Amazon EKS 節點 IAM 角色，建議您改為將其指派給與 Kubernetes aws-node服務帳戶相關聯的 IAM 角色。如需詳細資訊，請參閱the section called “為 IRSA 設定”。
3. 如果符合下列條件，我們建議封鎖 Pod 存取 IMDS：
   - 您計劃將 IAM 角色指派給所有 Kubernetes 服務帳戶，讓 Pod 僅擁有所需的最低許可。
   - 叢集中沒有任何 Pod 因其他原因需要存取 Amazon EC2 執行個體中繼資料服務 (IMDS)，例如擷取目前 AWS 區域。

   如需詳細資訊，請參閱限制存取指派給工作節點的執行個體設定檔。

建立自我管理的 Ubuntu Linux 節點

> **Note**
>
> 受管節點群組可能會為您的使用案例提供一些優勢。如需詳細資訊，請參閱the section called “受管節點群組”。

本主題說明如何在向 Amazon EKS 叢集註冊的 Amazon Elastic Kubernetes Service (EKS) 或 Amazon Elastic Kubernetes Service (EKS) 節點上啟動 Ubuntu 的 Auto Scaling 群組。Ubuntu 和 Ubuntu Pro for EKS 是以官方 Ubuntu Minimal LTS 為基礎，包括與 共同開發的自訂 AWS 核心 AWS，並專門為

EKS 建置。Ubuntu Pro 透過支援 EKS 延長支援期間、核心即時修補、FIPS 合規以及執行無限制 Pro 容器的能力來新增額外的安全涵蓋範圍。

節點加入叢集後，您可以將容器化應用程式部署到叢集。如需詳細資訊，請參閱 上的 Ubuntu AWS eksctl 文件和 文件中的自訂 AMI 支援。

> **Important**
>
> - Amazon EKS 節點為標準 Amazon EC2 執行個體，會根據一般 Amazon EC2 執行個體價格向您收取這些節點的費用。如需詳細資訊，請參閱 Amazon EC2 定價。
> - 您可以在 AWS Outposts 上的 Amazon EKS 擴充叢集中啟動 Ubuntu 節點，但無法在 AWS Outposts 的本機叢集中啟動它們。如需詳細資訊，請參閱 AWS Outposts 上的 Amazon EKS。
> - 您可以使用 x86或 Arm 處理器部署至 Amazon EC2 執行個體。不過，具有 Inferentia 晶片的執行個體可能需要先安裝 Neuron SDK。

此程序需要 eksctl 版本 0.210.0 或更新版本。您可使用以下命令檢查您的版本：

```
eksctl version
```

如需如何安裝或升級 的指示eksctl，請參閱 eksctl 文件中的安裝。注意：此程序僅適用於使用 建立的叢集eksctl。

1. 將以下內容複製到您的裝置。使用您叢集的名稱取代 my-cluster。此名稱僅能使用英數字元 (區分大小寫) 和連字號。它必須以字母字元開頭，且長度不可超過 100 個字元。將 ng-ubuntu 取代為您的節點群組名稱。節點群組名稱不能超過 63 個字元。它必須以字母或數字開頭，但剩餘字元也可以包含連字符和底線。若要在 Arm 執行個體上部署，請以 Arm 執行個體類型取代 m5.large。使用 Amazon EC2 SSH 金鑰對名稱取代 my-ec2-keypair-name，您可以在節點啟動後使用該金鑰對來透過 SSH 連接至節點。如果您還沒有 Amazon EC2 金鑰對，您可以在 中建立一個金鑰對 AWS Management Console。如需詳細資訊，請參閱《Amazon EC2 使用者指南》中的 Amazon EC2 金鑰對。 Amazon EC2 將所有剩餘的###取代為您自己的值。進行替換後，請執行修改後的命令來建立 ubuntu.yaml 檔案。

   > **Important**
   >
   > 若要將節點群組部署至 AWS Outposts、 AWS Wavelength 或 AWS Local Zone 子網路，請勿在建立叢集時傳遞 AWS Outposts、 AWS Wavelength 或 AWS Local Zone 子網路。您必

須在下列範例中指定子網路。如需詳細資訊，請參閱 eksctl 文件中的從組態檔案建立節點群組和組態檔案結構描述。將 *region-code* 取代為您的叢集所在的 AWS 區域。

```
cat >ubuntu.yaml <<EOF
---
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: my-cluster
  region: region-code
  version: '1.33'

iam:
  withOIDC: true

nodeGroups:
  - name: ng-ubuntu
    instanceType: m5.large
    desiredCapacity: 3
    amiFamily: Ubuntu2204
    iam:
       attachPolicyARNs:
          - arn:aws: iam::aws:policy/AmazonEKSWorkerNodePolicy
          - arn:aws: iam::aws:policy/AmazonEC2ContainerRegistryReadOnly
          - arn:aws: iam::aws:policy/AmazonSSMManagedInstanceCore
          - arn:aws: iam::aws:policy/AmazonEKS_CNI_Policy
    ssh:
        allow: true
        publicKeyName: my-ec2-keypair-name
EOF
```

若要建立 Ubuntu Pro 節點群組，只需將 amiFamily值變更為 UbuntuPro2204即可。

2. 使用下列命令部署節點。

```
eksctl create nodegroup --config-file=ubuntu.yaml
```

範例輸出如下。

建立節點時，會有數行輸出。輸出的最後幾行之一類似於以下的範例行。

```
[#]  created 1 nodegroup(s) in cluster "my-cluster"
```

3. （選用）部署範例應用程式以測試 Ubuntu 節點。
4. 如果符合下列條件，我們建議封鎖 Pod 存取 IMDS：
   - 您計劃將 IAM 角色指派給所有 Kubernetes 服務帳戶，讓 Pod 僅擁有所需的最低許可。
   - 叢集中沒有任何 Pod 因其他原因需要存取 Amazon EC2 執行個體中繼資料服務 (IMDS)，例如擷取目前 AWS 區域。

   如需詳細資訊，請參閱限制存取指派給工作節點的執行個體設定檔。

更新叢集的自我管理節點

全新 Amazon EKS 最佳化 AMI 發行時，請考量以新 AMI 取代自我管理節點群組中的節點。同樣地，如果更新 Amazon EKS 叢集的 Kubernetes 版本，請更新節點以使用具相同 Kubernetes 版本的節點。

> **Important**
>
> 本主題涵蓋自我管理節點的節點更新。如果您使用的是受管節點群組，請參閱 the section called “更新”。

有兩種基本方法可以將叢集中的自我管理節點群組更新成使用新的 AMI：

將應用程式遷移至新的節點群組

建立新的節點群組，並將您的 Pod 遷移至該群組。遷移到新的節點群組比僅更新現有 AWS CloudFormation 堆疊中的 AMI ID 更優雅。這是因為遷移程序會將舊節點群組污點為 ，NoSchedule並在新堆疊準備好接受現有的 Pod 工作負載後耗盡節點。

更新 an AWS CloudFormation 節點堆疊

更新現有節點群組的 AWS CloudFormation 堆疊，以使用新的 AMI。使用 建立的節點群組不支援此方法eksctl。

將應用程式遷移至新的節點群組

此主題會說明如何建立新的節點群組，將現有的應用程式從容遷移至新群組，接著從叢集移除舊的節點群組。您可以使用 `eksctl` 或 AWS Management Console遷移至新的節點群組。

- the section called “eksctl”
- the section called “AWS Management Console 和 AWS CLI”

eksctl

使用 將應用程式遷移至新的節點群組 **eksctl**

如需使用 eksctl 進行遷移的詳細資訊，請參閱 eksctl 文件中的未受管節點群組。

此程序需要 eksctl 版本 0.210.0 或更新版本。您可使用以下命令檢查您的版本：

```
eksctl version
```

如需有關安裝或更新 eksctl 的指示，請參閱 eksctl 文件中的 Installation 一節。

> **Note**
>
> 此程序只適用於使用 eksctl 所建立的叢集和節點群組。

1. 擷取現有節點群組的名稱，將 *my-cluster* 取代為您的叢集名稱。

   ```
   eksctl get nodegroups --cluster=my-cluster
   ```

   範例輸出如下。

   ```
   CLUSTER      NODEGROUP          CREATED               MIN SIZE      MAX SIZE
    DESIRED CAPACITY     INSTANCE TYPE     IMAGE ID
   default      standard-nodes   2019-05-01T22:26:58Z  1             4             3
                  t3.medium         ami-05a71d034119ffc12
   ```

2. 使用以下命令啟動具備 eksctl 的新節點群組。在 命令中，將每個###取代為您自己的值。版本編號不能晚於控制平面的 Kubernetes 版本。此外，它不能比控制平面的 Kubernetes 版本早兩個以上的次要版本。建議使用與控制平面相同的版本。

   如果符合下列條件，我們建議封鎖 Pod 存取 IMDS：

   - 您計劃將 IAM 角色指派給所有 Kubernetes 服務帳戶，讓 Pod 僅擁有所需的最低許可。
   - 叢集中沒有任何 Pod 因其他原因需要存取 Amazon EC2 執行個體中繼資料服務 (IMDS)，例如擷取目前 AWS 區域。

如需詳細資訊，請參閱限制存取指派給工作節點的執行個體設定檔。

若要封鎖對 IMDS 的 Pod 存取，請將 `--disable-pod-imds`選項新增至下列命令。

> **Note**
>
> 如需更多可用旗標及其描述的資訊，請參閱 https://eksctl.io/。

```
eksctl create nodegroup \
  --cluster my-cluster \
  --version 1.33 \
  --name standard-nodes-new \
  --node-type t3.medium \
  --nodes 3 \
  --nodes-min 1 \
  --nodes-max 4 \
  --managed=false
```

3. 之前的命令完成時，使用下列命令來確認所有節點已達到 `Ready` 狀態：

```
kubectl get nodes
```

4. 使用下列命令來刪除原始的節點群組。在 命令中，將每個###取代為您的叢集和節點群組名稱：

```
eksctl delete nodegroup --cluster my-cluster --name standard-nodes-old
```

AWS Management Console 和 AWS CLI

使用 AWS Management Console 和 AWS CLI 將應用程式遷移至新的節點群組

1. 按照建立自我管理 Amazon Linux 節點中概述的步驟啟動新的節點群組。
2. 當堆疊已完成建立時，從主控台將其選取，然後選擇 Outputs (輸出)。
3. 記錄已建立節點群組的 NodeInstanceRole。您需要這樣才能新增 Amazon EKS 節點至叢集。

> **Note**
>
> 如果已將任何額外的 IAM 政策連接至舊節點群組 IAM 角色，則請將相同政策連接至新節點群組 IAM 角色，以便在新群組維持該功能。例如，如果為 Kubernetes Cluster Autoscaler 新增許可，則適用此操作。

4. 更新兩個節點群組的安全群組，使其能夠互相通訊。如需詳細資訊，請參閱the section called “安全群組要求”。

   a. 記錄兩個節點群組的安全群組 ID。這在 AWS CloudFormation 堆疊輸出中顯示為 NodeSecurityGroup 值。

   您可以使用下列 AWS CLI 命令，從堆疊名稱取得安全群組 IDs。在這些命令中， `oldNodes`是舊版節點堆疊的 AWS CloudFormation 堆疊名稱，而 `newNodes`是您要遷移至的堆疊名稱。將每個`###`取代為您自己的值。

   ```
   oldNodes="old_node_CFN_stack_name"
   newNodes="new_node_CFN_stack_name"

   oldSecGroup=$(aws cloudformation describe-stack-resources --stack-name $oldNodes \
   --query 'StackResources[?
   ResourceType==`AWS::EC2::SecurityGroup`].PhysicalResourceId' \
   --output text)
   newSecGroup=$(aws cloudformation describe-stack-resources --stack-name $newNodes \
   --query 'StackResources[?
   ResourceType==`AWS::EC2::SecurityGroup`].PhysicalResourceId' \
   --output text)
   ```

   b. 新增輸入規則至每個節點安全群組，使其接受彼此的流量。

   下列 AWS CLI 命令會將傳入規則新增至每個安全群組，以允許來自其他安全群組的所有通訊協定上的所有流量。當您將工作負載遷移至新群組時，此組態可讓每個節點群組中的 Pod 彼此通訊。

   ```
   aws ec2 authorize-security-group-ingress --group-id $oldSecGroup \
   --source-group $newSecGroup --protocol -1
   aws ec2 authorize-security-group-ingress --group-id $newSecGroup \
   --source-group $oldSecGroup --protocol -1
   ```

5. 編輯 `aws-auth` configmap 以在 RBAC 中對應新的節點執行個體角色。

```
kubectl edit configmap -n kube-system aws-auth
```

為新節點群組新增 mapRoles 項目。

```
apiVersion: v1
data:
  mapRoles: |
    - rolearn: ARN of instance role (not instance profile)
      username: system:node:{{EC2PrivateDNSName}}
      groups:
        - system:bootstrappers
        - system:nodes>
    - rolearn: arn:aws: iam::111122223333:role/nodes-1-16-NodeInstanceRole-
U11V27W93CX5
      username: system:node:{{EC2PrivateDNSName}}
      groups:
        - system:bootstrappers
        - system:nodes
```

將*###### ########### ###### ARN* 取代為您在上一個步驟中記錄的 NodeInstanceRole 值。接著，儲存並關閉檔案以套用更新的 configmap。

6. 注意節點狀態並等待新的節點加入叢集和達到 Ready 狀態。

```
kubectl get nodes --watch
```

7. （選用）如果您使用的是 Kubernetes Cluster Autoscaler，請將部署縮減為零 (0) 個複本，以避免衝突的擴展動作。

```
kubectl scale deployments/cluster-autoscaler --replicas=0 -n kube-system
```

8. 使用以下命令污染每個您想要用 NoSchedule 移除的節點。這樣就不會在您要取代的節點上排程或重新排程新的 Pod。如需詳細資訊，請參閱 Kubernetes 文件中的污點和容錯。

```
kubectl taint nodes node_name key=value:NoSchedule
```

如果您要將節點升級至新的 Kubernetes 版本，您可以使用下列程式碼片段來識別特定 Kubernetes 版本（在此案例中為 1.31) 的所有節點並進行污點。版本編號不能晚於控制平面的 Kubernetes 版

本。它也不能比控制平面的 Kubernetes 版本早兩個以上的次要版本。建議使用與控制平面相同的版本。

```
K8S_VERSION=1.31
nodes=$(kubectl get nodes -o jsonpath="{.items[?(@.status.nodeInfo.kubeletVersion==
\"v$K8S_VERSION\")].metadata.name}")
for node in ${nodes[@]}
do
    echo "Tainting $node"
    kubectl taint nodes $node key=value:NoSchedule
done
```

9. 判斷叢集的 DNS 供應商。

```
kubectl get deployments -l k8s-app=kube-dns -n kube-system
```

範例輸出如下。此叢集使用 CoreDNS 進行 DNS 解析，但您的叢集可以kube-dns改為傳回）：

```
NAME      DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
coredns   1         1         1            1           31m
```

10如果您目前的部署執行少於 2 個複本，請將部署擴增為 2 個複本。kubedns 如果您先前的命令輸出傳回 *coredns*，請將 coredns 取代為 。

```
kubectl scale deployments/coredns --replicas=2 -n kube-system
```

11使用以下命令清空您想要從叢集移除的每個節點：

```
kubectl drain node_name --ignore-daemonsets --delete-local-data
```

如果您要將節點升級至新的 Kubernetes 版本，請使用下列程式碼片段識別並耗盡特定 Kubernetes 版本（在本例中為 *1.31*) 的所有節點。

```
K8S_VERSION=1.31
nodes=$(kubectl get nodes -o jsonpath="{.items[?(@.status.nodeInfo.kubeletVersion==
\"v$K8S_VERSION\")].metadata.name}")
for node in ${nodes[@]}
do
    echo "Draining $node"
    kubectl drain $node --ignore-daemonsets --delete-local-data
```

```
done
```

12.在舊的節點完成消耗後，請撤銷您先前授權的安全群組傳入規則。然後，刪除 AWS CloudFormation 堆疊以終止執行個體。

> **Note**
>
> 如果您將任何其他 IAM 政策連接至舊節點群組 IAM 角色，例如新增 Kubernetes Cluster Autoscaler 的許可，請先從角色分離這些其他政策，然後才能刪除 AWS CloudFormation 堆疊。

a. 撤銷您先前為節點安全群組建立的傳入規則。在這些命令中，`oldNodes`是舊版節點堆疊的 AWS CloudFormation 堆疊名稱，而 `newNodes`是您要遷移至的堆疊名稱。

```
oldNodes="old_node_CFN_stack_name"
newNodes="new_node_CFN_stack_name"

oldSecGroup=$(aws cloudformation describe-stack-resources --stack-name $oldNodes \
--query 'StackResources[?
ResourceType==`AWS::EC2::SecurityGroup`].PhysicalResourceId' \
--output text)
newSecGroup=$(aws cloudformation describe-stack-resources --stack-name $newNodes \
--query 'StackResources[?
ResourceType==`AWS::EC2::SecurityGroup`].PhysicalResourceId' \
--output text)
aws ec2 revoke-security-group-ingress --group-id $oldSecGroup \
--source-group $newSecGroup --protocol -1
aws ec2 revoke-security-group-ingress --group-id $newSecGroup \
--source-group $oldSecGroup --protocol -1
```

b. 開啟 AWS CloudFormation 主控台。

c. 選取舊的節點堆疊。

d. 選擇 刪除。

e. 在 Delete stack (刪除堆疊) 確認對話方塊中，選擇 Delete stack (刪除堆疊)。

13.編輯 `aws-auth` configmap 以從 RBAC 移除舊的節點執行個體角色。

```
kubectl edit configmap -n kube-system aws-auth
```

刪除舊節點群組的 mapRoles 項目。

```
apiVersion: v1
data:
  mapRoles: |
    - rolearn: arn:aws: iam::111122223333:role/nodes-1-16-NodeInstanceRole-
W70725MZQFF8
      username: system:node:{{EC2PrivateDNSName}}
      groups:
        - system:bootstrappers
        - system:nodes
    - rolearn: arn:aws: iam::111122223333:role/nodes-1-15-NodeInstanceRole-
U11V27W93CX5
      username: system:node:{{EC2PrivateDNSName}}
      groups:
        - system:bootstrappers
        - system:nodes>
```

儲存並關閉檔案以套用更新的 configmap。

14.(選用) 如果您使用的是 Kubernetes Cluster Autoscaler，請將部署縮減回 1 個複本。

> **Note**
>
> 您也必須適當標記新的 Auto Scaling 群組 (例如 k8s.io/cluster-autoscaler/enabled,k8s.io/cluster-autoscaler/my-cluster)，並更新 Cluster Autoscaler 部署的命令，以指向新標記的 Auto Scaling 群組。如需詳細資訊，請參閱 Cluster Autoscaler on AWS。

```
kubectl scale deployments/cluster-autoscaler --replicas=1 -n kube-system
```

15.（選用）確認您使用的是最新版本的 Kubernetes 專用 Amazon VPC CNI 外掛程式。您可能需要更新 CNI 版本，才能使用最新支援的執行個體類型。如需詳細資訊，請參閱the section called “Amazon VPC CNI”。

16.如果您的叢集使用 kube-dns進行 DNS 解析（請參閱[migrate-determine-dns-step]），請將kube-dns部署擴展為一個複本。

```
kubectl scale deployments/kube-dns --replicas=1 -n kube-system
```

更新 an AWS CloudFormation 節點堆疊

本主題說明如何使用新的 AMI 更新現有的 AWS CloudFormation 自我管理節點堆疊。您可以使用此程序在叢集更新後將節點更新至新版本的 Kubernetes。否則，您可以針對現有 Kubernetes 版本更新至最新的 Amazon EKS 最佳化 AMI。

> **Important**
>
> 本主題涵蓋自我管理節點的節點更新。如需搭配受管節點群組使用簡化節點生命週期的詳細資訊，請參閱 the section called “更新”。

最新的預設 Amazon EKS node AWS CloudFormation 範本已設定為在移除舊執行個體之前，在叢集中啟動具有新 AMI 的執行個體，一次一個。此組態可確保在滾動更新期間，您始終擁有叢集中 Auto Scaling 群組所需的作用中執行個體計數。

> **Note**
>
> 使用 建立的節點群組不支援此方法eksctl。如果透過 eksctl 建立叢集或節點群組，則請參閱 the section called “遷移”。

1. 判斷叢集的 DNS 供應商。

   ```
   kubectl get deployments -l k8s-app=kube-dns -n kube-system
   ```

   範例輸出如下。此叢集正在使用 CoreDNS 進行 DNS 解析，但您的叢集可能會kube-dns改為傳回。視您使用kubectl的 版本而定，您的輸出可能看起來不同。

   ```
   NAME      DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
   coredns   1         1         1            1           31m
   ```

2. 如果您目前的部署執行少於 2 個複本，請將部署擴增為 2 個複本。kube-dns 如果您先前的命令輸出傳回 *coredns### coredns* 取代為 。

   ```
   kubectl scale deployments/coredns --replicas=2 -n kube-system
   ```

3. （選用） 如果您使用的是 Kubernetes Cluster Autoscaler，請將部署縮減至零 (0) 個複本，以避免衝突的擴展動作。

```
kubectl scale deployments/cluster-autoscaler --replicas=0 -n kube-system
```

4. 判斷目前節點群組的執行個體類型和所需的執行個體計數。稍後當您更新群組的 AWS CloudFormation 範本時，您會輸入這些值。
   a. 前往 https://console.aws.amazon.com/ec2/ 開啟 Amazon EC2 主控台。
   b. 在左側導覽窗格中，選擇 Launch Configurations (啟動組態)，並記下現有節點啟動組態的執行個體類型。
   c. 在左側導覽窗格中，選擇 Auto Scaling Groups (Auto Scaling 群組)，並注意現有節點 Auto Scaling 群組的 Desired (所需) 執行個體計數。
5. 開啟 AWS CloudFormation 主控台。
6. 選取您的節點群組堆疊，然後選擇 Update (更新)。
7. 選取 Replace current template (取代目前的範本)，然後選取 Amazon S3 URL。
8. 對於 Amazon S3 URL，請將下列 URL 貼到文字區域，以確保您使用的是最新版本的 node AWS CloudFormation 範本。然後選擇 Next (下一步)：

```
https://s3.us-west-2.amazonaws.com/amazon-eks/cloudformation/2022-12-23/amazon-eks-nodegroup.yaml
```

9. 在 Specify stack details (指定堆疊詳細資訊) 頁面上，填寫下列參數，然後選擇 Next (下一步)：
   - NodeAutoScalingGroupDesiredCapacity：輸入在先前步驟中記錄的所需執行個體計數。或者，在堆疊更新時，輸入欲擴展的所需節點數量。
   - NodeAutoScalingGroupMaxSize：輸入節點 Auto Scaling 群組可以擴增的最大節點數。此值必須至少超過所需容量一個節點。這樣才能夠在更新期間執行節點滾動更新，而不必減少節點計數。
   - NodeInstanceType：選擇在先前步驟中記錄的執行個體類型。或者，為節點選擇不同的執行個體類型。在選擇不同的執行個體類型之前，請檢閱選擇最佳的 Amazon EC2 節點執行個體類型。每個 Amazon EC2 執行個體類型都支援最大數量的彈性網路介面 (網路介面)，且每個網路介面支援最大數量的 IP 地址。由於每個工作者節點和 Pod 都會指派自己的 IP 地址，因此請務必選擇執行個體類型，以支援您要在每個 Amazon EC2 節點上執行的 Pod 數量上限。如需執行個體類型支援的網路介面和 IP 地址數量清單，請參閱每種執行個體類型每個網路介面的 IP 地址。例如，`m5.large`執行個體類型支援工作者節點和 Pod 最多 30 個 IP 地址。

     > **Note**
     >
     > 最新版本的 Kubernetes 專用 Amazon VPC CNI 外掛程式支援的執行個體類型會顯示在 GitHub 上的 vpc_ip_resource_limit.go。您可能需要更新 Kubernetes 版本的 Amazon

VPC CNI 外掛程式，才能使用最新支援的執行個體類型。如需詳細資訊，請參閱the section called “Amazon VPC CNI”。

> **Important**
>
> 有些執行個體類型可能無法在所有 AWS 區域中使用。

- NodeImageIdSSMParam – 您要更新的 AMI ID 的 Amazon EC2 Systems Manager 參數。下列值使用適用於 Kubernetes 版本 的最新 Amazon EKS 最佳化 AMI1.33。

```
/aws/service/eks/optimized-ami/1.33/amazon-linux-2/recommended/image_id
```

  您可以使用相同 eks/latest/userguide/platform-versions.html【platform-version，type="documentation"】 取代 *1.33*。或者比控制平面上執行的 Kubernetes 版本早一個版本號的版本。建議您將節點保持在與控制平面相同的版本。您也可以將 *amazon-linux-2* 取代為不同的 AMI 類型。如需詳細資訊，請參閱the section called “取得最新的 IDs”。

> **Note**
>
> 使用 Amazon EC2 Systems Manager 參數可讓您在未來更新節點，而無需查詢和指定 AMI ID。如果您的 AWS CloudFormation 堆疊使用此值，任何堆疊更新一律會針對您指定的 Kubernetes 版本啟動最新的建議 Amazon EKS 最佳化 AMI。即使您未變更範本中的任何值，也是如此。

- NodeImageId：若要使用您自己的自訂 AMI，請輸入要使用的 AMI 的 ID。

> **Important**
>
> 此值會覆寫 NodeImageIdSSMParam 指定的任何值。如果您想要使用 NodeImageIdSSMParam 值，請確定 NodeImageId 的值為空白。

- DisableIMDSv1：在預設情況下，每個節點都支援執行個體中繼資料服務版本 1 (IMDSv1) 和 IMDSv2。不過，您可以停用 IMDSv1。如果您不希望節點群組中排程的任何節點或任何 Pod 使用 IMDSv1，請選取 true。如需 IMDS 的詳細資訊，請參閱設定執行個體中繼資料服務。如果您已為服務帳戶實作 IAM 角色，請直接將必要的許可指派給需要存取 AWS 服務的所有 Pod。如此一來，叢集中的 Pod 就不需要基於其他原因存取 IMDS，例如擷取目前 AWS 區域。然後，您也

可以針對不使用主機聯網的 Pod，停用對 IMDSv2 的存取。如需詳細資訊，請參閱限制存取指派給工作節點的執行個體設定檔。

10. (選用) 在 Options (選項) 頁面上，為堆疊資源加上標籤。選擇下一步。

11. 在 Review (檢閱) 頁面上檢閱您的資訊，確認該堆疊可建立 IAM 資源，然後選擇 Update stack (更新堆疊)。

> **Note**
>
> 叢集中每個節點的更新，都需要幾分鐘的時間。請等待所有節點更新完成再執行後續步驟。

12. 如果您叢集的 DNS 提供者是 kube-dns，請將kube-dns部署擴展為一個複本。

```
kubectl scale deployments/kube-dns --replicas=1 -n kube-system
```

13. (選用) 如果您使用的是 Kubernetes Cluster Autoscaler，請將部署縮減回您想要的複本數量。

```
kubectl scale deployments/cluster-autoscaler --replicas=1 -n kube-system
```

14. (選用) 確認您使用的是最新版本的 Kubernetes 專用 Amazon VPC CNI 外掛程式。您可能需要更新 Kubernetes 版本的 Amazon VPC CNI 外掛程式，才能使用最新支援的執行個體類型。如需詳細資訊，請參閱the section called “Amazon VPC CNI”。

使用 AWS Fargate 簡化運算管理

本主題討論如何使用 Amazon EKS 在 AWS Fargate 上執行 Kubernetes Pod。Fargate 是一種為容器提供隨需、適當大小運算容量的科技。使用 Fargate，您不需要自行佈建、設定或擴展虛擬機器群組，即可執行容器。您也不需要選擇伺服器類型、決定何時擴展節點群組，或最佳化叢集封裝。

您可以控制哪些 Pod 從 Fargate 開始，以及它們如何使用 Fargate 設定檔執行。Fargate 描述檔會定義為 Amazon EKS 叢集的一部分。Amazon EKS 使用 Kubernetes 提供的 AWS 上游可擴展模型建置的控制器，將 Kubernetes 與 Fargate 整合。這些控制器作為 Amazon EKS 受管 Kubernetes 控制平面的一部分執行，並負責將原生 Kubernetes Pod 排程到 Fargate。Fargate 控制器包括新的排程器，除了數個變換與驗證許可控制器之外，也會隨著預設 Kubernetes 排程器執行。當您啟動符合 Fargate 上執行條件的 Pod 時，叢集中執行的 Fargate 控制器會辨識、更新並將 Pod 排程到 Fargate。

本主題說明在 Fargate 上執行之 Pod 的不同元件，並指出搭配 Amazon EKS 使用 Fargate 的特殊考量。

AWS Fargate 考量事項

以下是在 Amazon EKS 上使用 Fargate 的考慮因素。

- 在 Fargate 上執行的每個 Pod 都有自己的運算界限。它們不會與其他 Pod 共用基礎核心、CPU 資源、記憶體資源或彈性網路界面。
- Network Load Balancer 和 Application Load Balancer (ALB) 可以僅與具有 IP 目標的 Fargate 搭配使用。如需詳細資訊，請參閱the section called “建立 Network Load Balancer”及the section called “應用程式負載平衡”。
- Fargate 的公開服務僅能在目標類型 IP 模式下執行，不能在節點 IP 模式下執行。若要檢查在受管節點上和在 Fargate 上執行的服務的連線狀態，建議透過服務名稱來連線。
- Pod 在排定在 Fargate 上執行時，必須符合 Fargate 設定檔。不符合 Fargate 描述檔的 Pod 可能會卡在 中`Pending`。如果存在相符的 Fargate 設定檔，您可以刪除已建立的待定 Pod，以將它們重新排程到 Fargate。
- Fargate 不支援協助程式集。如果您的應用程式需要協助程式，請重新設定該協助程式在 Pod 中做為附屬容器執行。
- Fargate 不支援特權容器。
- 在 Fargate 上執行`HostNetwork`的 Pod 無法在 Pod 資訊清單中指定 `HostPort`或 。
- 預設`nofile`和`nproc`軟性限制為 1024，而 Fargate Pod 的硬性限制為 65535。
- GPUs 目前無法在 Fargate 上使用。
- 在 Fargate 上執行的 Pod 僅支援私有子網路 （具有 AWS 服務的 NAT 閘道存取權，但不支援直接路由至網際網路閘道），因此叢集的 VPC 必須具有可用的私有子網路。如需沒有傳出網際網路存取權的叢集，請參閱 the section called “私有叢集”。
- 您可以使用調整 Pod 資源搭配 Vertical Pod Autoscaler 來設定 Fargate Pod 的 CPU 和記憶體初始正確大小，然後使用 Scale Pod 部署搭配 Horizontal Pod Autoscaler 來擴展這些 Pod。如果您希望 Vertical Pod Autoscaler 自動重新部署 Pod 到具有較大 CPU 和記憶體組合的 Fargate，請將 Vertical Pod Autoscaler 的 模式設定為 `Auto`或 `Recreate` ，以確保功能正確。如需詳細資訊，請參閱 GitHub 上的 Vertical Pod Autoscaler 文件。
- 您的 VPC 必須啟用 DNS 解析和 DNS 主機名稱。如需詳細資訊，請參閱檢視與更新 VPC 的 DNS 支援。
- Amazon EKS Fargate 透過在虛擬機器 (VM) 中隔離每個 Pod，為 Kubernetes 應用程式增加深度防禦功能。此 VM 界限可防止在容器逸出時存取其他 Pod 使用的主機型資源，這是攻擊容器化應用程式並存取容器外資源的常用方法。

使用 Amazon EKS 不會變更您在共同責任模型下的責任。您應該審慎考慮叢集安全性和控管控制項的組態。隔離應用程式最安全的方法是永遠在個別的叢集中執行。

- Fargate 描述檔支援從 VPC 次要 CIDR 區塊指定子網路。您可能想要指定次要 CIDR 區塊。這是因為子網路中可用的 IP 地址數量有限。因此，叢集中也可以建立有限數量的 Pod。透過對 Pod 使用不同的子網路，您可以增加可用 IP 地址的數量。如需詳細資訊，請參閱為 VPC 新增 IPv4 CIDR 區塊。
- 部署到 Fargate 節點的 Pod 無法使用 Amazon EC2 執行個體中繼資料服務 (IMDS)。如果您有部署到 Fargate 且需要 IAM 登入資料的 Pod，請使用服務帳戶的 IAM 角色將它們指派給您的 Pod。如果您的 Pod 需要存取透過 IMDS 提供的其他資訊，則必須將此資訊硬式編碼為 Pod 規格。這包括部署 Pod 的區域 AWS 或可用區域。
- 您無法將 Fargate Pod 部署至 AWS Outposts、 AWS Wavelength 或 AWS Local Zones。
- Amazon EKS 必須定期修補 Fargate Pod，以確保其安全。我們會以降低影響的方式嘗試更新，但有時候如果 Pod 未成功移出，則必須將其刪除。您可採取一些措施以盡可能地減少中斷。如需詳細資訊，請參閱the section called “作業系統修補事件”。
- 適用於 Amazon EKS 的 Amazon VPC CNI 外掛程式會安裝在 Fargate 叢集上。您無法為具有 Fargate 節點的 Amazon EKS 叢集使用替代 CNI 外掛程式。
- 在 Fargate 上執行的 Pod 會自動掛載 Amazon EFS 檔案系統，而不需要手動驅動程式安裝步驟。您無法搭配 Fargate 節點使用動態持久性磁碟區佈建，但您可以使用靜態佈建。
- Amazon EKS 不支援 Fargate Spot。
- 您無法將 Amazon EBS 磁碟區掛載至 Fargate Pod。
- 您可以在 Fargate 節點上執行 Amazon EBS CSI 控制器，但 Amazon EBS CSI 節點 DaemonSet 只能在 Amazon EC2 執行個體上執行。
- 標記 Kubernetes 任務`Completed`或 之後`Failed`，任務建立的 Pod 通常會繼續存在。此行為可讓您檢視日誌和結果，但使用 Fargate，如果您之後未清除任務，則會產生費用。

  若要在任務完成或失敗後自動刪除相關的 Pod，您可以使用time-to-live(TTL) 控制器指定時段。下列範例顯示`.spec.ttlSecondsAfterFinished`在您的任務資訊清單中指定 。

```
apiVersion: batch/v1
kind: Job
metadata:
  name: busybox
spec:
  template:
    spec:
```

```
    containers:
    - name: busybox
      image: busybox
      command: ["/bin/sh", "-c", "sleep 10"]
    restartPolicy: Never
ttlSecondsAfterFinished: 60 # <-- TTL controller
```

Fargate 比較表

條件	AWS Fargate
可以部署到 AWS Outpost	否
可以部署到 AWS Local Zone	否
可以執行需要 Windows 的容器	否
可以執行需要 Linux 的容器	是
可以執行需要 Inferentia 晶片的工作負載	否
可以執行需要 GPU 的工作負載	否
可以執行需要 Arm 處理器的工作負載	否
可以執行 AWS Bottlerocket	否
Pod 與其他 Pod 共用核心執行期環境	否 – 每個 Pod 都有一個專用核心
Pod 與其他 Pod 共用 CPU、記憶體、儲存和網路資源。	否 – 每個 Pod 都有專用資源，可以獨立調整大小以最大化資源使用率。
Pod 可以使用比 Pod 規格中請求更多的硬體和記憶體	否 – 不過，您可以使用較大的 vCPU 和記憶體組態重新部署 Pod。
必須部署和管理 Amazon EC2 執行個體	否
必須保護、維護和修補 Amazon EC2 執行個體的作業系統	否

條件	AWS Fargate
可以在部署節點時提供引導引數，例如額外的 kubelet 引數。	否
可以將 IP 地址從與指派給節點的 IP 地址不同的 CIDR 區塊指派給 Pod。	否
可以對節點執行 SSH	否 – 沒有節點主機作業系統可供 SSH 使用。
可以將自己的自訂 AMI 部署至節點	否
可以將您自己的自訂 CNI 部署至節點	否
必須自行更新節點 AMI	否
必須自行更新節點 Kubernetes 版本	否 – 您不管理節點。
可以搭配 Pod 使用 Amazon EBS 儲存	否
可以搭配 Pod 使用 Amazon EFS 儲存	是
可以使用 Amazon FSx for Lustre 儲存搭配 Pod	否
可以針對服務使用 Network Load Balancer	是，使用建立網路負載平衡器時
Pod 可以在公有子網路中執行	否
可以將不同的 VPC 安全群組指派給個別 Pod	是
可以執行 Kubernetes DaemonSets	否
Pod 資訊清單中`HostNetwork` 的支援`HostPort`和	否
AWS 區域可用性	部分 Amazon EKS 支援的區域
可以在 Amazon EC2 專用主機上執行容器	否

條件	AWS Fargate
定價	獨立 Fargate 記憶體和 CPU 組態的成本。每個 Pod 都有自己的成本。如需詳細資訊，請參閱 AWS Fargate 定價。

為您的叢集開始使用 AWS Fargate

本主題說明如何開始使用 Amazon EKS 叢集在 AWS Fargate 上執行 Pod。

如果使用 CIDR 區塊限制對叢集公有端點的存取，則建議您也啟用私有端點存取。如此一來，Fargate Pod 就可以與叢集通訊。若不啟用私有端點，您指定用於公有存取的 CIDR 區塊必須包含來自 VPC 的傳出來源。如需詳細資訊，請參閱the section called “叢集端點存取”。

先決條件

現有的叢集。如果您還沒有 Amazon EKS 叢集，請參閱 開始使用。

步驟 1：確保現有節點可以與 Fargate Pod 通訊

如果您使用的是沒有節點的新叢集，或是只有受管節點群組的叢集 （請參閱 the section called “受管節點群組”），您可以跳到 the section called “步驟 2：建立 Fargate Pod 執行角色”。

假設您使用的現有叢集已有與其相關聯的節點。請確定這些節點上的 Pod 可以與 Fargate 上執行的 Pod 自由通訊。在 Fargate 上執行的 Pod 會自動設定為針對與其相關聯的叢集使用叢集安全群組。確保叢集中的任何現有節點都可以傳送和接收來自叢集安全群組的流量。受管節點群組也會自動設定為使用叢集安全群組，因此您不需要修改或檢查它們是否有此相容性 （請參閱 the section called “受管節點群組”）。

對於使用 `eksctl`或 Amazon EKS Managed AWS CloudFormation 範本建立的現有節點群組，您可以將叢集安全群組手動新增至節點。您還可以修改節點群組的 Auto Scaling 群組啟動範本，從而將叢集安全群組連接至執行個體。如需詳細資訊，請參閱《Amazon VPC 使用者指南》中的變更執行個體的安全群組。

您可以在叢集的網路區段 AWS Management Console 下的 中，檢查叢集的安全群組。或者，您可以使用下列 CLI AWS 命令來執行此操作。使用此命令時，請以您叢集的名稱取代 `<my-cluster>`。

```
aws eks describe-cluster --name <my-cluster> --query
 cluster.resourcesVpcConfig.clusterSecurityGroupId
```

步驟 2：建立 Fargate Pod 執行角色

當您的叢集在 AWS Fargate 上建立 Pod 時，在 Fargate 基礎設施上執行的元件必須代表您呼叫 AWS APIs。Amazon EKS Pod 執行角色提供執行此操作的 IAM 許可。若要建立 AWS Fargate Pod 執行角色，請參閱 the section called “Pod 執行 IAM 角色”。

> **Note**
>
> 如果您`eksctl`使用 `--fargate`選項使用 建立叢集，您的叢集已有 Pod 執行角色，您可以在具有 模式的 IAM 主控台中找到該角色`eksctl-my-cluster-FargatePodExecutionRole-ABCDEFGHIJKL`。同樣地，如果您使用 `eksctl`建立 Fargate 設定檔，則 會在尚未`eksctl`建立 Pod 執行角色時建立 Pod 執行角色。

步驟 3：為您的叢集建立 Fargate 設定檔

在排程叢集中 Fargate 上執行的 Pod 之前，您必須定義 Fargate 設定檔，指定哪些 Pod 在啟動時使用 Fargate。如需詳細資訊，請參閱the section called “定義設定檔”。

> **Note**
>
> 如果您`eksctl`使用 `--fargate`選項使用 建立叢集，則已為您的叢集建立 Fargate 設定檔，其中包含 `kube-system`和 `default` 命名空間中所有 Pod 的選擇器。使用下列程序來為您想要搭配 Fargate 使用的任何其他命名空間建立 Fargate 描述檔。

您可以使用下列任一工具建立 Fargate 設定檔：

- the section called “`eksctl`”
- the section called “AWS Management Console”

`eksctl`

此程序需要 `eksctl` 版本 `0.210.0` 或更新版本。您可使用以下命令檢查您的版本：

```
eksctl version
```

如需有關安裝或更新 `eksctl` 的指示，請參閱 `eksctl` 文件中的 Installation 一節。

使用 建立 Fargate 設定檔 **eksctl**

使用以下 eksctl 命令建立您的 Fargate 設定檔，並以您自己的值取代每一個 <example value>。您需要指定命名空間。不過，不需要 --labels選項。

```
eksctl create fargateprofile \
    --cluster <my-cluster> \
    --name <my-fargate-profile> \
    --namespace <my-kubernetes-namespace> \
    --labels <key=value>
```

對於 <my-kubernetes-namespace> 和 <key=value> 標籤可以使用某些萬用字元。如需詳細資訊，請參閱the section called “Fargate 設定檔萬用字元”。

AWS Management Console

使用 建立 Fargate 設定檔 AWS Management Console

1. 開啟 Amazon EKS 主控台。
2. 選擇要為其建立 Fargate 設定檔的叢集。
3. 選擇 Compute (運算) 索引標籤。
4. 在 Fargate profiles (Fargate 設定檔) 下，選擇 Add Fargate profile (新增 Fargate 設定檔)。
5. 在 Configure Fargate profile (設定 Fargate 設定檔) 頁面上，執行以下操作：
   a. 對於 Name (名稱)，輸入您 Fargate 描述檔的名稱。名稱必須是唯一的。
   b. 針對 Pod 執行角色，選擇要搭配 Fargate 設定檔使用的 Pod 執行角色。只會顯示具有 eks-fargate-pods.amazonaws.com 服務委託人的 IAM 角色。如果您沒有看到列出的任何角色，則必須建立一個角色。如需詳細資訊，請參閱the section called “Pod 執行 IAM 角色”。
   c. 視需要修改選取的子網路。

      > **Note**
      > 在 Fargate 上執行的 Pod 僅支援私有子網路。

   d. 對於 Tags (標籤)，您可以選擇標記 Fargate 設定檔。這些標籤不會傳播到與設定檔相關聯的其他資源，例如 Pod。
   e. 選擇下一步。
6. 在設定 Pod 選擇頁面上，執行下列動作：
   a. 針對命名空間，輸入要符合 Pod 的命名空間。

- 您可以使用特定的命名空間進行比對，例如 kube-system 或 default。
- 您可以使用某些萬用字元 (例如 prod-*) 來比對多個命名空間 (例如 prod-deployment 和 prod-test)。如需詳細資訊，請參閱the section called “Fargate 設定檔萬用字元”。

b. (選用) 將 Kubernetes 標籤新增至選取器。特別是將它們新增至指定命名空間中 Pod 需要比對的 Pod。

- 您可以將標籤新增至infrastructure: fargate選擇器，以便只有具有 Kubernetes 標籤的指定命名空間中的 Pod infrastructure: fargate 符合選擇器。
- 您可以使用某些萬用字元 (例如 key?: value?) 來比對多個命名空間 (例如 keya: valuea 和 keyb: valueb)。如需詳細資訊，請參閱the section called “Fargate 設定檔萬用字元”。

c. 選擇下一步。

7. 在 Review and create (檢閱和建立) 頁面，檢閱您 Fargate 設定檔的資訊並選擇 Create (建立)。

步驟 4：更新 CoreDNS

CoreDNS 預設為在 Amazon EKS 叢集上的 Amazon EC2 基礎設施上執行。如果您只想在叢集中的 Fargate 上執行 Pod，請完成下列步驟。

> **Note**
>
> 如果透過 --fargate 選項使用 eksctl 建立叢集，您可以直接跳至 the section called “後續步驟”。

1. 使用下列命令建立 CoreDNS 的 Fargate 設定檔。<my-cluster> 將 取代為您的叢集名稱，<111122223333>將 取代為您的 帳戶 ID，<AmazonEKSFargatePodExecutionRole>將取代為您的 Pod 執行角色名稱，將 <000000000000000b>、 <000000000000000a>和 <000000000000000c>取代為您的私有子網路 IDs。如果您沒有 Pod 執行角色，您必須先建立一個 （請參閱 the section called “步驟 2：建立 Fargate Pod 執行角色”)。

> **Important**
>
> 角色 ARN 不能包含 以外的路徑/。例如，如果您的角色名稱是 development/apps/AmazonEKSFargatePodExecutionRole，則您必須在指定角色的 ARN 時將其變更為 AmazonEKSFargatePodExecutionRole。角色 ARN 的格式必須是 arn:aws:iam::<111122223333>:role/<AmazonEKSFargatePodExecutionRole>。

```
aws eks create-fargate-profile \
    --fargate-profile-name coredns \
    --cluster-name <my-cluster> \
    --pod-execution-role-arn arn:aws: iam::<111122223333>:role/
<AmazonEKSFargatePodExecutionRole> \
    --selectors namespace=kube-system,labels={k8s-app=kube-dns} \
    --subnets subnet-<000000000000000a> subnet-<000000000000000b> subnet-
<000000000000000c>
```

2. 觸發coredns部署的推展。

```
kubectl rollout restart -n kube-system deployment coredns
```

後續步驟

- 您可以透過以下工作流程開始遷移現有的應用程式，以在 Fargate 上執行。
  a. the section called “建立 Fargate 設定檔” 符合您應用程式的 Kubernetes 命名空間和 Kubernetes 標籤。
  b. 刪除並重新建立任何現有的 Pod，以便在 Fargate 上排程這些 Pod。修改 <namespace>和 <deployment-type>以更新您的特定 Pod。

  ```
  kubectl rollout restart -n <namespace> deployment <deployment-type>
  ```

- 部署 the section called “應用程式負載平衡”以允許在 Fargate 上執行之 Pod 的傳入物件。
- 您可以使用 the section called “Vertical Pod Autoscaler”來設定 Fargate Pod 的 CPU 和記憶體初始正確大小，然後使用 the section called “Horizontal Pod Autoscaler”來擴展這些 Pod。如果您希望 Vertical Pod Autoscaler 自動重新部署 Pod 到具有更高 CPU 和記憶體組合的 Fargate，請將 Vertical Pod Autoscaler 模式設定為 Auto或 Recreate。這是為了確認功能可以運作正確。如需詳細資訊，請參閱 GitHub 上的 Vertical Pod Autoscaler 文件。
- 請遵循以下說明設定 AWS Distro for OpenTelemetry (ADOT) 收集器，以便監控應用程式。

定義啟動時使用 AWS Fargate 的 Pod

在叢集中排程 Fargate 上的 Pod 之前，您必須定義至少一個 Fargate 設定檔，指定啟動時使用 Fargate 的 Pod。

身為管理員，您可以使用 Fargate 設定檔來宣告在 Fargate 上執行的 Pod。您可以透過設定檔的選取器來執行此操作。您最多可以將五個選取器新增至每個設定檔。每個選取器必須包含一個命名空間。選取器還可以包括標籤。標籤欄位由多個選用鍵/值對組成。與選取器相符的 Pod 會排程在 Fargate 上執行。比對 Pod 時，會使用命名空間和在選取器中指定的標籤。如果命名空間選擇器定義沒有標籤，Amazon EKS 會嘗試使用 設定檔，將在該命名空間中執行的所有 Pod 排程到 Fargate。如果to-be-scheduled Pod 符合 Fargate 設定檔中的任何選擇器，則該 Pod 會排程在 Fargate 上。

如果 Pod 符合多個 Fargate 設定檔，您可以透過將下列 Kubernetes 標籤新增至 Pod 規格來指定 Pod 使用的設定檔：`eks.amazonaws.com/fargate-profile: my-fargate-profile`。Pod 必須符合該設定檔中的選擇器，才能排程到 Fargate。Kubernetes 親和性/反親和性規則不適用，而且 Amazon EKS Fargate Pod 不需要。

建立 Fargate 設定檔時，您必須指定 Pod 執行角色。此執行角色適用於使用設定檔在 Fargate 基礎設施上執行的 Amazon EKS 元件。它會新增至叢集的 Kubernetes 角色型存取控制 (RBAC) 以進行授權。這樣一來，在 Fargate 基礎設施上執行的 `kubelet` 就可以註冊到您的 Amazon EKS 叢集，並作為節點出現在您的叢集中。Pod 執行角色也提供 Fargate 基礎設施的 IAM 許可，以允許對 Amazon ECR 映像儲存庫的讀取存取權。如需詳細資訊，請參閱the section called “Pod 執行 IAM 角色”。

Fargate 設定檔無法變更。不過，您可以建立新的已更新設定檔來取代現有設定檔，然後刪除原始的設定檔。

> **Note**
>
> 刪除設定檔時，使用 Fargate 設定檔執行的任何 Pod 都會停止並進入擱置狀態。

如果在叢集中的任何 Fargate 設定檔處於 `DELETING` 狀態，您必須等候該 Fargate 設定檔刪除後，才能在該叢集建立任何其他設定檔。

> **Note**
>
> Fargate 目前不支援 Kubernetes topologySpreadConstraints。

Amazon EKS 和 Fargate 會將 Pod 分散到 Fargate 設定檔中定義的每個子網路。不過，最終可能會產生不均勻的散佈結果。如果必須產生均勻的散佈結果，請使用兩個 Fargate 設定檔。即使在您想要部署兩個複本，但不想要任何停機時間的情況下，甚至分散也很重要。我們建議每個設定檔只定義一個子網路。

Fargate 描述檔元件

下列元件包含在 Fargate 描述檔中。

Pod 執行角色

當您的叢集在 AWS Fargate 上建立 Pod `kubelet`時，在 Fargate 基礎設施上執行的 必須代表您呼叫 AWS APIs。例如，它必須透過呼叫從 Amazon ECR 提取容器映像。Amazon EKS Pod 執行角色提供執行此操作的 IAM 許可。

建立 Fargate 設定檔時，您必須指定要與 Pod 搭配使用的 Pod 執行角色。此角色會新增至叢集的 Kubernetes 角色型存取控制 (RBAC) 以進行授權。如此一來`kubelet`，在 Fargate 基礎設施上執行的 就可以向 Amazon EKS 叢集註冊，並顯示為節點。如需詳細資訊，請參閱the section called “Pod 執行 IAM 角色”。

子網路

使用此設定檔在 中啟動 Pod 的子網路 IDs。目前，在 Fargate 上執行的 Pod 不會指派公有 IP 地址。因此，此參數僅接受私有子網路 (不具網際網路閘道的直接路由)。

選取器

Pod 使用此 Fargate 設定檔時要比對的選取器。您最多可在一個 Fargate 設定檔中指定五個選取器。選取器具有下列元件：

- 命名空間：您必須指定選取器的命名空間。選擇器僅符合在此命名空間中建立的 Pod。不過，您可以建立多個選取器來定義多個命名空間。
- 標籤：您可以選擇指定符合選取器的 Kubernetes 標籤。選擇器只比對具有選擇器中指定之所有標籤的 Pod。

Fargate 設定檔萬用字元

除了 Kubernetes 允許的字元之外，您還可以在命名空間、標籤索引鍵`*`和標籤值的選擇器條件`?`中使用和 ：

- `*` 代表無、一個或多個字元。例如，`prod*` 可以代表 `prod` 和 `prod-metrics`。
- `?` 代表單一字元 (例如，`value?` 可以代表 `valuea`)。不過，它不能代表 `value` 和 `value-a`，因為 `?` 只能代表確切一個字元。

這些萬用字元適用於任何位置和組合 (例如，`prod*`、`*dev`，以及 `frontend*?`)。其他萬用字元和模式比對形式 (例如規則運算式) 則不支援。

如果 Pod 規格中有多個命名空間和標籤相符的設定檔，Fargate 會根據設定檔名稱的英數字元排序來挑選設定檔。例如，如果設定檔 A （名稱為 beta-workload) 和設定檔 B （名稱為 prod-workload) 都具有要啟動的 Pod 的相符選擇器，則 Fargate 會為 Pod 挑選設定檔 A (beta-workload)。Pod 在 Pod 上具有設定檔 A 的標籤 （例如，eks.amazonaws.com/fargate-profile=beta-workload)。

如果您想要將現有的 Fargate Pod 遷移到使用萬用字元的新設定檔，有兩種方法可以執行此操作：

- 使用相符的選取器建立新的設定檔，然後刪除舊設定檔。標有舊設定檔的 Pod 會重新排程為新的相符設定檔。
- 如果您想要遷移工作負載，但不確定每個 Fargate Pod 上有哪些 Fargate 標籤，您可以使用下列方法。建立新的設定檔，其名稱在同一叢集上的設定檔中按英數字元順序排列在第一位。然後，回收需要遷移至新設定檔的 Fargate Pod。

建立 Fargate 設定檔

本節說明如何建立 Fargate 設定檔。您還必須已建立 Pod 執行角色，以用於 Fargate 設定檔。如需詳細資訊，請參閱the section called “Pod 執行 IAM 角色”。在 Fargate 上執行的 Pod 僅支援具有 NAT 閘道存取 AWS 服務的私有子網路，但不支援直接路由至網際網路閘道。這樣您叢集的 VPC 必須具有可用的私有子網路。

您可以使用下列項目建立設定檔：

- the section called “eksctl”
- the section called “AWS Management Console”

eksctl

使用 建立 Fargate 設定檔 **eksctl**

使用下列eksctl命令建立 Fargate 設定檔，以您自己的值取代每個範例值。您必須指定命名空間。不過， --labels選項並非必要項目。

```
eksctl create fargateprofile \
    --cluster my-cluster \
    --name my-fargate-profile \
    --namespace my-kubernetes-namespace \
    --labels key=value
```

對於 my-kubernetes-namespace 和 key=value 標籤可以使用某些萬用字元。如需詳細資訊，請參閱the section called “Fargate 設定檔萬用字元”。

AWS Management Console

使用 建立 Fargate 設定檔 AWS Management Console

1. 開啟 Amazon EKS 主控台。
2. 選擇要為其建立 Fargate 設定檔的叢集。
3. 選擇 Compute (運算) 索引標籤。
4. 在 Fargate profiles (Fargate 設定檔) 下，選擇 Add Fargate profile (新增 Fargate 設定檔)。
5. 在 Configure Fargate profile (設定 Fargate 設定檔) 頁面上，執行以下作業：
   a. 對於 Name (名稱)，輸入 Fargate 設定檔的唯一名稱，例如 my-profile。
   b. 針對 Pod 執行角色，選擇要搭配 Fargate 設定檔使用的 Pod 執行角色。只會顯示具有 eks-fargate-pods.amazonaws.com 服務委託人的 IAM 角色。如果您沒有看到任何列出的角色，則必須建立一個角色。如需詳細資訊，請參閱the section called “Pod 執行 IAM 角色”。
   c. 視需要修改選取的子網路。

      > **Note**
      >
      > 在 Fargate 上執行的 Pod 僅支援私有子網路。

   d. 對於 Tags (標籤)，您可以選擇標記 Fargate 設定檔。這些標籤不會傳播到與設定檔相關聯的其他資源，例如 Pod。
   e. 選擇 Next (下一步)。
6. 在設定 Pod 選擇頁面上，執行下列動作：
   a. 針對命名空間，輸入要比對的 Pod 命名空間。
      - 您可以使用特定的命名空間進行比對，例如 kube-system 或 default。
      - 您可以使用某些萬用字元 (例如 prod-*) 來比對多個命名空間 (例如 prod-deployment 和 prod-test)。如需詳細資訊，請參閱the section called “Fargate 設定檔萬用字元”。
   b. (選用) 將 Kubernetes 標籤新增至選取器。具體而言，將它們新增至指定命名空間中的 Pod 需要比對的 Pod。
      - 您可以將標籤新增至infrastructure: fargate選取器，以便只有指定命名空間中也有 Kubernetes infrastructure: fargate 標籤的 Pod 符合選取器。

- 您可以使用某些萬用字元 (例如 `key?: value?`) 來比對多個命名空間 (例如 `keya: valuea` 和 `keyb: valueb`)。如需詳細資訊，請參閱the section called “Fargate 設定檔萬用字元”。

c. 選擇 Next (下一步)。

7. 在 Review and create (檢閱和建立) 頁面，檢閱您 Fargate 設定檔的資訊並選擇 Create (建立)。

刪除 Fargate 設定檔

本主題說明如何刪除 Fargate 設定檔。當您刪除 Fargate 設定檔時，任何已排程至 Fargate 且具有該設定檔的 Pod 都會遭到刪除。如果這些 Pod 符合另一個 Fargate 設定檔，則會在 Fargate 上與該設定檔一起排程。如果它們不再符合任何 Fargate 描述檔，則不會排程到 Fargate，並且可能會保持待定狀態。

一個叢集中一次只能有一個 Fargate 描述檔可以處於 `DELETING` 狀態。請等候 Fargate 描述檔完成刪除動作，才可以刪除該叢集中的任何其他描述檔。

您可以使用下列任何工具刪除設定檔：

- the section called “`eksctl`”
- the section called “AWS Management Console”
- the section called “AWS CLI”

`eksctl`

使用 刪除 Fargate 設定檔 **`eksctl`**

使用下列命令從叢集刪除設定檔。將每個###取代為您自己的值。

```
eksctl delete fargateprofile  --name my-profile --cluster my-cluster
```

AWS Management Console

使用 刪除 Fargate 設定檔 AWS Management Console

1. 開啟 Amazon EKS 主控台。
2. 在左側導覽窗格中選擇叢集。在叢集清單中，選擇您要從中刪除 Fargate 設定檔的叢集。
3. 選擇 Compute (運算) 索引標籤。

4. 選擇要刪除的 Fargate 設定檔，然後選擇 Delete (刪除)。
5. 在 Delete Fargate profile (刪除 Fargate 設定檔) 頁面上，輸入設定檔名稱，然後選擇 Delete (刪除)。

AWS CLI

使用 CLI 刪除 Fargate AWS 設定檔

使用下列命令從叢集刪除設定檔。將每個###取代為您自己的值。

```
aws eks delete-fargate-profile --fargate-profile-name my-profile --cluster-name my-cluster
```

了解 Fargate Pod 組態詳細資訊

本節說明在 AWS Fargate 上執行 Kubernetes Pod 的一些唯一 Pod 組態詳細資訊。

Pod CPU 和記憶體

使用 Kubernetes，您可以定義請求、最低 vCPU 數量，以及配置給 Pod 中每個容器的記憶體資源。Kubernetes 會排程 Pod，以確保運算資源上至少有每個 Pod 的請求資源可用。如需詳細資訊，請參閱 Kubernetes 文件中的管理容器的運算資源。

> **Note**
>
> 由於 Amazon EKS Fargate 每個節點只執行一個 Pod，因此不會在資源較少的情況下移出 Pod。所有 Amazon EKS Fargate Pod 都以保證的優先順序執行，因此請求的 CPU 和記憶體必須等於所有容器的限制。如需詳細資訊，請參閱《Kubernetes 說明文件》中的設定 Pod 的服務品質。

在 Fargate 上排程 Pod 時，Pod 規格內的 vCPU 和記憶體保留會決定要為 Pod 佈建多少 CPU 和記憶體。

- 任何 Init 容器的最大請求用於判斷 Init 請求 vCPU 和記憶體需求。
- 所有長期執行容器的請求會加總，以判斷長期執行請求 vCPU 和記憶體需求。
- 針對 vCPU 和記憶體請求選擇前面兩個值中較大的值，以用於您的 Pod。

- Fargate 為每個 Pod 的記憶體保留新增了 256 MB，以用於所需的 Kubernetes 元件 (`kubelet`、`kube-proxy`和 `containerd`)。

Fargate 四捨五入至下列運算組態，最符合 vCPU 和記憶體請求的總和，以確保 Pod 永遠擁有執行所需的資源。

如果您未指定 vCPU 和記憶體組合，則會使用最小的可用組合 (.25 vCPU 和 0.5 GB 記憶體）。

下表顯示可在 Fargate 上執行的 Pod 使用的 vCPU 和記憶體組合。

vCPU 數值	記憶體數值
.25 vCPU	0.5 GB、1 GB、2 GB
.5 vCPU	1 GB、2 GB、3 GB、4 GB
1 vCPU	2 GB、3 GB、4 GB、5 GB、6 GB、7 GB、8 GB
2 vCPU	介於 4 GB 與 16 GB 之間，以 1 GB 為單位遞增
4 vCPU	介於 8 GB 與 30 GB 之間，以 1 GB 為單位遞增
8 vCPU	介於 16 GB 與 60 GB 之間，以 4 GB 為單位遞增
16 vCPU	介於 32 GB 與 120 GB 之間，以 8 GB 為單位遞增

為 Kubernetes 元件預訂的額外記憶體，可能會導致 Fargate 任務佈建了超出原先請求數量的 vCPU。例如，對 1 個 vCPU 和 8 GB 記憶體的請求將會新增 256 MB 至其記憶體請求，並且會佈建具有 2 個 vCPU 和 9 GB 記憶體的 Fargate 任務，因為沒有任何具有 1 個 vCPU 和 9 GB 記憶體的可用任務。

在 Fargate 上執行的 Pod 大小與 Kubernetes 與 報告的節點大小之間沒有關聯`kubectl get nodes`。報告的節點大小通常大於 Pod 的容量。您可以使用下列命令來驗證 Pod 容量。將`###`取代為您 Pod 的命名空間，並將 *`Pod-name`* 取代為您的 Pod 名稱。

```
kubectl describe pod --namespace default pod-name
```

範例輸出如下。

```
[...]
annotations:
    CapacityProvisioned: 0.25vCPU 0.5GB
[...]
```

`CapacityProvisioned` 註釋代表強制執行的 Pod 容量，並決定在 Fargate 上執行的 Pod 成本。如需運算組態的定價資訊，請參閱 AWS Fargate 定價。

Fargate 儲存

在 Fargate 上執行的 Pod 會自動掛載 Amazon EFS 檔案系統，而不需要手動驅動程式安裝步驟。您無法搭配 Fargate 節點使用動態持久性磁碟區佈建，但您可以使用靜態佈建。如需詳細資訊，請參閱 GitHub 上的 Amazon EFS CSI 驅動程式。

佈建時，在 Fargate 上執行的每個 Pod 都會收到預設的 20 GiB 暫時性儲存。Pod 停止後，會刪除此類型的儲存。在 Fargate 上啟動的新 Pod 預設會啟用暫時性儲存磁碟區的加密。暫時性 Pod 儲存會使用 AWS Fargate 受管金鑰，以 AES-256 加密演算法加密。

> **Note**
>
> 在 Fargate 上執行的 Amazon EKS Pod 預設可用儲存體小於 20 GiB。這是因為 `kubelet`和 Pod 內載入的其他 Kubernetes 模組會使用一些空間。

您可以增加暫時性儲存的總量，最多可達 175 GiB。若要使用 Kubernetes 設定大小，請指定 Pod 中每個容器`ephemeral-storage`的資源請求。當 Kubernetes 排程 Pod 時，可確保每個 Pod 的資源請求總和小於 Fargate 任務的容量。如需詳細資訊，請參閱 Kubernetes 文件中的 Pod 和容器的資源管理。

Amazon EKS Fargate 佈建的暫時性儲存空間比系統使用目的之要求更多。例如，100 GiB 的請求將佈建具有 115 GiB 暫時性儲存的 Fargate 任務。

設定 AWS Fargate 作業系統修補事件的動作

Amazon EKS 會定期修補 AWS Fargate 節點的作業系統，以確保其安全。作為修補程序的一部分，我們會回收節點以安裝作業系統修補程式。以對您的服務產生最小影響的方式嘗試更新。不過，如果未成功移出 Pod，有時必須刪除它們。您可以採取以下措施，以最大限度地減少潛在的中斷：

- 設定適當的 Pod 中斷預算 (PDBs)，以控制同時關閉的 Pod 數量。

- 建立 Amazon EventBridge 規則，在刪除 Pod 之前處理失敗的移出。
- 在您收到的通知中張貼的移出日期之前，手動重新啟動受影響的 Pod。
- 在 AWS 使用者通知中建立通知組態。

Amazon EKS 與 Kubernetes 社群緊密合作，以盡快提供錯誤修正和安全性修補程式。所有 Fargate Pod 都從最新的 Kubernetes 修補程式版本開始，可從 Amazon EKS 取得叢集的 Kubernetes 版本。如果您有具有較舊修補程式版本的 Pod，Amazon EKS 可能會回收它以將其更新為最新版本。這可確保您的 Pod 配備最新的安全性更新。如此一來，如果有重大的常見漏洞與暴露 (CVE) 問題，您就可以隨時掌握最新情況，以降低安全風險。

更新 AWS Fargate 作業系統時，Amazon EKS 會傳送通知給您，其中包含受影響的資源以及即將移出 Pod 的日期。如果提供的移出日期不方便，您可以選擇在通知中發佈的移出日期之前手動重新啟動受影響的 Pod。在您收到通知之前建立的任何 Pod 都會遭到移出。如需如何手動重新啟動 Pod 的進一步說明，請參閱 Kubernetes 文件。

若要限制當 Pod 回收時一次關閉的 Pod 數量，您可以設定 Pod 中斷預算 (PDBs)。您可以使用 PDB 根據每個應用程式的要求定義最低可用性，同時仍允許進行更新。PDB 的最低可用性必須小於 100%。如需詳細資訊，請參閱 Kubernetes 文件中的為您的應用程式指定中斷預算。

Amazon EKS 使用 Eviction API 安全地耗盡 Pod，同時遵守您為應用程式設定的 PDBs。可用區域將 Pod 移出，以最大限度地減少影響。如果移出成功，新的 Pod 會取得最新的修補程式，而且不需要進一步的動作。

當 Pod 的移出失敗時，Amazon EKS 會將事件傳送至您的帳戶，其中包含移出失敗的 Pod 詳細資訊。您可以在計劃終止時間之前對訊息執行操作。具體時間根據修補程式的緊迫性而有所不同。到了時，Amazon EKS 會再次嘗試移出 Pod。但是，這次如果移出失敗，則不會傳送新事件。如果移出再次失敗，您現有的 Pod 會定期刪除，以便新的 Pod 可以擁有最新的修補程式。

以下是 Pod 移出失敗時收到的範例事件。它包含有關叢集、Pod 名稱、Pod 命名空間、Fargate 設定檔和排程終止時間的詳細資訊。

```
{
    "version": "0",
    "id": "12345678-90ab-cdef-0123-4567890abcde",
    "detail-type": "EKS Fargate Pod Scheduled Termination",
    "source": "aws.eks",
    "account": "111122223333",
    "time": "2021-06-27T12:52:44Z",
    "region": "region-code",
```

```
    "resources": [
        "default/my-database-deployment"
    ],
    "detail": {
        "clusterName": "my-cluster",
        "fargateProfileName": "my-fargate-profile",
        "podName": "my-pod-name",
        "podNamespace": "default",
        "evictErrorMessage": "Cannot evict pod as it would violate the pod's disruption
 budget",
        "scheduledTerminationTime": "2021-06-30T12:52:44.832Z[UTC]"
    }
}
```

此外，擁有多個與 Pod 相關聯的 PDBs 可能會導致移出失敗事件。此事件傳回以下錯誤訊息。

```
"evictErrorMessage": "This pod has multiple PodDisruptionBudget, which the eviction
 subresource does not support",
```

您可以根據此事件建立所需的動作。例如，您可以調整 Pod 中斷預算 (PDB)，以控制如何移出 Pod。更具體地說，假設您從指定可用 Pod 目標百分比的 PDB 開始。在升級期間強制終止 Pod 之前，您可以將 PDB 調整為不同百分比的 Pod。若要接收此事件，您必須在叢集所屬 AWS 的帳戶和 AWS 區域中建立 Amazon EventBridge 規則。規則必須使用以下自訂模式。如需詳細資訊，請參閱《Amazon EventBridge 使用者指南》中的建立對事件做出反應的 Amazon EventBridge 規則。

```
{
  "source": ["aws.eks"],
  "detail-type": ["EKS Fargate Pod Scheduled Termination"]
}
```

可以為事件設定合適的目標以進行擷取。如需可用目標的完整清單，請參閱《Amazon EventBridge 使用者指南》中的 Amazon EventBridge 目標。您也可以在 AWS 使用者通知中建立通知組態。使用 AWS Management Console 建立通知時，請在事件規則下，為 AWS 服務名稱選擇 Elastic Kubernetes Service (EKS)，並為事件類型選擇 EKS Fargate Pod 排程終止。如需詳細資訊，請參閱《AWS 使用者通知使用者指南》中的使用者通知入門。 AWS

如需 EKS Pod 移除的常見問題，請參閱 re：Post 中的 FAQs Fargate Pod 移除通知。 AWS

收集 AWS Fargate 應用程式和用量指標

您可以收集 AWS Fargate 的系統指標和 CloudWatch 用量指標。

應用程式指標

對於在 Amazon EKS 和 AWS Fargate 上執行的應用程式，您可以使用 AWS Distro for OpenTelemetry (ADOT)。ADOT 允許您收集系統指標並將其傳送到 CloudWatch Container Insights 儀表板。若要在 Fargate 上執行的應用程式開始使用 ADOT，請參閱 ADOT 文件中的使用 CloudWatch Container Insights 搭配 AWS Distro for OpenTelemetry。

用量指標

您可以使用 CloudWatch 用量指標來提供帳戶資源用量的可見性。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 Fargate 的服務配額詳細資訊，請參閱 the section called “Service Quotas”。

AWS Fargate 會在 `AWS/Usage` 命名空間中發佈下列指標。

指標	描述
`ResourceCount`	您的帳戶中正在執行的特定資源總數。資源由與指標相關聯的維度定義。

下列維度用於精簡 AWS Fargate 發佈的使用指標。

維度	描述
`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。 目前， AWS Fargate 會傳回 Fargate 隨需用量的相關資訊。Fargate 隨需用量的資源值為 `OnDemand`。 【備註】 ====

維度	描述
	Fargate 隨需用量結合使用 Fargate 的 Amazon EKS Pod、使用 Fargate 啟動類型的 Amazon ECS 任務，以及使用FARGATE容量提供者的 Amazon ECS 任務。 ====
Class	正在追蹤的資源類別。目前， AWS Fargate 不使用類別維度。

建立 CloudWatch 警示來監控 Fargate 資源用量指標

AWS Fargate 提供對應至 Fargate 隨需資源用量 AWS 服務配額的 CloudWatch 用量指標。在 Service Quotas 主控台中，您可以在圖表上一目了然地查看使用情況。您也可以設定警示，在您的用量接近服務配額時發出警示。如需詳細資訊，請參閱the section called “收集指標”。

使用下列步驟，根據其中一個 Fargate 資源用量指標來建立 CloudWatch 警示。

1. 開啟 Service Quotas 主控台，網址為 https://console.aws.amazon.com/servicequotas/。
2. 在左側導覽窗格中，選擇 AWS 服務。
3. 從 AWS 服務清單中，搜尋並選取 AWS Fargate。
4. 在 Service quotas (服務配額) 清單中，選擇您要為其建立警示的 Fargate 用量配額。
5. 在 Amazon CloudWatch alarms (Amazon CloudWatch 警示) 區段中，選擇 Create (建立)。
6. 針對 Alarm threshold (警示閾值)，選擇您想要多少百分比的套用配額值設為警示值。
7. 針對 Alarm name (警示名稱)，輸入警示的名稱，然後選擇 Create (建立)。

為您的叢集啟動 AWS Fargate 記錄

Amazon EKS on Fargate 提供了基於 Fluent Bit 的內建日誌路由器。這表示您未明確執行 Fluent Bit 容器做為附屬項目，但 Amazon 會為您執行。您只需設定日誌路由器即可。組態會透過必須符合以下條件的專用 ConfigMap 生效：

- 名稱 aws-logging
- 在名為 aws-observability 的專用命名空間中建立

- 不能超過 5300 個字元。

建立 後ConfigMap，Amazon EKS on Fargate 會自動偵測它，並使用它設定日誌路由器。Fargate 使用 AWS 適用於 Fluent Bit 的 版本，Fluent Bit 是由 管理的上游合規分發 AWS。如需詳細資訊，請參閱 GitHub 上的適用於 Fluent Bit 的AWS。

日誌路由器可讓您在 使用廣泛的服務 AWS 進行日誌分析和儲存。您可以直接將日誌從 Fargate 串流至 Amazon CloudWatch 和 Amazon OpenSearch Service。您也可以透過 Amazon Data Firehose 將日誌串流到目的地，例如 Amazon S3、Amazon Amazon Kinesis Data Streams 和合作夥伴工具。https://aws.amazon.com/kinesis/data-firehose/

- 現有的 Fargate 設定檔，指定您部署 Fargate Pod 的現有 Kubernetes 命名空間。如需詳細資訊，請參閱 the section called “步驟 3：為您的叢集建立 Fargate 設定檔”。
- 現有的 Fargate Pod 執行角色。如需詳細資訊，請參閱 the section called “步驟 2：建立 Fargate Pod 執行角色”。

日誌路由器組態

> **Important**
>
> 若要成功發佈日誌，您的叢集所在的 VPC 必須具有對日誌目的地的網路存取權。這主要涉及使用者為其 VPC 自訂輸出規則。如需使用 CloudWatch 的範例，請參閱《Amazon CloudWatch Logs 使用者指南》中的將 CloudWatch Logs 與介面 VPC 端點搭配使用。Amazon CloudWatch

在下列步驟中，將每個###取代為您自己的值。

1. 建立名為 aws-observability 的專用 Kubernetes 命名空間。
   a. 將下列內容儲存到電腦上名為 aws-observability-namespace.yaml 的檔案中。name 的值必須為 aws-observability，且需要 aws-observability: enabled 標籤。

```
kind: Namespace
apiVersion: v1
metadata:
  name: aws-observability
  labels:
```

```
    aws-observability: enabled
```

b. 建立命名空間。

```
kubectl apply -f aws-observability-namespace.yaml
```

2. 建立具有 Fluent Conf 資料值的 ConfigMap，將容器日誌寄送至目的地。Fluent Conf 是 Fluent Bit，這是一種快速輕量型日誌處理器組態語言，用於將容器日誌路由到您選擇的日誌目的地。如需詳細資訊，請參閱 Fluent Bit 文件中的組態檔案。

> **Important**
>
> 一般 Fluent Conf 中包含的主要區段是 Service、Input、Filter 和 Output。然而，Fargate 日誌路由器僅接受：
>
> - Filter 和 Output 區段。
> - Parser 區段。
>
> 如果您提供任何其他區段，則這些區段將被拒絕。

Fargate 日誌路由器管理 Service 和 Input 區段。它具有以下Input區段，無法修改，也不需要在您的 中ConfigMap。但是，您可以從中取得洞察，例如記憶體緩衝區限制和套用於日誌的標籤。

```
[INPUT]
    Name tail
    Buffer_Max_Size 66KB
    DB /var/log/flb_kube.db
    Mem_Buf_Limit 45MB
    Path /var/log/containers/*.log
    Read_From_Head On
    Refresh_Interval 10
    Rotate_Wait 30
    Skip_Long_Lines On
    Tag kube.*
```

建立 ConfigMap 時，請考慮下列 Fargate 用來驗證欄位的規則：

- 應該在每個相應的索引鍵下指定 [FILTER]、[OUTPUT] 和 [PARSER]。例如，[FILTER] 必須位於 filters.conf 之下。您可以在 filters.conf 下有一或多個 [FILTER]。[OUTPUT] 和 [PARSER] 區段也應該位於其相應的索引鍵之下。透過指定多個 [OUTPUT] 區段中，您可以同時將日誌路由到不同的目的地。

- Fargate 驗證每個區段所需的索引鍵。Name 和 match 對於每個 [FILTER] 和 [OUTPUT] 都是必要項目。Name 和 format 對於每個 [PARSER] 都是必要項目。索引鍵不區分大小寫。
- ${ENV_VAR} 中不允許 等環境變數ConfigMap。
- 在每個 filters.conf、output.conf 和 parsers.conf 中每個指令或鍵值對的縮排必須是相同的。鍵值對必須比指令縮排更多。
- Fargate 會根據以下支援的篩選條件進行驗證：grep、parser、record_modifier、rewrite_tag、throttle、nest、modify 和 kubernetes。
- Fargate 會根據以下支援的輸出進行驗證：es、firehose、kinesis_firehose、cloudwatch、cloudwatch_logs 和 kinesis。
- 中至少必須提供一個支援的Output外掛程式ConfigMap，才能啟用記錄。 Filter 和 Parser 不需要啟用記錄。

您也可以使用所需的組態在 Amazon EC2 上執行 Fluent Bit，以對因驗證而產生的任何問題進行故障診斷。使用下列範例之一建立您的 ConfigMap。

> **Important**
>
> Amazon EKS Fargate 記錄不支援 的動態組態ConfigMap。對 的任何變更只會ConfigMap套用至新的 Pod。變更不會套用至現有的 Pod。

使用您所需的日誌目的地範例建立 ConfigMap。

> **Note**
>
> 您也可以將 Amazon Kinesis Data Streams 用於您的日誌目的地。如果您使用 Kinesis Data Streams，請確定 Pod 執行角色已獲 kinesis:PutRecords 許可。如需詳細資訊，請參閱 Fluent Bit：官方手冊中的 Amazon Kinesis Data Streams 許可。

Example

CloudWatch

使用 CloudWatch 時，您有兩個輸出選項：

- 用 C 編寫的輸出外掛程式

- 用 Golang 編寫的輸出外掛程式

以下範例顯示如何使用 cloudwatch_logs 外掛程式將日誌傳送至 CloudWatch。

a. 將下列內容儲存到名為 aws-logging-cloudwatch-configmap.yaml 的檔案中。將 *region-code* 取代為您的叢集所在的 AWS 區域。[OUTPUT] 下的參數為必要項目。

```
kind: ConfigMap
apiVersion: v1
metadata:
  name: aws-logging
  namespace: aws-observability
data:
  flb_log_cw: "false"  # Set to true to ship Fluent Bit process logs to
 CloudWatch.
  filters.conf: |
    [FILTER]
        Name parser
        Match *
        Key_name log
        Parser crio
    [FILTER]
        Name kubernetes
        Match kube.*
        Merge_Log On
        Keep_Log Off
        Buffer_Size 0
        Kube_Meta_Cache_TTL 300s
  output.conf: |
    [OUTPUT]
        Name cloudwatch_logs
        Match   kube.*
        region region-code
        log_group_name my-logs
        log_stream_prefix from-fluent-bit-
        log_retention_days 60
        auto_create_group true
  parsers.conf: |
    [PARSER]
        Name crio
        Format Regex
        Regex ^(?<time>[^ ]+) (?<stream>stdout|stderr) (?<logtag>P|F) (?
<log>.*)$
        Time_Key    time
```

```
        Time_Format %Y-%m-%dT%H:%M:%S.%L%z
```

b. 將清單檔案套用至叢集。

```
kubectl apply -f aws-logging-cloudwatch-configmap.yaml
```

Amazon OpenSearch Service

如果您想要將日誌傳送至 Amazon OpenSearch Service，您可以使用 es 輸出，這是以 C 撰寫的外掛程式。下列範例示範如何使用外掛程式將日誌傳送至 OpenSearch。

a. 將下列內容儲存到名為 `aws-logging-opensearch-configmap.yaml` 的檔案中。將每個*###*取代為您自己的值。

```
kind: ConfigMap
apiVersion: v1
metadata:
  name: aws-logging
  namespace: aws-observability
data:
  output.conf: |
    [OUTPUT]
      Name  es
      Match *
      Host  search-example-gjxdcilagiprbglqn42jsty66y.region-
code.es.amazonaws.com
      Port  443
      Index example
      Type  example_type
      AWS_Auth On
      AWS_Region region-code
      tls   On
```

b. 將清單檔案套用至叢集。

```
kubectl apply -f aws-logging-opensearch-configmap.yaml
```

Firehose

將日誌傳送至 Firehose 時，您有兩個輸出選項：

- kinesis_firehose – 以 C 撰寫的輸出外掛程式。
- firehose – 以 Golang 編寫的輸出外掛程式。

下列範例示範如何使用 kinesis_firehose 外掛程式將日誌傳送至 Firehose。

a. 將下列內容儲存到名為 aws-logging-firehose-configmap.yaml 的檔案中。將 *region-code* 取代為您的叢集所在的 AWS 區域。

```
kind: ConfigMap
apiVersion: v1
metadata:
  name: aws-logging
  namespace: aws-observability
data:
  output.conf: |
    [OUTPUT]
     Name  kinesis_firehose
     Match *
     region region-code
     delivery_stream my-stream-firehose
```

b. 將清單檔案套用至叢集。

```
kubectl apply -f aws-logging-firehose-configmap.yaml
```

3. 設定 Fargate Pod 執行角色的許可，將日誌傳送至目的地。

a. 將目的地的 IAM 政策下載至您的電腦。

Example

CloudWatch

將 CloudWatch IAM 政策下載到您的電腦。您也可以在 GitHub 上檢視政策。

```
curl -O https://raw.githubusercontent.com/aws-samples/amazon-eks-fluent-
logging-examples/mainline/examples/fargate/cloudwatchlogs/permissions.json
```

Amazon OpenSearch Service

將 OpenSearch IAM 政策下載到您的電腦。您也可以在 GitHub 上檢視政策。

```
curl -O https://raw.githubusercontent.com/aws-samples/amazon-eks-
fluent-logging-examples/mainline/examples/fargate/amazon-elasticsearch/
permissions.json
```

確認 OpenSearch Dashboards 的存取控制設定正確。OpenSearch Dashboards `all_access role`中的 需要映射 Fargate Pod 執行角色和 IAM 角色。必須為 `security_manager` 角色進行相同的映射。您可以新增先前的映射，方法是選取 `Menu`、`Security`、`Roles`，然後選取個別的角色。如需詳細資訊，請參閱如何對 CloudWatch Logs 進行故障診斷，以便將其串流至我的 Amazon ES 網域？。

Firehose

將 Firehose IAM 政策下載至您的電腦。您也可以在 GitHub 上檢視政策。

```
curl -O https://raw.githubusercontent.com/aws-samples/amazon-eks-fluent-
logging-examples/mainline/examples/fargate/kinesis-firehose/permissions.json
```

b. 從您下載的政策檔案建立 IAM 政策。

```
aws iam create-policy --policy-name eks-fargate-logging-policy --policy-document
 file://permissions.json
```

c. 使用下列命令，將 IAM 政策連接至為 Fargate 設定檔指定的 Pod 執行角色。使用您的帳戶 ID 取代 *111122223333*。將 *AmazonEKSFargatePodExecutionRole* 取代為您的 Pod 執行角色（如需詳細資訊，請參閱 the section called “步驟 2：建立 Fargate Pod 執行角色”）。

```
aws iam attach-role-policy \
  --policy-arn arn:aws: iam::111122223333:policy/eks-fargate-logging-policy \
  --role-name AmazonEKSFargatePodExecutionRole
```

Kubernetes 篩選條件支援

Fluent Bit Kubernetes 篩選條件可讓您將 Kubernetes 中繼資料新增至您的日誌檔案。如需有關篩選條件的相關資訊，請參閱 Fluent Bit 說明文件中的 Kubernetes。您可以使用 API 伺服器端點來套用篩選條件。

```
filters.conf: |
    [FILTER]
        Name                kubernetes
        Match               kube.*
        Merge_Log           On
        Buffer_Size         0
```

```
        Kube_Meta_Cache_TTL 300s
```

Important

- Kube_URL、Kube_CA_File、Kube_Token_Command 和 Kube_Token_File 是服務擁有的組態參數，您無法指定這些參數。Amazon EKS Fargate 會將這些值填入。
- Kube_Meta_Cache_TTL 是 Fluent Bit 等待直到其可以與 API 伺服器通訊，以取得最新中繼資料所需的時間。如果Kube_Meta_Cache_TTL未指定 ，Amazon EKS Fargate 會附加預設值 30 分鐘，以減少 API 伺服器的負載。

將 Fluent Bit 程序日誌運送到您的帳戶

您可以選擇使用下列 將 Fluent Bit 程序日誌運送到 Amazon CloudWatchConfigMap。將 Fluent Bit 處理日誌傳送至 CloudWatch 需要額外的記錄擷取和儲存費用。將 *region-code* 取代為您的叢集所在的 AWS 區域。

```
kind: ConfigMap
apiVersion: v1
metadata:
  name: aws-logging
  namespace: aws-observability
  labels:
data:
  # Configuration files: server, input, filters and output
  # ======================================================
  flb_log_cw: "true"  # Ships Fluent Bit process logs to CloudWatch.

  output.conf: |
    [OUTPUT]
        Name cloudwatch
        Match kube.*
        region region-code
        log_group_name fluent-bit-cloudwatch
        log_stream_prefix from-fluent-bit-
        auto_create_group true
```

日誌位於與叢集位於相同區域的 CloudWatch AWS 中。日誌群組的名稱是 *my-cluster*-fluent-bit-logs，而 Fluent Bit logstream 的名稱是 fluent-bit-*podname*-*pod-namespace* 。

> **Note**
>
> - 只有當 Fluent Bit 程序成功啟動時，才會傳送程序日誌。若在啟動 Fluent Bit 時失敗，程序日誌就會遺失。您只能將程序日誌傳送至 CloudWatch。
> - 若要對傳送程序日誌至您的帳戶除錯，您可以套用先前的 `ConfigMap` 來取得程序日誌。Fluent Bit 無法啟動通常是由於您的 `ConfigMap` 開始時無法被 Fluent Bit 剖析或接受。

停止運送 Fluent Bit 程序日誌

將 Fluent Bit 處理日誌傳送至 CloudWatch 需要額外的記錄擷取和儲存費用。若要排除現有 `ConfigMap` 設定中的處理日誌，請執行下列步驟。

1. 在啟用 Fargate 記錄之後，找到為 Amazon EKS 叢集的 Fluent Bit 程序日誌自動建立的 CloudWatch 日誌群組。它遵循格式 *`my-cluster`*`-fluent-bit-logs`。
2. 刪除針對 CloudWatch 日誌群組中每個 Pod 程序日誌建立的現有 CloudWatch 日誌串流。
3. 編輯 `ConfigMap` 和設定 `flb_log_cw: "false"`。
4. 重新啟動叢集中的任何現有 Pod。

測試應用程式

1. 部署範例 Pod。
   a. 將下列內容儲存到電腦上名為 `sample-app.yaml` 的檔案中。

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: sample-app
  namespace: same-namespace-as-your-fargate-profile
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
```

```
    spec:
      containers:
        - name: nginx
          image: nginx:latest
          ports:
            - name: http
              containerPort: 80
```

b. 將清單檔案套用至叢集。

```
kubectl apply -f sample-app.yaml
```

2. 使用您在 ConfigMap 中設定的目的地檢視 NGINX 日誌。

大小考量因素

建議您為日誌路由器規劃最多 50 MB 的記憶體。如果您希望應用程式以非常高的輸送量產生日誌，則應該規劃最多 100 MB。

故障診斷

若要確認記錄功能是否因為某些原因而啟用或停用，例如無效的 ConfigMap，以及無效的原因，請使用 檢查您的 Pod 事件`kubectl describe pod` *pod-name* 。輸出可能包含 Pod 事件，以釐清日誌記錄是否已啟用，例如下列範例輸出。

```
[...]
Annotations:          CapacityProvisioned: 0.25vCPU 0.5GB
                      Logging: LoggingDisabled: LOGGING_CONFIGMAP_NOT_FOUND
[...]
Events:
  Type     Reason           Age        From
              Message
  ----     ------           ----       ----
              -------
  Warning  LoggingDisabled  <unknown>  fargate-scheduler
               Disabled logging because aws-logging configmap was not found. configmap
 "aws-logging" not found
```

Pod 事件是暫時性的，其時段取決於設定。您也可以使用 檢視 Pod 的註釋`kubectl describe pod` *pod-name* 。在 Pod 註釋中，有記錄功能是否啟用或停用以及原因的相關資訊。

選擇最佳的 Amazon EC2 節點執行個體類型

Amazon EC2 為工作節點提供多種執行個體類型選擇。每個執行個體類型皆提供不同的運算、記憶體、儲存與網路功能。每個執行個體也會依照這些功能分組為執行個體系列。如需清單，請參閱《Amazon EC2 使用者指南》中的可用執行個體類型。Amazon EKS 發佈了多種 Amazon EC2 AMI 變體以啟用支援。若要確保您選取的執行個體類型與 Amazon EKS 相容，請考慮以下條件。

- 所有 Amazon EKS AMIs 目前都不支援 `mac` 系列。
- Arm 和非加速 Amazon EKS AMIs 不支援 `g3`、`inf`、 `g4`和 `p` 系列。
- 加速的 Amazon EKS AMIs 不支援 `a`、`c`、`m`、 `hpc`和 `t` 系列。
- 對於 Arm 型執行個體，Amazon Linux 2023 (AL2023) 僅支援使用 Graviton2 或更新版本處理器的執行個體類型。AL2023 不支援`A1`執行個體。

在 Amazon EKS 支援的執行個體類型之間進行選擇時，請考慮每種類型的以下功能。

節點群組中的執行個體數量

一般而言，越少越大的執行個體越好，特別是如果您有許多 Daemonset。每個執行個體都需要對 API 伺服器進行 API 呼叫，因此擁有的執行個體越多，API 伺服器上的負載就越大。

作業系統

檢閱 Linux、Windows 以及 Bottlerocket 支援的執行個體類型。在建立 Windows 執行個體之前，請檢閱在 EKS 叢集上部署 Windows 節點。

硬體架構

您需要 x86 或 Arm 嗎？ 部署 Arm 執行個體之前，請檢閱 Amazon EKS 最佳化 Arm Amazon Linux AMIs。您需要建置在 Nitro 系統 (Linux 或 Windows) 上的執行個體，還是具有加速功能的執行個體？ 如果您需要加速功能，則只能將 Linux 與 Amazon EKS 搭配使用。

Pod 數量上限

由於每個 Pod 都會指派自己的 IP 地址，因此執行個體類型支援的 IP 地址數量是決定可在執行個體上執行之 Pod 數量的因素。若要手動判斷執行個體類型支援的 Pod 數量，請參閱 the section called “Amazon EKS 為每種 Amazon EC2 執行個體類型建議 Pod 數量上限”。

> **Note**
>
> 如果您使用的是 或更新版本的 Amazon EKS 最佳化 Amazon Linux 2 AMI`v20220406`，則可以使用新的執行個體類型，而無需升級至最新的 AMI。對於這些 AMIs，如果 AMI 未列在

eni-max-pods.txt 檔案中，則 AMI 會自動計算必要的max-pods值。預設情況下，Amazon EKS 可能不支援目前處於預覽版中的執行個體類型。此類類型的 max-pods 值仍需新增至我們 AMI 中的 eni-max-pods.txt。

AWS Nitro System 執行個體類型可選擇性地支援比非 Nitro System 執行個體類型更多的 IP 地址。不過，並非所有為執行個體指派的 IP 地址都可供 Pod 使用。要為您的執行個體指派大量的 IP 地址，您必須在叢集中安裝並適當設定 Amazon VPC CNI 附加元件 1.9.0 或更新版本。如需詳細資訊，請參閱the section called “增加 IP 地址”。要將最大數量的 IP 地址指派給執行個體，您必須在叢集中安裝版本 1.10.1 或更新版本的 Amazon VPC CNI 附加元件，並使用 IPv6 系列部署叢集。

IP 系列

您可以在使用叢集的 IPv4 系列時使用任何支援的執行個體類型，這可讓您的叢集將私有IPv4地址指派給您的 Pod 和服務。但是，如果您想在叢集中使用 IPv6 系列，則必須使用 AWS Nitro 系統執行個體類型或裸機類型。Windows 執行個體僅支援 IPv4。您的叢集必須是執行版本 1.10.1 或更新版本 Amazon VPC CNI 附加元件的叢集。如需有關使用 IPv6 的詳細資訊，請參閱 the section called “IPv6”。

您正在執行的 Amazon VPC CNI 附加元件版本

最新版的 Kubernetes 專用 Amazon VPC CNI 外掛程式支援這些執行個體類型。您可能需要更新 Amazon VPC CNI 附加元件版本，才能利用最新支援的執行個體類型。如需詳細資訊，請參閱the section called “Amazon VPC CNI”。最新版本支援與 Amazon EKS 搭配使用的最新功能。舊版不支援所有功能。您可以在 GitHub 上的變更日誌檢視不同版本支援的功能。

AWS 您要在 中建立節點的區域

並非所有 AWS 區域都提供所有執行個體類型。

您是否使用 Pod 的安全群組

如果您使用 Pod 的安全群組，則僅支援特定執行個體類型。如需詳細資訊，請參閱the section called “Pod 的安全群組”。

Amazon EKS 為每種 Amazon EC2 執行個體類型建議 Pod 數量上限

由於每個 Pod 都會指派自己的 IP 地址，因此執行個體類型支援的 IP 地址數量是決定可在執行個體上執行之 Pod 數量的因素。Amazon EKS 提供一個可以下載並執行的指令碼，以便判斷 Amazon EKS 建議在每種執行個體類型上所能執行的 Pod 數量上限。該指令碼會使用每個執行個體的硬體屬性和組

態選項來判斷 Pod 數量上限。您可以使用這些步驟中傳回的數字來啟用功能，例如從與執行個體不同的子網路將 IP 地址指派給 Pod，以及大幅增加執行個體的 IP 地址數量。如果您使用的是具有多個執行個體類型的受管節點群組，請使用適用於所有執行個體類型的值。

1. 下載可用來計算每種執行個體類型的 Pod 數量上限的指令碼。

```
curl -O https://raw.githubusercontent.com/awslabs/amazon-eks-ami/master/templates/al2/runtime/max-pods-calculator.sh
```

2. 將指令碼標記為電腦上的可執行檔。

```
chmod +x max-pods-calculator.sh
```

3. 執行指令碼，將 *m5.large* 取代為您計劃部署的執行個體類型，並將 *1.9.0-eksbuild.1* 取代為您的 Amazon VPC CNI 附加元件版本。若要判斷您的附加元件版本，請參閱使用 Amazon VPC CNI 將 IPs 指派給 Pod 中的更新程序。

```
./max-pods-calculator.sh --instance-type m5.large --cni-version 1.9.0-eksbuild.1
```

   範例輸出如下。

```
29
```

   您可以在指令碼中新增下列選項，查看使用選用功能時支援的 Pod 數量上限。

   - `--cni-custom-networking-enabled` – 當您想要從與執行個體不同的子網路指派 IP 地址時，請使用此選項。如需詳細資訊，請參閱the section called “自訂聯網”。將此選項新增至具有相同範例值的前一個指令碼會產生 `20`。
   - `--cni-prefix-delegation-enabled`：想要為每個彈性網路介面指派更大量的 IP 地址時，請使用此選項。此功能需要在 Nitro 系統上執行的 Amazon Linux 執行個體，以及 Amazon VPC CNI 附加元件 `1.9.0` 版或更新版本。如需詳細資訊，請參閱 the section called “增加 IP 地址”。將此選項新增至具有相同範例值的前一個指令碼會產生 `110`。

您還可以使用 `--help` 選項執行指令碼，以查看所有可用選項。

> **Note**
>
> 最大 Pods 計算器指令碼`110`會根據 Kubernetes 可擴展性閾值和建議設定，將傳回值限制為 。如果您的執行個體類型有超過 30 個 vCPU，則此限制會跳轉至 `250`，這是根據內部

Amazon EKS 可擴展性團隊測試而定的數字。如需詳細資訊，請參閱 Amazon VPC CNI 外掛程式增加每節點的 Pod 限制部落格文章。

EKS Auto 模式的考量事項

EKS Auto Mode 會將節點上的 Pod 數量限制為以下數量的較低者：

- 110 個 Pod 硬蓋
- 上述最大 Pod 計算的結果。

使用預先建置的最佳化映像建立節點

當您使用受管節點群組或自我管理節點時，可以使用預先建置的 Amazon EKS 最佳化 Amazon Machine Image (AMIs) 或您自己的自訂 AMIs 部署節點。如果您正在執行混合節點，請參閱 the section called “準備作業系統”。如需有關各種 Amazon EKS 最佳化 AMI 類型的詳細資訊，請參閱下列其中一個主題。如需如何建立自己的自訂 AMI 的詳細資訊，請參閱 the section called “自訂組建”。

使用 Amazon EKS Auto Mode 時，EKS 會管理 EC2 執行個體，包括選取和更新 AMI。

主題

- EKS AL2 和 AL2-Accelerated AMIs轉換功能指南
- 使用最佳化的 Amazon Linux AMIs 建立節點
- 使用最佳化 Bottlerocket AMIs建立節點
- 使用最佳化的 Ubuntu Linux AMIs建立節點
- 使用最佳化 Windows AMIs建立節點

EKS AL2 和 AL2-Accelerated AMIs轉換功能指南

AWS 將於 20AL2-optimized和 AL2-accelerated AMIs 的支援。雖然您可以在end-of-support (EOS) 日期 (2025 年 11 月 26 日） 之後繼續使用 EKS AL2 AMIs，但 EKS 將不再發行任何新的 Kubernetes 版本或 AL2 AMIs 更新，包括此日期之後的次要版本、修補程式和錯誤修正。我們建議您升級至 Amazon Linux 2023 (AL2023) 或 Bottlerocket AMIs：

- AL2023 啟用secure-by-default方法，其中包含預先設定的安全政策、允許模式的 SELinux、預設啟用IMDSv2-only 模式、最佳化開機時間，以及增強安全性和效能的改善套件管理，非常適合需要大量自訂的基礎設施，例如直接作業系統層級存取或廣泛的節點變更。
- Bottlerocket 透過其專門建置的容器最佳化設計，實現增強的安全性、更快的開機時間和更小的攻擊面，以提高效率，非常適合採用最少節點自訂的容器原生方法。

或者，您可以the section called “自訂組建”直到 EOS 日期 (2025 年 11 月 26 日），或使用 Amazon Linux 2 基本執行個體建置自訂 AMI，直到 Amazon Linux 2 EOS 日期 (2026 年 6 月 30 日）。如需詳細資訊，請參閱 AL2023 FAQs、 Bottlerocket FAQs，或參閱 the section called “升級至 AL2023”或 the section called “Bottlerocket” 文件以取得詳細的遷移指引。

遷移和支援FAQs

如何從 AL2 遷移到 AL2023 AMI？

我們建議您建立並實作遷移計畫，其中包含完整的應用程式工作負載測試和文件化復原程序，然後遵循 EKS 官方文件中從 Amazon Linux 2 升級到 Amazon Linux 2023 的step-by-step說明。

我可以為 EKS 最佳化 AL2 AMI 建置超過 EKS end-of-support (EOS) 日期的自訂 AL2 AMIs 嗎？

雖然我們建議移至官方支援和發佈的 AL2023 或 Bottlerocket EKS 最佳化 AMIs，但您可以建置自訂 EKS AL2-optimized和 AL2-accelerated AMIs，直到 AL2 AMI EOS 日期 (2025 年 11 月 26 日）。或者，您可以使用 Amazon Linux 2 基本執行個體建置自訂 AMI，直到 Amazon Linux 2 EOS 日期 (2026 年 6 月 30 日）。如需建置自訂 EKS AL2-optimized和 AL2-accelerated AMI step-by-step說明，請參閱在 EKS 官方文件中建置自訂 Amazon Linux AMI。

EKS Kubernetes 版本支援政策是否適用於 Amazon Linux 發行版本？

否。EKS AL2-optimized和 AL2-accelerated AMIs 的 EOS 日期與 EKS 的 Kubernetes 版本標準和延長支援時間表無關。即使您使用 EKS 延伸支援，仍需要遷移至 AL2023 或 Bottlerocket。

從 cgroupv1 到 cgroupv2 的轉移如何影響我的遷移？

Kubernetes 社群將`cgroupv1`支援 (AL2 使用） 移至維護模式，這表示不會新增任何新功能，而且只會提供重大安全性和主要錯誤修正。若要在 Kubernetes `cgroupv2`中採用 ，您需要確保作業系統、核心、容器執行時間和 Kubernetes 元件的相容性。這需要`cgroupv2`預設啟用 的 Linux 發行版本，例如 AL2023、 Bottlerocket、Red Hat Enterprise Linux (RHEL) 9+、Ubuntu 22.04+ 或 Debian 11+。這些分發隨附核心版本 ≥5.8，這是 Kubernetes `cgroupv2`支援的最低需求。若要進一步了解，請參閱關於 cgroup v2。

如果我在自訂 AL2 AMI 中需要 Neuron，該怎麼辦？

您無法在以 AL2-based AMIs 上原生執行完整 Neuron 支援的應用程式。若要在 AL2 AMI 上利用 AWS Neuron，您必須使用 Neuron 支援的容器與non-AL2 Linux 發行版本（例如 Ubuntu 22.04、Amazon Linux 2023 等）來容器化應用程式，然後在已安裝 Neuron 驅動程式 (`aws-neuronx-dkms`) 的 AL2-based AMI 上部署這些容器。

相容性和版本

AL2 AMIs 支援的 Kubernetes 版本

Kubernetes 1.32 版是 Amazon EKS 將發行 AL2 (Amazon Linux 2) AMIs最後一個版本。對於支援高達 1.32 的 https://docs.aws.amazon.com/eks/latest/userguide/kubernetes-versions.html Kubernetes 版本，EKS 將繼續發行 AL2 AMIs(AL2_ARM_64、AL2_x86_64) 和 AL2-accelerated AMIs (AL2_x86_64_GPU)，直到 2025 年 11 月 26 日為止。在此日期之後，EKS 將停止發行所有 Kubernetes 版本的 AL2-optimized和 AL2-accelerated AMIs。請注意，EKS AL2-optimized和 AL2-accelerated AMIs 的 EOS 日期與 EKS 的 Kubernetes 版本標準和延長支援時間表無關。

AL2, AL2023和 Bottlerocket AMIs NVIDIA 驅動程式比較

驅動程式分支	Amazon Linux 2 AMI	Amazon Linux 2023 AMI	Bottlerocket AMI	End-of-Life
R535	不支援	不支援	支援	2027 年 9 月
R550	支援	支援	不支援	2025 年 4 月
R560	不支援	支援	不支援	2025 年 3 月
R570	不支援	支援	即將推出	2026 年 2 月

若要進一步了解，請參閱 Nvidia 版本文件。

AL2, AL2023和 Bottlerocket AMIs NVIDIA CUDA 版本比較

CUDA 版本	AL2 支援	AL2023 支援	Bottlerocket 支援
10.1	支援	不支援	不支援

CUDA 版本	AL2 支援	AL2023 支援	Bottlerocket 支援
11.8	支援	支援	支援
12.0	不支援	支援	支援
12.5	不支援	支援	支援

若要進一步了解，請參閱 CUDA 版本文件。

ALAL2, AL2023 和 Bottlerocket AMIs 支援的驅動程式和 Linux 核心版本比較

元件	AL2 AMI 來源	AL2023 AMI 來源	Bottlerocket AMI 來源
基本作業系統相容性	RHEL7/CentOS 7	Fedora/CentOS 9	N/A
CUDA 工具組	CUDA 11.x–12.x	CUDA 12.5+	CUDA 11.x (12.5 即將推出）
NVIDIA GPU 驅動程式	R550	R565 (R570 即將推出）	R535 (R570 即將推出）
AWS Neuron 驅動程式	2.19	2.19+	2.20
Linux 核心	5.10	6.1、6.12	5.15、6.1 (6.12 即將推出）

AWS Neuron 與 AL2 AMIs相容性

從 AWS Neuron 2.20 版開始，EKS AL 型 AMIs 使用的 Neuron 執行期 (`aws-neuronx-runtime-lib`) 不再支援 Amazon Linux 2 (AL2)。Neuron 驅動程式 (`aws-neuronx-dkms`) 現在是唯一支援 Amazon Linux 2 的 AWS Neuron 套件。這表示您無法在以 AL2-based AMI 上原生執行 Neuron 支援的應用程式。若要在 AL2023 AMIs 上設定 Neuron，請參閱 AWS Neuron 設定指南。

Kubernetes 與 AL2 AMIs相容性

Kubernetes 社群已將`cgroupv1`支援 (AL2 使用） 移至維護模式。這表示不會新增任何功能，而且只會提供關鍵安全性和主要錯誤修正。任何依賴 cgroupv2 的 Kubernetes 功能，例如 MemoryQoS 和增

強資源隔離，都無法在 AL2 上使用。此外，Amazon EKS Kubernetes 1.32 版是支援 AL2 AMIs最後一個版本。為了維持與最新 Kubernetes 版本的相容性，我們建議遷移至 AL2023 或 Bottlerocket，該版本`cgroupv2`預設為啟用。

Linux 版本與 AL2 AMIs相容性

在 2026 年 6 月 30 日end-of-support(EOS) 日期之前， 都支援 Amazon Linux 2 (AL2)。 AWS 不過，隨著 AL2 的老化，更廣泛的 Linux 社群對新應用程式和功能的支援變得越來越有限。AL2 AMIs是以 Linux 核心 5.10 為基礎，而 AL2023 使用 Linux 核心 6.10。與 AL2023, AL2對更廣泛的 Linux 社群的支援有限。這表示許多上游 Linux 套件和工具需要向後移植，才能使用 AL2 的舊核心版本，某些現代 Linux 功能和安全改進因舊核心而無法使用，許多開放原始碼專案已棄用或限制對 5.10 等舊核心版本的支援。

AL2023 中未包含的已棄用套件

AL2023 中未包含或變更的一些最常見套件包括：

- Amazon Linux 2 中的某些來源二進位套件不再在 Amazon Linux 2023 中提供
- Amazon Linux 如何在 AL2023 中支援不同版本套件 （ 例如 amazon-linux-extras 系統) 的變更
- AL2023 不支援適用於 Enterprise Linux (EPEL) 的額外套件
- AL2023 不支援 32 位元應用程式
- AL2023 不會製作 mate-desktop 套件

若要進一步了解，請參閱比較 AL2 和 AL2023。

跨 AL2, AL2023 和 Bottlerocket 的 FIPS 驗證比較

Amazon Linux 2 (AL2)、Amazon Linux 2023 (AL2023) 和 Bottlerocket 支援聯邦資訊處理標準 (FIPS) 合規。

- AL2 已通過 FIPS 140-2 認證，而 AL2023 已通過 FIPS 140-3 認證。若要在 AL2023 上啟用 FIPS 模式，請在 Amazon EC2 執行個體上安裝必要的套件，並遵循在 AL2023 上啟用 FIPS 模式中的指示執行組態步驟。若要進一步了解，請參閱 AL2023 FAQs。
- Bottlerocket 特別為 FIPS 提供專門建置的變體，限制核心和使用者空間元件使用已提交至 FIPS 140-3 密碼編譯模組驗證計劃的密碼編譯模組。

EKS AMI 驅動程式和版本變更日誌

如需所有 EKS AMI 元件及其版本的完整清單，請參閱 GitHub 上的 Amazon EKS AMI 版本備註。

使用最佳化的 Amazon Linux AMIs 建立節點

Amazon EKS 最佳化 Amazon Linux AMIs 建置於 Amazon Linux 2 (AL2) 和 Amazon Linux 2023 (AL2023) 之上。它們設定為做為 Amazon EKS 節點的基礎映像。AMIs 已設定為使用 Amazon EKS，且包含下列元件：

- `kubelet`
- AWS IAM 驗證器
- `containerd`

> **Note**
>
> - 您可以選擇所需版本的索引標籤，在 Amazon Linux 安全中心追蹤 Amazon Linux 的安全或隱私權事件。您也可以訂閱適用的 RSS 摘要。安全與隱私權事件包含問題的概觀、哪些套件受到影響，以及如何更新您的執行個體以修正問題。
> - 部署加速或 Arm AMI 之前，請檢閱 Amazon EKS 最佳化加速 Amazon Linux AMIs和 中的資訊the section called “Amazon EKS 最佳化 Arm Amazon Linux AMI”。
> - Amazon EKS 不支援 Amazon EC2 P2執行個體，因為它們需要NVIDIA驅動程式 470 版或更早版本。
> - 版本 1.30或更新版本上叢集中任何新建立的受管節點群組，都會自動預設為使用 AL2023 做為節點作業系統。先前，新的節點群組預設為 AL2。建立新節點群組時，您可以選擇 AL2 做為 AMI 類型，以繼續使用 AL2。
> - 在 2025 年 11 月 26 日之後，Amazon EKS 將不再發佈 EKS 最佳化的 Amazon Linux 2 (AL2) AMIs。此外，Kubernetes 版本1.32是 Amazon EKS 將發行 AL2 AMIs最後一個版本。從版本1.33開始，Amazon EKS 將繼續發行 AL2023 和 Bottlerocket 型 AMIs。

Amazon EKS 最佳化加速 Amazon Linux AMI

Amazon EKS 最佳化加速 Amazon Linux AMIs 是以標準 Amazon EKS 最佳化 Amazon Linux AMIs 為基礎建置。它們設定為做為 Amazon EKS 節點的選用映像，以支援 GPU、Inferentia 和 Trainium 型工作負載。

除了標準 Amazon EKS 最佳化 AMI 組態之外，加速 AMIs 還包含下列項目：

- NVIDIA 驅動程式

- `nvidia-container-toolkit`
- AWS Neuron 驅動程式

如需加速 AMIs 中包含的最新元件清單，請參閱 GitHub 上的`amazon-eks-ami`版本。

> **Note**
>
> - 請務必在 node AWS CloudFormation 範本中指定適用的執行個體類型。使用 Amazon EKS 最佳化加速 AMIs，即表示您同意 NVIDIA 的雲端最終使用者授權合約 (EULA)。
> - Amazon EKS 最佳化加速 AMIs 先前稱為具有 GPU 支援的 Amazon EKS 最佳化 AMIs。
> - 舊版的 Amazon EKS 最佳化加速 AMIs 已安裝儲存`nvidia-docker`庫。Amazon EKS AMI 版本 `v20200529` 和更新版本中不再包含該儲存庫。

如需在 Amazon EKS 最佳化加速 Amazon Linux AMIs 上執行工作負載的詳細資訊，請參閱 the section called “設定 Linux GPU AMIs”。

Amazon EKS 最佳化 Arm Amazon Linux AMI

Arm 執行個體可為擴增和 Arm 型應用程式節省大量的成本，例如 Web 伺服器、容器化微型服務、快取機群和分散式資料存放區。將 Arm 節點新增至叢集時，請檢閱下列考量事項。

- 如果叢集是在 2020 年 8 月 17 日之前部署，您必須對重要的叢集附加元件資訊清單進行一次性升級。如此一來，Kubernetes 就可以為叢集中使用中的每個硬體架構提取正確的映像。如需更新叢集附加元件的詳細資訊，請參閱 the section called “步驟 1：準備升級”。如果您在 2020 年 8 月 17 日或之後部署叢集，則適用於 Kubernetes 附加元件的 CoreDNS`kube-proxy`、 和 Amazon VPC CNI 外掛程式已具備多架構功能。
- 部署至 Arm 節點的應用程式必須針對 Arm 進行編譯。
- 如果您在現有叢集中部署了 DaemonSets，或想要將它們部署到也想要部署 Arm 節點的新叢集，請確認您的 DaemonSet 可以在叢集中的所有硬體架構上執行。
- 您可以在相同的叢集中執行 Arm 節點群組和 x86 節點群組。如果您這麼做，請考慮將多架構容器映像部署到容器儲存庫，例如 Amazon Elastic Container Registry，然後將節點選擇器新增至資訊清單，以便 Kubernetes 知道可以部署 Pod 的硬體架構。如需詳細資訊，請參閱《Amazon ECR 使用者指南》中的推送多架構映像和適用於 Amazon ECR 的多架構容器映像簡介部落格一文。

其他資訊

如需使用 Amazon EKS 最佳化 Amazon Linux AMI 的詳細資訊，請參閱下列區段：

- 若要將 Amazon Linux 與受管節點群組搭配使用，請參閱 the section called “受管節點群組”。
- 若要啟動自我管理的 Amazon Linux 節點，請參閱 the section called “取得最新的 IDs”。
- 如需版本資訊，請參閱the section called “取得版本資訊”。
- 若要擷取 Amazon EKS 最佳化 Amazon Linux AMI 的最新 ID，請參閱 the section called “取得最新的 IDs”。
- 如需用於建置 Amazon EKS 最佳化 AMIs 的開放原始碼指令碼，請參閱 the section called “自訂組建”。

從 Amazon Linux 2 升級到 Amazon Linux 2023

Amazon EKS 最佳化 AMIs 提供以 AL2 和 AL2023 為基礎的兩個系列。AL2023 是新的 Linux 作業系統，旨在為您的雲端應用程式提供安全、穩定且高效能的環境。這是來自 Amazon Web Services 的新一代 Amazon Linux，可用於所有支援的 Amazon EKS 版本。

AL2023 提供數個優於 AL2 的改善。如需完整比較，請參閱《Amazon Linux 2023 使用者指南》中的比較 AL2 和 Amazon Linux 2023。 已從 AL2 新增、升級和移除數個套件。強烈建議您在升級之前使用 AL2023 測試您的應用程式。如需 AL2023 中所有套件變更的清單，請參閱《Amazon Linux 2023 版本備註》中的 Amazon Linux 2023 套件變更。

除了這些變更之外，您應該注意下列事項：

- AL2023 推出nodeadm使用 YAML 組態結構描述的新節點初始化程序。如果您使用自我管理節點群組或具有啟動範本的 AMI，您現在將需要在建立新的節點群組時明確提供額外的叢集中繼資料。最低必要參數的範例如下所示，其中cidr現在需要 apiServerEndpoint、certificateAuthority和服務：

```
---
apiVersion: node.eks.aws/v1alpha1
kind: NodeConfig
spec:
  cluster:
    name: my-cluster
    apiServerEndpoint: https://example.com
    certificateAuthority: Y2VydGlmaWNhdGVBdXRob3JpdHk=
    cidr: 10.100.0.0/16
```

在 AL2 中，從 Amazon EKS DescribeCluster API 呼叫中發現了這些參數的中繼資料。透過 AL2023，此行為已變更，因為在大型節點擴展期間，額外的 API 呼叫風險調節。如果您使用的是沒有啟動範本的受管節點群組，或是使用的是 Karpenter，則此變更不會影響您。如需 certificateAuthority和服務的詳細資訊cidr，請參閱《Amazon EKS API 參考DescribeCluster》中的 。

- 對於 AL2023， nodeadm也會變更格式，以使用 將參數套用至每個節點kubelet的NodeConfigSpec。在 AL2 中，這是使用 --kubelet-extra-args 參數完成的。這通常用於將標籤和污點新增至節點。以下範例顯示--node-labels將 maxPods和 套用至節點。

```
---
apiVersion: node.eks.aws/v1alpha1
kind: NodeConfig
spec:
  cluster:
    name: test-cluster
    apiServerEndpoint: https://example.com
    certificateAuthority: Y2VydGlmaWNhdGVBdXRob3JpdHk=
    cidr: 10.100.0.0/16
  kubelet:
    config:
      maxPods: 110
    flags:
      - --node-labels=karpenter.sh/capacity-type=on-demand,karpenter.sh/nodepool=test
```

- AL2023 需要 Amazon VPC CNI 版本 1.16.2 或更高版本。
- 根據預設IMDSv2，AL2023 需要 。 IMDSv2具有數種有助於改善安全狀態的優點。它使用工作階段導向的身分驗證方法，需要在簡單的 HTTP PUT 請求中建立秘密字符，才能啟動工作階段。工作階段的權杖可以在 1 秒到 6 小時之間的任何位置有效。如需如何從 轉換IMDSv1至 的詳細資訊IMDSv2，請參閱使用執行個體中繼資料服務第 2 版的轉換和取得 IMDSv2 的完整優點，以及在整個 AWS 基礎設施中停用 IMDSv1。如果您想要使用 IMDSv1，仍然可以使用執行個體中繼資料選項啟動屬性手動覆寫設定來執行此操作。

> **Note**
>
> 對於IMDSv2使用 AL2023 的 ，受管節點群組的預設跳轉計數可能會有所不同：
>
> - 不使用啟動範本時，預設值會設為 1。這表示容器將無法存取使用 IMDS 的節點登入資料。如果您需要容器存取節點的登入資料，您仍然可以使用自訂的 Amazon EC2 啟動範本來執行此操作。

    - 在啟動範本中使用自訂 AMI 時，預設值`HttpPutResponseHopLimit`會設為 2。您可以在啟動範本`HttpPutResponseHopLimit`中手動覆寫 。

  或者，您可以使用 Amazon EKS Pod Identity 來提供登入資料，而不是 IMDSv2。

- AL2023 具有新一代的統一控制群組階層 (cgroupv2)。 cgroupv2 用於實作容器執行期，並由 執行`systemd`。雖然 AL2023 仍包含可讓系統使用 執行的程式碼`cgroupv1`，但這不是建議或支援的組態。此組態將在未來的 Amazon Linux 主要版本中完全移除。
- `eksctl` 需要 版本 `0.176.0`或更高版本`eksctl`才能支援 AL2023。

對於先前現有的受管節點群組，您可以執行就地升級或藍/綠升級，具體取決於您使用啟動範本的方式：

- 如果您使用自訂 AMI 搭配受管節點群組，您可以在啟動範本中交換 AMI ID，以執行就地升級。在執行此升級策略之前，您應該先確保您的應用程式和任何使用者資料傳輸到 AL2023。
- 如果您使用受管節點群組搭配標準啟動範本，或搭配未指定 AMI ID 的自訂啟動範本，則需要使用藍/綠策略進行升級。藍/綠升級通常更為複雜，涉及建立全新的節點群組，其中會將 AL2023 指定為 AMI 類型。然後，需要仔細設定新的節點群組，以確保來自 AL2 節點群組的所有自訂資料都與新的作業系統相容。一旦使用應用程式測試和驗證新的節點群組，Pod 就可以從舊節點群組遷移到新的節點群組。一旦遷移完成，您就可以刪除舊節點群組。

如果您使用 Karpenter 且想要使用 AL2023，則需要使用 AL2023 修改 `EC2NodeClass` `amiFamily` 欄位。根據預設，漂移會在 Karpenter 中啟用。這表示變更`amiFamily`欄位後，Karpenter 會在可用時自動將工作者節點更新為最新的 AMI。

擷取 Amazon Linux AMI 版本資訊

Amazon EKS 最佳化的 Amazon Linux AMIs 是由 Kubernetes 版本和 AMI 的發行日期進行版本控制，格式如下：

```
k8s_major_version.k8s_minor_version.k8s_patch_version-release_date
```

每個 AMI 版本都包含各種版本的 kubelet、Linux 核心和容器化版本。加速AMIs 也包含各種版本的 NVIDIA 驅動程式。您可以在 GitHub 上的變更日誌中找到此版本資訊。

擷取建議的 Amazon Linux AMI IDs

部署節點時，您可以為預先建置的 Amazon EKS 最佳化 Amazon Machine Image (AMI) 指定 ID。若要擷取符合您所需組態的 AMI ID，請查詢 AWS Systems Manager 參數存放區 API。使用此 API 不需要手動查詢 Amazon EKS 最佳化 AMI IDs。如需詳細資訊，請參閱 GetParameter。您使用的 IAM 主體必須擁有 `ssm:GetParameter` IAM 許可，才能擷取 Amazon EKS 最佳化 AMI 中繼資料。

您可以使用下列命令擷取最新建議的 Amazon EKS 最佳化 Amazon Linux AMI 的影像 ID，該命令使用子參數 `image_id`。視需要對命令進行下列修改，然後執行修改後的命令：

- `<kubernetes-version>` 將 取代為支援的 eks/latest/userguide/platform-versions.html【platform-version，type="documentation"】。
- 將 *ami-type* 取代為下列其中一個選項。如需 Amazon EC2 執行個體類型的資訊，請參閱 Amazon EC2 執行個體類型。
  - 針對以 Amazon Linux *2023 (AL2023) ########## amazon-linux-2023/x86_64/standard*。AL2023 x86
  - 針對 AL2023*2023 ARM ###### amazon-linux-2023/arm64/standard*，例如以 AWS Graviton 為基礎的執行個體。
  - 針對最新核准的 *AL2023 NVIDIA ######## amazon-linux-2023/x86_64/nvidia*。AL2023 x86
  - 針對最新核准的 *AL2023 NVIDIA ######## amazon-linux-2023/arm64/nvidia*。AL2023 `arm64`
  - 針對最新的 *AL2023 Neuron ###### amazon-linux-2023/x86_64/neuron*。 AL2023 AWS
  - 針對以 Amazon Linux *2 (AL2) ########## amazon-linux-*2。AL2 x86
  - 針對 AL2 ARM 執行個體使用 *amazon-linux-2-arm64*，例如以 AWS Graviton 為基礎的執行個體。
  - 針對 NVIDIA GPU、Inferentia 和 Trainium x86型工作負載的 AL2 硬體加速型執行個體，請使用 *amazon-linux-2-gpu*。
- `<region-code>` 將 取代為您想要 AMI ID 的 Amazon EKS 支援 AWS 區域。

```
aws ssm get-parameter --name /aws/service/eks/optimized-ami/<kubernetes-version>/<ami-type>/recommended/image_id \
    --region <region-code> --query "Parameter.Value" --output text
```

以下是進行預留位置取代後的範例命令。

```
aws ssm get-parameter --name /aws/service/eks/optimized-ami/1.31/amazon-
linux-2023/x86_64/standard/recommended/image_id \
    --region us-west-2 --query "Parameter.Value" --output text
```

範例輸出如下。

```
ami-1234567890abcdef0
```

建置自訂 Amazon Linux AMI

> **Important**
>
> Amazon EKS 不會再於 2025 年 11 月 26 日之後發佈 EKS 最佳化的 Amazon Linux 2 (AL2) AMIs。此外，Kubernetes 版本1.32是 Amazon EKS 將發行 AL2 AMIs最後一個版本。從版本1.33開始，Amazon EKS 將繼續發行 AL2023 和 Bottlerocket 型 AMIs。

Amazon EKS 最佳化的 Amazon Linux (AL) AMIs 建置在 AL2 和 AL2023 之上，專門做為 Amazon EKS 叢集中的節點使用。Amazon EKS 在 Amazon EKS AMI Build Specification 儲存庫中提供開放原始碼建置指令碼，您可以使用下列方式：* 檢視 的組態kubelet、執行時間，以及 Kubernetes 的 AWS IAM Authenticator。* 從頭開始建置您自己的 AL 型 AMI。

此儲存庫包含在開機時間執行的專門引導指令碼和 nodeadm 指令碼。這些指令碼會設定執行個體的憑證資料、控制平面端點、叢集名稱等。指令碼被視為 Amazon EKS 最佳化 AMI 建置的真實來源，因此您可以遵循 GitHub 儲存庫來監控 AMIs的變更。

先決條件

- 安裝 AWS CLI
- 安裝 HashiCorp Packer 1.9.4 版以上
- 安裝 GNU Make

快速指南

本節顯示在您的 AWS 帳戶中建立自訂 AMI 的命令。若要進一步了解可用於自訂 AMI 的組態，請參閱 Amazon Linux 2023 頁面上的範本變數。

步驟 1. 設定您的環境

複製或分叉官方 Amazon EKS AMI 儲存庫。例如：

```
git clone https://github.com/awslabs/amazon-eks-ami.git
cd amazon-eks-ami
```

確認 Packer 已安裝：

```
packer --version
```

步驟 2. 建立自訂 AMI

以下是各種自訂 AMIs的範例命令。

基本 NVIDIA AL2 AMI：

```
make k8s=1.31 os_distro=al2 \
  enable_accelerator=nvidia \
  nvidia_driver_major_version=560 \
  enable_efa=true
```

基本 NVIDIA AL2023 AMI：

```
make k8s=1.31 os_distro=al2023 \
  enable_accelerator=nvidia \
  nvidia_driver_major_version=560 \
  enable_efa=true
```

符合 STIG 規範的 Neuron AL2023 AMI：

```
make k8s=1.31 os_distro=al2023 \
  enable_accelerator=neuron \
  enable_fips=true \
  source_ami_id=ami-0abcd1234efgh5678 \
  kms_key_id=alias/aws-stig
```

在您執行這些命令之後，Packer 將執行下列動作：* 啟動暫時 Amazon EC2 執行個體。* 安裝 Kubernetes 元件、驅動程式和組態。* 在 AWS 帳戶中建立 AMI。

步驟 3。檢視預設值

若要檢視預設值和其他選項，請執行下列命令：

```
make help
```

使用最佳化 Bottlerocket AMIs建立節點

Bottlerocket 是由 贊助和支援的開放原始碼 Linux 發行版本 AWS。Bottlerocket 專為託管容器工作負載而打造。透過 Bottlerocket，您可以自動化容器基礎設施的更新，以改善容器化部署的可用性並降低營運成本。Bottlerocket 僅包含執行容器的必要軟體，可改善資源用量、降低安全威脅，並降低管理開銷。Bottlerocket AMI 包含 `containerd`、 `kubelet`和 AWS IAM Authenticator。除了受管節點群組和自我管理節點之外，Karpenter 也支援 Bottlerocket。

優點

將 Bottlerocket 與 Amazon EKS 叢集搭配使用具有以下優勢：

- 較高的運作時間，營運成本較低且管理複雜性較低 – Bottlerocket 的資源使用量較小、開機時間較短，而且比其他 Linux 發行版本更不易受到安全威脅。Bottlerocket 的佔用空間較小，有助於透過使用較少的儲存、運算和聯網資源來降低成本。
- 透過自動作業系統更新提高安全性 – 對 Bottlerocket 的更新作為一個單元套用，必要時可以復原。這消除了損毀或失敗更新可能使系統處於無法使用狀態的風險。透過 Bottlerocket，只要安全性更新以最低破壞性的方式提供，就可以立即自動套用，並在發生故障時復原。
- 進階支援 – 在 Amazon EC2 上 AWS 提供的 Bottlerocket 組建涵蓋在相同的 AWS 支援計劃中，這些計劃也涵蓋 Amazon EC2、Amazon EKS 和 Amazon ECR 等 AWS 服務。

考量事項

針對 AMI 類型使用 Bottlerocket 時，請考慮下列事項：

- Bottlerocket 支援使用 `x86_64`和 `arm64` 處理器的 Amazon EC2 執行個體。Bottlerocket AMI 不建議搭配 Inferentia 晶片與 Amazon EC2 執行個體搭配使用。
- Bottlerocket 映像不包含 SSH 伺服器或 shell。您可以採用頻外存取方法來允許 SSH。這些方法啟用管理員容器，並傳遞一些引導組態步驟與使用者資料。如需詳細資訊，請參閱 GitHub 上 Bottlerocket 作業系統中的下列章節：
  - 探勘

  - 管理員容器
  - Kubernetes 設定
- Bottlerocket 使用不同的容器類型：
  - 在預設情況下，已啟用控制容器。此容器會執行 AWS Systems Manager 代理程式，您可以用來在 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 Systems Manager 使用者指南中的設定 Session Manager。 AWS
  - 如果在建立節點群組時提供 SSH 金鑰，則啟用管理容器。建議只在開發和測試案例中使用管理容器，我們不建議將其用於生產環境。如需詳細資訊，請參閱 GitHub 上的管理容器。

其他資訊

如需使用 Amazon EKS 最佳化 Bottlerocket AMIs的詳細資訊，請參閱下列章節：

- 如需 Bottlerocket 的詳細資訊，請參閱 Bottlerocket 文件。
- 如需版本資訊資源，請參閱 the section called “取得版本資訊”。
- 若要搭配受管節點群組使用 Bottlerocket，請參閱 the section called “受管節點群組”。
- 若要啟動自我管理的 Bottlerocket 節點，請參閱 the section called “Bottlerocket”。
- 若要擷取 Amazon EKS 最佳化 Bottlerocket AMIs 的最新 IDs，請參閱 the section called “取得最新的 IDs”。
- 如需合規性支援的詳細資訊，請參閱 the section called “合規支援”。

擷取 Bottlerocket AMI 版本資訊

每個 Bottlerocket AMI 版本都包含各種版本的 kubelet、 Bottlerocket 核心和容器化。加速 AMI 變體也包含各種版本的 NVIDIA 驅動程式。您可以在 Bottlerocket 文件的作業系統主題中找到此版本資訊。在此頁面上，導覽至適用的版本資訊子主題。

Bottlerocket 文件有時可能會落後於 GitHub 上提供的版本。您可以在 GitHub 的 版本中找到最新版本的變更清單。

擷取建議的 Bottlerocket AMI IDs

部署節點時，您可以為預先建置的 Amazon EKS 最佳化 Amazon Machine Image (AMI) 指定 ID。若要擷取符合您所需組態的 AMI ID，請查詢 AWS Systems Manager 參數存放區 API。使用此 API 不需要手動查詢 Amazon EKS 最佳化 AMI IDs。如需詳細資訊，請參閱 GetParameter。您使用的 IAM 主體必須擁有 `ssm:GetParameter` IAM 許可，才能擷取 Amazon EKS 最佳化 AMI 中繼資料。

您可以使用下列使用子參數 的 AWS CLI 命令，擷取最新建議的 Amazon EKS 最佳化 Bottlerocket AMI 的影像 IDimage_id。視需要對命令進行下列修改，然後執行修改後的命令：

- 將 *kubernetes-version* 取代為支援的 eks/latest/userguide/platform-versions.html【platform-version，type="documentation"】。
- 將 *-flavor* 取代為下列其中一個選項。
  - 針對沒有 GPU 的變體移除 *-Flavor*。
  - 將 *-nvidia* 用於啟用 GPU 的變體。
  - 針對啟用 FIPS 的變體使用 *-fips*。
- 將*##*取代為下列其中一個選項。
  - 針對x86以 為基礎的執行個體使用 *x86_64*。
  - 針對 ARM 執行個體使用 *arm64*。
- 將 *region-code* 取代為您想要 AMI ID 的 Amazon EKS 支援 AWS 區域。

```
aws ssm get-parameter --name /aws/service/bottlerocket/aws-k8s-kubernetes-version-
flavor/architecture/latest/image_id \
    --region region-code --query "Parameter.Value" --output text
```

以下是進行預留位置取代後的範例命令。

```
aws ssm get-parameter --name /aws/service/bottlerocket/aws-k8s-1.31/x86_64/latest/
image_id \
    --region us-west-2 --query "Parameter.Value" --output text
```

範例輸出如下。

```
ami-1234567890abcdef0
```

使用 Bottlerocket 滿足合規要求

Bottlerocket 符合各種組織定義的建議：

- Bottlerocket 有已定義的 CIS 基準。在預設組態中， Bottlerocket 映像具有 CIS 第 1 級組態描述檔所需的大部分控制項。您可以實作 CIS 第 2 級組態設定檔所需的控制項。如需詳細資訊，請參閱 AWS 部落格上的針對 CIS 基準驗證 Amazon EKS 最佳化 Bottlerocket AMI。

- 最佳化功能集和減少攻擊面，表示 Bottlerocket 執行個體需要較少的組態以滿足 PCI DSS 要求。Bottlerocket 的 CIS 基準是強化指引的絕佳資源，並支援您在 PCI DSS 要求 2.2 下的安全組態標準需求。您也可以利用 Fluent Bit 來支援您在 PCI DSS 要求 10.2 下的作業系統層級稽核記錄記錄需求。 會定期 AWS 發佈新的 （修補） Bottlerocket 執行個體，以協助您符合 PCI DSS 要求 6.2 （適用於 v3.2.1) 和要求 6.3.3 （適用於 v4.0)。
- Bottlerocket 是符合 HIPAA 資格的功能，授權與 Amazon EC2 和 Amazon EKS 的受管制工作負載搭配使用。如需詳細資訊，請參閱 HIPAA 合格服務參考。
- Bottlerocket AMIs已預先設定為使用 FIPS 140-3 驗證的密碼編譯模組。這包括 Amazon Linux 2023 核心加密 API 密碼編譯模組和 AWS-LC 密碼編譯模組。如需詳細資訊，請參閱the section called “Bottlerocket FIPS AMIs”。

讓您的工作者節點 FIPS 準備好 Bottlerocket FIPS AMIs

聯邦資訊處理標準 (FIPS) 出版物 140-3 是美國和加拿大政府標準，指定保護敏感資訊之密碼編譯模組的安全要求。Bottlerocket 提供具有 FIPS 核心AMIs，讓您更輕鬆地遵守 FIPS。

這些 AMIs已預先設定為使用 FIPS 140-3 驗證的密碼編譯模組。這包括 Amazon Linux 2023 核心加密 API 密碼編譯模組和 AWS-LC 密碼編譯模組。

使用 Bottlerocket FIPS AMIs可讓您的工作者節點「FIPS 就緒」，但不會自動「FIPS 相容」。如需詳細資訊，請參閱聯邦資訊處理標準 (FIPS) 140-3。

考量事項

- 如果您的叢集使用隔離的子網路，則可能無法存取 Amazon ECR FIPS 端點。這可能會導致節點引導失敗。請確定您的網路組態允許存取必要的 FIPS 端點。如需詳細資訊，請參閱 AWS PrivateLink 指南中的透過資源 VPC 端點存取資源。
- 如果您的叢集搭配 PrivateLink 使用子網路，映像提取將會失敗，因為 Amazon ECR FIPS 端點無法透過 PrivateLink 使用。

使用 Bottlerocket FIPS AMI 建立受管節點群組

Bottlerocket FIPS AMI 有兩種變體來支援您的工作負載：

- `BOTTLEROCKET_x86_64_FIPS`
- `BOTTLEROCKET_ARM_64_FIPS`

若要使用 Bottlerocket FIPS AMI 建立受管節點群組，請在建立過程中選擇適用的 AMI 類型。如需詳細資訊，請參閱the section called “建立”。

如需選取啟用 FIPS 的變體的詳細資訊，請參閱 the section called “取得最新的 IDs”。

停用不支援 AWS 區域的 FIPS 端點

Bottlerocket FIPS AMIs 直接在美國支援，包括 AWS GovCloud (US) 區域。對於可使用 AMIs 但無法直接支援的 AWS 區域，您仍然可以透過使用啟動範本建立受管節點群組來使用 AMIs。

Bottlerocket FIPS AMI 在引導期間倚賴 Amazon ECR FIPS 端點，這些端點在美國以外尚未正式推出。若要在沒有 Amazon ECR FIPS 端點可用的 AWS 區域中，為其 FIPS 核心使用 AMI，請執行下列步驟來停用 FIPS 端點：

1. 使用下列內容建立新的組態檔案，或將內容併入現有的組態檔案中。

```
[default]
use_fips_endpoint=false
```

1. 將檔案內容編碼為 Base64 格式。
2. 在啟動範本的 中UserData，使用 TOML 格式新增下列編碼字串：

```
[settings.aws]
config = "<your-base64-encoded-string>"
```

如需其他設定，請參閱 GitHub 上 Bottlerocket 的設定描述。

以下是啟動範本UserData中 的範例：

```
[settings]
motd = "Hello from eksctl!"
[settings.aws]
config = "W2RlZmF1bHRdCnVzZV9maXBzX2VuZHBvaW50PWZhbHNlCg==" # Base64-encoded string.
[settings.kubernetes]
api-server = "<api-server-endpoint>"
cluster-certificate = "<cluster-certificate-authority>"
cluster-name = "<cluster-name>"
...<other-settings>
```

如需使用使用者資料建立啟動範本的詳細資訊，請參閱 the section called “啟動範本”。

使用最佳化的 Ubuntu Linux AMIs建立節點

Canonical 已與 Amazon EKS 合作，建立了可供您用於叢集的節點 AMI。

Canonical 提供專為特定用途打造的 Kubernetes 節點作業系統映像。此最小化的 Ubuntu 映像已針對 Amazon EKS 最佳化，並包含與 共同開發的自訂 AWS 核心 AWS。如需詳細資訊，請參閱 Amazon Elastic Kubernetes Service (EKS) 上的 Ubuntu 和 the section called “Ubuntu Linux” 。如需支援的相關資訊，請參閱 AWS Premium Support FAQs中的第三方軟體區段。

使用最佳化 Windows AMIs建立節點

Windows Amazon EKS 最佳化 AMIs 建置在 Windows Server 2019 和 Windows Server 2022 之上。這些 AMI 已設定為 Amazon EKS 節點的基礎映像。預設情況下，AMI 會包括以下組件：

- kubelet
- kube-proxy
- AWS 適用於 Kubernetes 的 IAM Authenticator
- csi-proxy
- 容器化

> **Note**
>
> 您可以使用 Microsoft 安全性更新指南，追蹤 Windows Server 的安全性或隱私權事件。

適用於 Amazon EKS 提供針對有下列變化之 Windows 容器最佳化的 AMI：

- Amazon EKS 最佳化 Windows Server 2019 Core AMI
- Amazon EKS 最佳化 Windows Server 2019 Full AMI
- Amazon EKS 最佳化 Windows Server 2022 Core AMI
- Amazon EKS 最佳化 Windows Server 2022 完整 AMI

Important

- Amazon EKS 最佳化 Windows Server 20H2 Core AMI 已棄用。不再發佈此 AMI 的新版本。
- 為了確保您預設擁有最新的安全性更新，Amazon EKS 會在過去 4 個月內維持最佳化 AMIs。每個新的 AMI 將在初始發行後 4 個月內可用。在此期間之後，較舊AMIs 會設為私有，且無法再存取。我們鼓勵使用最新的 AMIs，以避免安全漏洞，並失去對已達支援生命週期結束的舊 AMIs 的存取權。雖然我們無法保證可以提供已設為私有AMIs 的存取權，但您可以透過向 AWS Support 提交票證來請求存取權。

版本發佈日曆

下表列出 Amazon EKS 上 Windows 版本的發佈和終止支援日期。如果結束日期為空白，是因為版本仍受支援。

Windows 版本	Amazon EKS 發佈	Amazon EKS 終止支援
Windows Server 2022 Core	10/17/2022	
Windows Server 2022 Full	10/17/2022	
Windows Server 20H2 Core	8/12/2021	8/9/2022
Windows Server 2004 Core	8/19/2020	12/14/2021
Windows Server 2019 Core	10/7/2019	
Windows Server 2019 Full	10/7/2019	
Windows Server 1909 Core	10/7/2019	12/8/2020

引導指令碼組態參數

當您建立 Windows 節點時，節點上有一個指令碼允許設定不同的參數。根據設定，可以在節點上類似於以下位置的地方找到該指令碼：`C:\Program Files\Amazon\EKS\Start-EKSBootstrap.ps1`。您可將自訂參數值指定為引導指令碼的引數，以指定自訂參數值。例如，您

可以更新啟動範本中的使用者資料。如需詳細資訊，請參閱the section called “Amazon EC2 使用者資料”。

該指令碼包括下列命令列參數：

- `-EKSClusterName`：指定此工作節點要加入的 Amazon EKS 叢集名稱。
- `-KubeletExtraArgs`：指定 kubelet 額外的引數 (選用)。
- `-KubeProxyExtraArgs`：指定 kube-proxy 額外的引數 (選用)。
- `-APIServerEndpoint`：指定 Amazon EKS 叢集 API 伺服器端點 (可選)。僅在搭配 `-Base64ClusterCA` 使用時有效。略過呼叫 `Get-EKSCluster`。
- `-Base64ClusterCA`：指定 Base64 編碼的叢集 CA 內容 (可選)。僅在搭配 `-APIServerEndpoint` 使用時有效。略過呼叫 `Get-EKSCluster`。
- `-DNSClusterIP`：覆寫要用於叢集內 DNS 查詢的 IP 地址 (選用)。根據主要介面的 IP 地址，預設為 `10.100.0.10` 或 `172.20.0.10`。
- `-ServiceCIDR` – 覆寫要從中處理叢集服務的 Kubernetes 服務 IP 地址範圍。根據主要介面的 IP 地址，預設為 `172.20.0.0/16` 或 `10.100.0.0/16`。
- `-ExcludedSnatCIDRs`：要從來源網路地址轉譯 (SNAT) 排除的 `IPv4` CIDR 清單。這表示 VPC 可定址的 Pod 私有 IP 不會轉譯為執行個體 ENI 用於傳出流量的主要`IPv4`地址 IP 地址。根據預設，會新增 Amazon EKS Windows 節點 VPC 的 `IPv4` CIDR。將 CIDR 指定給此參數也會另外排除指定的 CIDR。如需詳細資訊，請參閱the section called “傳出流量”。

除了命令列參數之外，您還可以指定一些環境變數參數。指定命令列參數時，它的優先順序高於個別的環境變數。環境變數應定義為機器 (或系統) 範圍，因為引導指令碼只會讀取機器範圍的變數。

此指令碼會考慮下列環境變數：

- `SERVICE_IPV4_CIDR`：請參閱 `ServiceCIDR` 命令列參數了解定義。
- `EXCLUDED_SNAT_CIDRS`：應為逗號分隔的字串。請參閱 `ExcludedSnatCIDRs` 命令列參數了解定義。

gMSA 身分驗證支援

Amazon EKS Windows Pod 允許不同類型的群組受管服務帳戶 (gMSA) 身分驗證。

- Amazon EKS 支援 Active Directory 網域身分進行身分驗證。如需加入網域的 gMSA 的詳細資訊，請參閱 AWS 部落格上的 Amazon EKS Windowspods 上的 Windows 身分驗證。

- Amazon EKS 提供外掛程式，可讓non-domain-joined節點擷取具有可攜式使用者身分的 gMSA 登入資料。如需無網域 gMSA 的詳細資訊，請參閱 AWS 部落格上的 Amazon EKS Windowspods 的無網域 Windows 身分驗證。

快取容器映像

Amazon EKS Windows 最佳化 AMIs 已快取containerd執行時間的特定容器映像。使用 Amazon 受管建置元件建置自訂 AMI 時，會快取容器映像。如需詳細資訊，請參閱the section called “使用 Amazon 受管建置元件”。

下列的快取容器映像適用於 containerd 執行期：

- amazonaws.com/eks/pause-windows
- mcr.microsoft.com/windows/nanoserver
- mcr.microsoft.com/windows/servercore

其他資訊

如需使用 Amazon EKS 最佳化 Windows AMIs 的詳細資訊，請參閱下列章節：

- 如需在 Amazon EKS 最佳化加速 Windows AMIs 上執行工作負載的詳細資訊，請參閱 the section called “設定 Windows GPU AMIs”。
- 若要搭配受管節點群組使用 Windows，請參閱 the section called “受管節點群組”。
- 若要啟動自我管理的 Windows 節點，請參閱 the section called “Windows”。
- 如需版本資訊，請參閱the section called “取得版本資訊”。
- 若要擷取 Amazon EKS 最佳化 Windows AMIs 的最新 IDs，請參閱 the section called “取得最新的 IDs”。
- 若要使用 Amazon EC2 Image Builder 建立自訂 Amazon EKS 最佳化 Windows AMIs，請參閱 the section called “自訂組建”。
- 如需最佳實務，請參閱《EKS 最佳實務指南》中的 Amazon EKS 最佳化 Windows AMI 管理。

使用 建立自我管理的 Windows Server 2022 節點 **eksctl**

您可以使用下列項目test-windows-2022.yaml做為建立自我管理 Windows Server 2022 節點的參考。將每個###取代為您自己的值。

> **Note**
>
> 您必須使用 0eksctl.116.0 版或更新版本來執行自我管理的 Windows Server 2022 節點。https://github.com/weaveworks/eksctl/releases/tag/v0.116.0

```
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: windows-2022-cluster
  region: region-code
  version: '1.33'

nodeGroups:
  - name: windows-ng
    instanceType: m5.2xlarge
    amiFamily: WindowsServer2022FullContainer
    volumeSize: 100
    minSize: 2
    maxSize: 3
  - name: linux-ng
    amiFamily: AmazonLinux2
    minSize: 2
    maxSize: 3
```

然後，可以使用以下命令建立節點群組。

```
eksctl create cluster -f test-windows-2022.yaml
```

擷取 Windows AMI 版本資訊

本主題列出 Amazon EKS 最佳化 Windows AMIs 的版本，及其對應的 kubelet、 containerd 和 csi-proxy 版本。

可透過程式設計方式擷取每個變體的 Amazon EKS 最佳化 AMI 中繼資料 (包括 AMI ID)。如需詳細資訊，請參閱the section called “取得最新的 IDs”。

AMI 是由 Kubernetes 版本和 AMI 發行日期進行版本化，格式如下：

```
k8s_major_version.k8s_minor_version-release_date
```

> **Note**
>
> Amazon EKS 受管節點群組支援 2022 年 11 月及後來的 Windows AMI 版本。

若要接收此特定文件頁面所有來源檔案變更的通知，您可以使用 RSS 讀取器訂閱下列 URL：

```
https://github.com/awsdocs/amazon-eks-user-guide/commits/mainline/latest/ug/nodes/eks-ami-versions-windows.adoc.atom
```

Amazon EKS 最佳化 Windows Server 2022 Core AMI

下表列出 Amazon EKS 最佳化 Windows Server 2022 Core AMI 的目前和先前版本。

Example

Kubernetes 1.33 版

AMI 版本	kubelet 版本	容器化版本	csi-proxy 版本	版本備註
1.33-2025.07.16	1.33.1	1.7.27	1.2.1	
1.33-2025.06.13	1.33.1	1.7.27	1.2.1	
1.33-2025.05.17	1.33.1	1.7.27	1.2.1	

Kubernetes 1.32 版

AMI 版本	kubelet 版本	容器化版本	csi-proxy 版本	版本備註
1.32-2025.07.16	1.32.5	1.7.27	1.2.1	
1.32-2025.06.13	1.32.5	1.7.27	1.2.1	已將 container

AMI 版本	kubelet 版本	容器化版本	csi-proxy 版本	版本備註
				d 升級至 1.7.27。
1.32-2025.05.17	1.32.5	1.7.20	1.2.1	
1.32-2025.04.14	1.32.1	1.7.20	1.1.3	
1.32-2025.03.14	1.32.1	1.7.20	1.1.3	
1.32-2025.02.18	1.32.1	1.7.20	1.1.3	
1.32-2025.01.15	1.32.0	1.7.20	1.1.3	

Kubernetes 1.31 版

AMI 版本	kubelet 版本	容器化版本	csi-proxy 版本	版本備註
1.31-2025.07.16	1.31.9	1.7.27	1.2.1	
1.31-2025.06.13	1.31.9	1.7.27	1.2.1	已將 containerd 升級至 1.7.27。
1.31-2025.05.17	1.31.9	1.7.20	1.2.1	
1.31-2025.04.14	1.31.5	1.7.20	1.1.3	

AMI 版本	kubelet 版本	容器化版本	csi-proxy 版本	版本備註
1.31-2025.03.14	1.31.5	1.7.20	1.1.3	
1.31-2025.02.15	1.31.5	1.7.20	1.1.3	
1.31-2025.01.15	1.31.4	1.7.20	1.1.3	
1.31-2025.01.01	1.31.4	1.7.20	1.1.3	包含 CVE-2024-9042 的修補程式。
1.31-2024.12.13	1.31.3	1.7.20	1.1.3	
1.31-2024.11.12	1.31.1	1.7.20	1.1.3	
1.31-2024.10.08	1.31.1	1.7.20	1.1.3	
1.31-2024.10.01	1.31.1	1.7.20	1.1.3	
1.31-2024.09.10	1.31.0	1.7.20	1.1.3	

Kubernetes 1.30 版

AMI 版本	kubelet 版本	容器化版本	csi-proxy 版本	版本備註
1.30-2025.07.16	1.30.13	1.7.27	1.2.1	

AMI 版本	kubelet 版本	容器化版本	csi-proxy 版本	版本備註
1.30-2025.06.13	1.30.13	1.7.27	1.2.1	已將 containerd 升級至 1.7.27。
1.30-2025.05.17	1.30.13	1.7.20	1.2.1	
1.30-2025.04.14	1.30.9	1.7.20	1.1.3	
1.30-2025.03.14	1.30.9	1.7.20	1.1.3	已將 containerd 升級至 1.7.20。
1.30-2025.02.15	1.30.9	1.7.14	1.1.3	
1.30-2025.01.15	1.30.8	1.7.14	1.1.3	
1.30-2025.01.01	1.30.8	1.7.14	1.1.3	包含 CVE-2024-9042 的修補程式。
1.30-2024.12.11	1.30.7	1.7.14	1.1.3	
1.30-2024.11.12	1.30.4	1.7.14	1.1.3	
1.30-2024.10.08	1.30.4	1.7.14	1.1.3	

AMI 版本	kubelet 版本	容器化版本	csi-proxy 版本	版本備註
1.30-2024.09.10	1.30.2	1.7.14	1.1.3	
1.30-2024.08.13	1.30.2	1.7.14	1.1.3	
1.30-2024.07.10	1.30.2	1.7.14	1.1.2	包含 CVE-2024-5321 的修補程式。
1.30-2024.06.17	1.30.0	1.7.14	1.1.2	已將 containerd 升級至 1.7.14。
1.30-2024.05.15	1.30.0	1.6.28	1.1.2	

Kubernetes 1.29 版

AMI 版本	kubelet 版本	容器化版本	csi-proxy 版本	版本備註
1.29-2025.07.16	1.29.15	1.7.27	1.2.1	
1.29-2025.06.13	1.29.15	1.7.27	1.2.1	已將 containerd 升級至 1.7.27。
1.29-2025.05.17	1.29.15	1.7.20	1.2.1	
1.29-2025.04.14	1.29.13	1.7.20	1.1.3	

AMI 版本	kubelet 版本	容器化版本	csi-proxy 版本	版本備註
1.29-2025.03.14	1.29.13	1.7.20	1.1.3	已將 containerd 升級至 1.7.20。
1.29-2025.02.15	1.29.13	1.7.14	1.1.3	
1.29-2025.01.15	1.29.12	1.7.14	1.1.3	
1.29-2025.01.01	1.29.12	1.7.14	1.1.3	包含 CVE-2024-9042 的修補程式。
1.29-2024.12.11	1.29.10	1.7.14	1.1.3	
1.29-2024.11.12	1.29.8	1.7.14	1.1.3	
1.29-2024.10.08	1.29.8	1.7.14	1.1.3	
1.29-2024.09.10	1.29.6	1.7.14	1.1.3	
1.29-2024.08.13	1.29.6	1.7.14	1.1.3	
1.29-2024.07.10	1.29.6	1.7.11	1.1.2	包含 CVE-2024-5321 的修補程式。

AMI 版本	kubelet 版本	容器化版本	csi-proxy 版本	版本備註
1.29-2024.06.17	1.29.3	1.7.11	1.1.2	
1.29-2024.05.15	1.29.3	1.7.11	1.1.2	已將 containerd 升級至 1.7.11。已將 kubelet 升級至 1.29.3。
1.29-2024.04.09	1.29.0	1.6.28	1.1.2	已將 containerd 升級至 1.6.28。csi-proxy 使用 重建 CNI 和 golang 1.22.1。
1.29-2024.03.12	1.29.0	1.6.25	1.1.2	
1.29-2024.02.13	1.29.0	1.6.25	1.1.2	
1.29-2024.02.06	1.29.0	1.6.25	1.1.2	修正kubelet垃圾回收程序錯誤刪除暫停映像的錯誤。

AMI 版本	kubelet 版本	容器化版本	csi-proxy 版本	版本備註
1.29-2024.01.11	1.29.0	1.6.18	1.1.2	排除 Windows Server 2022 Core AMI 上的獨立 Windows 更新 KB5034439。AMIs KB 僅適用於具有個別 WinRE 分割區的 Windows 安裝，該分割區不包含在任何 Amazon EKS Optimized Windows AMIs 中。

Kubernetes 1.28 版

AMI 版本	kubelet 版本	容器化版本	csi-proxy 版本	版本備註
1.28-2025.07.16	1.28.15	1.7.27	1.2.1	
1.28-2025.06.13	1.28.15	1.7.27	1.2.1	已將 containerd 升級至 1.7.27。
1.28-2025.05.17	1.28.15	1.7.20	1.2.1	
1.28-2025.04.14	1.28.15	1.7.20	1.1.3	

AMI 版本	kubelet 版本	容器化版本	csi-proxy 版本	版本備註
1.28-2025.03.14	1.28.15	1.7.20	1.1.3	已將 containerd 升級至 1.7.20。
1.28-2025.02.15	1.28.15	1.7.14	1.1.3	
1.28-2025.01.15	1.28.15	1.7.14	1.1.3	
1.28-2025-01-01	1.28.15	1.7.14	1.1.3	包含 CVE-2024-9042 的修補程式。
1.28-2024.12.11	1.28.15	1.7.14	1.1.3	
1.28-2024.11.12	1.28.13	1.7.14	1.1.3	
1.28-2024.10.08	1.28.13	1.7.14	1.1.3	
1.28-2024.09.10	1.28.11	1.7.14	1.1.3	
1.28-2024.08.13	1.28.11	1.7.14	1.1.3	
1.28-2024.07.10	1.28.11	1.7.11	1.1.2	包含 CVE-2024-5321 的修補程式。

AMI 版本	kubelet 版本	容器化版本	csi-proxy 版本	版本備註
1.28-2024.06.17	1.28.8	1.7.11	1.1.2	已將 containerd 升級至 1.7.11。
1.28-2024.05.14	1.28.8	1.6.28	1.1.2	已將 containerd 升級至 1.6.28。已將 kubelet 升級至 1.28.8。
1.28-2024.04.09	1.28.5	1.6.25	1.1.2	已將 containerd 升級至 1.6.25。csi-proxy 使用 重建 CNI 和 golang 1.22.1。
1.28-2024.03.12	1.28.5	1.6.18	1.1.2	
1.28-2024.02.13	1.28.5	1.6.18	1.1.2	

AMI 版本	kubelet 版本	容器化版本	csi-proxy 版本	版本備註
1.28-2024.01.11	1.28.5	1.6.18	1.1.2	排除 Windows Server 2022 Core AMI 上的獨立 Windows Update KB5034439。AMIs KB 僅適用於具有個別 WinRE 分割區的 Windows 安裝，該分割區不包含在任何 Amazon EKS Optimized Windows AMIs 中。
1.28-2023.12.12	1.28.3	1.6.18	1.1.2	
1.28-2023.11.14	1.28.3	1.6.18	1.1.2	包含 CVE-2023-5528 的修補程式。
1.28-2023.10.19	1.28.2	1.6.18	1.1.2	已將 containerd 升級至 1.6.18。已新增引導指令碼環境變數 (SERVICE_IPV4_CIDR 和 EXCLUDED_SNAT_CIDRS)。

AMI 版本	kubelet 版本	容器化版本	csi-proxy 版本	版本備註
1.28-2023-09.27	1.28.2	1.6.6	1.1.2	已修正 kubelet 的 安全性公告。
1.28-2023.09.12	1.28.1	1.6.6	1.1.2	

Kubernetes 1.27 版

AMI 版本	kubelet 版本	容器化版本	csi-proxy 版本	版本備註
1.27-2025.07.16	1.27.16	1.7.27	1.2.1	
1.27-2025.06.13	1.27.16	1.7.27	1.2.1	已將 containerd 升級至 1.7.27。
1.27-2025.05.17	1.27.16	1.7.20	1.2.1	
1.27-2025.04.14	1.27.16	1.7.20	1.1.3	
1.27-2025.03.14	1.27.16	1.7.20	1.1.3	已將 containerd 升級至 1.7.20。
1.27-2025.02.15	1.27.16	1.7.14	1.1.3	
1.27-2025.01.15	1.27.16	1.7.14	1.1.3	

AMI 版本	kubelet 版本	容器化版本	csi-proxy 版本	版本備註
1.27-2025.01.01	1.27.16	1.7.14	1.1.3	包含 CVE-2024-9042 的修補程式。
1.27-2024.12.11	1.27.16	1.7.14	1.1.3	
1.27-2024.11.12	1.27.16	1.7.14	1.1.3	
1.27-2024.10.08	1.27.16	1.7.14	1.1.3	
1.27-2024.09.10	1.27.15	1.7.14	1.1.3	
1.27-2024.08.13	1.27.15	1.7.14	1.1.3	
1.27-2024.07.10	1.27.15	1.7.11	1.1.2	包含 CVE-2024-5321 的修補程式。
1.27-2024.06.17	1.27.12	1.7.11	1.1.2	已將 containerd 升級至 1.7.11。
1.27-2024.05.14	1.27.12	1.6.28	1.1.2	已將 containerd 升級至 1.6.28。已將 kubelet 升級至 1.27.12。

AMI 版本	kubelet 版本	容器化版本	csi-proxy 版本	版本備註
1.27-2024.04.09	1.27.9	1.6.25	1.1.2	已將 containerd 升級至 1.6.25。csi-proxy 使用 重建 CNI 和 golang 1.22.1。
1.27-2024.03.12	1.27.9	1.6.18	1.1.2	
1.27-2024.02.13	1.27.9	1.6.18	1.1.2	
1.27-2024.01.11	1.27.9	1.6.18	1.1.2	排除 Windows Server 2022 Core AMI 上的獨立 Windows Update KB5034439。AMIs KB 僅適用於具有個別 WinRE 分割區的 Windows 安裝，該分割區不包含在任何 Amazon EKS Optimized Windows AMIs 中。
1.27-2023.12.12	1.27.7	1.6.18	1.1.2	

AMI 版本	kubelet 版本	容器化版本	csi-proxy 版本	版本備註
1.27-2023.11.14	1.27.7	1.6.18	1.1.2	包含 CVE-2023-5528 的修補程式。
1.27-2023.10.19	1.27.6	1.6.18	1.1.2	已將 containerd 升級至 1.6.18。已新增引導指令碼環境變數 (SERVICE_IPV4_CIDR 和 EXCLUDED_SNAT_CIDRS)。
1.27-2023-09.27	1.27.6	1.6.6	1.1.2	已修正 kubelet 的 安全性公告。
1.27-2023.09.12	1.27.4	1.6.6	1.1.2	升級 Amazon VPC CNI 外掛程式以使用 Kubernetes 連接器二進位檔，從 Kubernetes API 伺服器取得 Pod IP 地址。合併的提取請求 #100。

AMI 版本	kubelet 版本	容器化版本	csi-proxy 版本	版本備註
1.27-2023.08.17	1.27.4	1.6.6	1.1.2	包含 CVE-2023-3676 、CVE-2023-3893 和 CVE-2023-3955 的修補程式。
1.27-2023.08.08	1.27.3	1.6.6	1.1.1	
1.27-2023.07.11	1.27.3	1.6.6	1.1.1	
1.27-2023.06.20	1.27.1	1.6.6	1.1.1	已解決導致 DNS 字尾搜尋清單填入錯誤的問題。
1.27-2023.06.14	1.27.1	1.6.6	1.1.1	加入了對 CNI 中主機連接埠映射的支援。合併提取請求 #93。
1.27-2023.06.06	1.27.1	1.6.6	1.1.1	修正 containers-roadmap 問題 #2042，此為導致節點無法提取私有 Amazon ECR 映像的問題。
1.27-2023.05.17	1.27.1	1.6.6	1.1.1	

Kubernetes 1.26 版

AMI 版本	kubelet 版本	容器化版本	csi-proxy 版本	版本備註
1.26-2025.05.17	1.26.15	1.7.20	1.2.1	
1.26-2025.04.14	1.26.15	1.7.20	1.1.3	
1.26-2025.03.14	1.26.15	1.7.20	1.1.3	已將 containerd 升級至 1.7.20。
1.26-2025.02.15	1.26.15	1.7.14	1.1.3	
1.26-2025.01.15	1.26.15	1.7.14	1.1.3	
1.26-2024.12.11	1.26.15	1.7.14	1.1.3	
1.26-2024.11.12	1.26.15	1.7.14	1.1.3	
1.26-2024.10.08	1.26.15	1.7.14	1.1.3	
1.26-2024.09.10	1.26.15	1.7.14	1.1.3	
1.26-2024.08.13	1.26.15	1.7.14	1.1.3	
1.26-2024.07.10	1.26.15	1.7.11	1.1.2	包含 CVE-2024-

AMI 版本	kubelet 版本	容器化版本	csi-proxy 版本	版本備註
				5321 的修補程式。
1.26-2024.06.17	1.26.15	1.7.11	1.1.2	已將 containerd 升級至 1.7.11。
1.26-2024.05.14	1.26.15	1.6.28	1.1.2	已將 containerd 升級至 1.6.28。已將 kubelet 升級至 1.26.15。
1.26-2024.04.09	1.26.12	1.6.25	1.1.2	已將 containerd 升級至 1.6.25。csi-proxy 使用 重建 CNI 和 golang 1.22.1。
1.26-2024.03.12	1.26.12	1.6.18	1.1.2	
1.26-2024.02.13	1.26.12	1.6.18	1.1.2	

AMI 版本	kubelet 版本	容器化版本	csi-proxy 版本	版本備註
1.26-2024.01.11	1.26.12	1.6.18	1.1.2	排除 Windows Server 2022 Core AMI 上的獨立 Windows Update KB5034439。AMIs KB 僅適用於具有個別 WinRE 分割區的 Windows 安裝，該分割區不包含在任何 Amazon EKS Optimized Windows AMIs 中。
1.26-2023.12.12	1.26.10	1.6.18	1.1.2	
1.26-2023.11.14	1.26.10	1.6.18	1.1.2	包含 CVE-2023-5528 的修補程式。

AMI 版本	kubelet 版本	容器化版本	csi-proxy 版本	版本備註
1.26-2023.10.19	1.26.9	1.6.18	1.1.2	已將 containerd 升級至 1.6.18。已將 kubelet 升級至 1.26.9。已新增引導指令碼環境變數 (SERVICE_IPV4_CIDR 和 EXCLUDED_SNAT_CIDRS)。
1.26-2023.09.12	1.26.7	1.6.6	1.1.2	升級 Amazon VPC CNI 外掛程式以使用 Kubernetes 連接器二進位檔，從 Kubernetes API 伺服器取得 Pod IP 地址。合併的提取請求 #100。
1.26-2023.08.17	1.26.7	1.6.6	1.1.2	包含 CVE-2023-3676 、CVE-2023-3893 和 CVE-2023-3955 的修補程式。
1.26-2023.08.08	1.26.6	1.6.6	1.1.1	

AMI 版本	kubelet 版本	容器化版本	csi-proxy 版本	版本備註
1.26-2023.07.11	1.26.6	1.6.6	1.1.1	
1.26-2023.06.20	1.26.4	1.6.6	1.1.1	已解決導致 DNS 字尾搜尋清單填入錯誤的問題。
1.26-2023.06.14	1.26.4	1.6.6	1.1.1	升級 Kubernetes 至 1.26.4。加入了對 CNI 中主機連接埠映射的支援。合併提取請求 #93。
1.26-2023.05.09	1.26.2	1.6.6	1.1.1	修正節點重新啟動後，在 Pod 上造成網路連線問題 #1126 的錯誤。引入新的引導指令碼組態參數 (ExcludedSnatCIDRs)。
1.26-2023.04.26	1.26.2	1.6.6	1.1.1	
1.26-2023.04.11	1.26.2	1.6.6	1.1.1	新增了服務當機時 kubelet 和 kube-proxy 的復原機制。
1.26-2023.03.24	1.26.2	1.6.6	1.1.1	

Amazon EKS 最佳化 Windows Server 2022 Full AMI

下表列出 Amazon EKS 最佳化 Windows Server 2022 Full AMI 的目前和先前版本。

Example

Kubernetes 1.33 版

AMI 版本	kubelet 版本	容器化版本	csi-proxy 版本	版本備註
1.33-2025.07.16	1.33.1	1.7.27	1.2.1	
1.33-2025.06.13	1.33.1	1.7.27	1.2.1	
1.33-2025.05.17	1.33.1	1.7.27	1.2.1	

Kubernetes 1.32 版

AMI 版本	kubelet 版本	容器化版本	csi-proxy 版本	版本備註
1.32-2025.07.16	1.32.5	1.7.27	1.2.1	
1.32-2025.06.13	1.32.5	1.7.27	1.2.1	升級至 containrd 1.7.27
1.32-2025.05.17	1.32.5	1.7.20	1.2.1	
1.32-2025.04.14	1.32.1	1.7.20	1.1.3	
1.32-2025.03.14	1.32.1	1.7.20	1.1.3	

AMI 版本	kubelet 版本	容器化版本	csi-proxy 版本	版本備註
1.32-2025.02.18	1.32.1	1.7.20	1.1.3	
1.32-2025.01.01	1.32.0	1.7.20	1.1.3	

Kubernetes 1.31 版

AMI 版本	kubelet 版本	容器化版本	csi-proxy 版本	版本備註
1.31-2025.07.16	1.31.9	1.7.27	1.2.1	
1.31-2025.06.13	1.31.9	1.7.27	1.2.1	已將 containerd 升級至 1.7.27。
1.31-2025.05.17	1.31.9	1.7.20	1.2.1	
1.31-2025.04.14	1.31.5	1.7.20	1.1.3	
1.31-2025.03.14	1.31.5	1.7.20	1.1.3	
1.31-2025.02.15	1.31.5	1.7.20	1.1.3	
1.31-2025.01.15	1.31.4	1.7.20	1.1.3	
1.31-2025.01.01	1.31.4	1.7.20	1.1.3	包含 CVE-2024-

AMI 版本	kubelet 版本	容器化版本	csi-proxy 版本	版本備註
				9042 的修補程式。
1.31-2024.12.13	1.31.3	1.7.20	1.1.3	
1.31-2024.11.12	1.31.1	1.7.20	1.1.3	
1.31-2024.10.08	1.31.1	1.7.20	1.1.3	
1.31-2024.10.01	1.31.1	1.7.20	1.1.3	
1.31-2024.09.10	1.31.0	1.7.20	1.1.3	

Kubernetes 1.30 版

AMI 版本	kubelet 版本	容器化版本	csi-proxy 版本	版本備註
1.30-2025.07.16	1.30.13	1.7.27	1.2.1	
1.30-2025.06.13	1.30.13	1.7.27	1.2.1	已將 containerd 升級至 1.7.27。
1.30-2025.05.17	1.30.13	1.7.20	1.2.1	
1.30-2025.04.14	1.30.9	1.7.20	1.1.3	

AMI 版本	kubelet 版本	容器化版本	csi-proxy 版本	版本備註
1.30-2025.03.14	1.30.9	1.7.20	1.1.3	已將 contianerd 升級至 1.7.20。
1.30-2025.02.15	1.30.9	1.7.14	1.1.3	
1.30-2025.01.15	1.30.8	1.7.14	1.1.3	
1.30-2025.01.01	1.30.8	1.7.14	1.1.3	包含 CVE-2024-9042 的修補程式。
1.30-2024.12.11	1.30.7	1.7.14	1.1.3	
1.30-2024.11.12	1.30.4	1.7.14	1.1.3	
1.30-2024.10.08	1.30.4	1.7.14	1.1.3	
1.30-2024.09.10	1.30.2	1.7.14	1.1.3	
1.30-2024.08.13	1.30.2	1.7.14	1.1.3	
1.30-2024.07.10	1.30.2	1.7.14	1.1.2	包含 CVE-2024-5321 的修補程式。

AMI 版本	kubelet 版本	容器化版本	csi-proxy 版本	版本備註
1.30-2024.06.17	1.30.0	1.7.14	1.1.2	已將 containerd 升級至 1.7.14。
1.30-2024.05.15	1.30.0	1.6.28	1.1.2	

Kubernetes 1.29 版

AMI 版本	kubelet 版本	容器化版本	csi-proxy 版本	版本備註
1.29-2025.07.16	1.29.15	1.7.27	1.2.1	
1.29-2025.06.13	1.29.15	1.7.27	1.2.1	已將 containerd 升級至 1.7.27。
1.29-2025.05.17	1.29.15	1.7.20	1.2.1	
1.29-2025.04.14	1.29.13	1.7.20	1.1.3	
1.29-2025.03.14	1.29.13	1.7.20	1.1.3	已將 containerd 升級至 1.7.20。
1.29-2025.02.15	1.29.13	1.7.14	1.1.3	
1.29-2025.01.15	1.29.12	1.7.14	1.1.3	

AMI 版本	kubelet 版本	容器化版本	csi-proxy 版本	版本備註
1.29-2025.01.01	1.29.12	1.7.14	1.1.3	包含 CVE-2024-9042 的修補程式。
1.29-2024.12.11	1.29.10	1.7.14	1.1.3	
1.29-2024.11.12	1.29.8	1.7.14	1.1.3	
1.29-2024.10.08	1.29.8	1.7.14	1.1.3	
1.29-2024.09.10	1.29.6	1.7.14	1.1.3	
1.29-2024.08.13	1.29.6	1.7.14	1.1.3	
1.29-2024.07.10	1.29.6	1.7.11	1.1.2	包含 CVE-2024-5321 的修補程式。
1.29-2024.06.17	1.29.3	1.7.11	1.1.2	
1.29-2024.05.15	1.29.3	1.7.11	1.1.2	已將 containerd 升級至 1.7.11。已將 kubelet 升級至 1.29.3。

AMI 版本	kubelet 版本	容器化版本	csi-proxy 版本	版本備註
1.29-2024.04.09	1.29.0	1.6.28	1.1.2	已將 containerd 升級至 1.6.28。csi-proxy 使用 重建 CNI 和 golang 1.22.1。
1.29-2024.03.12	1.29.0	1.6.25	1.1.2	
1.29-2024.02.13	1.29.0	1.6.25	1.1.2	
1.29-2024.02.06	1.29.0	1.6.25	1.1.2	修正kubelet垃圾回收程序錯誤刪除暫停映像的錯誤。
1.29-2024.01.09	1.29.0	1.6.18	1.1.2	

Kubernetes 1.28 版

AMI 版本	kubelet 版本	容器化版本	csi-proxy 版本	版本備註
1.28-2025.07.16	1.28.15	1.7.27	1.2.1	
1.28-2025.06.13	1.28.15	1.7.27	1.2.1	已將 containerd 升級至 1.7.27。

AMI 版本	kubelet 版本	容器化版本	csi-proxy 版本	版本備註
1.28-2025.05.17	1.28.15	1.7.20	1.2.1	
1.28-2025.04.14	1.28.15	1.7.20	1.1.3	
1.28-2025.03.14	1.28.15	1.7.20	1.1.3	已將 containerd 升級至 1.7.20。
1.28-2025.02.15	1.28.15	1.7.14	1.1.3	
1.28-2025.01.15	1.28.15	1.7.14	1.1.3	
1.28-2025.01.01	1.28.15	1.7.14	1.1.3	包含 CVE-2024-9042 的修補程式。
1.28-2024.12.11	1.28.15	1.7.14	1.1.3	
1.28-2024.11.12	1.28.13	1.7.14	1.1.3	
1.28-2024.10.08	1.28.13	1.7.14	1.1.3	
1.28-2024.09.10	1.28.11	1.7.14	1.1.3	
1.28-2024.08.13	1.28.11	1.7.14	1.1.3	

AMI 版本	kubelet 版本	容器化版本	csi-proxy 版本	版本備註
1.28-2024.07.10	1.28.11	1.7.11	1.1.2	包含 CVE-2024-5321 的修補程式。
1.28-2024.06.17	1.28.8	1.7.11	1.1.2	已將 containerd 升級至 1.7.11。
1.28-2024.05.14	1.28.8	1.6.28	1.1.2	已將 containerd 升級至 1.6.28。已將 kubelet 升級至 1.28.8。
1.28-2024.04.09	1.28.5	1.6.25	1.1.2	已將 containerd 升級至 1.6.25。csi-proxy 使用 重建 CNI 和 golang 1.22.1。
1.28-2024.03.12	1.28.5	1.6.18	1.1.2	
1.28-2024.02.13	1.28.5	1.6.18	1.1.2	
1.28-2024.01.09	1.28.5	1.6.18	1.1.2	

AMI 版本	kubelet 版本	容器化版本	csi-proxy 版本	版本備註
1.28-2023.12.12	1.28.3	1.6.18	1.1.2	
1.28-2023.11.14	1.28.3	1.6.18	1.1.2	包含 CVE-2023-5528 的修補程式。
1.28-2023.10.19	1.28.2	1.6.18	1.1.2	已將 containerd 升級至 1.6.18。已新增引導指令碼環境變數 (SERVICE_IPV4_CIDR 和 EXCLUDED_SNAT_CIDRS)。
1.28-2023-09.27	1.28.2	1.6.6	1.1.2	已修正 kubelet 的 安全性公告。
1.28-2023.09.12	1.28.1	1.6.6	1.1.2	

Kubernetes 1.27 版

AMI 版本	kubelet 版本	容器化版本	csi-proxy 版本	版本備註
1.27-2025.07.16	1.27.16	1.7.27	1.2.1	

AMI 版本	kubelet 版本	容器化版本	csi-proxy 版本	版本備註
1.27-2025.06.13	1.27.16	1.7.27	1.2.1	已將 containerd 升級至 1.7.27。
1.27-2025.05.17	1.27.16	1.7.20	1.2.1	
1.27-2025.04.14	1.27.16	1.7.20	1.1.3	
1.27-2025.03.14	1.27.16	1.7.20	1.1.3	已將 containerd 升級至 1.7.20。
1.27-2025.02.15	1.27.16	1.7.14	1.1.3	
1.27-2025.01.15	1.27.16	1.7.14	1.1.3	
1.27-2025.01.01	1.27.16	1.7.14	1.1.3	包含 CVE-2024-9042 的修補程式。
1.27-2024.12.11	1.27.16	1.7.14	1.1.3	
1.27-2024.11.12	1.27.16	1.7.14	1.1.3	
1.27-2024.10.08	1.27.16	1.7.14	1.1.3	

AMI 版本	kubelet 版本	容器化版本	csi-proxy 版本	版本備註
1.27-2024.09.10	1.27.15	1.7.14	1.1.3	
1.27-2024.08.13	1.27.15	1.7.14	1.1.3	
1.27-2024.07.10	1.27.15	1.7.11	1.1.2	包含 CVE-2024-5321 的修補程式。
1.27-2024.06.17	1.27.12	1.7.11	1.1.2	已將 containerd 升級至 1.7.11。
1.27-2024.05.14	1.27.12	1.6.28	1.1.2	已將 containerd 升級至 1.6.28。已將 kubelet 升級至 1.27.12。
1.27-2024.04.09	1.27.9	1.6.25	1.1.2	已將 containerd 升級至 1.6.25。csi-proxy 使用 重建 CNI 和 golang 1.22.1。
1.27-2024.03.12	1.27.9	1.6.18	1.1.2	

AMI 版本	kubelet 版本	容器化版本	csi-proxy 版本	版本備註
1.27-2024.02.13	1.27.9	1.6.18	1.1.2	
1.27-2024.01.09	1.27.9	1.6.18	1.1.2	
1.27-2023.12.12	1.27.7	1.6.18	1.1.2	
1.27-2023.11.14	1.27.7	1.6.18	1.1.2	包含 CVE-2023-5528 的修補程式。
1.27-2023.10.19	1.27.6	1.6.18	1.1.2	已將 containerd 升級至 1.6.18。已新增引導指令碼環境變數 (SERVICE_IPV4_CIDR 和 EXCLUDED_SNAT_CIDRS)。
1.27-2023-09.27	1.27.6	1.6.6	1.1.2	已修正 kubelet 的 安全性公告。

AMI 版本	kubelet 版本	容器化版本	csi-proxy 版本	版本備註
1.27-2023.09.12	1.27.4	1.6.6	1.1.2	升級 Amazon VPC CNI 外掛程式以使用 Kubernetes 連接器二進位檔，從 Kubernetes API 伺服器取得 Pod IP 地址。合併的提取請求 #100。
1.27-2023.08.17	1.27.4	1.6.6	1.1.2	包含 CVE-2023-3676 、CVE-2023-3893 和 CVE-2023-3955 的修補程式。
1.27-2023.08.08	1.27.3	1.6.6	1.1.1	
1.27-2023.07.11	1.27.3	1.6.6	1.1.1	
1.27-2023.06.20	1.27.1	1.6.6	1.1.1	已解決導致 DNS 字尾搜尋清單填入錯誤的問題。
1.27-2023.06.14	1.27.1	1.6.6	1.1.1	加入了對 CNI 中主機連接埠映射的支援。合併提取請求 #93。

AMI 版本	kubelet 版本	容器化版本	csi-proxy 版本	版本備註
1.27-2023.06.06	1.27.1	1.6.6	1.1.1	修正 containers-roadmap 問題 #2042，此為導致節點無法提取私有 Amazon ECR 映像的問題。
1.27-2023.05.18	1.27.1	1.6.6	1.1.1	

Kubernetes 1.26 版

AMI 版本	kubelet 版本	容器化版本	csi-proxy 版本	版本備註
1.26-2025.05.17	1.26.15	1.7.20	1.2.1	
1.26-2025.04.14	1.26.15	1.7.20	1.1.3	
1.26-2025.03.14	1.26.15	1.7.20	1.1.3	已將 containerd 升級至 1.7.20。
1.26-2025.02.15	1.26.15	1.7.14	1.1.3	
1.26-2025.01.15	1.26.15	1.7.14	1.1.3	
1.26-2024.12.11	1.26.15	1.7.14	1.1.3	

AMI 版本	kubelet 版本	容器化版本	csi-proxy 版本	版本備註
1.26-2024.11.12	1.26.15	1.7.14	1.1.3	
1.26-2024.10.08	1.26.15	1.7.14	1.1.3	
1.26-2024.09.10	1.26.15	1.7.14	1.1.3	
1.26-2024.08.13	1.26.15	1.7.14	1.1.3	
1.26-2024.07.10	1.26.15	1.7.11	1.1.2	包含 CVE-2024-5321 的修補程式。
1.26-2024.06.17	1.26.15	1.7.11	1.1.2	已將 containerd 升級至 1.7.11。
1.26-2024.05.14	1.26.15	1.6.28	1.1.2	已將 containerd 升級至 1.6.28。已將 kubelet 升級至 1.26.15。

AMI 版本	kubelet 版本	容器化版本	csi-proxy 版本	版本備註
1.26-2024.04.09	1.26.12	1.6.25	1.1.2	已將 containerd 升級至 1.6.25。csi-proxy 使用 重建 CNI 和 golang 1.22.1。
1.26-2024.03.12	1.26.12	1.6.18	1.1.2	
1.26-2024.02.13	1.26.12	1.6.18	1.1.2	
1.26-2024.01.09	1.26.12	1.6.18	1.1.2	
1.26-2023.12.12	1.26.10	1.6.18	1.1.2	
1.26-2023.11.14	1.26.10	1.6.18	1.1.2	包含 CVE-2023-5528 的修補程式。

AMI 版本	kubelet 版本	容器化版本	csi-proxy 版本	版本備註
1.26-2023.10.19	1.26.9	1.6.18	1.1.2	已將 containerd 升級至 1.6.18。已將 kubelet 升級至 1.26.9。已新增引導指令碼環境變數 (SERVICE_IPV4_CIDR 和 EXCLUDED_SNAT_CIDRS)。
1.26-2023.09.12	1.26.7	1.6.6	1.1.2	升級 Amazon VPC CNI 外掛程式以使用 Kubernetes 連接器二進位檔，從 Kubernetes API 伺服器取得 Pod IP 地址。合併的提取請求 #100。
1.26-2023.08.17	1.26.7	1.6.6	1.1.2	包含 CVE-2023-3676 、CVE-2023-3893 和 CVE-2023-3955 的修補程式。
1.26-2023.08.08	1.26.6	1.6.6	1.1.1	

AMI 版本	kubelet 版本	容器化版本	csi-proxy 版本	版本備註
1.26-2023.07.11	1.26.6	1.6.6	1.1.1	
1.26-2023.06.20	1.26.4	1.6.6	1.1.1	已解決導致 DNS 字尾搜尋清單填入錯誤的問題。
1.26-2023.06.14	1.26.4	1.6.6	1.1.1	升級 Kubernetes 至 1.26.4。加入了對 CNI 中主機連接埠映射的支援。合併提取請求 #93。
1.26-2023.05.09	1.26.2	1.6.6	1.1.1	修正節點重新啟動後，在 Pod 上造成網路連線問題 #1126 的錯誤。引入新的引導指令碼組態參數 (ExcludedSnatCIDRs)。
1.26-2023.04.26	1.26.2	1.6.6	1.1.1	
1.26-2023.04.11	1.26.2	1.6.6	1.1.1	新增了服務當機時 kubelet 和 kube-proxy 的復原機制。
1.26-2023.03.24	1.26.2	1.6.6	1.1.1	

Amazon EKS 最佳化 Windows Server 2019 Core AMI

下表列出 Amazon EKS 最佳化 Windows Server 2019 Core AMI 的目前和先前版本。

Example

Kubernetes 1.33 版

AMI 版本	kubelet 版本	containerd 版本	csi-proxy 版本	版本備註
1.33-2025.07.16	1.33.1	1.7.27	1.2.1	
1.33-2025.06.13	1.33.1	1.7.27	1.2.1	
1.33-2025.05.17	1.33.1	1.7.27	1.2.1	

Kubernetes 1.32 版

AMI 版本	kubelet 版本	containerd 版本	csi-proxy 版本	版本備註
1.32-2025.07.16	1.32.5	1.7.27	1.2.1	
1.32-2025.06.13	1.32.5	1.7.27	1.2.1	已將 containerd 升級至 1.7.27。
1.32-2025.05.17	1.32.5	1.7.20	1.2.1	
1.32-2025.04.14	1.32.1	1.7.20	1.1.3	
1.32-2025.03.14	1.32.1	1.7.20	1.1.3	

AMI 版本	kubelet 版本	containerd 版本	csi-proxy 版本	版本備註
1.32-2025.02.18	1.32.1	1.7.20	1.1.3	
1.32-2025.01.15	1.32.4	1.7.20	1.1.3	

Kubernetes 1.31 版

AMI 版本	kubelet 版本	containerd 版本	csi-proxy 版本	版本備註
1.31-2025.07.16	1.31.9	1.7.27	1.2.1	
1.31-2025.06.13	1.31.9	1.7.27	1.2.1	已將 containerd 升級至 1.7.27。
1.31-2025.05.17	1.31.9	1.7.20	1.2.1	
1.31-2025.04.14	1.31.5	1.7.20	1.1.3	
1.31-2025.03.14	1.31.5	1.7.20	1.1.3	
1.31-2025.02.15	1.31.5	1.7.20	1.1.3	
1.31-2025.01.15	1.31.4	1.7.20	1.1.3	
1.31-2025.01.01	1.31.4	1.7.20	1.1.3	包含 CVE-2024-

AMI 版本	kubelet 版本	containerd 版本	csi-proxy 版本	版本備註
				9042 的修補程式。
1.31-2024.12.13	1.31.3	1.7.20	1.1.3	
1.31-2024.11.12	1.31.1	1.7.20	1.1.3	
1.31-2024.10.08	1.31.1	1.7.20	1.1.3	
1.31-2024.10.01	1.31.1	1.7.20	1.1.3	
1.31-2024.09.10	1.31.0	1.7.20	1.1.3	

Kubernetes 1.30 版

AMI 版本	kubelet 版本	containerd 版本	csi-proxy 版本	版本備註
1.30-2025.07.16	1.30.13	1.7.27	1.2.1	
1.30-2025.06.13	1.30.13	1.7.27	1.2.1	已將 containerd 升級至 1.7.27。
1.30-2025.05.17	1.30.13	1.7.20	1.2.1	
1.30-2025.04.14	1.30.9	1.7.20	1.1.3	

AMI 版本	kubelet 版本	containerd 版本	csi-proxy 版本	版本備註
1.30-2025.03.14	1.30.9	1.7.20	1.1.3	已將 containerd 升級至 1.7.20。
1.30-2025-02-15	1.30.9	1.7.14	1.1.3	
1.30-2025.01.15	1.30.8	1.7.14	1.1.3	
1.30-2025.01.01	1.30.8	1.7.14	1.1.3	包含 CVE-2024-9042 的修補程式。
1.30-2024.12.11	1.30.7	1.7.14	1.1.3	
1.30-2024.11.12	1.30.4	1.7.14	1.1.3	
1.30-2024.10.08	1.30.4	1.7.14	1.1.3	
1.30-2024.09.10	1.30.2	1.7.14	1.1.3	
1.30-2024.08.13	1.30.2	1.7.14	1.1.3	
1.30-2024.07.10	1.30.2	1.7.14	1.1.2	包含 CVE-2024-5321 的修補程式。

AMI 版本	kubelet 版本	containerd 版本	csi-proxy 版本	版本備註
1.30-2024.06.17	1.30.0	1.7.14	1.1.2	已將 containerd 升級至 1.7.14。
1.30-2024.05.15	1.30.0	1.6.28	1.1.2	

Kubernetes 1.29 版

AMI 版本	kubelet 版本	containerd 版本	csi-proxy 版本	版本備註
1.29-2025.07.16	1.29.15	1.7.27	1.2.1	
1.29-2025.06.13	1.29.15	1.7.27	1.2.1	已將 containerd 升級至 1.7.27。
1.29-2025.05.17	1.29.15	1.7.20	1.2.1	
1.29-2025.04.14	1.29.13	1.7.20	1.1.3	
1.29-2025.03.14	1.29.13	1.7.20	1.1.3	已將 containerd 升級至 1.7.20。
1.29-2025.02.15	1.29.13	1.7.14	1.1.3	
1.29-2025.01.15	1.29.12	1.7.14	1.1.3	

AMI 版本	kubelet 版本	containerd 版本	csi-proxy 版本	版本備註
1.29-2025.01.01	1.29.12	1.7.14	1.1.3	包含 CVE-2024-9042 的修補程式。
1.29-2024.12.11	1.29.10	1.7.14	1.1.3	
1.29-2024.11.12	1.29.8	1.7.14	1.1.3	
1.29-2024.10.08	1.29.8	1.7.14	1.1.3	
1.29-2024.09.10	1.29.6	1.7.14	1.1.3	
1.29-2024.08.13	1.29.6	1.7.14	1.1.3	
1.29-2024.07.10	1.29.6	1.7.11	1.1.2	包含 CVE-2024-5321 的修補程式。
1.29-2024.06.17	1.29.3	1.7.11	1.1.2	
1.29-2024.05.15	1.29.3	1.7.11	1.1.2	已將 containerd 升級至 1.7.11。已將 kubelet 升級至 1.29.3。

AMI 版本	kubelet 版本	containerd 版本	csi-proxy 版本	版本備註
1.29-2024.04.09	1.29.0	1.6.28	1.1.2	已將 containerd 升級至 1.6.28。csi-proxy 使用 重建 CNI 和 golang 1.22.1。
1.29-2024.03.13	1.29.0	1.6.25	1.1.2	
1.29-2024.02.13	1.29.0	1.6.25	1.1.2	
1.29-2024.02.06	1.29.0	1.6.25	1.1.2	已修正kubelet垃圾回收程序錯誤刪除暫停映像的錯誤。
1.29-2024.01.09	1.29.0	1.6.18	1.1.2	

Kubernetes 1.28 版

AMI 版本	kubelet 版本	containerd 版本	csi-proxy 版本	版本備註
1.28-2025.07.16	1.28.15	1.7.27	1.2.1	
1.28-2025.06.13	1.28.15	1.7.20	1.2.1	已將 containerd 升級至 1.7.27。

AMI 版本	kubelet 版本	containerd 版本	csi-proxy 版本	版本備註
1.28-2025.05.17	1.28.15	1.7.20	1.2.1	
1.28-2025.04.14	1.28.15	1.7.20	1.1.3	
1.28-2025.03.14	1.28.15	1.7.20	1.1.3	已將 containerd 升級至 1.7.20。
1.28-2025.02.15	1.28.15	1.7.14	1.1.3	
1.28-2025-01-15	1.28.15	1.7.14	1.1.3	
1.28-2025-01-01	1.28.15	1.7.14	1.1.3	包含 CVE-2024-9042 的修補程式。
1.28-2024.12.11	1.28.15	1.7.14	1.1.3	
1.28-2024.11.12	1.28.13	1.7.14	1.1.3	
1.28-2024.10.08	1.28.13	1.7.14	1.1.3	
1.28-2024.09.10	1.28.11	1.7.14	1.1.3	
1.28-2024.08.13	1.28.11	1.7.14	1.1.3	

AMI 版本	kubelet 版本	containerd 版本	csi-proxy 版本	版本備註
1.28-2024.07.10	1.28.11	1.7.11	1.1.2	包含 CVE-2024-5321 的修補程式。
1.28-2024.06.17	1.28.8	1.7.11	1.1.2	已將 containerd 升級至 1.7.11。
1.28-2024.05.14	1.28.8	1.6.28	1.1.2	已將 containerd 升級至 1.6.28。已將 kubelet 升級至 1.28.8。
1.28-2024.04.09	1.28.5	1.6.25	1.1.2	已將 containerd 升級至 1.6.25。csi-proxy 使用 重建 CNI 和 golang 1.22.1。
1.28-2024.03.13	1.28.5	1.6.18	1.1.2	
1.28-2024.02.13	1.28.5	1.6.18	1.1.2	
1.28-2024.01.09	1.28.5	1.6.18	1.1.2	

AMI 版本	kubelet 版本	containerd 版本	csi-proxy 版本	版本備註
1.28-2023.12.12	1.28.3	1.6.18	1.1.2	
1.28-2023.11.14	1.28.3	1.6.18	1.1.2	包含 CVE-2023-5528 的修補程式。
1.28-2023.10.19	1.28.2	1.6.18	1.1.2	已將 containerd 升級至 1.6.18。已新增引導指令碼環境變數 (SERVICE_IPV4_CIDR 和 EXCLUDED_SNAT_CIDRS)。
1.28-2023-09.27	1.28.2	1.6.6	1.1.2	已修正 kubelet 的 安全性公告。
1.28-2023.09.12	1.28.1	1.6.6	1.1.2	

Kubernetes 1.27 版

AMI 版本	kubelet 版本	containerd 版本	csi-proxy 版本	版本備註
1.27-2025.07.16	1.27.16	1.7.27	1.2.1	

AMI 版本	kubelet 版本	containerd 版本	csi-proxy 版本	版本備註
1.27-2025.06.13	1.27.16	1.7.27	1.2.1	已將 containerd 升級至 1.7.27。
1.27-2025.05.17	1.27.16	1.7.20	1.2.1	
1.27-2025.04.14	1.27.16	1.7.20	1.1.3	
1.27-2025.03.14	1.27.16	1.7.20	1.1.3	已將 containerd 升級至 1.7.20。
1.27-2025.02.15	1.27.16	1.7.14	1.1.3	
1.27-2025.01.15	1.27.16	1.7.14	1.1.3	
1.27-2025.01.01	1.27.16	1.7.14	1.1.3	包含 CVE-2024-9042 的修補程式。
1.27-2024.12.11	1.27.16	1.7.14	1.1.3	
1.27-2024.11.12	1.27.16	1.7.14	1.1.3	
1.27-2024.10.08	1.27.16	1.7.14	1.1.3	

AMI 版本	kubelet 版本	containerd 版本	csi-proxy 版本	版本備註
1.27-2024.09.10	1.27.15	1.7.14	1.1.3	
1.27-2024.08.13	1.27.15	1.7.14	1.1.3	
1.27-2024.07.10	1.27.15	1.7.11	1.1.2	包含 CVE-2024-5321 的修補程式。
1.27-2024.06.17	1.27.12	1.7.11	1.1.2	已將 containerd 升級至 1.7.11。
1.27-2024.05.14	1.27.12	1.6.28	1.1.2	已將 containerd 升級至 1.6.28。已將 kubelet 升級至 1.27.12。
1.27-2024.04.09	1.27.9	1.6.25	1.1.2	已將 containerd 升級至 1.6.25。csi-proxy 使用 重建 CNI 和 golang 1.22.1。
1.27-2024.03.13	1.27.9	1.6.18	1.1.2	

AMI 版本	kubelet 版本	containerd 版本	csi-proxy 版本	版本備註
1.27-2024.02.13	1.27.9	1.6.18	1.1.2	
1.27-2024.01.09	1.27.9	1.6.18	1.1.2	
1.27-2023.12.12	1.27.7	1.6.18	1.1.2	
1.27-2023.11.14	1.27.7	1.6.18	1.1.2	包含 CVE-2023-5528 的修補程式。
1.27-2023.10.19	1.27.6	1.6.18	1.1.2	已將 containerd 升級至 1.6.18。已新增引導指令碼環境變數 (SERVICE_IPV4_CIDR 和 EXCLUDED_SNAT_CIDRS)。
1.27-2023-09.27	1.27.6	1.6.6	1.1.2	已修正 kubelet 的 安全性公告。

AMI 版本	kubelet 版本	containerd 版本	csi-proxy 版本	版本備註
1.27-2023.09.12	1.27.4	1.6.6	1.1.2	升級 Amazon VPC CNI 外掛程式以使用 Kubernetes 連接器二進位檔，從 Kubernetes API 伺服器取得 Pod IP 地址。合併的提取請求 #100。
1.27-2023.08.17	1.27.4	1.6.6	1.1.2	包含 CVE-2023-3676 、CVE-2023-3893 和 CVE-2023-3955 的修補程式。
1.27-2023.08.08	1.27.3	1.6.6	1.1.1	
1.27-2023.07.11	1.27.3	1.6.6	1.1.1	
1.27-2023.06.20	1.27.1	1.6.6	1.1.1	已解決導致 DNS 字尾搜尋清單填入錯誤的問題。
1.27-2023.06.14	1.27.1	1.6.6	1.1.1	加入了對 CNI 中主機連接埠映射的支援。合併提取請求 #93。

AMI 版本	kubelet 版本	containerd 版本	csi-proxy 版本	版本備註
1.27-2023.06.06	1.27.1	1.6.6	1.1.1	修正 containers-roadmap 問題 #2042，此為導致節點無法提取私有 Amazon ECR 映像的問題。
11.27-2023.05.18	1.27.1	1.6.6	1.1.1	

Kubernetes 1.26 版

AMI 版本	kubelet 版本	containerd 版本	csi-proxy 版本	版本備註
1.26-2025.05.17	1.26.15	1.7.20	1.2.1	
1.26-2025.04.14	1.26.15	1.7.20	1.1.3	
1.26-2025.03.14	1.26.15	1.7.20	1.1.3	已將 containerd 升級至 1.7.20。
1.26-2025.02.15	1.26.15	1.7.14	1.1.3	
1.26-2025.01.15	1.26.15	1.7.14	1.1.3	
1.26-2024.12.11	1.26.15	1.7.14	1.1.3	

AMI 版本	kubelet 版本	containerd 版本	csi-proxy 版本	版本備註
1.26-2024.11.12	1.26.15	1.7.14	1.1.3	
1.26-2024.10.09	1.26.15	1.7.14	1.1.3	
1.26-2024.09.10	1.26.15	1.7.14	1.1.3	
1.26-2024.08.13	1.26.15	1.7.14	1.1.3	
1.26-2024.07.10	1.26.15	1.7.11	1.1.2	包含 CVE-2024-5321 的修補程式。
1.26-2024.06.17	1.26.15	1.7.11	1.1.2	已將 containerd 升級至 1.7.11。
1.26-2024.05.14	1.26.15	1.6.28	1.1.2	已將 containerd 升級至 1.6.28。已將 kubelet 升級至 1.26.15。

AMI 版本	kubelet 版本	containerd 版本	csi-proxy 版本	版本備註
1.26-2024.04.09	1.26.12	1.6.25	1.1.2	已將 containerd 升級至 1.6.25。csi-proxy 使用 重建 CNI 和 golang 1.22.1。
1.26-2024.03.13	1.26.12	1.6.18	1.1.2	
1.26-2024.02.13	1.26.12	1.6.18	1.1.2	
1.26-2024.01.09	1.26.12	1.6.18	1.1.2	
1.26-2023.12.12	1.26.10	1.6.18	1.1.2	
1.26-2023.11.14	1.26.10	1.6.18	1.1.2	包含 CVE-2023-5528 的修補程式。

AMI 版本	kubelet 版本	containerd 版本	csi-proxy 版本	版本備註
1.26-2023.10.19	1.26.9	1.6.18	1.1.2	已將 containerd 升級至 1.6.18。已將 kubelet 升級至 1.26.9。已新增引導指令碼環境變數 (SERVICE_IPV4_CIDR 和 EXCLUDED_SNAT_CIDRS)。
1.26-2023.09.12	1.26.7	1.6.6	1.1.2	升級 Amazon VPC CNI 外掛程式以使用 Kubernetes 連接器二進位檔，從 Kubernetes API 伺服器取得 Pod IP 地址。合併的提取請求 #100。
1.26-2023.08.17	1.26.7	1.6.6	1.1.2	包含 CVE-2023-3676 、CVE-2023-3893 和 CVE-2023-3955 的修補程式。
1.26-2023.08.08	1.26.6	1.6.6	1.1.1	

AMI 版本	kubelet 版本	containerd 版本	csi-proxy 版本	版本備註
1.26-2023.07.11	1.26.6	1.6.6	1.1.1	
1.26-2023.06.20	1.26.4	1.6.6	1.1.1	已解決導致 DNS 字尾搜尋清單填入錯誤的問題。
1.26-2023.06.14	1.26.4	1.6.6	1.1.1	已將 Kubernetes 升級到 1.26.4。加入了對 CNI 中主機連接埠映射的支援。合併提取請求 #93。
1.26-2023.05.09	1.26.2	1.6.6	1.1.1	修正節點重新啟動後，在 Pod 上造成網路連線問題 #1126 的錯誤。引入新的引導指令碼組態參數 (ExcludedSnatCIDRs)。
1.26-2023.04.26	1.26.2	1.6.6	1.1.1	
1.26-2023.04.11	1.26.2	1.6.6	1.1.1	新增了服務當機時 kubelet 和 kube-proxy 的復原機制。
1.26-2023.03.24	1.26.2	1.6.6	1.1.1	

Amazon EKS 最佳化 Windows Server 2019 Full AMI

下表列出 Amazon EKS 最佳化 Windows Server 2019 Full AMI 的目前和先前版本。

Example

Kubernetes 1.33 版

AMI 版本	kubelet 版本	containerd 版本	csi-proxy 版本	版本備註
1.33-2025.07.16	1.33.1	1.7.27	1.2.1	
1.33-2025.06.13	1.33.1	1.7.27	1.2.1	
1.33-2025.05.17	1.33.1	1.7.27	1.2.1	

Kubernetes 1.32 版

AMI 版本	kubelet 版本	containerd 版本	csi-proxy 版本	版本備註
1.32-2025.07.16	1.32.5	1.7.27	1.2.1	
1.32-2025.06.13	1.32.5	1.7.27	1.2.1	已將 containerd 升級至 1.7.27。
1.32-2025.05.17	1.32.5	1.7.20	1.2.1	
1.32-2025.04.14	1.32.1	1.7.20	1.1.3	
1.32-2025.03.14	1.32.1	1.7.20	1.1.3	

AMI 版本	kubelet 版本	containerd 版本	csi-proxy 版本	版本備註
1.32-2025.02.18	1.32.1	1.7.20	1.1.3	
1.32-2025.01.15	1.32.0	1.7.20	1.1.3	

Kubernetes 1.31 版

AMI 版本	kubelet 版本	containerd 版本	csi-proxy 版本	版本備註
1.31-2025.07.16	1.31.9	1.7.27	1.2.1	
1.31-2025.06.13	1.31.9	1.7.27	1.2.1	已將 containerd 升級至 1.7.27。
1.31-2025.05.17	1.31.9	1.7.20	1.2.1	
1.31-2025.04.14	1.31.5	1.7.20	1.1.3	
1.31-2025.03.14	1.31.5	1.7.20	1.1.3	
1.31-2025.02.15	1.31.5	1.7.20	1.1.3	
1.31-2025.01.15	1.31.4	1.7.20	1.1.3	
1.31-2025.01.01	1.31.4	1.7.20	1.1.3	包含 CVE-2024-

AMI 版本	kubelet 版本	containerd 版本	csi-proxy 版本	版本備註
				9042 的修補程式。
1.31-2024.12.13	1.31.3	1.7.20	1.1.3	
1.31-2024.11.12	1.31.1	1.7.20	1.1.3	
1.31-2024.10.08	1.31.1	1.7.20	1.1.3	
1.31-2024.10.01	1.31.1	1.7.20	1.1.3	
1.31-2024.09.10	1.31.0	1.7.20	1.1.3	

Kubernetes 1.30 版

AMI 版本	kubelet 版本	containerd 版本	csi-proxy 版本	版本備註
1.30-2025.07.16	1.30.13	1.7.27	1.2.1	
1.30-2025.06.13	1.30.13	1.7.27	1.2.1	已將 containerd 升級至 1.7.27。
1.30-2025.05.17	1.30.13	1.7.20	1.2.1	
1.30-2025.04.14	1.30.9	1.7.20	1.1.3	

AMI 版本	kubelet 版本	containerd 版本	csi-proxy 版本	版本備註
1.30-2025.03.14	1.30.9	1.7.20	1.1.3	已將 containerd 升級至 1.7.20。
1.30-2025.02.15	1.30.9	1.7.14	1.1.3	
1.30-2025.01.15	1.30.8	1.7.14	1.1.3	
1.30-2025.01.01	1.30.8	1.7.14	1.1.3	包含 CVE-2024-9042 的修補程式。
1.30-2024.12.11	1.30.7	1.7.14	1.1.3	
1.30-2024.11.12	1.30.4	1.7.14	1.1.3	
1.30-2024.10.08	1.30.4	1.7.14	1.1.3	
1.30-2024.09.10	1.30.2	1.7.14	1.1.3	
1.30-2024.08.13	1.30.2	1.7.14	1.1.3	
1.30-2024.07.10	1.30.2	1.7.14	1.1.2	包含 CVE-2024-5321 的修補程式。

AMI 版本	kubelet 版本	containerd 版本	csi-proxy 版本	版本備註
1.30-2024.06.17	1.30.0	1.7.14	1.1.2	已將 containerd 升級至 1.7.14。
1.30-2024.05.15	1.30.0	1.6.28	1.1.2	

Kubernetes 1.29 版

AMI 版本	kubelet 版本	containerd 版本	csi-proxy 版本	版本備註
1.29-2025.07.16	1.29.15	1.7.27	1.2.1	
1.29-2025.06.13	1.29.15	1.7.27	1.2.1	已將 containerd 升級至 1.7.27。
1.29-2025.05.17	1.29.15	1.7.20	1.2.1	
1.29-2025.04.14	1.29.13	1.7.20	1.1.3	
1.29-2025.03.14	1.29.13	1.7.20	1.1.3	已將 containerd 升級至 1.7.20。
1.29-2025.02.15	1.29.13	1.7.14	1.1.3	
1.29-2025.01.15	1.29.12	1.7.14	1.1.3	

AMI 版本	kubelet 版本	containerd 版本	csi-proxy 版本	版本備註
1.29-2025.01.01	1.29.12	1.7.14	1.1.3	包含 CVE-2024-9042 的修補程式。
1.29-2024.12.11	1.29.10	1.7.14	1.1.3	
1.29-2024.11.12	1.29.8	1.7.14	1.1.3	
1.29-2024.10.08	1.29.8	1.7.14	1.1.3	
1.29-2024.09.10	1.29.6	1.7.14	1.1.3	
1.29-2024.08.13	1.29.6	1.7.14	1.1.3	
1.29-2024.07.10	1.29.6	1.7.11	1.1.2	包含 CVE-2024-5321 的修補程式。
1.29-2024.06.17	1.29.3	1.7.11	1.1.2	
1.29-2024.05.15	1.29.3	1.7.11	1.1.2	已將 containerd 升級至 1.7.11。已將 kubelet 升級至 1.29.3。

AMI 版本	kubelet 版本	containerd 版本	csi-proxy 版本	版本備註
1.29-2024.04.09	1.29.0	1.6.28	1.1.2	已將 containerd 升級至 1.6.28。csi-proxy 使用 重建 CNI 和 golang 1.22.1。
1.29-2024.03.13	1.29.0	1.6.25	1.1.2	
1.29-2024.02.13	1.29.0	1.6.25	1.1.2	
1.29-2024.02.06	1.29.0	1.6.25	1.1.2	已修正kubelet垃圾回收程序錯誤刪除暫停映像的錯誤。
1.29-2024.01.09	1.29.0	1.6.18	1.1.2	

Kubernetes 1.28 版

AMI 版本	kubelet 版本	containerd 版本	csi-proxy 版本	版本備註
1.28-2025.07.16	1.28.15	1.7.27	1.2.1	
1.28-2025.06.13	1.28.15	1.7.27	1.2.1	已將 containerd 升級至 1.7.27。

AMI 版本	kubelet 版本	containerd 版本	csi-proxy 版本	版本備註
1.28-2025.05.17	1.28.15	1.7.20	1.2.1	
1.28-2025.04.14	1.28.15	1.7.20	1.1.3	
1.28-2025.03.14	1.28.15	1.7.20	1.1.3	已將 containerd 升級至 1.7.20。
1.28-2025.02.15	1.28.15	1.7.14	1.1.3	
1.28-2025-01-15	1.28.15	1.7.14	1.1.3	
1.28-2025-01-01	1.28.15	1.7.14	1.1.3	包含 CVE-2024-9042 的修補程式。
1.28-2024.12.11	1.28.15	1.7.14	1.1.3	
1.28-2024.11.12	1.28.13	1.7.14	1.1.3	
1.28-2024.10.08	1.28.13	1.7.14	1.1.3	
1.28-2024.09.10	1.28.11	1.7.14	1.1.3	
1.28-2024.08.13	1.28.11	1.7.14	1.1.3	

AMI 版本	kubelet 版本	containerd 版本	csi-proxy 版本	版本備註
1.28-2024.07.10	1.28.11	1.7.11	1.1.2	包含 CVE-2024-5321 的修補程式。
1.28-2024.06.17	1.28.8	1.7.11	1.1.2	已將 containerd 升級至 1.7.11。
1.28-2024.05.14	1.28.8	1.6.28	1.1.2	已將 containerd 升級至 1.6.28。已將 kubelet 升級至 1.28.8。
1.28-2024.04.09	1.28.5	1.6.25	1.1.2	已將 containerd 升級至 1.6.25。csi-proxy 使用 重建 CNI 和 golang 1.22.1。
1.28-2024.03.13	1.28.5	1.6.18	1.1.2	
1.28-2024.02.13	1.28.5	1.6.18	1.1.2	
1.28-2024.01.09	1.28.5	1.6.18	1.1.2	

AMI 版本	kubelet 版本	containerd 版本	csi-proxy 版本	版本備註
1.28-2023.12.12	1.28.3	1.6.18	1.1.2	
1.28-2023.11.14	1.28.3	1.6.18	1.1.2	包含 CVE-2023-5528 的修補程式。
1.28-2023.10.19	1.28.2	1.6.18	1.1.2	已將 containerd 升級至 1.6.18。已新增引導指令碼環境變數 (SERVICE_IPV4_CIDR 和 EXCLUDED_SNAT_CIDRS)。
1.28-2023-09.27	1.28.2	1.6.6	1.1.2	已修正 kubelet 的 安全性公告。
1.28-2023.09.12	1.28.1	1.6.6	1.1.2	

Kubernetes 1.27 版

AMI 版本	kubelet 版本	containerd 版本	csi-proxy 版本	版本備註
1.27-2025.07.16	1.27.16	1.7.27	1.2.1	

AMI 版本	kubelet 版本	containerd 版本	csi-proxy 版本	版本備註
1.27-2025.06.13	1.27.16	1.7.27	1.2.1	已將 containerd 升級至 1.7.27。
1.27-2025.05.17	1.27.16	1.7.20	1.2.1	
1.27-2025.04.14	1.27.16	1.7.20	1.1.3	
1.27-2025.03.14	1.27.16	1.7.20	1.1.3	已將 containerd 升級至 1.7.20。
1.27-2025.02.15	1.27.16	1.7.14	1.1.3	
1.27-2025.01.15	1.27.16	1.7.14	1.1.3	
1.27-2025.01.01	1.27.16	1.7.14	1.1.3	包含 CVE-2024-9042 的修補程式。
1.27-2024.12.11	1.27.16	1.7.14	1.1.3	
1.27-2024.11.12	1.27.16	1.7.14	1.1.3	
1.27-2024.10.08	1.27.16	1.7.14	1.1.3	

AMI 版本	kubelet 版本	containerd 版本	csi-proxy 版本	版本備註
1.27-2024.09.10	1.27.15	1.7.14	1.1.3	
1.27-2024.08.13	1.27.15	1.7.14	1.1.3	
1.27-2024.07.10	1.27.15	1.7.11	1.1.2	包含 CVE-2024-5321 的修補程式。
1.27-2024.06.17	1.27.12	1.7.11	1.1.2	已將 containerd 升級至 1.7.11。
1.27-2024.05.14	1.27.12	1.6.28	1.1.2	已將 containerd 升級至 1.6.28。已將 kubelet 升級至 1.27.12。
1.27-2024.04.09	1.27.9	1.6.25	1.1.2	已將 containerd 升級至 1.6.25。csi-proxy 使用 重建 CNI 和 golang 1.22.1。
1.27-2024.03.13	1.27.9	1.6.18	1.1.2	

AMI 版本	kubelet 版本	containerd 版本	csi-proxy 版本	版本備註
1.27-2024.02.13	1.27.9	1.6.18	1.1.2	
1.27-2024.01.09	1.27.9	1.6.18	1.1.2	
1.27-2023.12.12	1.27.7	1.6.18	1.1.2	
1.27-2023.11.14	1.27.7	1.6.18	1.1.2	包含 CVE-2023-5528 的修補程式。
1.27-2023.10.19	1.27.6	1.6.18	1.1.2	已將 containerd 升級至 1.6.18。已新增引導指令碼環境變數 (SERVICE_IPV4_CIDR 和 EXCLUDED_SNAT_CIDRS)。
1.27-2023-09.27	1.27.6	1.6.6	1.1.2	已修正 kubelet 的 安全性公告。

AMI 版本	kubelet 版本	containerd 版本	csi-proxy 版本	版本備註
1.27-2023.09.12	1.27.4	1.6.6	1.1.2	升級 Amazon VPC CNI 外掛程式以使用 Kubernetes 連接器二進位檔，從 Kubernetes API 伺服器取得 Pod IP 地址。合併的提取請求 #100。
1.27-2023.08.17	1.27.4	1.6.6	1.1.2	包含 CVE-2023-3676 、CVE-2023-3893 和 CVE-2023-3955 的修補程式。
1.27-2023.08.08	1.27.3	1.6.6	1.1.1	
1.27-2023.07.11	1.27.3	1.6.6	1.1.1	
1.27-2023.06.20	1.27.1	1.6.6	1.1.1	已解決導致 DNS 字尾搜尋清單填入錯誤的問題。
1.27-2023.06.14	1.27.1	1.6.6	1.1.1	加入了對 CNI 中主機連接埠映射的支援。合併提取請求 #93。

AMI 版本	kubelet 版本	containerd 版本	csi-proxy 版本	版本備註
1.27-2023.06.06	1.27.1	1.6.6	1.1.1	修正 containers-roadmap 問題 #2042，此為導致節點無法提取私有 Amazon ECR 映像的問題。
1.27-2023.05.17	1.27.1	1.6.6	1.1.1	

Kubernetes 1.26 版

AMI 版本	kubelet 版本	containerd 版本	csi-proxy 版本	版本備註
1.26-2025.05.17	1.26.15	1.7.20	1.2.1	
1.26-2025.04.14	1.26.15	1.7.20	1.1.3	
1.26-2025.03.14	1.26.15	1.7.20	1.1.3	已將 containerd 升級至 1.7.20。
1.26-2025.02.15	1.26.15	1.7.14	1.1.3	
1.26-2025.01.15	1.26.15	1.7.14	1.1.3	
1.26-2024.12.11	1.26.15	1.7.14	1.1.3	

AMI 版本	kubelet 版本	containerd 版本	csi-proxy 版本	版本備註
1.26-2024.11.12	1.26.15	1.7.14	1.1.3	
1.26-2024.10.08	1.26.15	1.7.14	1.1.3	
1.26-2024.09.10	1.26.15	1.7.14	1.1.3	
1.26-2024.08.13	1.26.15	1.7.14	1.1.3	
1.26-2024.07.10	1.26.15	1.7.11	1.1.2	包含 CVE-2024-5321 的修補程式。
1.26-2024.06.17	1.26.15	1.7.11	1.1.2	已將 containerd 升級至 1.7.11。
1.26-2024.05.14	1.26.15	1.6.28	1.1.2	已將 containerd 升級至 1.6.28。已將 kubelet 升級至 1.26.15。

AMI 版本	kubelet 版本	containerd 版本	csi-proxy 版本	版本備註
1.26-2024.04.09	1.26.12	1.6.25	1.1.2	已將 containerd 升級至 1.6.25。csi-proxy 使用 重建 CNI 和 golang 1.22.1。
1.26-2024.03.13	1.26.12	1.6.18	1.1.2	
1.26-2024.02.13	1.26.12	1.6.18	1.1.2	
1.26-2024.01.09	1.26.12	1.6.18	1.1.2	
1.26-2023.12.12	1.26.10	1.6.18	1.1.2	
1.26-2023.11.14	1.26.10	1.6.18	1.1.2	包含 CVE-2023-5528 的修補程式。

AMI 版本	kubelet 版本	containerd 版本	csi-proxy 版本	版本備註
1.26-2023.10.19	1.26.9	1.6.18	1.1.2	已將 containerd 升級至 1.6.18。已將 kubelet 升級至 1.26.9。已新增引導指令碼環境變數 (SERVICE_IPV4_CIDR 和 EXCLUDED_SNAT_CIDRS)。
1.26-2023.09.12	1.26.7	1.6.6	1.1.2	升級 Amazon VPC CNI 外掛程式以使用 Kubernetes 連接器二進位檔，從 Kubernetes API 伺服器取得 Pod IP 地址。合併的提取請求 #100。
1.26-2023.08.17	1.26.7	1.6.6	1.1.2	包含 CVE-2023-3676 、CVE-2023-3893 和 CVE-2023-3955 的修補程式。
1.26-2023.08.08	1.26.6	1.6.6	1.1.1	

AMI 版本	kubelet 版本	containerd 版本	csi-proxy 版本	版本備註
1.26-2023.07.11	1.26.6	1.6.6	1.1.1	
1.26-2023.06.20	1.26.4	1.6.6	1.1.1	已解決導致 DNS 字尾搜尋清單填入錯誤的問題。
1.26-2023.06.14	1.26.4	1.6.6	1.1.1	已將 Kubernetes 升級到 1.26.4。加入了對 CNI 中主機連接埠映射的支援。合併提取請求 #93。
1.26-2023.05.09	1.26.2	1.6.6	1.1.1	修正節點重新啟動後，在 Pod 上造成網路連線問題 #1126 的錯誤。引入新的引導指令碼組態參數 (ExcludedSnatCIDRs)。
1.26-2023.04.26	1.26.2	1.6.6	1.1.1	
1.26-2023.04.11	1.26.2	1.6.6	1.1.1	新增了服務當機時 kubelet 和 kube-proxy 的復原機制。
1.26-2023.03.24	1.26.2	1.6.6	1.1.1	

擷取建議的 Microsoft Windows AMI IDs

部署節點時，您可以為預先建置的 Amazon EKS 最佳化 Amazon Machine Image (AMI) 指定 ID。若要擷取符合您所需組態的 AMI ID，請查詢 AWS Systems Manager 參數存放區 API。使用此 API 不需要手動查詢 Amazon EKS 最佳化 AMI IDs。如需詳細資訊，請參閱 GetParameter。您使用的 IAM 主體必須擁有 `ssm:GetParameter` IAM 許可，才能擷取 Amazon EKS 最佳化 AMI 中繼資料。

您可以使用下列命令擷取最新建議的 Amazon EKS 最佳化 Windows AMI 的影像 ID，該命令使用 子參數 `image_id`。視需要對命令進行下列修改，然後執行修改後的命令：

- 將*##*版本取代為下列其中一個選項。
  - 針對 Windows Server *2022* 使用 2022。
  - 針對 Windows Server *2019* 使用 2019。
- 將 *installation-option* 取代為下列其中一個選項。如需詳細資訊，請參閱 Windows Server 中的什麼是 Server Core 安裝選項。
  - 使用 *Core* 以較小的攻擊面進行最少的安裝。
  - 使用*##*來包含 Windows 桌面體驗。
- 將 *kubernetes-version* 取代為支援的 eks/latest/userguide/platform-versions.html【platform-version，type="documentation"】。
- 將 *region-code* 取代為您想要 AMI ID 的 Amazon EKS 支援 AWS 區域。

```
aws ssm get-parameter --name /aws/service/ami-windows-latest/Windows_Server-release-
English-installation-option-EKS_Optimized-kubernetes-version/image_id \
    --region region-code --query "Parameter.Value" --output text
```

以下是進行預留位置取代後的範例命令。

```
aws ssm get-parameter --name /aws/service/ami-windows-latest/Windows_Server-2022-
English-Core-EKS_Optimized-k8s-n-2/image_id \
    --region us-west-2 --query "Parameter.Value" --output text
```

範例輸出如下。

```
ami-1234567890abcdef0
```

使用映像建置器建置自訂 Windows AMI

您可以使用 EC2 Image Builder，透過下列其中一個選項建立自訂 Amazon EKS 最佳化 Windows AMIs：

- 使用 Amazon EKS 最佳化 Windows AMI 做為基礎
- 使用 Amazon 受管建置元件

兩種方法都需要您建立自己的 Image Builder 配方。如需詳細資訊，請參閱《Image Builder 使用者指南》中的建立映像配方的新版本。

> **Important**
>
> 下列 eks 專用的 Amazon 受管元件包含 CVE-2024-5321 的修補程式。
>
> - 1.26.4 和更高版本
> - 1.27.2 和更高版本
> - 1.28.2 和更高版本
> - 1.29.2 和更高版本
> - 1.30.1 和更高版本

使用 Amazon EKS 最佳化 Windows AMI 做為基礎

此選項是建置自訂 Windows AMIs的建議方法。我們提供的 Amazon EKS 最佳化 Windows AMIs 比 Amazon 受管建置元件更頻繁更新。

1. 啟動新的 Image Builder 配方。
   a. 在 https：//https://console.aws.amazon.com/imagebuilder 開啟 EC2 Image Builder 主控台。
   b. 在左側導覽窗格中，選擇映像配方。
   c. 選擇建立映像配方。
2. 在配方詳細資訊區段中，輸入名稱和版本。
3. 在基礎映像區段中指定 Amazon EKS 最佳化 Windows AMI 的 ID。
   a. 選擇輸入自訂 AMI ID。
   b. 擷取您需要的 Windows 作業系統版本的 AMI ID。如需詳細資訊，請參閱the section called “取得最新的 IDs”。

c. 輸入自訂 AMI ID。如果找不到 AMI ID，請確定 AMI ID AWS 的區域與主控台右上角顯示 AWS 的區域相符。

4. (選用) 若要取得最新的安全性更新，請在建置元件：區段中新增 `update-windows` 元件。

   a. 從依名稱尋找元件搜尋方塊右側的下拉式清單中，選擇 Amazon 受管。

   b. 在依名稱尋找元件搜尋方塊中，輸入 `update-windows`。

   c. 選取**`update-windows`**搜尋結果的核取方塊。此元件包含作業系統的最新 Windows 修補程式。

5. 完成具備您所需之組態的剩餘映像配方輸入。如需詳細資訊，請參閱《Image Builder 使用者指南》中的建立映像配方的新版本 (主控台)。

6. 選擇建立配方。

7. 在新的或現有的映像管道中使用新的映像配方。映像管道成功執行後，您的自訂 AMI 將被列為輸出映像並可供使用。如需詳細資訊，請參閱使用 EC2 Image Builder 主控台精靈建立映像管道。

使用 Amazon 受管建置元件

使用 Amazon EKS 最佳化 Windows AMI 做為基礎時，您可以改用 Amazon 受管建置元件。此選項可能會落後於最新支援的 Kubernetes 版本。

1. 啟動新的 Image Builder 配方。

   a. 開啟 EC2 Image Builder 主控台，網址為 https：//https://console.aws.amazon.com/imagebuilder。

   b. 在左側導覽窗格中，選擇映像配方。

   c. 選擇建立映像配方。

2. 在配方詳細資訊區段中，輸入名稱和版本。

3. 在基礎映像區段中，決定您將用來建立自訂 AMI 的選項：

   - 選取受管映像：對於您的映像作業系統 (OS) 選擇 Windows。然後對於映像來源，選擇下列其中一個選項：
     - 快速入門 (Amazon 受管) – 在映像名稱下拉式清單中，選擇 Amazon EKS 支援的 Windows Server 版本。如需詳細資訊，請參閱the section called “Windows”。
     - 我擁有的映像：對於映像名稱，請使用您自己的授權選擇您自己的映像 ARN。您提供的映像尚未安裝 Amazon EKS 元件。
   - 輸入自訂 AMI ID：對於 AMI ID，請使用您自己的授權輸入 AMI 的 ID。您提供的映像尚未安裝 Amazon EKS 元件。

4. 在建置元件 – 視窗區段中，執行下列操作：

a. 從依名稱尋找元件搜尋方塊右側的下拉式清單中，選擇 Amazon 受管。

b. 在依名稱尋找元件搜尋方塊中，輸入 `eks`。

c. 選取**`eks-optimized-ami-windows`**搜尋結果的核取方塊，即使傳回的結果可能不是您想要的版本。

d. 在依名稱尋找元件搜尋方塊中，輸入 `update-windows`。

e. 選取 update-windows 搜尋結果的核取方塊。此元件包含作業系統的最新 Windows 修補程式。

5. 在選取元件區段中，執行下列操作：

a. 選擇 **`eks-optimized-ami-windows`** 的版本控制選項。

b. 選擇指定元件版本。

c. 在元件版本欄位中，輸入 *`version.x`*，將*`##`*取代為支援的 Kubernetes 版本。為版本編號的一部分輸入 *`x`* 表示 使用最新的元件版本，該版本也與您明確定義的版本部分相符。請注意控制台輸出，因為它會告知您所需的版本是否可作為受管元件使用。請記住，建置元件可能無法使用最新的 Kubernetes 版本。如需有關可用版本的詳細資訊，請參閱 the section called “擷取 `eks-optimized-ami-windows` 元件版本的相關資訊”。

6. 完成具備您所需之組態的剩餘映像配方輸入。如需詳細資訊，請參閱《Image Builder 使用者指南》中的建立映像配方的新版本 (主控台)。

7. 選擇建立配方。

8. 在新的或現有的映像管道中使用新的映像配方。映像管道成功執行後，您的自訂 AMI 將被列為輸出映像並可供使用。如需詳細資訊，請參閱使用 EC2 Image Builder 主控台精靈建立映像管道。

擷取 `eks-optimized-ami-windows` 元件版本的相關資訊

您可以擷取與每個元件安裝之內容相關的特定資訊。例如，您可以確認安裝的 `kubelet` 版本。這些元件會在 Amazon EKS 支援的 Windows 作業系統版本上進行功能測試。如需詳細資訊，請參閱the section called “版本發佈日曆”。未列為支援或已終止支援的任何其他 Windows 作業系統版本可能與 元件不相容。

1. 在 https：//https://console.aws.amazon.com/imagebuilder 開啟 EC2 Image Builder 主控台。

2. 在左側導覽窗格中選擇元件。

3. 從依名稱尋找元件搜尋方塊右側的下拉式清單中，將我擁有的變更為 Quick start (Amazon 受管)。

4. 在 Find components by name (依名稱尋找元件) 方塊中，輸入 `eks`。

5. (選用) 如果您使用的是最新版本，請選擇兩次，將版本欄以遞減順序排序。

6. 選擇具有所需版本**`eks-optimized-ami-windows`**的連結。

結果頁面中的描述會顯示特定資訊。

啟用節點自動修復並調查節點運作狀態問題

節點運作狀態是指節點的操作狀態和功能，以有效地執行工作負載。運作狀態良好的節點會維持預期的連線能力、有足夠的資源，而且可以成功執行 Pod，而不會中斷。如需取得節點詳細資訊的詳細資訊，請參閱 the section called “檢視節點運作狀態”和 the section called “取得節點日誌”。

為了協助維護正常運作的節點，Amazon EKS 提供節點監控代理程式和節點自動修復。

> **Important**
>
> 節點監控代理程式和節點自動修復僅適用於 Linux。這些功能不適用於 Windows。

節點監控代理程式

節點監控代理程式會自動讀取節點日誌，以偵測特定運作狀態問題。它會剖析節點日誌，以偵測失敗並顯示工作者節點的各種狀態資訊。專用的 `NodeCondition`會針對偵測到的每個問題類別套用到工作者節點，例如儲存和聯網問題。可觀測性儀表板提供偵測到之運作狀態問題的描述。如需詳細資訊，請參閱the section called “節點運作狀態問題”。

節點監控代理程式會納入做為所有 Amazon EKS Auto Mode 叢集的功能。對於其他叢集類型，您可以將監控代理程式新增為 Amazon EKS 附加元件。如需詳細資訊，請參閱the section called “建立附加元件”。

節點自動修復

節點自動修復是一項額外功能，可持續監控節點的運作狀態、自動回應偵測到的問題，並盡可能取代節點。這有助於叢集的整體可用性，只需最少的手動介入。如果運作狀態檢查失敗，節點會自動封鎖，因此節點上不會排程新的 Pod。

節點自動修復本身可以對 `kubelet`和手動刪除的任何節點物件`Ready`的條件做出反應。與節點監控代理程式配對時，節點自動修復可以對其他情況下無法偵測到的更多條件做出反應。這些其他條件包括 `KernelReady`、 `NetworkingReady`和 `StorageReady`。

此自動化節點復原會自動解決間歇性節點問題，例如無法加入叢集、無回應的 kubelet 和加速器（裝置）錯誤。改善的可靠性有助於減少應用程式停機時間並改善叢集操作。節點自動修復無法處理報告的某些問題，例如 `DiskPressure`、 `MemoryPressure`和 `PIDPressure`。Amazon EKS 會等待

10 分鐘再對 AcceleratedHardwareReady 採取行動NodeConditions，所有其他條件則為 30 分鐘。

在兩種情況下，受管節點群組也會基於安全考量自動停用節點修復。這兩種情況下，先前進行中的任何修復操作都會繼續。

- 如果已透過應用程式復原控制器 (ARC) 觸發叢集的區域轉移，則所有後續修復操作都會停止。
- 如果您的節點群組有超過五個節點，且節點群組中有超過 20% 的節點處於運作狀態不良狀態，則修復操作會停止。

您可以在建立或編輯受管節點群組時啟用節點自動修復。

- 使用 Amazon EKS 主控台時，請啟用受管節點群組的啟用節點自動修復核取方塊。如需詳細資訊，請參閱the section called “建立”。
- 使用 AWS CLI 時，請將 `--node-repair-config enabled=true` 新增至 `eks create nodegroup`或 `eks update-nodegroup-config`命令。
- 如需eksctlClusterConfig使用受管節點群組搭配節點自動修復的範例，請參閱 GitHub 上的 44-node-repair.yaml。

節點運作狀態問題

下表說明節點監控代理程式可偵測到的節點運作狀態問題。問題有兩種類型：

- 條件 – 需要修復動作的終端問題，例如執行個體替換或重新啟動。啟用自動修復時，Amazon EKS 會執行修復動作，以取代節點或重新啟動。如需詳細資訊，請參閱the section called “節點條件”。
- 事件 – 暫時性問題或次佳節點組態。不會發生自動修復動作。如需詳細資訊，請參閱the section called “節點事件”。

核心節點運作狀態問題

名稱	嚴重性	描述
ForkFailedOutOfPID	條件	由於系統缺少程序 IDs 或記憶體，因此分支或執行呼叫失敗，這可能是因為殭屍程序或實體記憶體耗盡所致。

名稱	嚴重性	描述
AppBlocked	事件	任務因排程而遭到封鎖很長一段時間，通常是因為輸入或輸出遭到封鎖所致。
AppCrash	事件	節點上的應用程式已當機。
ApproachingKernelPidMax	事件	程序數目接近目前 kernel.pid_max 設定可用的 PIDs 數目上限，之後就無法再啟動任何程序。
ApproachingMaxOpenFiles	事件	根據目前的核心設定，開啟的檔案數量接近可能開啟的檔案數量上限，在此之後開啟新檔案將會失敗。
ConntrackExceededKernel	事件	連線追蹤超過核心的最大值，無法建立新的連線，這可能會導致封包遺失。
ExcessiveZombieProcesses	事件	無法完全回收的程序會累積大量，這表示應用程式問題，並可能導致達到系統程序限制。
KernelBug	事件	Linux 核心本身偵測到並報告核心錯誤，但有時可能是因為具有高 CPU 或記憶體用量的節點而導致事件處理延遲。
LargeEnvironment	事件	此程序的環境變數數量大於預期，可能是由許多將 enableServiceLinks 設為 true 的服務所造成，這可能會導致效能問題。

名稱	嚴重性	描述
RapidCron	事件	Cron 任務在此節點上的執行速度比每五分鐘快，如果任務耗用大量資源，可能會影響效能。
SoftLockup	事件	CPU 停滯了一段時間。

網路節點運作狀態問題

名稱	嚴重性	描述
InterfaceNotRunning	條件	此界面似乎未執行，或有網路問題。
InterfaceNotUp	條件	此界面似乎沒有運作，或發生網路問題。
IPAMDNotReady	條件	IPAMD 無法連線至 API 伺服器。
IPAMDNotRunning	條件	找不到正在執行`aws-k8s-agent` 的程序。
MissingLoopbackInterface	條件	此執行個體缺少迴路界面，導致服務失敗，取決於本機連線。
BandwidthInExceeded	事件	由於傳入彙總頻寬超過執行個體的上限，因此封包已排入佇列或捨棄。
BandwidthOutExceeded	事件	由於傳出彙總頻寬超過執行個體的上限，因此封包已排入佇列或捨棄。

名稱	嚴重性	描述
ConntrackExceeded	事件	連線追蹤超過執行個體的上限，而且無法建立新的連線，這可能會導致封包遺失。
IPAMDNoIPs	事件	IPAM-D 沒有 IP 地址。
IPAMDRepeatedlyRestart	事件	IPAMD 服務中發生多次重新啟動。
KubeProxyNotReady	事件	Kube-proxy 無法監看或列出資源。
LinkLocalExceeded	事件	由於本機代理服務的流量 PPS 超過網路介面上限，因此封包遭到捨棄。
MissingDefaultRoutes	事件	缺少預設路由規則。
MissingIPRules、 MissingIPRoutes	事件	路由表中缺少下列 Pod IPs的路由規則。
NetworkSysctl	事件	此節點的網路 sysctl 設定可能不正確。
PortConflict	事件	如果 Pod 使用 hostPort，它可以寫入覆寫主機已繫結連接埠的 iptables 規則，從而可能阻止 API 伺服器存取 `kubelet`。
PPSExceeded	事件	由於雙向 PPS 超過執行個體的最大值，因此封包已排入佇列或捨棄。
UnexpectedRejectRule	事件	在 iptable 中找到非預期的 `REJECT`或 `DROP`規則，可能封鎖預期的流量。

Neuron 節點運作狀態問題

名稱	嚴重性	描述
NeuronDMAError	條件	DMA 引擎發生無法復原的錯誤。
NeuronHBMUncorrectableError	條件	HBM 遇到無法修正的錯誤，並產生不正確的結果。
NeuronNCUncorrectableError	條件	偵測到 Neuron Core 無法修正的記憶體錯誤。
NeuronSRAMUncorrectableError	條件	晶片上 SRAM 遇到同位錯誤並產生不正確的結果。

NVIDIA 節點運作狀態問題

如果啟用自動修復，列出的修復動作會在偵測到問題後 10 分鐘開始。如需 XID 錯誤的詳細資訊，請參閱 NVIDIA GPU 部署和管理文件中的 Xid 錯誤。如需個別 XID 訊息的詳細資訊，請參閱 NVIDIA GPU 部署和管理文件中的了解 Xid 訊息。

名稱	嚴重性	描述	修復動作
NvidiaDoubleBitError	條件	GPU 驅動程式產生雙位元錯誤。	Replace (取代)
NvidiaNVLinkError	條件	GPU 驅動程式回報 NVLink 錯誤。	Replace (取代)
NvidiaXID13Error	條件	有圖形引擎例外狀況。	重新開機
NvidiaXID31Error	條件	有可疑的硬體問題。	重新開機
NvidiaXID48Error	條件	驅動程式會回報雙位元 ECC 錯誤。	重新開機

名稱	嚴重性	描述	修復動作
NvidiaXID63Error	條件	有頁面淘汰或資料列重新映射。	重新開機
NvidiaXID64Error	條件	嘗試淘汰頁面或執行節點重新映射失敗。	重新開機
NvidiaXID74Error	條件	從 GPU 到另一個 GPU 或 NVSwitch 透過 NVLink 連線時發生問題。這可能表示連結本身發生硬體故障，或可能表示連結遠端的裝置發生問題。	Replace (取代)
NvidiaXID79Error	條件	GPU 驅動程式嘗試透過 PCI Express 連線存取 GPU，並發現無法存取 GPU。	Replace (取代)
NvidiaXID94Error	條件	發生 ECC 記憶體錯誤。	重新開機
NvidiaXID95Error	條件	發生 ECC 記憶體錯誤。	重新開機
NvidiaXID119Error	條件	GSP 回應來自驅動程式中其他位元的 RPC 請求逾時。	Replace (取代)
NvidiaXID120Error	條件	GSP 已及時回應，但發生錯誤。	Replace (取代)
NvidiaXID121Error	條件	C2C 是晶片互連。它可在 CPUs、加速器等之間共用記憶體。	Replace (取代)

名稱	嚴重性	描述	修復動作
NvidiaXID140Error	條件	GPU 驅動程式可能已在 GPU 記憶體中觀察到無法修正的錯誤，以中斷 GPU 驅動程式標記頁面以進行動態頁面離線或資料列重新映射的能力。	Replace (取代)
NvidiaPageRetirement	事件	GPU 驅動程式已將記憶體頁面標記為淘汰。如果在同一地址遇到單一雙位元錯誤或兩個單一位元錯誤，則可能會發生這種情況。	無
NvidiaXID【Code】警告	事件	除了此清單中定義的 XIDs 之外，任何 XID 的出現都會產生此事件。	無
DCGMError	條件	與資料中心 GPU Manager (DCGM) 主機程序的連線遺失或無法建立。	無
DCGMDiagnosticError	條件	發出執行 DCGM 作用中診斷的 。	無
DCGMDiagnosticFailure	條件	DCGM 作用中診斷測試套件的測試案例失敗。	無

執行期節點運作狀態問題

名稱	嚴重性	描述
PodStuckTerminating	條件	Pod 正在或停滯終止時間過長，這可能是由於防止 Pod 狀態惡化的 CRI 錯誤所造成。
%sRepeatedRestart	事件	重新啟動節點上任何系統化的服務（使用標題大小寫單位名稱進行格式化）。
ContainerRuntimeFailed	事件	容器執行時間無法建立容器，如果重複發生，可能與任何回報的問題有關。
KubeletFailed	事件	kubelet 進入失敗狀態。
LivenessProbeFailures	事件	偵測到活體探查失敗，如果重複發生，可能表示應用程式碼問題或逾時值不足。
ReadinessProbeFailures	事件	偵測到整備探查失敗，如果重複發生，可能表示應用程式碼問題或逾時值不足。
ServiceFailedToStart	事件	系統化單位無法啟動。

儲存節點運作狀態問題

名稱	嚴重性	描述
XFSSmallAverageClusterSize	條件	XFS 平均叢集大小很小，表示過度的可用空間分段，即使可用節點或可用空間，仍可能導致檔案無法建立。

名稱	嚴重性	描述
EtcHostsMountFailed	事件	在 kubelet-container 操作/var/lib/kubelet/pods 期間重新掛載使用者資料，導致產生的 kubelet 掛載/etc/hosts 失敗。
IODelays	事件	在程序中偵測到輸入或輸出延遲，如果過多，可能表示輸入輸出佈建不足。
KubeletDiskUsageSlow	事件	Kubelet 在嘗試存取檔案系統時回報磁碟使用量緩慢，可能表示磁碟輸入輸出不足或檔案系統問題。

檢視節點的運作狀態

本主題說明可用於監控 Amazon EKS 叢集中節點運作狀態的工具和方法。此資訊涵蓋節點條件、事件和偵測案例，可協助您識別和診斷節點層級的問題。使用此處所述的命令和模式來檢查節點運作狀態資源、解譯狀態條件，並分析節點事件以進行操作故障診斷。

您可以使用所有節點的 Kubernetes 命令來取得一些節點運作狀態資訊。而且，如果您透過 Amazon EKS Auto Mode 或 Amazon EKS 受管附加元件使用節點監控代理程式，您會收到更多種類的節點訊號以協助故障診斷。節點監控代理程式偵測到的運作狀態問題的描述，也可在可觀測性儀表板中找到。如需詳細資訊，請參閱the section called “節點運作狀態”。

節點條件

節點條件代表需要修復動作的終端問題，例如執行個體替換或重新啟動。

若要取得所有節點的條件：

```
kubectl get nodes -o 'custom-
columns=NAME:.metadata.name,CONDITIONS:.status.conditions[*].type,STATUS:.status.conditions[*].
```

取得特定節點的詳細條件

```
kubectl describe node node-name
```

正常運作節點的條件輸出範例：

```
  - lastHeartbeatTime: "2024-11-21T19:07:40Z"
    lastTransitionTime: "2024-11-08T03:57:40Z"
    message: Monitoring for the Networking system is active
    reason: NetworkingIsReady
    status: "True"
    type: NetworkingReady
```

聯網問題之運作狀態不佳節點的範例條件：

```
  - lastHeartbeatTime: "2024-11-21T19:12:29Z"
    lastTransitionTime: "2024-11-08T17:04:17Z"
    message: IPAM-D has failed to connect to API Server which could be an issue with
      IPTable rules or any other network configuration.
    reason: IPAMDNotReady
    status: "False"
    type: NetworkingReady
```

節點事件

節點事件指出暫時性問題或次佳組態。

取得節點監控代理程式報告的所有事件

當節點監控代理程式可用時，您可以執行下列命令。

```
kubectl get events --field-selector=reportingComponent=eks-node-monitoring-agent
```

輸出範例：

```
LAST SEEN   TYPE      REASON       OBJECT
 MESSAGE
4s          Warning   SoftLockup   node/ip-192-168-71-251.us-west-2.compute.internal
 CPU stuck for 23s
```

取得所有節點的事件

```
kubectl get events --field-selector involvedObject.kind=Node
```

取得特定節點的事件

```
kubectl get events --field-selector involvedObject.kind=Node,involvedObject.name=node-name
```

即時監看事件

```
kubectl get events -w --field-selector involvedObject.kind=Node
```

範例事件輸出：

```
LAST SEEN   TYPE      REASON           OBJECT         MESSAGE
2m          Warning   MemoryPressure   Node/node-1    Node experiencing memory pressure
5m          Normal    NodeReady        Node/node-1    Node became ready
```

常見故障診斷命令

```
# Get comprehensive node status
kubectl get node node-name -o yaml

# Watch node status changes
kubectl get nodes -w

# Get node metrics
kubectl top node
```

使用 kubectl 和 S3 擷取受管節點的節點日誌

了解如何擷取具有節點監控代理程式之 Amazon EKS 受管節點的節點日誌。

先決條件

請確定您有下列項目：

- 具有節點監控代理程式的現有 Amazon EKS 叢集。如需詳細資訊，請參閱the section called “節點運作狀態”。

- 安裝並設定 kubectl命令列工具，以與您的叢集通訊。
- 已安裝並登入的 AWS CLI 具有足夠的許可，可建立 S3 儲存貯體和物件。
- 已安裝最新版本的 Python 3
- 已安裝適用於 Python 3、Boto 3 的 AWS SDK。

步驟 1：建立 S3 儲存貯體目的地 （選用）

如果您還沒有儲存日誌的 S3 儲存貯體，請建立一個。使用下列 AWS CLI 命令。儲存貯體預設為private存取控制清單。以您選擇的唯一*######## bucket-name*。

```
aws s3api create-bucket --bucket bucket-name
```

步驟 2：為 HTTP Put 建立預先簽章的 S3 URL

Amazon EKS 會透過對您指定的 URL 執行 HTTP PUT 操作來傳回節點日誌。在本教學課程中，我們將產生預先簽章的 S3 HTTP PUT URL。

日誌會以 gzip tarball 傳回，副.tar.gz檔名為 。

> **Note**
>
> 您必須使用 AWS API 或 SDK 來建立預先簽章的 S3 上傳 URL，以供 EKS 上傳日誌檔案。您無法使用 CLI 建立預先簽章的 S3 AWS 上傳 URL。

1. 決定您要在儲存貯體中存放日誌的位置。例如，您可以使用 2024-11-12/logs1.tar.gz做為金鑰。
2. 將下列 Python 程式碼儲存至檔案 presign-upload.py。取代 <bucket-name>和 <key>。金鑰應以 結尾.tar.gz。

   ```
   import boto3; print(boto3.client('s3').generate_presigned_url(
      ClientMethod='put_object',
      Params={'Bucket': '<bucket-name>', 'Key': '<key>'},
      ExpiresIn=1000
   ))
   ```

3. 使用 執行指令碼

```
python presign-upload.py
```

4. 請注意 URL 輸出。在下一個步驟中使用此值做為 `http-put-destination`。

如需詳細資訊，請參閱產生預先簽章的 URL，以在適用於 Python 的Boto3 開發套件文件中上傳檔案。AWS Boto3

步驟 3：建立 NodeDiagnostic 資源

識別您要從中收集日誌的節點名稱。

建立使用節點名稱做為資源名稱的資訊`NodeDiagnostic`清單，並提供 HTTP PUT URL 目的地。

```
apiVersion: eks.amazonaws.com/v1alpha1
kind: NodeDiagnostic
metadata:
    name: node-name
spec:
    logCapture:
        destination: http-put-destination
```

將清單檔案套用至叢集。

```
kubectl apply -f nodediagnostic.yaml
```

您可以透過描述 `NodeDiagnostic` 資源來檢查集合的狀態：

- 狀態為 `Success`或 `SuccessWithErrors`表示任務已完成，且日誌上傳到提供的目的地 (`SuccessWithErrors` 表示部分日誌可能遺失）
- 如果狀態為失敗，請確認上傳 URL 格式正確且未過期。

```
kubectl describe nodediagnostics.eks.amazonaws.com/node-name
```

步驟 4：從 S3 下載日誌

等待大約一分鐘，然後再嘗試下載日誌。然後，使用 S3 CLI 下載日誌。

```
# Once NodeDiagnostic shows Success status, download the logs
aws s3 cp s3://bucket-name/key ./node-logs.tar.gz
```

步驟 5：清除 NodeDiagnostic 資源

- NodeDiagnostic 資源不會自動刪除。取得日誌成品後，您應該自行清除這些項目

```
# Delete the NodeDiagnostic resource
kubectl delete nodediagnostics.eks.amazonaws.com/node-name
```

Amazon EKS 混合節點概觀

透過 Amazon EKS 混合節點，您可以使用內部部署和邊緣基礎設施做為 Amazon EKS 叢集中的節點。 AWS 管理 Amazon EKS 叢集的 AWS託管 Kubernetes 控制平面，並管理在內部部署或邊緣環境中執行的混合節點。這可統一您環境中的 Kubernetes 管理，並將 AWS 內部部署和邊緣應用程式的 Kubernetes 控制平面管理卸載至 。

Amazon EKS 混合節點可與任何內部部署硬體或虛擬機器搭配使用，將 Amazon EKS 的效率、可擴展性和可用性帶入您的應用程式需要執行的任何地方。您可以搭配 Amazon EKS 混合節點使用各種 Amazon EKS 功能，包括 Amazon EKS 附加元件、Amazon EKS Pod Identity、叢集存取項目、叢集洞見和擴充的 Kubernetes 版本支援。Amazon EKS 混合節點原生整合了 AWS Systems Manager、AWS IAM Roles Anywhere、Amazon Managed Service for Prometheus 和 Amazon CloudWatch 等 AWS 服務，以進行集中式監控、記錄和身分管理。

使用 Amazon EKS 混合節點時，無需預付承諾或最低費用，而且當您混合節點的 vCPU 資源連接到 Amazon EKS 叢集時，會按小時向您收費。如需定價的詳細資訊，請參閱 Amazon EKS 定價。

功能

EKS 混合節點具有下列高階功能：

- 受管 Kubernetes 控制平面： AWS 管理 EKS 叢集的 AWS託管 Kubernetes 控制平面，以及管理在內部部署或邊緣環境中執行的混合節點。這可統一您環境中的 Kubernetes 管理，並將 AWS 內部部署和邊緣應用程式的 Kubernetes 控制平面管理卸載至 。透過將 Kubernetes 控制平面移至 AWS，您可以為應用程式節省內部部署容量，並信任 Kubernetes 控制平面隨工作負載擴展。

- 一致的 EKS 體驗：EKS 混合節點支援大多數 EKS 功能，可在內部部署和雲端環境中提供一致的 EKS 體驗，包括 EKS 附加元件、EKS Pod Identity、叢集存取項目、叢集洞察、擴充 Kubernetes 版本支援等。the section called “設定附加元件” 如需 EKS 混合節點支援之 EKS 附加元件的詳細資訊，請參閱 。
- 集中式可觀測性和身分管理：EKS 混合節點與 AWS Systems Manager、 AWS IAM Roles Anywhere、Amazon Managed Service for Prometheus 和 Amazon CloudWatch 等 AWS 服務原生整合，以進行集中式監控、記錄和身分管理。
- Burst-to-cloud或新增內部部署容量：單一 EKS 叢集可用於在 AWS 區域、 AWS 本機區域或 AWS Outpost 中執行混合節點和節點至burst-to-cloud，或將內部部署容量新增至 EKS 叢集。如需詳細資訊，請參閱混合模式叢集的考量事項。
- 彈性基礎設施：EKS 混合節點遵循自有基礎設施方法，且與您用於混合節點的基礎設施無關。您可以在實體或虛擬機器以及 x86 和 ARM 架構上執行混合節點，讓您可以跨不同的基礎設施類型遷移在混合節點上執行的內部部署工作負載。
- 彈性聯網：使用 EKS 混合節點，EKS 控制平面和混合節點之間的通訊會透過您在叢集建立期間傳遞的 VPC 和子網路路由，該子網路以 EKS 中的現有機制為基礎，用於控制平面到節點聯網。這對於將內部部署網路連線到 VPC 的偏好方法來說非常靈活 AWS。有多種文件記錄的選項可供使用，包括 AWS Site-to-Site VPN、 AWS Direct Connect 或您自己的 VPN 解決方案，您可以選擇最適合您的使用案例的方法。

限制

- 每個叢集最多支援 15 CIDRs 和 15 CIDRs。

考量事項

- EKS 混合節點可與新的或現有的 EKS 叢集搭配使用。
- EKS 混合節點適用於所有 AWS 區域， AWS GovCloud (US) 區域和 AWS 中國區域除外。
- EKS 混合節點必須在您的內部部署環境與 之間具有可靠的連線 AWS。EKS 混合節點不適用於中斷、中斷、間歇性或受限 (DDIL) 環境。如果您在 DDIL 環境中執行 ，請考慮使用 Amazon EKS Anywhere。請參閱 EKS 混合節點的最佳實務，以取得混合節點在網路中斷連線案例期間的行為資訊。
- 不支援在雲端基礎設施上執行 EKS 混合節點，包括 AWS 區域、 AWS 本機區域、 AWS Outposts 或其他雲端。如果您在 Amazon EC2 執行個體上執行混合節點，需支付混合節點費用。

- 混合節點的計費會在節點加入 EKS 叢集時開始，並在節點從叢集中移除時停止。如果您未使用混合節點，請務必從 EKS 叢集中移除它們。

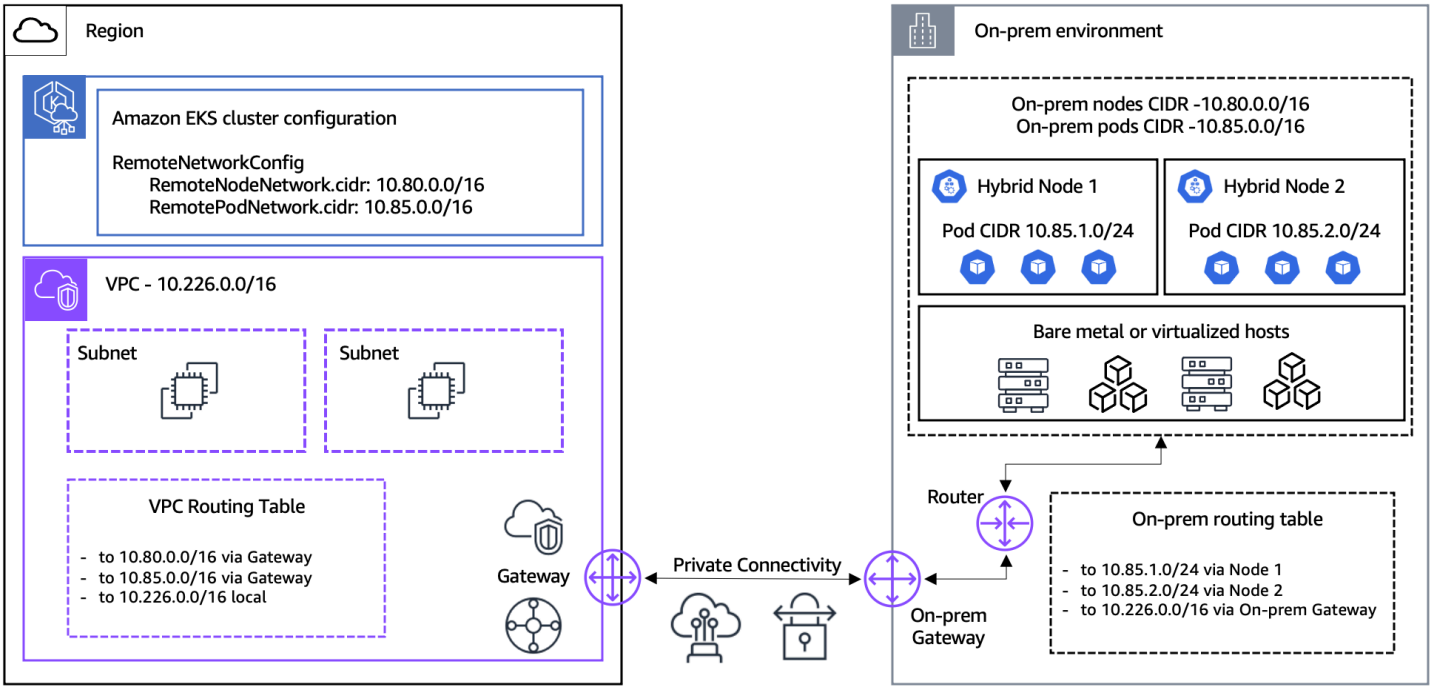
其他資源

- EKS 混合節點研討會：在示範環境中部署 EKS 混合節點的Step-by-step說明。
- AWS re：Invent： EKS 混合節點： AWS re：Invent 工作階段會向客戶介紹 EKS 混合節點啟動，顯示他們在其環境中如何使用 EKS 混合節點。
- AWS re：Post：EKS 混合節點的叢集聯網：文章說明為 EKS 混合節點設定聯網的各種方法。
- AWS 部落格：使用 EKS 混合節點在環境中執行 GenAI 推論：部落格文章說明如何使用 EKS 混合節點在環境中執行 GenAI 推論。

混合節點的先決條件設定

若要使用 Amazon EKS 混合節點，您必須擁有內部部署環境往返的私有連線 AWS、具有支援作業系統的裸機伺服器或虛擬機器，以及已設定的 AWS IAM Roles Anywhere 或 AWS Systems Manager (SSM) 混合啟用。您負責在整個混合節點生命週期中管理這些先決條件。

- 從現場部署環境往返的混合網路連線 AWS
- 實體或虛擬機器形式的基礎設施
- 與混合節點相容的作業系統
- 已設定內部部署 IAM 登入資料提供者



混合網路連線

Amazon EKS 控制平面和混合節點之間的通訊會透過 VPC 和您在叢集建立期間傳遞的子網路進行路由，該子網路在 Amazon EKS 中建置於控制平面到節點聯網的現有機制。有數個文件記錄的選項可讓您將內部部署環境與您的 VPC 連接，包括 AWS Site-to-Site VPN、 AWS Direct Connect 或您自己的 VPN 連接。如需如何將這些解決方案用於混合網路連線的詳細資訊，請參閱 AWS Site-to-Site VPN 和 AWS Direct Connect 使用者指南。

為了獲得最佳體驗，我們建議您擁有至少 100 Mbps 的可靠網路連線能力，以及最多 200 毫秒的混合節點往返延遲。 AWS 這是適用於大多數使用案例的一般指引，但不是嚴格的要求。頻寬和延遲需求可能會因混合節點的數量和工作負載特性而有所不同，例如應用程式映像大小、應用程式彈性、監控和記錄組態，以及存取儲存在其他服務中的資料的應用程式相依性 AWS 。建議您在部署至生產環境之前，先使用自己的應用程式和環境進行測試，以驗證您的聯網設定是否符合工作負載的需求。

內部部署網路組態

您必須啟用從 Amazon EKS 控制平面到內部部署環境的傳入網路存取，以允許 Amazon EKS 控制平面與`kubelet`在混合節點上執行的 通訊，也可以與在混合節點上執行的 Webhook 通訊。此外，您必須為在其上執行的混合節點和元件啟用傳出網路存取，才能與 Amazon EKS 控制平面通訊。您可以設定此通訊，讓 AWS Direct Connect、 AWS Site-to-Site VPN 或您自己的 VPN 連線保持完全私有。

您用於內部部署節點和 Pod 網路的無類別網域間路由 (CIDR) 範圍必須使用 IPv4 RFC-1918 地址範圍。您的內部部署路由器必須設定通往內部部署節點和選用 Pod 的路由。the section called “內部部署

聯網組態" 如需現場部署網路需求的詳細資訊，請參閱 ，包括必須在防火牆和現場部署環境中啟用的必要連接埠和通訊協定完整清單。

EKS 叢集組態

為了將延遲降至最低，建議您在最接近內部部署或邊緣環境的 AWS 區域中建立 Amazon EKS 叢集。您可以在 Amazon EKS 叢集建立期間，透過兩個 API 欄位傳遞內部部署節點和 Pod CIDRs：`RemoteNodeNetwork`和 `RemotePodNetwork`。您可能需要與您的內部部署網路團隊討論，以識別您的內部部署節點和 Pod CIDRs。如果您使用 CNI 的浮水印網路，節點 CIDR 會從內部部署網路配置，而 Pod CIDR 會從您使用的容器網路界面 (CNI) 配置。Cilium 和 Calico 預設使用浮水印網路。

您透過 和 `RemotePodNetwork` 欄位設定的現場部署節點`RemoteNodeNetwork`和 Pod CIDRs，會用來設定 Amazon EKS 控制平面，將流量透過 VPC 路由至 ，`kubelet`以及在混合節點上執行的 Pod。您的內部部署節點和 Pod CIDRs 不能彼此重疊、您在叢集建立期間傳遞的 VPC CIDR，或 Amazon EKS 叢集的服務 IPv4 組態。

建議您對 Amazon EKS Kubernetes API 伺服器端點使用公有或私有端點存取。如果您選擇「公有和私有」，Amazon EKS Kubernetes API 伺服器端點一律會解析為在 VPC 外部執行之混合節點的公 IPs，以防止混合節點加入叢集。當您使用公有端點存取時，Kubernetes API 伺服器端點會解析為公 IPs，而從混合節點到 Amazon EKS 控制平面的通訊將透過網際網路路由。當您選擇私有端點存取時，Kubernetes API 伺服器端點會解析為私有 IPs，而且從混合節點到 Amazon EKS 控制平面的通訊將透過您的私有連線連結路由，在大多數情況下是 AWS Direct Connect AWS Site-to-Site VPN。

VPC 組態

您必須設定在 Amazon EKS 叢集建立期間傳遞的 VPC，其路由表中包含內部部署節點的路由，以及選擇性地將虛擬私有閘道 (VGW) 或傳輸閘道 (TGW) 做為目標的 Pod 網路。以下所示為範例。`REMOTE_POD_CIDR` 使用內部部署網路的值取代 `REMOTE_NODE_CIDR`和 。

目的地	目標	描述
10.226.0.0/16	本機	VPC 內 VPC 路由的本機流量
REMOTE_NODE_CIDR	tgw-abcdef123456	內部部署節點 CIDR，將流量路由到 TGW
REMODE_POD_CIDR	tgw-abcdef123456	內部部署 Pod CIDR，將流量路由到 TGW

安全群組組態

當您建立叢集時，Amazon EKS 會建立名為 的安全群組`eks-cluster-sg-<cluster-name>-<uniqueID>`。您無法變更此叢集安全群組的傳入規則，但可以限制傳出規則。您必須將額外的安全群組新增至叢集，才能啟用在混合節點上執行的 kubelet 和選用 Webhook，以聯絡 Amazon EKS 控制平面。此額外安全群組所需的傳入規則如下所示。`REMOTE_POD_CIDR` 使用現場部署網路的值取代`REMOTE_NODE_CIDR`和 。

名稱	安全群組規則 ID	IP 版本	Type	通訊協定	連接埠範圍	來源
內部部署節點傳入	sgr-abcdef123456	IPv4	HTTPS	TCP	443	REMOTE_NODE_CIDR
內部部署 Pod 傳入	sgr-abcdef654321	IPv4	HTTPS	TCP	443	REMOTE_POD_CIDR

基礎設施

您必須擁有可用於混合節點的裸機伺服器或虛擬機器。混合節點與基礎基礎設施無關，並支援 x86 和 ARM 架構。Amazon EKS 混合節點遵循「使用您自己的基礎設施」方法，負責佈建和管理用於混合節點的裸機伺服器或虛擬機器。雖然沒有嚴格的最低資源需求，但我們建議您為混合節點使用至少具有 1 個 vCPU 和 1GiB RAM 的主機。

作業系統

Bottlerocket、Amazon Linux 2023 (AL2023)、Ubuntu 和 RHEL 會持續經過驗證，以用作混合節點的節點作業系統。Bottlerocket 僅支援 VMware vSphere AWS環境。在 Amazon EC2 外部執行時，AWS 支援計劃不涵蓋 AL2023。 Amazon EC2 AL2023 只能在內部部署虛擬化環境中使用，如需詳細資訊，請參閱 Amazon Linux 2023 使用者指南。 AWS 支援與 Ubuntu 和 RHEL 作業系統整合的混合節點，但不支援作業系統本身。

您負責作業系統佈建和管理。當您第一次測試混合節點時，在已佈建的主機上執行 Amazon EKS 混合節點 CLI (`nodeadm`) 最簡單。對於生產部署，我們建議您在黃金作業系統映像`nodeadm`中包含 ，並將它設定為執行為系統化服務，以在主機啟動時自動將主機加入 Amazon EKS 叢集。

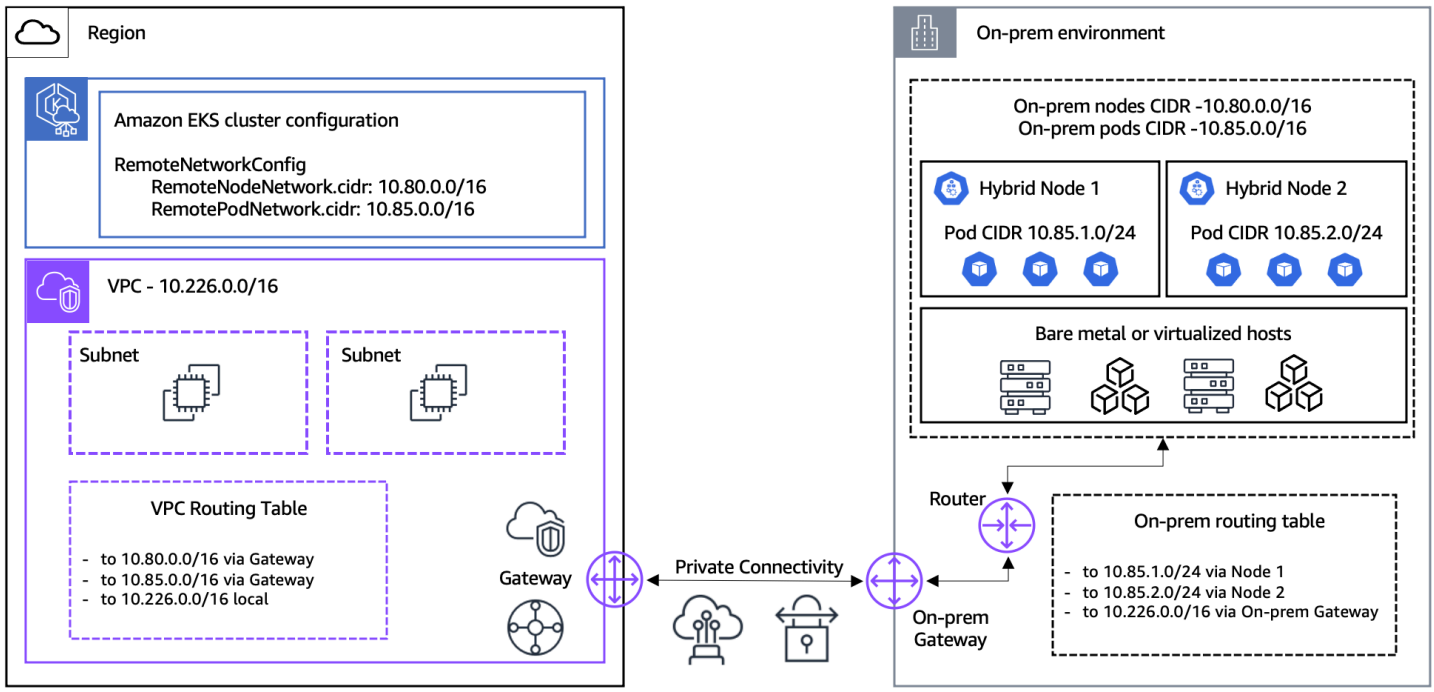
內部部署 IAM 登入資料提供者

Amazon EKS 混合節點使用由 AWS SSM 混合啟用或 IAM Roles Anywhere AWS 佈建的臨時 IAM 登入資料來驗證 Amazon EKS 叢集。您必須將 AWS SSM 混合啟用或 AWS IAM Roles Anywhere 與 Amazon EKS 混合節點 CLI () 搭配使用nodeadm。如果您現有的公有金鑰基礎設施 (PKI) 沒有憑證授權機構 (CA) 和內部部署環境的憑證，建議您使用 AWS SSM 混合啟用。如果您有現有的 PKI 和現場部署憑證，請使用 AWS IAM Roles Anywhere。

與在 Amazon EC2 上執行之節點the section called “節點 IAM 角色”的 類似，您將建立混合節點 IAM 角色，其中包含將混合節點加入 Amazon EKS 叢集所需的許可。如果您使用的是 AWS IAM Roles Anywhere，請設定信任政策，允許 AWS IAM Roles Anywhere 擔任混合節點 IAM 角色，並使用混合節點 AWS IAM 角色做為可擔任的角色來設定 IAM Roles Anywhere 設定檔。如果您使用的是 AWS SSM，請設定信任政策，允許 AWS SSM 擔任混合節點 IAM 角色，並使用混合節點 IAM 角色建立混合啟用。如需the section called “準備登入資料”如何建立具備必要許可的混合節點 IAM 角色，請參閱 。

準備混合節點的聯網

本主題概述您在建立 Amazon EKS 叢集和連接混合節點之前必須設定的聯網設定。本指南假設您已符合使用 AWS Site-to-Site VPN、AWS Direct Connect 或您自己的 VPN 解決方案進行混合網路連線的先決條件要求。



內部部署聯網組態

最低網路需求

為了獲得最佳體驗，我們建議您擁有至少 100 Mbps 的可靠網路連線能力，以及最多 200 毫秒的混合節點往返延遲。 AWS 這是適用於大多數使用案例的一般指引，但不是嚴格的要求。頻寬和延遲需求可能會因混合節點的數量和工作負載特性而有所不同，例如應用程式映像大小、應用程式彈性、監控和記錄組態，以及存取儲存在其他服務中的資料的應用程式相依性 AWS 。建議您在部署至生產環境之前，先使用自己的應用程式和環境進行測試，以驗證您的聯網設定是否符合工作負載的需求。

內部部署節點和 Pod CIDRs

識別您將用於混合節點的節點和 Pod CIDRs，以及在它們上執行的工作負載。如果您為 CNI 使用浮水印網路，節點 CIDR 會從內部部署網路配置，而 Pod CIDR 會從容器網路界面 (CNI) 配置。當您使用 和 `RemotePodNetwork` 欄位建立 EKS 叢集時，您可以將內部部署節點 CIDRs `RemoteNodeNetwork`和 Pod CIDRs 做為輸入傳遞。您的內部部署節點 CIDRs 必須可在內部部署網路上路由。如需現場部署 Pod CIDR 可路由性的資訊，請參閱下一節。

內部部署節點和 Pod CIDR 區塊必須符合下列要求：

1. 位於下列其中一個 `IPv4` RFC-1918 範圍內：`10.0.0.0/8`、 `172.16.0.0/12`或 `192.168.0.0/16`。
2. 不會彼此重疊、EKS 叢集的 VPC CIDR 或 Kubernetes 服務 `IPv4` CIDR。

內部部署 Pod 網路路由

使用 EKS 混合節點時，通常建議您在內部部署網路上讓內部部署 Pod CIDRs 可路由，以便在雲端和內部部署環境之間啟用完整的叢集通訊和功能。

可路由 Pod 網路

如果您能夠在內部部署網路上讓 Pod 網路可路由，請遵循下列指引。

1. `RemotePodNetwork` 使用內部部署 Pod CIDR、使用內部部署 Pod CIDR 設定 VPC 路由表，以及使用內部部署 Pod CIDR 設定 EKS 叢集安全群組的欄位。
2. 您可以使用多種技術，讓您的內部部署 Pod CIDR 可在內部部署網路上路由，包括邊界閘道協定 (BGP)、靜態路由或其他自訂路由解決方案。BGP 是建議的解決方案，因為它比需要自訂或手動路由組態的替代解決方案更具可擴展性且更容易管理。 AWS 支援 Cilium 和 Calico 的 BGP 功能來公告 Pod CIDRs，如需詳細資訊，請參閱 the section called “可路由遠端 Pod CIDRs” the section called “設定 CNI”和 。

3. Webhook 可以在混合節點上執行，因為 EKS 控制平面能夠與指派給 Webhook 的 Pod IP 地址進行通訊。
4. 在雲端節點上執行的工作負載可以直接與在同一 EKS 叢集中混合節點上執行的工作負載通訊。
5. AWS 其他服務，例如 AWS Application Load Balancer 和 Amazon Managed Service for Prometheus，能夠與混合節點上執行的工作負載進行通訊，以平衡網路流量和湊集 Pod 指標。

無法路由的 Pod 網路

如果您無法在內部部署網路上讓 Pod 網路可路由，請遵循下列指引。

1. Webhook 無法在混合節點上執行，因為 Webhook 需要從 EKS 控制平面連線至指派給 Webhook 的 Pod IP 地址。在此情況下，我們建議您在與混合節點相同的 EKS 叢集中的雲端節點上執行 Webhook，the section called “設定 Webhook”如需詳細資訊，請參閱 。
2. 雲端節點上執行的工作負載在使用雲端節點的 VPC CNI 和混合節點的 Cilium 或 Calico 時，無法直接與混合節點上執行的工作負載通訊。
3. 使用服務流量分佈將流量保留在其來源區域的本機。如需服務流量分佈的詳細資訊，請參閱 the section called “設定服務流量分佈”。
4. 將 CNI 設定為在 Pod 流量離開內部部署主機時，使用輸出遮罩或網路位址轉譯 (NAT)。根據預設，這會在 Cilium 中啟用。Calico `natOutgoing`需要設定為 `true`。
5. Application AWS Load Balancer 和 Amazon Managed Service for Prometheus 等 AWS 其他服務無法與混合節點上執行的工作負載通訊。

在混合節點安裝和升級期間需要存取

在安裝程序期間，您必須在主機上安裝混合節點相依性時，擁有下列網域的存取權。當您建置作業系統映像時，此程序可以完成一次，也可以在每個主機上執行。這包括初始安裝和升級混合節點的 Kubernetes 版本時。

元件	URL	通訊協定	連線埠
EKS 節點成品 (S3)	https://hybrid-assets.eks.amazonaws.com	HTTPS	443

元件	URL	通訊協定	連線埠
EKS 服務端點	https://eks.*region*.amazonaws.com	HTTPS	443
ECR 服務端點	https://api.ecr.*region*.amazonaws.com	HTTPS	443
EKS ECR 端點	如需區域端點the section called “檢視 Amazon 映像登錄檔”，請參閱 。	HTTPS	443
SSM 二進位端點 [1]	https：//amazon-ssm-*region*.s3.*region*.amazonaws.com	HTTPS	443
SSM 服務端點 [1]	https://ssm.*region*.amazonaws.com	HTTPS	443
IAM Anywhere 二進位端點 [2]	https://rolesanywhere.amazonaws.com	HTTPS	443
IAM Anywhere 服務端點 [2]	https://rolesanywhere.*region*.amazonaws.com	HTTPS	443

Note

[1] 只有在您為內部部署 AWS IAM 憑證提供者使用 AWS SSM 混合啟用時，才需要存取 SSM 端點。

[2] 只有在您為內部部署 AWS IAM 憑證提供者使用 AWS IAM Roles Anywhere 時，才需要存取 IAM 端點。

持續叢集操作所需的存取

持續叢集操作需要您現場部署防火牆的下列網路存取。

> **Important**
>
> 根據您選擇的 CNI，您需要為 CNI 連接埠設定其他網路存取規則。如需詳細資訊，請參閱 Cilium 文件和 Calico 文件。

Type	通訊協定	Direction	連線埠	來源	目的地	用量
HTTPS	TCP	傳出	443	遠端節點 CIDR (s)	EKS IPs [1]	kubelet 到 Kubernetes API 伺服器
HTTPS	TCP	傳出	443	遠端 Pod CIDR (s)	EKS 叢集 IPs [1]	Pod 到 Kubernetes API 伺服器
HTTPS	TCP	傳出	443	遠端節點 CIDR (s)	SSM 服務端點	SSM 混合啟用每隔 5 分鐘重新整理登入資料和 SSM 活動訊號
HTTPS	TCP	傳出	443	遠端節點 CIDR (s)	IAM Anywhere 服務端點	IAM Roles Anywhere 登入資料重新整理
HTTPS	TCP	傳出	443	遠端 Pod CIDR (s)	STS 區域端點	Pod 到 STS 端點，僅 IRSA 需要

Type	通訊協定	Direction	連線埠	來源	目的地	用量
HTTPS	TCP	傳出	443	遠端節點 CIDR (s)	Amazon EKS Auth 服務端點	節點到 Amazon EKS 身分驗證端點，僅 Amazon EKS Pod 身分需要
HTTPS	TCP	傳入	10250	EKS 叢集 IPs [1]	遠端節點 CIDR (s)	Kubernetes API 伺服器到 kubelet
HTTPS	TCP	傳入	Webhook 連接埠	EKS 叢集 IPs [1]	遠端 Pod CIDR (s)	Kubernetes API 伺服器到 Webhook
HTTPS	TCP、UDP	傳入、傳出	53	遠端 Pod CIDR (s)	遠端 Pod CIDR (s)	Pod 到 CoreDNS。如果您在雲端中執行至少 1 個 CoreDNS 複本，您必須允許 DNS 流量到執行 CoreDNS 的 VPC。
使用者定義	使用者定義	傳入、傳出	應用程式連接埠	遠端 Pod CIDR (s)	遠端 Pod CIDR (s)	Pod 到 Pod

> **Note**
>
> [1] EKS 叢集IPs。請參閱 Amazon EKS 彈性網路介面的下一節。

Amazon EKS 網路介面

Amazon EKS 會將網路介面連接到您在叢集建立期間傳遞的 VPC 中的子網路，以啟用 EKS 控制平面與 VPC 之間的通訊。在 Amazon EC2 主控台或使用 CLI 建立叢集之後，可以找到 Amazon EKS 建立的網路介面。 AWS 在 EKS 叢集上套用變更時，會刪除原始網路介面並建立新的網路介面，例如 Kubernetes 版本升級。您可以針對叢集建立期間傳遞的子網路使用限制子網路大小來限制 Amazon EKS 網路介面的 IP 範圍，這可讓您更輕鬆地設定內部部署防火牆，以允許對此已知、限制的一組 IPs 的傳入/傳出連線。若要控制要在哪些子網路中建立網路介面，您可以限制在建立叢集時指定的子網路數量，也可以在建立叢集之後更新子網路。

Amazon EKS 佈建的網路介面具有格式 的描述`Amazon EKS` *`your-cluster-name`* 。如需可用來尋找 Amazon AWS EKS 佈建之網路介面 IP 地址的 CLI 命令，請參閱下列範例。`VPC_ID` 將 取代為您在叢集建立期間傳遞的 VPC ID。

```
aws ec2 describe-network-interfaces \
--query 'NetworkInterfaces[?(VpcId == VPC_ID && contains(Description,Amazon
 EKS))].PrivateIpAddress'
```

AWS VPC 和子網路設定

Amazon EKS 的現有 VPC 和子網路需求適用於具有混合節點的叢集。此外，您的 VPC CIDR 無法與內部部署節點和 Pod CIDRs 重疊。您必須在 VPC 路由表中為內部部署節點和選用的 Pod CIDRs 設定路由。必須設定這些路由，將流量路由到您用於混合網路連線的閘道，這通常是虛擬私有閘道 (VGW) 或傳輸閘道 (TGW)。如果您使用 TGW 或 VGW 將 VPC 連接到內部部署環境，則必須為 VPC 建立 TGW 或 VGW 附件。您的 VPC 必須支援 DNS 主機名稱和 DNS 解析。

下列步驟使用 AWS CLI。您也可以在 中 AWS Management Console 或透過 AWS CloudFormation、AWS CDK 或 Terraform 等其他界面建立這些資源。

步驟 1：建立 VPC

1. 執行下列命令來建立 VPC。`VPC_CIDR` 將 取代為 IPv4 RFC-1918（私有） non-RFC-1918（公有）CIDR 範圍（例如)`10.0.0.0/16`。注意：預設會為 VPC 啟用 DNS 解析，這是 EKS 要求。

   ```
   aws ec2 create-vpc --cidr-block VPC_CIDR
   ```

2. 為您的 VPC 啟用 DNS 主機名稱。請注意，預設會為 VPC 啟用 DNS 解析。VPC_ID 以您在上一個步驟中建立的 VPC ID 取代 。

```
aws ec2 modify-vpc-attribute --vpc-id VPC_ID --enable-dns-hostnames
```

步驟 2：建立子網路

建立至少 2 個子網路。Amazon EKS 會將這些子網路用於叢集網路介面。如需詳細資訊，請參閱子網路需求和考量事項。

1. 您可以使用下列命令找到 AWS 區域的可用區域。將 取代us-west-2為您的區域。

```
aws ec2 describe-availability-zones \
     --query 'AvailabilityZones[?(RegionName == us-west-2)].ZoneName'
```

2. 建立子網。VPC_ID 將 取代為 VPC 的 ID。SUBNET_CIDR 將 取代為子網路的 CIDR 區塊 （ 例如 10.0.1.0/24)。AZ 將 取代為要建立子網路的可用區域 （ 例如 us-west-2a)。您建立的子網路必須至少位於 2 個不同的可用區域。

```
aws ec2 create-subnet \
    --vpc-id VPC_ID \
    --cidr-block SUBNET_CIDR \
    --availability-zone AZ
```

（ 選用 ） 步驟 3：使用 Amazon VPC Transit Gateway (TGW) 或 AWS Direct Connect 虛擬私有閘道 (VGW) 連接 VPC

如果您使用的是 TGW 或 VGW，請將 VPC 連接到 TGW 或 VGW。如需詳細資訊，請參閱 Amazon VPC Transit Gateways 中的 Amazon VPC 附件或 AWS Direct Connect 虛擬私有閘道關聯。

轉換閘道

執行下列命令來連接 Transit Gateway。VPC_ID 將 取代為 VPC 的 ID。SUBNET_ID2 使用您在上一個步驟中建立IDs 取代 SUBNET_ID1和 。TGW_ID 以 TGW 的 ID 取代 。

```
aws ec2 create-transit-gateway-vpc-attachment \
```

```
    --vpc-id VPC_ID \
    --subnet-ids SUBNET_ID1 SUBNET_ID2 \
    --transit-gateway-id TGW_ID
```

虛擬私有閘道

執行下列命令來連接 Transit Gateway。VPN_ID 以 VGW 的 ID 取代 。VPC_ID 將 取代為 VPC 的 ID。

```
aws ec2 attach-vpn-gateway \
    --vpn-gateway-id VPN_ID \
    --vpc-id VPC_ID
```

（選用）步驟 4：建立路由表

您可以修改 VPC 的主要路由表，也可以建立自訂路由表。下列步驟會建立自訂路由表，其中包含內部部署節點和 Pod CIDRs路由。如需詳細資訊，請參閱子網路路由表。VPC_ID 將 取代為 VPC 的 ID。

```
aws ec2 create-route-table --vpc-id VPC_ID
```

步驟 5：建立內部部署節點和 Pod 的路由

在路由表中為每個內部部署遠端節點建立路由。您可以修改 VPC 的主要路由表，或使用您在上一個步驟中建立的自訂路由表。

以下範例示範如何為您的內部部署節點和 Pod CIDRs 建立路由。在範例中，傳輸閘道 (TGW) 用於連接 VPC 與內部部署環境。如果您有多個內部部署節點和 Pod CIDRs，請為每個 CIDR 重複這些步驟。

- 如果您使用的是網際網路閘道或虛擬私有閘道 (VGW)，請將 取代`--transit-gateway-id`為 `--gateway-id`。
- RT_ID 以您在上一個步驟中建立的路由表 ID 取代 。
- REMOTE_NODE_CIDR 以您將用於混合節點的 CIDR 範圍取代 。
- REMOTE_POD_CIDR 將 取代為您在混合節點上執行的 Pod 所使用的 CIDR 範圍。Pod CIDR 範圍對應至容器網路界面 (CNI) 組態，這最常使用內部部署的浮水印網路。如需詳細資訊，請參閱the section called “設定 CNI”。
- TGW_ID 將 取代為 TGW 的 ID。

遠端節點網路

```
aws ec2 create-route \
    --route-table-id RT_ID \
    --destination-cidr-block REMOTE_NODE_CIDR \
    --transit-gateway-id TGW_ID
```

遠端 Pod 網路

```
aws ec2 create-route \
    --route-table-id RT_ID \
    --destination-cidr-block REMOTE_POD_CIDR \
    --transit-gateway-id TGW_ID
```

（選用）步驟 6：將子網路與路由表建立關聯

如果您在上一個步驟中建立了自訂路由表，請將您在上一個步驟中建立的每個子網路與您的自訂路由表建立關聯。如果您要修改 VPC 主路由表，子網路會自動與 VPC 的主路由表建立關聯，您可以略過此步驟。

針對您在先前步驟中建立的每個子網路執行下列命令。RT_ID 將 取代為您在上一個步驟中建立的路由表。SUBNET_ID 將 取代為子網路的 ID。

```
aws ec2 associate-route-table --route-table-id RT_ID --subnet-id SUBNET_ID
```

叢集安全群組組態

持續叢集操作需要 EKS 叢集安全群組的下列存取權。當您使用設定的遠端節點和 Pod 網路建立或更新叢集時，Amazon EKS 會自動為混合節點建立必要的輸入安全群組規則。

Type	通訊協定	Direction	連線埠	來源	目的地	用量
HTTPS	TCP	傳入	443	遠端節點 CIDR (s)	N/A	Kubelet 到 Kubernetes API 伺服器
HTTPS	TCP	傳入	443	遠端 Pod CIDR (s)	N/A	當 CNI 未針對 Pod

Type	通訊協定	Direction	連線埠	來源	目的地	用量
						流量使用 NAT 時，需要存取 K8s API 伺服器的 Pod。
HTTPS	TCP	傳出	10250	N/A	遠端節點 CIDR (s)	Kubernetes API 伺服器到 Kubelet
HTTPS	TCP	傳出	Webhook 連接埠	N/A	遠端 Pod CIDR (s)	Kubernetes API 伺服器到 Webhook（如果在混合節點上執行 Webhook)

Important

安全群組規則限制：Amazon EC2 安全群組預設最多有 60 個傳入規則。如果您的叢集安全群組接近此限制，則安全群組傳入規則可能不適用。在這種情況下，可能需要手動新增缺少的輸入規則。

CIDR 清除責任：如果您從 EKS 叢集中移除遠端節點或 Pod 網路，EKS 不會自動移除對應的安全群組規則。您有責任從安全群組規則手動移除未使用的遠端節點或 Pod 網路。

如需有關 Amazon EKS 建立的叢集安全群組的詳細資訊，請參閱 the section called “安全群組要求”。

（選用）手動安全群組組態

如果您需要建立其他安全群組或修改自動建立的規則，您可以使用下列命令做為參考。根據預設，以下命令會建立允許所有傳出存取的安全群組。您可以限制傳出存取只包含上述規則。如果您考慮限制傳出規則，建議您先徹底測試所有應用程式和 Pod 連線，再將已變更的規則套用至生產叢集。

- 在第一個命令中，SG_NAME將 取代為安全群組的名稱
- 在第一個命令中，將 取代VPC_ID為您在上一個步驟中建立的 VPC ID
- 在第二個命令中，將 取代SG_ID為您在第一個命令中建立的安全群組 ID
- 在第二個命令中，將 REMOTE_NODE_CIDR和 REMOTE_POD_CIDR 取代為混合節點和內部部署網路的值。

```
aws ec2 create-security-group \
    --group-name SG_NAME \
    --description "security group for hybrid nodes" \
    --vpc-id VPC_ID
```

```
aws ec2 authorize-security-group-ingress \
    --group-id SG_ID \
    --ip-permissions '[{"IpProtocol": "tcp", "FromPort": 443, "ToPort": 443,
 "IpRanges": [{"CidrIp": "REMOTE_NODE_CIDR"}, {"CidrIp": "REMOTE_POD_CIDR"}]}]'
```

準備混合節點的作業系統

Bottlerocket、Amazon Linux 2023 (AL2023)、Ubuntu 和 RHEL 會持續經過驗證，以用作混合節點的節點作業系統。Bottlerocket 僅支援 VMware vSphere AWS環境。在 Amazon EC2 外部執行時，AWS 支援計劃不涵蓋 AL2023。 Amazon EC2 AL2023 只能在內部部署虛擬化環境中使用，如需詳細資訊，請參閱 Amazon Linux 2023 使用者指南。 AWS 支援與 Ubuntu 和 RHEL 作業系統整合的混合節點，但不支援作業系統本身。

您負責作業系統佈建和管理。當您第一次測試混合節點時，最容易在已佈建的主機上執行 Amazon EKS 混合節點 CLI (nodeadm)。對於生產部署，我們建議您在作業系統映像nodeadm中包含 ，並將它設定為執行為系統化服務，以在主機啟動時自動將主機加入 Amazon EKS 叢集。如果您在 vSphere 上使用 Bottlerocket 做為節點作業系統，則不需要使用 ，nodeadm因為 Bottlerocket 已包含混合節點所需的相依性，並在主機啟動時自動連線到您設定的叢集。

版本相容性

下表代表相容且經過驗證可用作混合節點節點之節點作業系統的作業系統版本。如果您使用的是不包含在此資料表中的其他作業系統變體或版本，則 AWS Support 不會涵蓋混合節點與作業系統變體或版本的相容性。混合節點與基礎基礎設施無關，並支援 x86 和 ARM 架構。

作業系統	版本
Amazon Linux	Amazon Linux 2023 (AL2023)
Bottlerocket	v1.37.0 及更高版本執行 Kubernetes v1.28 及更高版本的 VMware 變體
Ubuntu	Ubuntu 20.04、Ubuntu 22.04、Ubuntu 24.04
Red Hat Enterprise Linux	RHEL 8、RHEL 9

作業系統考量事項

一般

- Amazon EKS 混合節點 CLI (nodeadm) 可用來簡化混合節點元件和相依性的安裝和組態。您可以在作業系統映像建置管道期間或每個現場部署主機的執行時間執行`nodeadm install`程序。如需`nodeadm`安裝 元件的詳細資訊，請參閱 the section called “混合節點 nodeadm”。
- 如果您在內部部署環境中使用代理連線到網際網路，安裝和升級程序需要額外的作業系統組態，才能將套件管理員設定為使用代理。如需說明，請參閱 the section called “設定代理”。

Bottlerocket

- 連接 Bottlerocket 節點的步驟和工具與其他作業系統的步驟不同，並分別涵蓋在 中the section called “使用 Bottlerocket 連接混合節點”，而不是 中的步驟the section called “連接混合節點”。
- Bottlerocket 的步驟不使用混合節點 CLI 工具 `nodeadm`。
- EKS 混合節點僅支援 Bottlerocket 1.37.0 版及更高版本的 VMware 變體。Bottlerocket 的 VMware 變體適用於 Kubernetes 1.28 版及更高版本。不支援其他 Bottlerocket 變體做為混合節點作業系統。注意： Bottlerocket 的 VMware 變體僅適用於 x86_64 架構。

Containerd

- Containerd 是標準 Kubernetes 容器執行期，是混合節點以及所有 Amazon EKS 節點運算類型的相依性。Amazon EKS 混合節點 CLI (nodeadm) 會嘗試在`nodeadm install`程序期間安裝容器。您可以使用`--containerd-source`命令列選項，在`nodeadm install`執行時間設定容器安裝。有效選項為 `none`、 `distro`和 `docker`。如果您使用的是 RHEL， `distro` 不是有效的選項，您可以`nodeadm`設定 從 Docker 的儲存庫安裝容器化建置，也可以手動安裝容器化。使用 AL2023 或

Ubuntu 時， nodeadm預設為從作業系統分佈安裝容器。如果您不希望 nodeadm 安裝 containerd，請使用 `--containerd-source none`選項。

Ubuntu

- 如果您使用的是 Ubuntu 20.04，則必須使用 AWS Systems Manager 混合啟用做為登入資料提供者。Ubuntu 20.04 不支援 AWS IAM Roles Anywhere。
- 如果您使用的是 Ubuntu 24.04，您可能需要更新容器版本，或變更 AppArmor 組態以採用允許 Pod 正確終止的修正，請參閱 Ubuntu #2065423。需要重新開機才能將變更套用至 AppArmor 設定檔。最新版本的 Ubuntu 24.04 在其套件管理員中具有更新的容器版本，並具有修正（容器版本 1.7.19+)。

RHEL

- 如果您使用的是 RHEL 8，則必須使用 AWS Systems Manager 混合啟用做為登入資料提供者。RHEL 8 不支援 AWS IAM Roles Anywhere。

ARM

- 如果您使用的是 ARM 硬體，則需要具有密碼編譯延伸 (ARMv8.2+crypto) 的 ARMv8.2 相容處理器，才能執行 EKS kube-proxy 附加元件的 1.31 版和更新版本。ARMv8.2+crypto) Raspberry Pi 5 之前的所有 Raspberry Pi 系統，以及 Cortex-A72 型處理器都不符合此要求。作為解決方法，您可以繼續使用 EKS kube-proxy 附加元件的 1.30 版，直到 2026 年 7 月擴展支援結束為止，請參閱 eks/latest/userguide/kubernetes-versions.html【Kubernetes Release calendar，type="documentation"】，或使用上游的自訂 kube-proxy 映像。
- kube-proxy 日誌中的下列錯誤訊息指出此不相容：

```
Fatal glibc error: This version of Amazon Linux requires a newer ARM64 processor
 compliant with at least ARM architecture 8.2-a with Cryptographic extensions. On EC2
 this is Graviton 2 or later.
```

建置作業系統映像

Amazon EKS 提供範例 Packer 範本，可用來建立包含 的作業系統映像，nodeadm並將其設定為在主機啟動時執行。建議此程序以避免在每個主機上個別提取混合節點相依性，並自動化混合節點引導程

序。您可以使用範例 Packer 範本搭配 Ubuntu 22.04、Ubuntu 24.04、RHEL 8 或 RHEL 9 ISO 映像，並可使用這些格式輸出映像：OVA、Qcow2 或 raw。

先決條件

使用範例 Packer 範本之前，您必須在執行 Packer 的機器上安裝下列項目。

- Packer 1.11.0 版或更新版本。如需安裝 Packer 的指示，請參閱 Packer 文件中的安裝 Packer。
- 如果建置 OVAs，VMware vSphere 外掛程式 1.4.0 或更高版本
- 如果是建置Qcow2或原始映像，QEMU 外掛程式 1.x 版

設定環境變數

在執行 Packer 建置之前，請在執行 Packer 的電腦上設定下列環境變數。

一般

必須設定下列環境變數，以使用所有作業系統和輸出格式建置映像。

環境變數	Type	描述
PKR_SSH_PASSWORD	字串	Packer 在佈建時使用 `ssh_username` 和 `ssh_password` 變數，將 SSH 傳送至建立的機器。這需要符合用於在個別作業系統的開始或使用者資料檔案中建立初始使用者的密碼。預設值會設定為「建置器」或「ubuntu」，視作業系統而定。設定密碼時，請務必在對應的 `ks.cfg`或 `user-data` 檔案中將其變更為相符。
ISO_URL	字串	要使用的 ISO URL。可以是從伺服器下載的 Web 連結，也可以是本機檔案的絕對路徑
ISO_CHECKSUM	字串	所提供 ISO 的相關檢查總和。

環境變數	Type	描述
CREDENTIAL_PROVIDER	字串	混合節點的登入資料提供者。SSM 混合啟用和 IAM Roles Anywhere 的有效值為 `iam` `ssm`（預設值）
K8S_VERSION	字串	混合節點的 Kubernetes 版本（例如 `1.31`）。如需支援的 Kubernetes 版本，請參閱 eks/latest/userguide/kubernetes-versions.html【Amazon EKS 支援的版本，type="documentation"】。
NODEADM_ARCH	字串	的架構`nodeadm install`。選取 `amd` 或 `arm`。

RHEL

如果您使用的是 RHEL，則必須設定下列環境變數。

環境變數	Type	描述
RH_USERNAME	字串	RHEL 訂閱管理員使用者名稱
RH_PASSWORD	字串	RHEL 訂閱管理員密碼
RHEL_VERSION	字串	正在使用 Rhel iso 版本。有效值為 8 或 9。

Ubuntu

不需要 Ubuntu 特定的環境變數。

vSphere

如果您要建置 VMware vSphere OVA，則必須設定下列環境變數。

環境變數	Type	描述
VSPHERE_SERVER	字串	vSphere 伺服器地址
VSPHERE_USER	字串	vSphere 使用者名稱
VSPHERE_PASSWORD	字串	vSphere 密碼
VSPHERE_DATACENTER	字串	vSphere 資料中心名稱
VSPHERE_CLUSTER	字串	vSphere 叢集名稱
VSPHERE_DATASTORE	字串	vSphere 資料存放區名稱
VSPHERE_NETWORK	字串	vSphere 網路名稱
VSPHERE_OUTPUT_FOLDER	字串	範本的 vSphere 輸出資料夾

QEMU

環境變數	Type	描述
PACKER_OUTPUT_FORMAT	字串	QEMU 建置器的輸出格式。有效值為 qcow2 和 raw。

驗證範本

執行建置之前，請在設定環境變數後，使用以下命令驗證範本。template.pkr.hcl 如果您為範本使用不同的名稱，請取代 。

```
packer validate template.pkr.hcl
```

建置映像

使用以下命令建置映像，並使用 -only旗標指定映像的目標和作業系統。template.pkr.hcl 如果您為範本使用不同的名稱，請取代 。

vSphere OVAs

> **Note**
>
> 如果您將 RHEL 與 vSphere 搭配使用，則需要將啟動檔案轉換為 OEMDRV 映像，並將其做為 ISO 傳遞以從中開機。如需詳細資訊，請參閱 EKS 混合節點 GitHub 儲存庫中的 Packer Readme。

Ubuntu 22.04 OVA

```
packer build -only=general-build.vsphere-iso.ubuntu22 template.pkr.hcl
```

Ubuntu 24.04 OVA

```
packer build -only=general-build.vsphere-iso.ubuntu24 template.pkr.hcl
```

RHEL 8 OVA

```
packer build -only=general-build.vsphere-iso.rhel8 template.pkr.hcl
```

RHEL 9 OVA

```
packer build -only=general-build.vsphere-iso.rhel9 template.pkr.hcl
```

QEMU

> **Note**
>
> 如果您要為不符合建置器主機的特定主機 CPU 建置映像，請參閱 QEMU 文件以取得符合主機 CPU 的名稱，並在執行下列命令時，使用具有主機 CPU 名稱的 `-cpu`旗標。

Ubuntu 22.04 Qcow2 / 原始

```
packer build -only=general-build.qemu.ubuntu22 template.pkr.hcl
```

Ubuntu 24.04 Qcow2 / 原始

```
packer build -only=general-build.qemu.ubuntu24 template.pkr.hcl
```

RHEL 8 Qcow2/原始

```
packer build -only=general-build.qemu.rhel8 template.pkr.hcl
```

RHEL 9 Qcow2/原始

```
packer build -only=general-build.qemu.rhel9 template.pkr.hcl
```

透過使用者資料傳遞 nodeadm 組態

您可以透過 cloud-init 傳遞使用者資料nodeadm中的 組態，以在主機啟動時設定混合節點並自動連線至 EKS 叢集。以下是使用 VMware vSphere 做為混合節點基礎設施時，如何達成此目的的範例。

1. 遵循 GitHub 上 govc readme 中的指示安裝 govc CLI。
2. 在上一節中執行 Packer 建置並佈建範本後，您可以使用下列項目複製範本以建立多個不同的節點。您必須複製要建立且將用於混合節點之每個新 VM 的範本。將以下命令中的變數取代為您環境的值。當您透過 metadata.yaml 檔案插入 VMs 的名稱NODE_NAME時，以下命令VM_NAME中的 會做為您的 。

   ```
   govc vm.clone -vm "/PATH/TO/TEMPLATE" -ds="YOUR_DATASTORE" \
       -on=false -template=false -folder=/FOLDER/TO/SAVE/VM "VM_NAME"
   ```

3. 複製每個新 VMs 的範本後，metadata.yaml請為您的 VMs 建立 userdata.yaml和 。您的 VMs 可以共用相同的 userdata.yamlmetadata.yaml，而且您將在以下步驟中以每個 VM 為基礎填入這些 VM。nodeadm 組態會在 的 write_files區段中建立和定義userdata.yaml。以下範例使用 AWS SSM 混合啟用做為混合節點的內部部署憑證提供者。如需nodeadm組態的詳細資訊，請參閱 the section called “混合節點 nodeadm”。

   userdata.yaml：

   ```
   #cloud-config
   users:
     - name: # username for login. Use 'builder' for RHEL or 'ubuntu' for Ubuntu.
       passwd: # password to login. Default is 'builder' for RHEL.
       groups: [adm, cdrom, dip, plugdev, lxd, sudo]
       lock-passwd: false
       sudo: ALL=(ALL) NOPASSWD:ALL
       shell: /bin/bash

   write_files:
   ```

```
  - path: /usr/local/bin/nodeConfig.yaml
    permissions: '0644'
    content: |
      apiVersion: node.eks.aws/v1alpha1
      kind: NodeConfig
      spec:
          cluster:
              name: # Cluster Name
              region: # AWS region
          hybrid:
              ssm:
                  activationCode: # Your ssm activation code
                  activationId: # Your ssm activation id

runcmd:
  - /usr/local/bin/nodeadm init -c file:///usr/local/bin/nodeConfig.yaml >> /var/log/
nodeadm-init.log 2>&1
```

metadata.yaml：

metadata.yaml 為您的環境建立 。將"$NODE_NAME"變數格式保留在 檔案中，因為這會在後續步驟中填入值。

```
instance-id: "$NODE_NAME"
local-hostname: "$NODE_NAME"
network:
  version: 2
  ethernets:
    nics:
      match:
        name: ens*
      dhcp4: yes
```

4. 使用下列命令將 userdata.yaml和 metadata.yaml 檔案新增為gzip+base64字串。應該為您建立的每個 VMs 執行下列命令。VM_NAME 將 取代為您更新之 VM 的名稱。

```
export NODE_NAME="VM_NAME"
export USER_DATA=$(gzip -c9 <userdata.yaml | base64)

govc vm.change -dc="YOUR_DATASTORE" -vm "$NODE_NAME" -e
 guestinfo.userdata="${USER_DATA}"
```

```
govc vm.change -dc="YOUR_DATASTORE" -vm "$NODE_NAME" -e
 guestinfo.userdata.encoding=gzip+base64

envsubst '$NODE_NAME' < metadata.yaml > metadata.yaml.tmp
export METADATA=$(gzip -c9 <metadata.yaml.tmp | base64)

govc vm.change -dc="YOUR_DATASTORE" -vm "$NODE_NAME" -e
 guestinfo.metadata="${METADATA}"
govc vm.change -dc="YOUR_DATASTORE" -vm "$NODE_NAME" -e
 guestinfo.metadata.encoding=gzip+base64
```

5. 開啟新 VMs的電源，這應該會自動連接到您設定的 EKS 叢集。

```
govc vm.power -on "${NODE_NAME}"
```

準備混合節點的登入資料

Amazon EKS 混合節點使用由 AWS SSM 混合啟用或 IAM Roles Anywhere AWS 佈建的臨時 IAM 登入資料來驗證 Amazon EKS 叢集。您必須將 AWS SSM 混合啟用或 AWS IAM Roles Anywhere 與 Amazon EKS 混合節點 CLI () 搭配使用nodeadm。您不應該同時使用 AWS SSM 混合啟用和 AWS IAM Roles Anywhere。如果您現有的公有金鑰基礎設施 (PKI) 沒有憑證授權機構 (CA) 和內部部署環境的憑證，建議您使用 AWS SSM 混合啟用。如果您有現有的 PKI 和現場部署憑證，請使用 AWS IAM Roles Anywhere。

混合節點 IAM 角色

在將混合節點連接到 Amazon EKS 叢集之前，您必須建立 IAM 角色，以搭配混合節點憑證的 AWS SSM 混合啟用或 AWS IAM Roles Anywhere 使用。建立叢集後，您將使用此角色搭配 Amazon EKS 存取項目或 aws-auth ConfigMap 項目，將 IAM 角色映射至 Kubernetes 角色型存取控制 (RBAC)。如需將混合節點 IAM 角色與 Kubernetes RBAC 建立關聯的詳細資訊，請參閱 the section called “準備叢集存取”。

混合節點 IAM 角色必須具有下列許可。

- nodeadm 使用 eks:DescribeCluster動作來收集用於將混合節點連線至叢集之叢集相關資訊的許可。如果您未啟用 eks:DescribeCluster動作，則必須在執行nodeadm初始化nodeadm時傳遞節點組態中的 Kubernetes API 端點、叢集 CA 套件和服務 IPv4 CIDR。
- kubelet 使用 Amazon Elastic Container Registry (Amazon ECR) 中容器映像的許可，如 AmazonEC2ContainerRegistryPullOnly 政策所定義。

- 如果使用 AWS SSM，則nodeadm初始化使用 AWS SSM 混合啟用的許可，如 aws-managed-policy/latest/reference/AmazonSSMManagedInstanceCore.html 政策所定義。
- 如果使用 AWS SSM，則使用 `ssm:DeregisterManagedInstance`動作和 `ssm:DescribeInstanceInformation`動作`nodeadm uninstall`取消註冊執行個體的許可。
- (選用) Amazon EKS Pod 身分識別代理程式使用 `eks-auth:AssumeRoleForPodIdentity` 動作擷取 Pod 憑證的許可。

設定 AWS SSM 混合啟用

在設定 AWS SSM 混合啟用之前，您必須建立並設定混合節點 IAM 角色。如需詳細資訊，請參閱the section called “建立混合節點 IAM 角色”。遵循 Systems Manager 使用者指南中的建立混合啟用中的指示，向 Systems Manager 註冊節點，為您的混合節點建立 AWS SSM 混合啟用。 AWS 當您向 Amazon EKS 叢集將主機註冊為混合節點nodeadm時，會搭配使用您收到的啟用碼和 ID。您可以在建立並準備混合節點的 Amazon EKS 叢集之後，稍後再返回此步驟。

> **Important**
>
> Systems Manager 會立即將啟用代碼和 ID 傳回主控台或命令視窗，視您如何建立啟用而定。複製此資訊，並將其存放在安全的地方。如果您離開主控台或關閉命令視窗，您可能會遺失此資訊。如果您遺失此資訊，您必須建立新的啟用。

根據預設， AWS SSM 混合啟用會處於作用中狀態 24 小時。您也可以在建立時間戳記格式的混合啟用`--expiration-date`時指定 ，例如 `2024-08-01T00:00:00`。當您使用 AWS SSM 做為登入資料提供者時，混合節點的節點名稱無法設定，且由 AWS SSM 自動產生。您可以在 Fleet Manager 下的 AWS Systems Manager 主控台中檢視和管理 AWS SSM 受管執行個體。每個 AWS 區域每個帳戶最多可註冊 1，000 個標準混合啟用節點，無需額外費用。不過，登錄超過 1,000 個混合節點需要啟用進階執行個體層。使用 Amazon EKS 混合節點定價中未包含的 advanced-instances 方案需支付費用。如需詳細資訊，請參閱 AWS Systems Manager 定價。

請參閱以下範例，了解如何使用混合節點 IAM 角色建立 AWS SSM 混合啟用。當您針對混合節點登入資料使用 AWS SSM 混合啟用時，混合節點的名稱將具有 格式，`mi-012345678abcdefgh`且 AWS SSM 佈建的臨時登入資料有效期為 1 小時。使用 AWS SSM 做為登入資料提供者時，您無法變更節點名稱或登入資料持續時間。 AWS SSM 會自動輪換臨時憑證，輪換不會影響節點或應用程式的狀態。

我們建議您每個 EKS 叢集使用一個 AWS SSM 混合啟用，將混合節點 IAM 角色的 AWS SSM `ssm:DeregisterManagedInstance`許可範圍限定為只能取消註冊與您的 AWS SSM 混合啟用相

關聯的執行個體。在此頁面上的範例中，使用具有 EKS 叢集 ARN 的標籤，可用於將您的 AWS SSM 混合啟用映射至 EKS 叢集。或者，您也可以根據您的許可界限和要求，使用偏好的標籤和方法來界定 AWS SSM 許可範圍。以下命令中的 REGISTRATION_LIMIT選項是整數，用於限制可使用 AWS SSM 混合啟用的機器數量 （ 例如 10)

```
aws ssm create-activation \
     --region AWS_REGION \
     --default-instance-name eks-hybrid-nodes \
     --description "Activation for EKS hybrid nodes" \
     --iam-role AmazonEKSHybridNodesRole \
     --tags Key=EKSClusterARN,Value=arn:aws: eks:AWS_REGION:AWS_ACCOUNT_ID:cluster/
CLUSTER_NAME \
     --registration-limit REGISTRATION_LIMIT
```

如需 AWS SSM 混合啟用可用組態設定的詳細資訊，請參閱建立混合啟用以向 Systems Manager 註冊節點的說明。

隨處設定 AWS IAM 角色

請遵循《IAM Roles Anywhere 使用者指南》中的 IAM Roles Anywhere 入門中的指示，設定您要用於混合節點 IAM 角色臨時 IAM 憑證的信任錨點和設定檔。建立設定檔時，您可以建立設定檔，而無需新增任何角色。您可以建立此描述檔，返回這些步驟來建立混合節點 IAM 角色，然後在建立後將角色新增至您的描述檔。或者，您可以在此頁面稍後使用 AWS CloudFormation 步驟來完成混合節點的 IAM Roles Anywhere 設定。

當您將混合節點 IAM 角色新增至設定檔時，請在 IAM Roles Anywhere 主控台的編輯設定檔頁面底部的自訂角色工作階段名稱面板中選取接受自訂角色工作階段名稱。 AWS 這對應至 CreateProfile API 的 acceptRoleSessionName 欄位。這可讓您在引導程序nodeadm期間傳遞至 的組態中，為混合節點提供自訂節點名稱。在nodeadm init程序期間傳遞自訂節點名稱是必要的。您可以在建立設定檔之後，更新您的設定檔以接受自訂角色工作階段名稱。

您可以透過 IAM Roles Anywhere AWS 設定檔的 durationSeconds 欄位，使用 IAM Roles Anywhere AWS 設定憑證有效性持續時間。預設持續時間為 1 小時，最多 12 小時。混合節點 IAM 角色上的MaxSessionDuration設定必須大於 IAM Roles Anywhere AWS 設定檔上的durationSeconds設定。如需 的詳細資訊MaxSessionDuration，請參閱 UpdateRole API 文件。

您從憑證授權單位 (CA) 產生的每個機器憑證和金鑰，必須放置在每個混合節點的 /etc/iam/pki目錄中，其中包含憑證和server.key金鑰server.pem的檔案名稱。

建立混合節點 IAM 角色

若要執行本節中的步驟，使用 AWS 主控台或 CLI 的 IAM AWS 主體必須具有下列許可。

- `iam:CreatePolicy`
- `iam:CreateRole`
- `iam:AttachRolePolicy`
- 如果使用 AWS IAM Roles Anywhere
  - `rolesanywhere:CreateTrustAnchor`
  - `rolesanywhere:CreateProfile`
  - `iam:PassRole`

AWS CloudFormation

如果您尚未安裝和設定 AWS CLI。請參閱安裝或更新至最新版本的 AWS CLI。

AWS SSM 混合啟用的步驟

CloudFormation 堆疊會使用上述許可建立混合節點 IAM 角色。CloudFormation 範本不會建立 AWS SSM 混合啟用。

1. 下載混合節點的 AWS SSM CloudFormation 範本：

```
curl -OL 'https://raw.githubusercontent.com/aws/eks-hybrid/refs/heads/main/example/
hybrid-ssm-cfn.yaml'
```

2. `cfn-ssm-parameters.json` 使用下列選項建立 ：
   a. `ROLE_NAME` 將 取代為混合節點 IAM 角色的名稱。根據預設，如果您未指定名稱，CloudFormation 範本會使用 `AmazonEKSHybridNodesRole` 做為其建立的角色名稱。
   b. `TAG_KEY` 將 取代為您在建立 AWS SSM 混合啟用時使用的 AWS SSM 資源標籤金鑰。標籤索引鍵和標籤值的組合用於 的條件`ssm:DeregisterManagedInstance`，只允許混合節點 IAM 角色取消註冊與您的 AWS SSM 混合啟用相關聯的 AWS SSM 受管執行個體。在 CloudFormation 範本中， `TAG_KEY` 預設為 `EKSClusterARN`。
   c. `TAG_VALUE` 將 取代為您建立 AWS SSM 混合啟用時使用的 AWS SSM 資源標籤值。標籤索引鍵和標籤值的組合用於 的條件`ssm:DeregisterManagedInstance`，只允許混合節點 IAM 角色取消註冊與您的 AWS SSM 混合啟用相關聯的 AWS SSM 受管執行個體。如果您使用的是預設值

TAG_KEY EKSClusterARN，請將您的 EKS 叢集 ARN 做為 傳遞TAG_VALUE。EKS 叢集 ARNs 的格式為 arn:aws: eks:AWS_REGION:AWS_ACCOUNT_ID:cluster/CLUSTER_NAME。

```
{
  "Parameters": {
    "RoleName": "ROLE_NAME",
    "SSMDeregisterConditionTagKey": "TAG_KEY",
    "SSMDeregisterConditionTagValue": "TAG_VALUE"
  }
}
```

3. 部署 CloudFormation 堆疊。STACK_NAME 將 取代為 CloudFormation 堆疊的名稱。

```
aws cloudformation deploy \
    --stack-name STACK_NAME \
    --template-file hybrid-ssm-cfn.yaml \
    --parameter-overrides file://cfn-ssm-parameters.json \
    --capabilities CAPABILITY_NAMED_IAM
```

AWS IAM Roles Anywhere 的步驟

CloudFormation 堆疊會使用您設定的憑證授權機構 AWS (CA) 建立 IAM Roles Anywhere 信任錨點、建立 AWS IAM Roles Anywhere 設定檔，以及使用先前概述的許可建立混合節點 IAM 角色。

1. 設定憑證授權單位 (CA)
   a. 若要使用 AWS Private CA 資源，請開啟 AWS Private Certificate Authority 主控台。請遵循 AWS Private CA 使用者指南中的指示。
   b. 若要使用外部 CA，請遵循 CA 提供的指示。您會在後續步驟中提供憑證內文。
   c. 從公有 CAs發出的憑證不能用作信任錨點。
2. 下載混合節點的 AWS IAM Roles Anywhere CloudFormation 範本

```
curl -OL 'https://raw.githubusercontent.com/aws/eks-hybrid/refs/heads/main/example/
hybrid-ira-cfn.yaml'
```

3. cfn-iamra-parameters.json 使用下列選項建立 ：
   a. ROLE_NAME 將 取代為混合節點 IAM 角色的名稱。根據預設，如果您未指定名稱，CloudFormation 範本會使用 AmazonEKSHybridNodesRole 做為其建立的角色名稱。

b. CERT_ATTRIBUTE 將 取代為可唯一識別您主機的每個機器憑證屬性。您使用的憑證屬性必須符合您在將混合節點連接至叢集時用於nodeadm組態的 nodeName。如需更多資訊，請參閱the section called “混合節點 nodeadm”。根據預設，CloudFormation 範本會使用 ${aws:PrincipalTag/x509Subject/CN}做為 CERT_ATTRIBUTE，其對應至每個機器憑證的 CN 欄位。或者，您可以將 $(aws:PrincipalTag/x509SAN/Name/CN}做為 傳遞CERT_ATTRIBUTE。

c. CA_CERT_BODY 以無換行符號的 CA 憑證主體取代 。CA_CERT_BODY 必須是隱私權增強郵件 (PEM) 格式。如果您有 PEM 格式的 CA 憑證，請先移除換行和 BEGIN CERTIFICATE 和 END CERTIFICATE 換行，再將 CA 憑證內文放入 cfn-iamra-parameters.json 檔案。

```
{
  "Parameters": {
    "RoleName": "ROLE_NAME",
    "CertAttributeTrustPolicy": "CERT_ATTRIBUTE",
    "CABundleCert": "CA_CERT_BODY"
  }
}
```

4. 部署 CloudFormation 範本。STACK_NAME 將 取代為 CloudFormation 堆疊的名稱。

```
aws cloudformation deploy \
    --stack-name STACK_NAME \
    --template-file hybrid-ira-cfn.yaml \
    --parameter-overrides file://cfn-iamra-parameters.json
    --capabilities CAPABILITY_NAMED_IAM
```

AWS CLI

如果您尚未安裝和設定 AWS CLI。請參閱安裝或更新至最新版本的 AWS CLI。

建立 EKS 描述叢集政策

1. 使用下列內容建立名為 eks-describe-cluster-policy.json 的檔案：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
```

```
                "eks:DescribeCluster"
            ],
            "Resource": "*"
        }
    ]
}
```

2. 使用下列命令建立政策：

```
aws iam create-policy \
    --policy-name EKSDescribeClusterPolicy \
    --policy-document file://eks-describe-cluster-policy.json
```

AWS SSM 混合啟用的步驟

1. 使用下列內容建立名為 eks-hybrid-ssm-policy.json 的檔案。此政策會授予兩個動作 ssm:DescribeInstanceInformation和 的許可ssm:DeregisterManagedInstance。此政策會根據您在信任政策中指定的資源標籤，將ssm:DeregisterManagedInstance許可限制為與您的 SSM 混合啟用相關聯的 AWS SSM AWS 受管執行個體。
   a. AWS_REGION 將 取代為 AWS SSM 混合啟用 AWS 的區域。
   b. 將 取代AWS_ACCOUNT_ID為 AWS 您的帳戶 ID。
   c. TAG_KEY 將 取代為您在建立 AWS SSM 混合啟用時使用的 AWS SSM 資源標籤金鑰。標籤索引鍵和標籤值的組合用於 的條件ssm:DeregisterManagedInstance，只允許混合節點 IAM 角色取消註冊與您的 AWS SSM 混合啟用相關聯的 AWS SSM 受管執行個體。在 CloudFormation 範本中， TAG_KEY 預設為 EKSClusterARN。
   d. TAG_VALUE 將 取代為您建立 AWS SSM 混合啟用時使用的 AWS SSM 資源標籤值。標籤索引鍵和標籤值的組合用於 的條件ssm:DeregisterManagedInstance，只允許混合節點 IAM 角色取消註冊與您的 AWS SSM 混合啟用相關聯的 AWS SSM 受管執行個體。如果您使用的是預設值 TAG_KEY EKSClusterARN，請將您的 EKS 叢集 ARN 做為 傳遞TAG_VALUE。EKS 叢集 ARNs 的格式為 arn:aws: eks:AWS_REGION:AWS_ACCOUNT_ID:cluster/CLUSTER_NAME。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "ssm:DescribeInstanceInformation",
            "Resource": "*"
```

```
        },
        {
            "Effect": "Allow",
            "Action": "ssm:DeregisterManagedInstance",
            "Resource": "arn:aws: ssm:AWS_REGION:AWS_ACCOUNT_ID:managed-instance/
*",
            "Condition": {
                "StringEquals": {
                    "ssm:resourceTag/TAG_KEY": "TAG_VALUE"
                }
            }
        }
    ]
}
```

2. 使用下列命令建立政策

```
aws iam create-policy \
    --policy-name EKSHybridSSMPolicy \
    --policy-document file://eks-hybrid-ssm-policy.json
```

3. 建立名為 eks-hybrid-ssm-trust.json 的檔案。AWS_REGION 將 取代為 AWS SSM 混合啟用 AWS 的區域，並將 AWS_ACCOUNT_ID取代為 AWS 您的帳戶 ID。

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"",
         "Effect":"Allow",
         "Principal":{
            "Service":"ssm.amazonaws.com"
         },
         "Action":"sts:AssumeRole",
         "Condition":{
            "StringEquals":{
               "aws:SourceAccount":"AWS_ACCOUNT_ID"
            },
            "ArnEquals":{
               "aws:SourceArn":"arn:aws: ssm:AWS_REGION:AWS_ACCOUNT_ID:*"
            }
         }
      }
```

```
    ]
}
```

4. 使用下列命令建立角色。

```
aws iam create-role \
    --role-name AmazonEKSHybridNodesRole \
    --assume-role-policy-document file://eks-hybrid-ssm-trust.json
```

5. 連接EKSHybridSSMPolicy您在先前步驟中建立的 EKSDescribeClusterPolicy和 。將 取代AWS_ACCOUNT_ID為 AWS 您的帳戶 ID。

```
aws iam attach-role-policy \
    --role-name AmazonEKSHybridNodesRole \
    --policy-arn arn:aws: iam::AWS_ACCOUNT_ID:policy/EKSDescribeClusterPolicy
```

```
aws iam attach-role-policy \
    --role-name AmazonEKSHybridNodesRole \
    --policy-arn arn:aws: iam::AWS_ACCOUNT_ID:policy/EKSHybridSSMPolicy
```

6. 連接 AmazonEC2ContainerRegistryPullOnly和 AmazonSSMManagedInstanceCore AWS 受管政策。

```
aws iam attach-role-policy \
    --role-name AmazonEKSHybridNodesRole \
    --policy-arn arn:aws: iam::aws:policy/AmazonEC2ContainerRegistryPullOnly
```

```
aws iam attach-role-policy \
    --role-name AmazonEKSHybridNodesRole \
    --policy-arn arn:aws: iam::aws:policy/AmazonSSMManagedInstanceCore
```

IAM Roles Anywhere AWS 的步驟

若要使用 AWS IAM Roles Anywhere，您必須在建立混合節點 AWS IAM 角色之前設定 IAM Roles Anywhere 信任錨點。如需說明，請參閱 the section called “隨處設定 AWS IAM 角色”。

1. 建立名為 eks-hybrid-iamra-trust.json 的檔案。TRUST_ANCHOR ARN 以您在the section called “隨處設定 AWS IAM 角色”步驟中建立的信任錨點 ARN 取代 。此信任政策中的 條件會限制 AWS IAM Roles Anywhere 擔任混合節點 IAM 角色的能力，只有在角色工作階段名稱符合安裝在混

合節點上 x509 憑證中的 CN 時，才能交換臨時 IAM 憑證。或者，您也可以使用其他憑證屬性來唯一識別您的節點。您在信任政策中使用的憑證屬性必須對應nodeName至您在nodeadm組態中設定的 。如需更多資訊，請參閱the section called “混合節點 nodeadm”。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "rolesanywhere.amazonaws.com"
            },
            "Action": [
                "sts:TagSession",
                "sts:SetSourceIdentity"
            ],
            "Condition": {
                "ArnEquals": {
                    "aws:SourceArn": "TRUST_ANCHOR_ARN"
                }
            }
        },
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "rolesanywhere.amazonaws.com"
            },
            "Action": "sts:AssumeRole",
            "Condition": {
                "StringEquals": {
                    "sts:RoleSessionName": "${aws:PrincipalTag/x509Subject/CN}"
                },
                "ArnEquals": {
                    "aws:SourceArn": "TRUST_ANCHOR_ARN"
                }
            }
        }
    ]
}
```

2. 使用下列命令建立角色。

```
aws iam create-role \
```

```
    --role-name AmazonEKSHybridNodesRole \
    --assume-role-policy-document file://eks-hybrid-iamra-trust.json
```

3. 連接EKSDescribeClusterPolicy您在先前步驟中建立的 。將 取代AWS_ACCOUNT_ID為 AWS 您的帳戶 ID。

```
aws iam attach-role-policy \
    --role-name AmazonEKSHybridNodesRole \
    --policy-arn arn:aws: iam::AWS_ACCOUNT_ID:policy/EKSDescribeClusterPolicy
```

4. 連接 AmazonEC2ContainerRegistryPullOnly AWS 受管政策

```
aws iam attach-role-policy \
    --role-name AmazonEKSHybridNodesRole \
    --policy-arn arn:aws: iam::aws:policy/AmazonEC2ContainerRegistryPullOnly
```

AWS Management Console

建立 EKS 描述叢集政策

1. 開啟 Amazon IAM 主控台
2. 在左側導覽窗格中選擇 Policies (政策)。
3. 在政策頁面上，選擇建立政策。
4. 在指定許可頁面的選取服務面板中，選擇 EKS。
   a. 篩選 DescribeCluster 的動作，然後選取 DescribeCluster Read 動作。
   b. 選擇下一步。
5. 在檢閱和建立頁面上
   a. 輸入政策的政策名稱，例如 EKSDescribeClusterPolicy。
   b. 選擇建立政策。

AWS SSM 混合啟用的步驟

1. 開啟 Amazon IAM 主控台
2. 在左側導覽窗格中選擇 Policies (政策)。
3. 在政策頁面上，選擇建立政策。

4. 在指定許可頁面上的政策編輯器右上角導覽中，選擇 JSON。貼上下列程式碼片段。AWS_REGION 將 取代為 AWS SSM 混合啟用 AWS 的區域，並將 取代AWS_ACCOUNT_ID為 AWS 您的帳戶 ID。將 TAG_KEY和 取代TAG_VALUE為您在建立 AWS SSM 混合啟用時使用的 AWS SSM 資源標籤金鑰。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "ssm:DescribeInstanceInformation",
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "ssm:DeregisterManagedInstance",
            "Resource": "arn:aws: ssm:AWS_REGION:AWS_ACCOUNT_ID:managed-instance/*",
            "Condition": {
                "StringEquals": {
                    "ssm:resourceTag/TAG_KEY": "TAG_VALUE"
                }
            }
        }
    ]
}
```

   a. 選擇下一步。

5. 在檢閱和建立頁面上。

   a. 輸入政策的政策名稱，例如 EKSHybridSSMPolicy

   b. 選擇建立政策。

6. 在左側導覽窗格中，選擇 Roles (角色)。

7. 在 Roles (角色) 頁面上，選擇 Create role (建立角色)。

8. 在 Select trusted entity (選取信任的實體) 頁面上，執行以下作業：

   a. 在信任的實體類型區段中，選擇自訂信任政策。將以下內容貼到自訂信任政策編輯器中。AWS_REGION 將 取代為 AWS SSM 混合啟用 AWS 的區域，並將 AWS_ACCOUNT_ID取代為 AWS 您的帳戶 ID。

```
{
   "Version":"2012-10-17",
```

```
   "Statement":[
      {
         "Sid":"",
         "Effect":"Allow",
         "Principal":{
            "Service":"ssm.amazonaws.com"
         },
         "Action":"sts:AssumeRole",
         "Condition":{
            "StringEquals":{
               "aws:SourceAccount":"AWS_ACCOUNT_ID"
            },
            "ArnEquals":{
               "aws:SourceArn":"arn:aws: ssm:AWS_REGION:AWS_ACCOUNT_ID:*"
            }
         }
      }
   ]
}
```

b. 選擇下一步。

9. 在新增許可頁面上，連接自訂政策或執行下列動作：

   a. 在篩選政策方塊中，輸入 EKSDescribeClusterPolicy或您在上面建立的政策名稱。選取搜尋結果中政策名稱左側的核取方塊。

   b. 在篩選政策方塊中，輸入 EKSHybridSSMPolicy或您在上面建立的政策名稱。選取搜尋結果中政策名稱左側的核取方塊。

   c. 在 Filter policies (篩選政策) 方塊中，輸入 AmazonEC2ContainerRegistryPullOnly。選取搜尋結果AmazonEC2ContainerRegistryPullOnly中左側的核取方塊。

   d. 在 Filter policies (篩選政策) 方塊中，輸入 AmazonSSMManagedInstanceCore。選取搜尋結果AmazonSSMManagedInstanceCore中左側的核取方塊。

   e. 選擇下一步。

10. 在 Name, review, and create (命名、檢閱和建立) 頁面上，執行以下作業：

    a. 針對 Role name (角色名稱)，為您的角色輸入唯一名稱 (例如 AmazonEKSHybridNodesRole)。

    b. 針對 Description (描述)，請以描述性文字 (例如 Amazon EKS - Hybrid Nodes role) 取代目前的文字。

    c. 選擇建立角色。

IAM Roles Anywhere AWS 的步驟

若要使用 AWS IAM Roles Anywhere，您必須在建立混合節點 AWS IAM 角色之前設定 IAM Roles Anywhere 信任錨點。如需說明，請參閱 the section called “隨處設定 AWS IAM 角色”。

1. 開啟 Amazon IAM 主控台
2. 在左側導覽窗格中，選擇 Roles (角色)。
3. 在 Roles (角色) 頁面上，選擇 Create role (建立角色)。
4. 在 Select trusted entity (選取信任的實體) 頁面上，執行以下作業：
   a. 在信任的實體類型區段中，選擇自訂信任政策。將以下內容貼到自訂信任政策編輯器中。TRUST_ANCHOR_ARN 以您在the section called “隨處設定 AWS IAM 角色”步驟中建立的信任錨點 ARN 取代 。此信任政策中的 條件會限制 AWS IAM Roles Anywhere 擔任混合節點 IAM 角色的能力，只有在角色工作階段名稱符合安裝在混合節點上 x509 憑證中的 CN 時，才能交換臨時 IAM 憑證。或者，您也可以使用其他憑證屬性來唯一識別您的節點。您在信任政策中使用的憑證屬性必須對應至您在 nodeadm 組態中設定的 nodeName。如需更多資訊，請參閱the section called “混合節點 nodeadm”。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "rolesanywhere.amazonaws.com"
            },
            "Action": [
                "sts:TagSession",
                "sts:SetSourceIdentity"
            ],
            "Condition": {
                "ArnEquals": {
                    "aws:SourceArn": "TRUST_ANCHOR_ARN"
                }
            }
        },
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "rolesanywhere.amazonaws.com"
            },
```

```
            "Action": "sts:AssumeRole",
            "Condition": {
                "StringEquals": {
                    "sts:RoleSessionName": "${aws:PrincipalTag/x509Subject/CN}"
                },
                "ArnEquals": {
                    "aws:SourceArn": "TRUST_ANCHOR_ARN"
                }
            }
        }
    ]
}
```

b. 選擇下一步。

5. 在新增許可頁面上，連接自訂政策或執行下列動作：

   a. 在篩選政策方塊中，輸入 EKSDescribeClusterPolicy或您在上面建立的政策名稱。選取搜尋結果中政策名稱左側的核取方塊。

   b. 在 Filter policies (篩選政策) 方塊中，輸入 AmazonEC2ContainerRegistryPullOnly。選取搜尋結果AmazonEC2ContainerRegistryPullOnly中左側的核取方塊。

   c. 選擇下一步。

6. 在 Name, review, and create (命名、檢閱和建立) 頁面上，執行以下作業：

   a. 針對 Role name (角色名稱)，為您的角色輸入唯一名稱 (例如 AmazonEKSHybridNodesRole)。

   b. 針對 Description (描述)，請以描述性文字 (例如 Amazon EKS - Hybrid Nodes role) 取代目前的文字。

   c. 選擇建立角色。

使用混合節點建立 Amazon EKS 叢集

本主題提供可用選項的概觀，並說明建立啟用混合節點的 Amazon EKS 叢集時應考量的事項。EKS 混合節點具有與具有雲端節點的 Amazon EKS 叢集相同的 eks/latest/userguide/kubernetes-versions.html【Kubernetes 版本支援，type="documentation"】，包括標準和延伸支援。

如果您不打算使用 EKS 混合節點，請參閱 中的主要 Amazon EKS 建立叢集文件the section called “建立叢集”。

先決條件

- 已完成 the section called “先決條件”。在建立啟用混合節點的叢集之前，您必須識別內部部署節點和選用的 Pod CIDRs、根據 EKS 需求建立的 VPC 和子網路，以及混合節點需求，以及具有內部部署和選用 Pod CIDRs 傳入規則的安全群組。如需這些先決條件的詳細資訊，請參閱 the section called “準備聯網”。
- 在您裝置上安裝和設定的最新版本 AWS 命令列界面 (AWS CLI)。若要檢查您目前的版本，請使用 `aws --version`。適用於 macOS 的 yum、apt-get 或 Homebrew 等套件管理員通常是最新版本 CLI AWS 後面的幾個版本。若要安裝最新版本，請參閱《 AWS 命令列界面使用者指南》中的安裝或更新至最新版本的 AWS CLI 和設定 AWS CLI 的設定。
- IAM 主體，具有建立 IAM 角色和連接政策，以及建立和描述 EKS 叢集的許可

考量事項

- 您的叢集必須使用 `API`或 `API_AND_CONFIG_MAP`進行叢集身分驗證模式。
- 您的叢集必須使用 IPv4 地址系列。
- 您的叢集必須使用公有或私有叢集端點連線。您的叢集無法使用「公有和私有」叢集端點連線，因為 Amazon EKS Kubernetes API 伺服器端點會解析為在 VPC 外部執行混合節點的公IPs。
- 具有混合節點的 EKS 叢集支援 OIDC 身分驗證。
- 您可以新增、變更或移除現有叢集的混合節點組態。如需詳細資訊，請參閱the section called “現有叢集”。

步驟 1：建立叢集 IAM 角色

如果您已有叢集 IAM 角色，或者您要使用 `eksctl` or AWS CloudFormation 建立叢集，則可以略過此步驟。根據預設，`eksctl` AWS CloudFormation 範本會為您建立叢集 IAM 角色。

1. 執行下列命令以建立 IAM 信任政策 JSON 檔案。

```
cat >eks-cluster-role-trust-policy.json <<EOF
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "eks.amazonaws.com"
```

```
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF
```

2. 建立 Amazon EKS 叢集 IAM 角色。如有必要，請在 eks-cluster-role-trust-policy.json 前面加上您在上一步中寫入檔案的電腦的路徑。命令會將您在上一步驟中建立的信任策略與角色相關聯。若要建立 IAM 角色，必須為建立角色的 IAM 主體指派以下 iam:CreateRole 動作 (許可)。

```
aws iam create-role \
    --role-name myAmazonEKSClusterRole \
    --assume-role-policy-document file://"eks-cluster-role-trust-policy.json"
```

3. 您可以指派 Amazon EKS 受管政策或建立自己的自訂政策。如需了解在自訂政策中必須使用的最低許可，請參閱 the section called “節點 IAM 角色”。將名為 AmazonEKSClusterPolicy 的 Amazon EKS 受管 IAM 政策連接到角色。若要將 IAM 政策連接至 IAM 主體，必須為連接政策的 IAM 實體指派以下 IAM 動作之一 (許可)：iam:AttachUserPolicy 或 iam:AttachRolePolicy。

```
aws iam attach-role-policy \
    --policy-arn arn:aws: iam::aws:policy/AmazonEKSClusterPolicy \
    --role-name myAmazonEKSClusterRole
```

步驟 2：建立啟用混合節點的叢集

您可以使用下列方式建立叢集：

- eksctl
- AWS CloudFormation
- AWS CLI
- AWS Management Console

建立啟用混合節點的叢集 - eksctl

您需要安裝最新版本的eksctl命令列工具。如需有關安裝或更新 eksctl 的指示，請參閱 eksctl 文件中的安裝一節。

1. 建立 cluster-config.yaml 以定義啟用混合節點的 Amazon EKS IPv4 叢集。在您的 中進行下列取代cluster-config.yaml。如需設定的完整清單，請參閱 eksctl 文件。

   a. 使用您的叢集名稱取代 CLUSTER_NAME。此名稱僅能使用英數字元 (區分大小寫) 和連字號。它必須以英數字元開頭，且長度不可超過 100 個字元。名稱在您要建立叢集 AWS 的區域和 AWS 帳戶中必須是唯一的。

   b. AWS_REGION 將 取代為您要在其中建立叢集 AWS 的區域。

   c. K8S_VERSION 將 取代為任何 eks/latest/userguide/kubernetes-versions.html【Amazon EKS 支援的版本，type="documentation"】。

   d. ira 根據您在 的步驟中設定的登入資料提供者，CREDS_PROVIDER將 取代為 ssm或 the section called “準備登入資料”。

   e. CA_BUNDLE_CERT 如果您的登入資料提供者設定為 ira，則取代 ，這會使用 AWS IAM Roles Anywhere 做為登入資料提供者。CA_BUNDLE_CERT 是憑證授權單位 (CA) 憑證內文，取決於您選擇的 CA。憑證必須是隱私權增強郵件 (PEM) 格式。

   f. GATEWAY_ID 將 取代為要連接到 VPC 的虛擬私有閘道或傳輸閘道的 ID。

   g. REMOTE_NODE_CIDRS 將 取代為混合節點的現場部署節點 CIDR。

   h. 針對在混合節點上執行的工作負載REMOTE_POD_CIDRS，將 取代為內部部署 Pod CIDR，或者如果您未在混合節點上執行 Webhook，請從組態中移除該行。REMOTE_POD_CIDRS 如果您的 CNI 不使用網路位址轉譯 (NAT)，或在 Pod 流量離開內部部署主機時偽裝 Pod IP 地址，您必須設定您的 。REMOTE_POD_CIDRS 如果您在混合節點上執行 Webhook，您必須設定 ，the section called “設定 Webhook”如需詳細資訊，請參閱 。

   i. 您的內部部署節點和 Pod CIDR 區塊必須符合下列要求：

      i. 位於其中一個 IPv4 RFC-1918 範圍內：172.16.0.0/12、 10.0.0.0/8或 192.168.0.0/16。

      ii. 不會彼此、叢集VPC CIDR的 或 Kubernetes 服務 IPv4 CIDR 重疊

```
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: CLUSTER_NAME
  region: AWS_REGION
  version: "K8S_VERSION"

remoteNetworkConfig:
  iam:
    provider: CREDS_PROVIDER # default SSM, can also be set to IRA
```

```
    # caBundleCert: CA_BUNDLE_CERT
  vpcGatewayID: GATEWAY_ID
  remoteNodeNetworks:
  - cidrs: ["REMOTE_NODE_CIDRS"]
  remotePodNetworks:
  - cidrs: ["REMOTE_POD_CIDRS"]
```

2. 執行以下命令：

```
eksctl create cluster -f cluster-config.yaml
```

叢集佈建需要幾分鐘的時間。建立叢集時，會出現幾行輸出。輸出的最後一行類似於下面的範例行。

```
[#]  EKS cluster "CLUSTER_NAME" in "REGION" region is ready
```

3. 繼續進行 the section called “步驟 3：更新 kubeconfig”。

建立啟用混合節點的叢集 - AWS CloudFormation

CloudFormation 堆疊會使用RemotePodNetwork您指定的 和 建立 EKS 叢集 IAM 角色RemoteNodeNetwork和 EKS 叢集。修改 CloudFormation 範本 如果您需要自訂 CloudFormation 範本中未公開的 EKS 叢集設定。

1. 下載 CloudFormation 範本。

```
curl -OL 'https://raw.githubusercontent.com/aws/eks-hybrid/refs/heads/main/example/
hybrid-eks-cfn.yaml'
```

2. 建立 `cfn-eks-parameters.json`，並為每個值指定您的組態。
   a. `CLUSTER_NAME`：要建立的 EKS 叢集名稱
   b. `CLUSTER_ROLE_NAME`：要建立之 EKS 叢集 IAM 角色的名稱。範本中的預設值為「EKSClusterRole」。
   c. `SUBNET1_ID`：您在先決條件步驟中建立的第一個子網路 ID
   d. `SUBNET2_ID`：您在先決條件步驟中建立的第二個子網路 ID
   e. `SG_ID`：您在先決條件步驟中建立的安全群組 ID
   f. `REMOTE_NODE_CIDRS`：混合節點的內部部署節點 CIDR

g. REMOTE_POD_CIDRS：在混合節點上執行之工作負載的內部部署 Pod CIDR。REMOTE_POD_CIDRS 如果您的 CNI 不使用網路位址轉譯 (NAT)，或在 Pod 流量離開內部部署主機時偽裝 Pod IP 地址，您必須設定您的 。REMOTE_POD_CIDRS 如果您在混合節點上執行 Webhook，您必須設定 ，the section called “設定 Webhook”如需詳細資訊，請參閱 。

h. 您的內部部署節點和 Pod CIDR 區塊必須符合下列要求：

   i. 位於其中一個 IPv4 RFC-1918 範圍內：172.16.0.0/12、 10.0.0.0/8或 192.168.0.0/16。

   ii. 不會彼此、叢集VPC CIDR的 或 Kubernetes 服務 IPv4 CIDR 重疊。

i. CLUSTER_AUTH：叢集的叢集身分驗證模式。有效值為 API 和 API_AND_CONFIG_MAP。範本中的預設值為 API_AND_CONFIG_MAP。

j. CLUSTER_ENDPOINT：叢集的叢集端點連線。有效值為「公有」和「私有」。範本中的預設值為私有，這表示您只能從 VPC 內連線至 Kubernetes API 端點。

k. K8S_VERSION：要用於叢集的 Kubernetes 版本。請參閱 eks/latest/userguide/kubernetes-versions.html【Amazon EKS 支援的版本，type="documentation"】。

```
{
  "Parameters": {
    "ClusterName": "CLUSTER_NAME",
    "ClusterRoleName": "CLUSTER_ROLE_NAME",
    "SubnetId1": "SUBNET1_ID",
    "SubnetId2": "SUBNET2_ID",
    "SecurityGroupId" "SG_ID",
    "RemoteNodeCIDR": "REMOTE_NODE_CIDRS",
    "RemotePodCIDR": "REMOTE_POD_CIDRS",
    "ClusterAuthMode": "CLUSTER_AUTH",
    "ClusterEndpointConnectivity": "CLUSTER_ENDPOINT",
    "K8sVersion": "K8S_VERSION"
  }
 }
```

3. 部署 CloudFormation 堆疊。STACK_NAME 將 取代為 CloudFormation 堆疊的名稱，並將 AWS_REGION取代為建立叢集的 AWS 區域。

```
aws cloudformation deploy \
    --stack-name STACK_NAME \
    --region AWS_REGION \
    --template-file hybrid-eks-cfn.yaml \
    --parameter-overrides file://cfn-eks-parameters.json \
```

```
    --capabilities CAPABILITY_NAMED_IAM
```

叢集佈建需要幾分鐘的時間。您可以使用下列命令來檢查堆疊的狀態。STACK_NAME 將 取代為 CloudFormation 堆疊的名稱，並將 AWS_REGION取代為建立叢集的 AWS 區域。

```
aws cloudformation describe-stacks \
    --stack-name STACK_NAME \
    --region AWS_REGION \
    --query 'Stacks[].StackStatus'
```

4. 繼續進行 the section called “步驟 3：更新 kubeconfig”。

建立啟用混合節點的叢集 - AWS CLI

1. 執行下列命令來建立啟用混合節點的 EKS 叢集。執行 命令之前，請將以下內容取代為您的設定。如需設定的完整清單，請參閱 the section called “建立叢集” 文件。
   a. CLUSTER_NAME：要建立的 EKS 叢集名稱
   b. AWS_REGION：建立叢集 AWS 的區域。
   c. K8S_VERSION：要用於叢集的 Kubernetes 版本。請參閱 Amazon EKS 支援的版本。
   d. ROLE_ARN：您為叢集設定的 Amazon EKS 叢集角色。如需詳細資訊，請參閱 Amazon EKS 叢集 IAM 角色。
   e. SUBNET1_ID：您在先決條件步驟中建立的第一個子網路 ID
   f. SUBNET2_ID：您在先決條件步驟中建立的第二個子網路 ID
   g. SG_ID：您在先決條件步驟中建立的安全群組 ID
   h. 您可以使用 API和 API_AND_CONFIG_MAP進行叢集存取身分驗證模式。在下列命令中，叢集存取身分驗證模式設定為 API_AND_CONFIG_MAP。
   i. 您可以使用 endpointPublicAccess和 endpointPrivateAccess 參數來啟用或停用叢集 Kubernetes API 伺服器端點的公有和私有存取。在下列命令中endpointPublicAccess， 設定為 falseendpointPrivateAccess， 設定為 true。
   j. REMOTE_NODE_CIDRS：混合節點的內部部署節點 CIDR。
   k. REMOTE_POD_CIDRS （選用）：在混合節點上執行工作負載的內部部署 Pod CIDR。
   l. 您的內部部署節點和 Pod CIDR 區塊必須符合下列要求：
      i. 位於其中一個 IPv4 RFC-1918 範圍內：172.16.0.0/12、 10.0.0.0/8或 192.168.0.0/16。
      ii. 不會彼此、Amazon EKS 叢集VPC CIDR的 或 Kubernetes 服務 IPv4 CIDR 重疊。

```
aws eks create-cluster \
    --name CLUSTER_NAME \
    --region AWS_REGION \
    --kubernetes-version K8S_VERSION \
    --role-arn ROLE_ARN \
    --resources-vpc-config
 subnetIds=SUBNET1_ID,SUBNET2_ID,securityGroupIds=SG_ID,endpointPrivateAccess=true,endpo
 \
    --access-config authenticationMode=API_AND_CONFIG_MAP \
    --remote-network-config '{"remoteNodeNetworks":[{"cidrs":
["REMOTE_NODE_CIDRS"]}],"remotePodNetworks":[{"cidrs":["REMOTE_POD_CIDRS"]}]}'
```

2. 佈建叢集需要幾分鐘才能完成。您可以使用下列命令來查詢叢集的狀態。CLUSTER_NAME 將 取代為您建立的叢集名稱，並將 AWS_REGION 取代為您建立叢集的 AWS 區域。在傳回的輸出為 之前，請勿繼續進行下一個步驟ACTIVE。

```
aws eks describe-cluster \
    --name CLUSTER_NAME \
    --region AWS_REGION \
    --query "cluster.status"
```

3. 繼續進行 the section called “步驟 3：更新 kubeconfig”。

建立啟用混合節點的叢集 - AWS Management Console

1. 在 Amazon EKS 主控台開啟 Amazon EKS 主控台。
2. 選取 Add cluster (新增叢集)，然後選取 Create (建立)。
3. 在 Configure cluster (設定叢集) 頁面上，輸入下列欄位：
   a. Name (名稱)：叢集的名稱。名稱只能包含英數字元 （ 區分大小寫 ）、連字號和底線。它必須以英數字元開頭，且長度不可超過 100 個字元。名稱在您要建立叢集 AWS 的區域和 AWS 帳戶中必須是唯一的。
   b. 叢集 IAM 角色 – 選擇您建立的 Amazon EKS 叢集 IAM 角色，以允許 Kubernetes 控制平面代表您管理 AWS 資源。
   c. Kubernetes version (Kubernetes 版本) – 您的叢集使用的 Kubernetes 版本。我們建議選取最新版本，除非您需要較早版本。
   d. 升級政策 - 選擇擴充或標準。

i. 延伸：此選項在發行日期後 26 個月內支援 Kubernetes 版本。延長支援期間會產生額外的每小時成本，從標準支援期間結束後開始。延伸支援結束時，您的叢集將自動升級至下一個版本。

ii. 標準：此選項在發行日期後支援 Kubernetes 版本 14 個月。無需額外費用。當標準支援結束時，您的叢集將自動升級至下一個版本。

e. 叢集存取 - 選擇允許或不允許叢集管理員存取，然後選擇身分驗證模式。啟用混合節點的叢集支援下列身分驗證模式。

i. EKS API：叢集只會從 EKS 存取項目 APIsIAM 主體。

ii. EKS API 和 ConfigMap：叢集將從 EKS 存取項目 APIs和 ConfigMap 中取得已驗證的 `aws-auth` IAM 主體。

f. Secrets encryption (秘密加密)：(選用) 選擇使用 KMS 金鑰啟用 Kubernetes 秘密的秘密加密。您也可以在建立叢集後啟用此功能。啟用此功能之前，請確定您已熟悉 中的資訊the section called “啟用秘密加密”。

g. ARC 區域轉移 - 如果啟用，EKS 會向 ARC 區域轉移註冊您的叢集，讓您使用區域轉移將應用程式流量從可用區域轉移。

h. Tags (標籤) – (選用) 將任何標籤新增到您的叢集。如需詳細資訊，請參閱the section called “標記您的 資源”。

i. 完成此頁面後，請選擇下一步。

4. 在 Specify networking (指定網路) 頁面上，選取下列欄位的值：

a. VPC – 選擇符合 the section called “VPC 和子網要求”和 Amazon EKS 混合節點需求的現有 VPC。選擇 VPC 之前，建議您先熟悉檢視 VPC、子網路和混合節點的 Amazon EKS 網路需求中的所有需求和考量事項。您無法在叢集建立後變更要使用的 VPC。如果沒有列出 VPC，則您需要先建立一個。如需詳細資訊，請參閱 the section called “建立 VPC”和 Amazon EKS 混合節點聯網需求。

b. Subnets (子網路)：根據預設，前一個欄位指定的 VPC 中的所有可用子網路會預先選取。您必須選取至少兩個。

c. 安全群組:(選用) 指定一或多個您希望 Amazon EKS 與其建立的網路介面相關聯的安全群組。您指定的至少一個安全群組必須具有內部部署節點的傳入規則，以及選用的 Pod CIDRs。如需詳細資訊，請參閱 Amazon EKS 混合節點聯網需求。無論您是否選擇任何安全群組，Amazon EKS 都會建立一個安全群組，以支援您的叢集和 VPC 之間的通訊。Amazon EKS 將此安全群組及您選擇的任何群組與其建立的網路介面相關聯。如需 Amazon EKS 建立的叢集安全群組的詳細資訊，請參閱the section called “安全群組要求”您可以在 Amazon EKS 建立的叢集安全群組中修改規則。

d. 選擇叢集 IP 地址系列 – 您必須為啟用混合節點的叢集選擇 IPv4。

e. （選用）選擇設定 Kubernetes 服務 IP 地址範圍，並指定服務 IPv4 範圍。

f. 選擇設定遠端網路以啟用混合節點，並為混合節點指定您的內部部署節點和 Pod CIDRs。

g. 如果您的 CNI 在 Pod 流量離開現場部署主機時不使用網路位址轉譯 (NAT) 或偽裝 Pod IP 地址，則必須設定遠端 Pod CIDR。如果您在混合節點上執行 Webhook，則必須設定遠端 Pod CIDR。

h. 您的內部部署節點和 Pod CIDR 區塊必須符合下列要求：

   i. 位於其中一個 IPv4 RFC-1918 範圍內：`172.16.0.0/12`、 `10.0.0.0/8`或 `192.168.0.0/16`。

   ii. 不會彼此、叢集`VPC CIDR`的 或 Kubernetes 服務 IPv4 CIDR 重疊

i. 針對 Cluster endpoint access (叢集端點存取)，選取一個選項。建立叢集後，您可以變更此選項。對於啟用混合節點的叢集，您必須選擇公有或私有。在選取非預設選項之前，請務必熟悉這些選項及其含義。如需詳細資訊，請參閱the section called “叢集端點存取”。

j. 完成此頁面後，請選擇下一步。

5. （選用）在設定可觀測性頁面上，選擇要開啟的指標和控制平面記錄選項。根據預設，系統會關閉每個日誌類型。

   a. 如需 Prometheus 指標選項的詳細資訊，請參閱 the section called “Prometheus 指標”。

   b. 如需 EKS 控制項記錄選項的詳細資訊，請參閱 the section called “控制平面日誌”。

   c. 完成此頁面後，請選擇下一步。

6. 在 Select add-ons (選取附加元件) 頁面上，選擇您要新增至叢集的附加元件。

   a. 您可以視需要選擇任意數量的 Amazon EKS 附加元件和 AWS Marketplace 附加元件。與混合節點不相容的 Amazon EKS 附加元件會標記為「與混合節點不相容」，而且附加元件具有防止在混合節點上執行的反親和性規則。如需詳細資訊，請參閱設定混合節點的附加元件。如果您想要安裝的 AWS Marketplace 附加元件未列出，您可以在搜尋方塊中輸入文字來搜尋可用的 AWS Marketplace 附加元件。您也可以依 category (類別)、vendor (廠商) 或 pricing model (定價模式) 進行搜尋，然後從搜尋結果中選擇附加元件。

   b. 預設會安裝一些附加元件，例如 CoreDNS 和 kube-proxy。如果您停用任何預設附加元件，這可能會影響您執行 Kubernetes 應用程式的能力。

   c. 當您完成此頁面時，請選擇 `Next`。

7. 在設定選取的附加元件設定頁面上，選取您要安裝的版本。

   a. 建立叢集後，您隨時皆可更新至更新版本。您可以在建立叢集後更新每個附加元件的組態。如需有關設定附加元件的詳細資訊，請參閱the section called “更新 附加元件”。如需與混合節點相容的附加元件版本，請參閱 the section called “設定附加元件”。

   b. 完成此頁面後，請選擇下一步。

8. 在 Review and create (檢閱並建立) 頁面上，檢閱您在先前頁面上輸入或選取的資訊。如需變更，請選擇 Edit (編輯)。當您滿意時，請選擇建立。在佈建叢集時，Status (狀態) 欄位顯示 CREATING (正在建立)。叢集佈建需要幾分鐘的時間。
9. 繼續進行 the section called “步驟 3：更新 kubeconfig”。

步驟 3：更新 kubeconfig

如果您使用 eksctl 建立叢集，則可以略過此步驟。這是因為 eksctl 已為您完成此步驟。將新內容新增至組態檔案kubectl，讓 kubectl 與您的叢集通訊。如需建立和更新檔案的詳細資訊，請參閱 the section called “使用 kubectl 存取叢集”。

```
aws eks update-kubeconfig --name CLUSTER_NAME --region AWS_REGION
```

範例輸出如下。

```
Added new context arn:aws: eks:AWS_REGION:111122223333:cluster/CLUSTER_NAME to /home/
username/.kube/config
```

透過執行以下命令確認與叢集的通訊。

```
kubectl get svc
```

範例輸出如下。

```
NAME         TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
kubernetes   ClusterIP   10.100.0.1   <none>        443/TCP   28h
```

步驟 4：叢集設定

下一步，請參閱 the section called “準備叢集存取” 以啟用混合節點的存取，以加入叢集。

在現有的 Amazon EKS 叢集上啟用混合節點或修改組態

本主題提供可用選項的概觀，並說明新增、變更或移除 Amazon EKS 叢集的混合節點組態時應考量的事項。

若要讓 Amazon EKS 叢集使用混合節點，請在RemoteNetworkConfig組態中新增現場部署節點的 IP 地址 CIDR 範圍和選用的 Pod 網路。EKS 使用此 CIDRs 清單來啟用叢集與內部部署網

路之間的連線。如需更新叢集組態時的完整選項清單，請參閱《Amazon EKS API 參考》中的 UpdateClusterConfig。

您可以對叢集中的 EKS 混合節點聯網組態執行下列任何動作：

- 新增遠端網路組態，以在現有叢集中啟用 EKS 混合節點。
- 新增、變更或移除現有叢集中的遠端節點網路或遠端 Pod 網路。
- 移除所有遠端節點網路 CIDR 範圍，以停用現有叢集中的 EKS 混合節點。

先決條件

- 為混合節點啟用 Amazon EKS 叢集之前，請確定您的環境符合 中所述的要求the section called “先決條件”，並在 the section called “準備聯網”、 the section called “準備作業系統”和 中詳細說明the section called “準備登入資料”。
- 您的叢集必須使用 IPv4 地址系列。
- 您的叢集必須使用 API或 API_AND_CONFIG_MAP進行叢集身分驗證模式。修改叢集身分驗證模式的程序如 所述the section called “身分驗證方式”。
- 建議您對 Amazon EKS Kubernetes API 伺服器端點使用公有或私有端點存取，但不能同時使用兩者。如果您選擇「公有和私有」，Amazon EKS Kubernetes API 伺服器端點一律會解析為在 VPC 外部執行之混合節點的公IPs，以防止混合節點加入叢集。修改叢集網路存取的程序如 所述the section called “叢集端點存取”。
- 在您裝置上安裝和設定的最新版本 AWS 命令列界面 (AWS CLI)。若要檢查您目前的版本，請使用 aws --version。適用於 macOS 的 yum、apt-get 或 Homebrew 等套件管理員通常是最新版本 CLI AWS 後面的幾個版本。若要安裝最新版本，請參閱《 AWS 命令列界面使用者指南》中的安裝或更新至最新版本的 AWS CLI 和設定 AWS CLI 的設定。
- IAM 主體，具有在您的 Amazon EKS 叢集上呼叫 UpdateClusterConfig 的許可。
- 將附加元件更新為與混合節點相容的版本。如需與混合節點相容的附加元件版本，請參閱 the section called “設定附加元件”。
- 如果您執行的附加元件與混合節點不相容，請確定附加元件 DaemonSet 或 部署具有下列親和性規則，以防止部署到混合節點。如果尚未存在，請新增下列親和性規則。

```
affinity:
  nodeAffinity:
    requiredDuringSchedulingIgnoredDuringExecution:
      nodeSelectorTerms:
      - matchExpressions:
```

```
        - key: eks.amazonaws.com/compute-type
          operator: NotIn
          values:
          - hybrid
```

考量事項

`remoteNetworkConfig` JSON 物件在更新期間有下列行為：

- 您未指定的任何現有組態部分都不會變更。如果您未指定 `remoteNodeNetworks`或 `remotePodNetworks`，則該部分將保持不變。
- 如果您要修改 CIDRs 的 `remoteNodeNetworks`或 `remotePodNetworks`清單，則必須在最終組態中指定您想要的 CIDRs 完整清單。當您指定 `remoteNodeNetworks`或 `remotePodNetworks` CIDR 清單的變更時，EKS 會在更新期間取代原始清單。
- 您的內部部署節點和 Pod CIDR 區塊必須符合下列要求：
  1. 在其中一個 IPv4 RFC-1918 範圍內：10.0.0.0/8、172.16.0.0/12 或 192.168.0.0/16。
  2. 不會彼此重疊、Amazon EKS 叢集 VPC 的所有 CIDRs 或 Kubernetes 服務 IPv4 CIDR。

在現有叢集上啟用混合節點

您可以使用下列方式在現有叢集中啟用 EKS 混合節點：

- AWS CloudFormation
- AWS CLI
- AWS Management Console

在現有叢集中啟用 EKS 混合節點 - AWS CloudFormation

1. 若要在叢集中啟用 EKS 混合節點，請將 `RemoteNodeNetwork`和 （選用） `RemotePodNetwork`新增至 CloudFormation 範本並更新堆疊。請注意， `RemoteNodeNetwork` 是最多包含一個`Cidrs`項目的清單，而 `Cidrs`是多個 IP CIDR 範圍的清單。

```
RemoteNetworkConfig:
  RemoteNodeNetworks:
    - Cidrs: [RemoteNodeCIDR]
  RemotePodNetworks:
    - Cidrs: [RemotePodCIDR]
```

2. 繼續進行the section called “準備叢集存取”。

在現有叢集中啟用 EKS 混合節點 - AWS CLI

1. 執行下列命令，為您的 EKS 叢集RemoteNetworkConfig啟用 EKS 混合節點的 。執行 命令之前，請將以下內容取代為您的設定。如需設定的完整清單，請參閱《Amazon EKS API 參考》中的UpdateClusterConfig。
   a. CLUSTER_NAME：要更新的 EKS 叢集名稱。
   b. AWS_REGION：EKS 叢集執行所在的 AWS 區域。
   c. REMOTE_NODE_CIDRS：混合節點的內部部署節點 CIDR。
   d. REMOTE_POD_CIDRS （選用）：在混合節點上執行工作負載的內部部署 Pod CIDR。

```
aws eks update-cluster-config \
    --name CLUSTER_NAME \
    --region AWS_REGION \
    --remote-network-config '{"remoteNodeNetworks":[{"cidrs":
["REMOTE_NODE_CIDRS"]}],"remotePodNetworks":[{"cidrs":["REMOTE_POD_CIDRS"]}]}'
```

2. 更新叢集需要幾分鐘的時間。您可以使用下列命令來查詢叢集的狀態。CLUSTER_NAME 將 取代為您修改的叢集名稱，並將 AWS_REGION取代為執行叢集的 AWS 區域。在傳回的輸出為 之前，請勿繼續進行下一個步驟ACTIVE。

```
aws eks describe-cluster \
    --name CLUSTER_NAME \
    --region AWS_REGION \
    --query "cluster.status"
```

3. 繼續進行the section called “準備叢集存取”。

在現有叢集中啟用 EKS 混合節點 - AWS Management Console

1. 在 Amazon EKS 主控台開啟 Amazon EKS 主控台。
2. 選擇叢集名稱以顯示您叢集的資訊。
3. 選擇聯網索引標籤，然後選擇管理。
4. 在下拉式清單中，選擇遠端網路。
5. 選擇設定遠端網路以啟用混合節點，並為混合節點指定您的內部部署節點和 Pod CIDRs。
6. 選擇 Save changes (儲存變更) 以完成操作。等待叢集狀態回到作用中。

7. 繼續進行the section called “準備叢集存取”。

更新現有叢集中的混合節點組態

您可以使用下列任一方式在現有混合叢集`remoteNetworkConfig`中修改 ：

- AWS CloudFormation
- AWS CLI
- AWS Management Console

更新現有叢集 - AWS CloudFormation 中的混合組態

1. 使用新的網路 CIDR 值更新您的 CloudFormation 範本。

```
RemoteNetworkConfig:
  RemoteNodeNetworks:
    - Cidrs: [NEW_REMOTE_NODE_CIDRS]
  RemotePodNetworks:
    - Cidrs: [NEW_REMOTE_POD_CIDRS]
```

> **Note**
>
> 更新 RemoteNodeNetworks 或 RemotePodNetworks CIDR 清單時，請包含所有 CIDRs（新和現有）。EKS 會在更新期間取代整個清單。從更新請求省略這些欄位會保留其現有的組態。

2. 使用修改過的範本更新您的 CloudFormation 堆疊，並等待堆疊更新完成。

更新現有叢集 - AWS CLI 中的混合組態

1. 若要修改遠端網路 CIDRs，請執行下列命令。將值取代為您的設定：

```
aws eks update-cluster-config
--name CLUSTER_NAME
--region AWS_REGION
--remote-network-config '{"remoteNodeNetworks":[{"cidrs":
["NEW_REMOTE_NODE_CIDRS"]}],"remotePodNetworks":[{"cidrs":
["NEW_REMOTE_POD_CIDRS"]}]}'
```

> **Note**
>
> 更新 `remoteNodeNetworks` 或 `remotePodNetworks` CIDR 清單時，請包含所有 CIDRs（新和現有）。EKS 會在更新期間取代整個清單。從更新請求省略這些欄位會保留其現有的組態。

2. 等待叢集狀態返回 ACTIVE，然後再繼續。

更新現有叢集中的混合組態 - AWS Management Console

1. 在 Amazon EKS 主控台開啟 Amazon EKS 主控台。
2. 選擇叢集名稱以顯示您叢集的資訊。
3. 選擇聯網索引標籤，然後選擇管理。
4. 在下拉式清單中，選擇遠端網路。
5. `Remote pod networks - Optional` 視需要更新 `Remote node networks`和 CIDRs。
6. 選擇儲存變更並等待叢集狀態回到作用中。

在現有叢集中停用混合節點

您可以使用下列方式停用現有叢集中的 EKS 混合節點：

- AWS CloudFormation
- AWS CLI
- AWS Management Console

在現有叢集中停用 EKS 混合節點 - AWS CloudFormation

1. 若要停用叢集中的 EKS 混合節點，請在 CloudFormation 範本中`RemotePodNetworks`將 `RemoteNodeNetworks`和 設定為空陣列，並更新堆疊。

```
RemoteNetworkConfig:
  RemoteNodeNetworks: []
  RemotePodNetworks: []
```

在現有叢集中停用 EKS 混合節點 - AWS CLI

1. 執行下列命令，RemoteNetworkConfig從 EKS 叢集中移除 。執行 命令之前，請將以下內容取代為您的設定。如需設定的完整清單，請參閱《Amazon EKS API 參考》中的 UpdateClusterConfig。

   a. CLUSTER_NAME：要更新的 EKS 叢集名稱。

   b. AWS_REGION：EKS 叢集執行所在的 AWS 區域。

   ```
   aws eks update-cluster-config \
       --name CLUSTER_NAME \
       --region AWS_REGION \
       --remote-network-config '{"remoteNodeNetworks":[],"remotePodNetworks":[]}'
   ```

2. 更新叢集需要幾分鐘的時間。您可以使用下列命令來查詢叢集的狀態。CLUSTER_NAME 將 取代為您修改的叢集名稱，並將 AWS_REGION取代為執行叢集的 AWS 區域。在傳回的輸出為 之前，請勿繼續進行下一個步驟ACTIVE。

   ```
   aws eks describe-cluster \
       --name CLUSTER_NAME \
       --region AWS_REGION \
       --query "cluster.status"
   ```

在現有叢集中停用 EKS 混合節點 - AWS Management Console

1. 在 Amazon EKS 主控台開啟 Amazon EKS 主控台。
2. 選擇叢集名稱以顯示您叢集的資訊。
3. 選擇聯網索引標籤，然後選擇管理。
4. 在下拉式清單中，選擇遠端網路。
5. 選擇設定遠端網路以啟用混合節點，並移除 Remote node networks和 下的所有 CIDRsRemote pod networks - Optional。
6. 選擇 Save changes (儲存變更) 以完成操作。等待叢集狀態回到作用中。

準備混合節點的叢集存取

將混合節點連線至 Amazon EKS 叢集之前，您必須使用 Kubernetes 許可啟用混合節點 IAM 角色，才能加入叢集。如需如何建立混合節點 IAM 角色的資訊，the section called “準備登入資料”請參閱 。Amazon EKS 支援兩種將 IAM 主體與 Kubernetes 角色型存取控制 (RBAC)、Amazon EKS 存取

項目和 aws-auth ConfigMap 建立關聯的方式。如需 Amazon EKS 存取管理的詳細資訊，請參閱the section called “Kubernetes API 存取”。

使用以下程序，將您的混合節點 IAM 角色與 Kubernetes 許可建立關聯。若要使用 Amazon EKS 存取項目，您的叢集必須使用 API或 API_AND_CONFIG_MAP 身分驗證模式建立。若要使用 aws-auth ConfigMap，您的叢集必須使用API_AND_CONFIG_MAP身分驗證模式建立。啟用混合節點的 Amazon EKS 叢集不支援CONFIG_MAP僅限 身分驗證模式。

使用混合節點 IAM 角色的 Amazon EKS 存取項目

名為 HYBRID_LINUX 的混合節點有 Amazon EKS 存取項目類型，可與 IAM 角色搭配使用。使用此存取項目類型時，使用者名稱會自動設定為 system：node：{{SessionName}}。如需建立存取項目的詳細資訊，請參閱 the section called “建立存取項目”。

AWS CLI

1. 您必須在裝置上安裝並設定最新版本的 AWS CLI。若要檢查您目前的版本，請使用 aws --version。適用於 macOS 的 yum、apt-get 或 Homebrew 等套件管理員通常是最新版本 CLI AWS 後面的幾個版本。若要安裝最新版本，請參閱《 AWS 命令列界面使用者指南》中的使用 aws 設定安裝 和 Quick configuration。
2. 使用以下命令建立您的存取項目。將 CLUSTER_NAME 取代為您的叢集名稱，並將 HYBRID_NODES_ROLE_ARN 取代為您在 的步驟中建立的角色 ARNthe section called “準備登入資料”。

```
aws eks create-access-entry --cluster-name CLUSTER_NAME \
    --principal-arn HYBRID_NODES_ROLE_ARN \
    --type HYBRID_LINUX
```

AWS Management Console

1. 在 Amazon EKS 主控台開啟 Amazon EKS 主控台。
2. 選擇啟用混合節點的叢集名稱。
3. 選擇存取索引標籤。
4. 選擇建立存取項目。
5. 針對 IAM 主體，選取您在 的步驟中建立的混合節點 IAM 角色the section called “準備登入資料”。
6. 針對類型，選取混合 Linux。

7. (選用) 您可以使用標籤為存取項目指派標籤。例如，為了更輕鬆地找到具有相同標籤的所有資源而指定標籤。
8. 選擇略過以檢閱和建立。您無法將政策新增至混合 Linux 存取項目，或變更其存取範圍。
9. 檢查存取項目的組態。如果有任何內容看起來不正確，請選擇上一步以返回上一步並修正錯誤。如果組態正確，請選擇建立。

針對混合節點 IAM 角色使用 aws-auth ConfigMap

在下列步驟中，您將使用您在 aws-auth 的步驟中建立的混合節點 IAM 角色的 ARN 來建立或更新 ConfigMapthe section called “準備登入資料”。

1. 檢查叢集是否有現有的 aws-auth ConfigMap。請注意，如果您使用的是特定kubeconfig檔案，請使用 --kubeconfig旗標。

```
kubectl describe configmap -n kube-system aws-auth
```

2. 如果顯示 aws-auth ConfigMap，請視需要更新。
   a. 開啟 ConfigMap 進行編輯。

```
kubectl edit -n kube-system configmap/aws-auth
```

   b. 視需要新增 mapRoles 個項目。HYBRID_NODES_ROLE_ARN 將 取代為混合節點 IAM 角色的 ARN。請注意，{{SessionName}}是在 ConfigMap 中儲存的正確範本格式。請勿將其取代為其他值。

```
data:
  mapRoles: |
  - groups:
    - system:bootstrappers
    - system:nodes
    rolearn: HYBRID_NODES_ROLE_ARN
    username: system:node:{{SessionName}}
```

   c. 儲存檔案並結束您的文字編輯器。
3. 如果叢集沒有現有的 aws-auth ConfigMap，請使用下列命令建立它。HYBRID_NODES_ROLE_ARN 將 取代為混合節點 IAM 角色的 ARN。請注意，{{SessionName}}是在 ConfigMap 中儲存的正確範本格式。請勿將其取代為其他值。

```
kubectl apply -f=/dev/stdin <<-EOF
```

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: aws-auth
  namespace: kube-system
data:
  mapRoles: |
  - groups:
    - system:bootstrappers
    - system:nodes
    rolearn: HYBRID_NODES_ROLE_ARN
    username: system:node:{{SessionName}}
EOF
```

在混合節點上執行內部部署工作負載

在啟用混合節點的 EKS 叢集中，您可以使用與 AWS 雲端中使用的相同 Amazon EKS 叢集、功能和工具，在自己的基礎設施上執行內部部署和邊緣應用程式。

下列各節包含使用混合節點的step-by-step說明。

主題

- 連接混合節點
- 使用 Bottlerocket 連接混合節點
- 升級叢集的混合節點
- 混合節點的修補程式安全性更新
- 移除混合節點

連接混合節點

> **Note**
>
> 下列步驟適用於執行 Bottlerocket 以外相容作業系統的混合節點。如需連接執行 Bottlerocket 之混合節點的步驟，請參閱 the section called “使用 Bottlerocket 連接混合節點”。

本主題說明如何將混合節點連線至 Amazon EKS 叢集。混合節點加入叢集後，它們會在 Amazon EKS 主控台和 Kubernetes 相容工具中顯示為未就緒狀態，例如 kubectl。完成此頁面的步驟後，請繼續 the section called “設定 CNI” ，讓您的混合節點準備好執行應用程式。

先決條件

將混合節點連線至 Amazon EKS 叢集之前，請確定您已完成先決條件步驟。

- 您可以從內部部署環境連線至託管 Amazon EKS 叢集 AWS 的區域。如需詳細資訊，請參閱the section called “準備聯網”。
- 您在內部部署主機上安裝了適用於混合節點的相容作業系統。如需詳細資訊，請參閱the section called “準備作業系統”。
- 您已建立混合節點 IAM 角色，並設定內部部署憑證提供者 (AWS Systems Manager 混合啟用或 AWS IAM Roles Anywhere)。如需詳細資訊，請參閱the section called “準備登入資料”。
- 您已建立啟用混合節點的 Amazon EKS 叢集。如需詳細資訊，請參閱the section called “建立叢集”。
- 您已將混合節點 IAM 角色與 Kubernetes 角色型存取控制 (RBAC) 許可建立關聯。如需詳細資訊，請參閱the section called “準備叢集存取”。

步驟 1：在每個內部部署主機上安裝混合節點 CLI (**nodeadm**)

如果您在預先建置的作業系統映像中包含 Amazon EKS 混合節點 CLI (nodeadm)，則可以略過此步驟。如需 混合節點版本的詳細資訊nodeadm，請參閱 the section called “混合節點 nodeadm”。

的混合節點版本nodeadm託管在 Amazon CloudFront 前端的 Amazon S3 中。 Amazon CloudFront 若要在每個現場部署主機nodeadm上安裝 ，您可以從現場部署主機執行下列命令。

對於 x86_64 主機：

```
curl -OL 'https://hybrid-assets.eks.amazonaws.com/releases/latest/bin/linux/amd64/
nodeadm'
```

對於 ARM 主機

```
curl -OL 'https://hybrid-assets.eks.amazonaws.com/releases/latest/bin/linux/arm64/
nodeadm'
```

將可執行檔許可新增至每個主機上下載的二進位檔。

```
chmod +x nodeadm
```

步驟 2：使用 安裝混合節點相依性 **nodeadm**

如果您要在預先建置的作業系統映像中安裝混合節點相依性，您可以略過此步驟。nodeadm install 命令可用來安裝混合節點所需的所有相依性。混合節點相依性包括 containerd、kubelet、kubectl 和 AWS SSM 或 AWS IAM Roles Anywhere 元件。the section called “混合節點 nodeadm” 如需 所安裝元件和檔案位置的詳細資訊，請參閱 nodeadm install。the section called “準備聯網” 如需必須在現場部署防火牆中允許之網域的詳細資訊，請參閱 以取得混合節點nodeadm install。

執行以下命令，在您的內部部署主機上安裝混合節點相依性。以下命令必須與在主機上具有 sudo/root 存取權的使用者一起執行。

> **Important**
>
> 混合節點 CLI (nodeadm) 必須與在主機上具有 sudo/root 存取權的使用者一起執行。

- K8S_VERSION 將 取代為 Amazon EKS 叢集的 Kubernetes 次要版本，例如 1.31。如需支援的 Kubernetes 版本清單，請參閱 eks/latest/userguide/kubernetes-versions.html【Amazon EKS 支援的版本，type="documentation"】。
- CREDS_PROVIDER 將 取代為您正在使用的現場部署憑證提供者。有效值ssm適用於 AWS SSM，iam-ra適用於 AWS IAM Roles Anywhere。

```
nodeadm install K8S_VERSION --credential-provider CREDS_PROVIDER
```

步驟 3：將混合節點連接至您的叢集

在將混合節點連線至叢集之前，請確定您已允許在內部部署防火牆和叢集的安全群組中，Amazon EKS 控制平面往返混合節點通訊所需的存取權。在此步驟中，大多數問題都與防火牆組態、安全群組組態或混合節點 IAM 角色組態相關。

> **Important**
>
> 混合節點 CLI (nodeadm) 必須與在主機上具有 sudo/root 存取權的使用者一起執行。

1. 使用部署的值在每個主機上建立nodeConfig.yaml檔案。如需可用組態設定的完整說明，請參閱 the section called “混合節點 nodeadm”。如果您的混合節點 IAM 角色沒有 eks:DescribeCluster動作的許可，您必須在 的叢集區段中傳遞 Kubernetes API 端點、叢集 CA 套件和 Kubernetes 服務 IPv4 CIDRnodeConfig.yaml。

   a. 如果您使用現場部署憑證提供者的 AWS SSM 混合啟用，請使用下列nodeConfig.yaml範例。

      i. 使用您叢集的名稱取代 CLUSTER_NAME。

      ii. 將 取代AWS_REGION為託管叢集的 AWS 區域。例如 us-west-2。

      iii. ACTIVATION_CODE 以您在建立 AWS SSM 混合啟用時收到的啟用代碼取代 。如需詳細資訊，請參閱the section called “準備登入資料”。

      iv. ACTIVATION_ID 以您在建立 AWS SSM 混合啟用時收到的啟用 ID 取代 。您可以從 AWS Systems Manager 主控台或 CLI AWS aws ssm describe-activations命令擷取此資訊。

      ```
      apiVersion: node.eks.aws/v1alpha1
      kind: NodeConfig
      spec:
        cluster:
          name: CLUSTER_NAME
          region: AWS_REGION
        hybrid:
          ssm:
            activationCode: ACTIVATION_CODE
            activationId: ACTIVATION_ID
      ```

   b. 如果您為內部部署憑證提供者使用 AWS IAM Roles Anywhere，請使用下列nodeConfig.yaml範例。

      i. 使用您叢集的名稱取代 CLUSTER_NAME。

      ii. 將 取代AWS_REGION為託管叢集的 AWS 區域。例如 us-west-2。

      iii. NODE_NAME 將 取代為您的節點名稱。如果您使用 "sts:RoleSessionName": "${aws:PrincipalTag/x509Subject/CN}" 資源條件設定混合節點 IAM 角色的信任政策，節點名稱必須符合主機上憑證的 CN。nodeName 您使用的 長度不得超過 64 個字元。

      iv. TRUST_ANCHOR_ARN 以您在準備混合節點登入資料的步驟中設定的信任錨的 ARN 取代 。

      v. PROFILE_ARN 以您在 的步驟中設定的信任錨點 ARN 取代 the section called “準備登入資料”。

      vi. ROLE_ARN 將 取代為混合節點 IAM 角色的 ARN。

vii.CERTIFICATE_PATH 使用磁碟中節點憑證的路徑取代 。如果您未指定，則預設值為 /etc/iam/pki/server.pem。

viiiKEY_PATH 將 取代為憑證私有金鑰在磁碟中的路徑。如果您未指定，則預設值為 /etc/iam/pki/server.key。

```
apiVersion: node.eks.aws/v1alpha1
kind: NodeConfig
spec:
  cluster:
    name: CLUSTER_NAME
    region: AWS_REGION
  hybrid:
    iamRolesAnywhere:
      nodeName: NODE_NAME
      trustAnchorArn: TRUST_ANCHOR_ARN
      profileArn: PROFILE_ARN
      roleArn: ROLE_ARN
      certificatePath: CERTIFICATE_PATH
      privateKeyPath: KEY_PATH
```

2. 使用 執行 nodeadm init命令nodeConfig.yaml，將混合節點連線至 Amazon EKS 叢集。

```
nodeadm init -c file://nodeConfig.yaml
```

如果上述命令成功完成，您的混合節點已加入您的 Amazon EKS 叢集。您可以在 Amazon EKS 主控台中透過導覽至叢集的運算索引標籤 (確保 IAM 主體具有檢視的許可) 或使用 來驗證這一點kubectl get nodes。

> **Important**
>
> 您的節點將具有 狀態 Not Ready，這是由於混合節點上缺少執行的 CNI 所致。如果您的節點未加入叢集，請參閱 the section called “故障診斷”。

步驟 4：設定混合節點的 CNI

若要讓您的混合節點準備好執行應用程式，請繼續執行 上的步驟the section called “設定 CNI”。

使用 Bottlerocket 連接混合節點

本主題說明如何將執行 Bottlerocket 的混合節點連接到 Amazon EKS 叢集。Bottlerocket 是由 贊助和支援的開放原始碼 Linux 發行版本 AWS。Bottlerocket 專為託管容器工作負載而打造。透過 Bottlerocket，您可以自動化容器基礎設施的更新，以改善容器化部署的可用性並降低營運成本。Bottlerocket 僅包含執行容器的必要軟體，可改善資源用量、降低安全威脅並降低管理開銷。

EKS 混合節點僅支援 Bottlerocket 1.37.0 版及更高版本的 VMware 變體。Bottlerocket 的 VMware 變體適用於 Kubernetes 1.28 版及更高版本。這些變體的作業系統映像包括 kubelet、containerd、aws-iam-authenticator 和其他 EKS 混合節點的軟體先決條件。您可以使用 Bottlerocket 設定檔案來設定這些元件，該檔案包含 Bottlerocket 引導和管理容器的 base64 編碼使用者資料。設定這些設定可讓 Bottlerocket 使用您的混合節點憑證提供者來驗證叢集的混合節點。混合節點加入叢集後，它們會在 `Not Ready` Amazon EKS 主控台和 Kubernetes 相容工具中顯示 狀態，例如 `kubectl`。完成此頁面上的步驟後，請繼續 the section called “設定 CNI” ，讓您的混合節點準備好執行應用程式。

先決條件

將混合節點連線至 Amazon EKS 叢集之前，請確定您已完成先決條件步驟。

- 您可以從內部部署環境連線至託管 Amazon EKS 叢集的 AWS 區域。如需詳細資訊，請參閱the section called “準備聯網”。
- 您已建立混合節點 IAM 角色，並設定內部部署憑證提供者 (AWS Systems Manager 混合啟用或 AWS IAM Roles Anywhere)。如需詳細資訊，請參閱the section called “準備登入資料”。
- 您已建立啟用混合節點的 Amazon EKS 叢集。如需詳細資訊，請參閱the section called “建立叢集”。
- 您已將混合節點 IAM 角色與 Kubernetes 角色型存取控制 (RBAC) 許可建立關聯。如需詳細資訊，請參閱the section called “準備叢集存取”。

步驟 1：建立 Bottlerocket 設定 TOML 檔案

若要設定混合節點的 Bottlerocket，您需要建立具有必要組態`settings.toml`的檔案。TOML 檔案的內容會根據您使用的登入資料提供者 (SSM 或 IAM Roles Anywhere) 而有所不同。佈建 Bottlerocket 執行個體時，此檔案會以使用者資料的形式傳遞。

SSM

如果您使用 AWS Systems Manager 做為登入資料提供者，請使用下列內容建立`settings.toml`檔案：

```
[settings.kubernetes]
cluster-name = "<cluster-name>"
api-server = "<api-server-endpoint>"
cluster-certificate = "<cluster-certificate-authority>"
hostname-override = "<hostname>"
provider-id = "eks-hybrid:///<region>/<cluster-name>/<hostname>"
authentication-mode = "aws"

[settings.network]
hostname = "<hostname>"

[settings.aws]
region = "<region>"

[settings.kubernetes.node-labels]
"eks.amazonaws.com/compute-type" = "hybrid"
"eks.amazonaws.com/hybrid-credential-provider" = "ssm"

[settings.host-containers.admin]
enabled = true
user-data = "<base64-encoded-admin-container-userdata>"

[settings.bootstrap-containers.eks-hybrid-setup]
mode = "always"
user-data = "<base64-encoded-bootstrap-container-userdata>"

[settings.host-containers.control]
enabled = true
```

將預留位置取代為下列值：

- <cluster-name>：Amazon EKS 叢集的名稱。
- <api-server-endpoint>：叢集的 API 伺服器端點。
- <cluster-certificate-authority>：叢集的 base64 編碼 CA 套件。
- <region>：託管叢集 AWS 的區域，例如 "us-east-1"。
- <hostname>： Bottlerocket 執行個體的主機名稱，也會設定為節點名稱。這可以是您選擇的任何唯一值，但必須遵循 Kubernetes 物件命名慣例。此外，您使用的主機名稱不能超過 64 個字元。注意：使用 SSM 提供者時，在向 SSM 註冊執行個體之後，此主機名稱和節點名稱將由受管執行個體 ID （例如 mi-* ID) 取代。

- <base64-encoded-admin-container-userdata>： Bottlerocket 管理容器組態的 base64 編碼內容。啟用管理員容器可讓您使用 SSH 連線至 Bottlerocket 執行個體，以進行系統探索和偵錯。雖然這不是必要的設定，但我們建議您啟用它，以便進行故障診斷。如需使用管理容器進行身分驗證的詳細資訊，請參閱 Bottlerocket 管理容器文件。管理員容器採用 JSON 格式的 SSH 使用者和金鑰輸入，例如，

```
{
  "user": "<ssh-user>",
  "ssh": {
    "authorized-keys": [
      "<ssh-authorized-key>"
    ]
  }
}
```

- <base64-encoded-bootstrap-container-userdata>： Bottlerocket 引導容器組態的 base64 編碼內容。如需其組態的詳細資訊，請參閱 Bottlerocket 引導容器文件。引導容器負責將執行個體註冊為 AWS SSM 受管執行個體，並將其聯結為 Amazon EKS 叢集上的 Kubernetes 節點。傳遞到引導容器的使用者資料採用命令調用的形式，接受 做為您先前建立的 SSM 混合啟用碼和 ID 的輸入：

```
eks-hybrid-ssm-setup --activation-id=<activation-id> --activation-code=<activation-
code> --region=<region>
```

IAM Roles Anywhere

如果您使用 AWS IAM Roles Anywhere 做為登入資料提供者，請使用下列內容建立settings.toml檔案：

```
[settings.kubernetes]
cluster-name = "<cluster-name>"
api-server = "<api-server-endpoint>"
cluster-certificate = "<cluster-certificate-authority>"
hostname-override = "<hostname>"
provider-id = "eks-hybrid:///<region>/<cluster-name>/<hostname>"
authentication-mode = "aws"

[settings.network]
hostname = "<hostname>"
```

```
[settings.aws]
region = "<region>"
config = "<base64-encoded-aws-config-file>"

[settings.kubernetes.node-labels]
"eks.amazonaws.com/compute-type" = "hybrid"
"eks.amazonaws.com/hybrid-credential-provider" = "iam-ra"

[settings.host-containers.admin]
enabled = true
user-data = "<base64-encoded-admin-container-userdata>"

[settings.bootstrap-containers.eks-hybrid-setup]
mode = "always"
user-data = "<base64-encoded-bootstrap-container-userdata>"
```

將預留位置取代為下列值：

- <cluster-name>：Amazon EKS 叢集的名稱。
- <api-server-endpoint>：叢集的 API 伺服器端點。
- <cluster-certificate-authority>：叢集的 base64 編碼 CA 套件。
- <region>：託管叢集 AWS 的區域，例如 "us-east-1"
- <hostname>： Bottlerocket 執行個體的主機名稱，也會設定為節點名稱。這可以是您選擇的任何唯一值，但必須遵循 Kubernetes 物件命名慣例。此外，您使用的主機名稱不能超過 64 個字元。注意：使用 IAM-RA 提供者時，如果您使用 "sts:RoleSessionName": "${aws:PrincipalTag/x509Subject/CN}" 資源條件設定混合節點 IAM 角色的信任政策，節點名稱必須符合主機上憑證的 CN。
- <base64-encoded-aws-config-file>：組態檔案的 base64 AWS 編碼內容。檔案的內容應如下所示：

```
[default]
credential_process = aws_signing_helper credential-process --certificate /root/.aws/
node.crt --private-key /root/.aws/node.key --profile-arn <profile-arn> --role-arn
 <role-arn> --trust-anchor-arn <trust-anchor-arn> --role-session-name <role-session-
name>
```

- <base64-encoded-admin-container-userdata>： Bottlerocket 管理容器組態的 base64 編碼內容。啟用管理員容器可讓您使用 SSH 連線至 Bottlerocket 執行個體，以進行系統探索和偵錯。雖然這不是必要的設定，但我們建議您啟用它，以便進行故障診斷。如需使用管理容器進行身分驗證的詳細資訊，請參閱 Bottlerocket 管理容器文件。管理員容器採用 JSON 格式的 SSH 使用者和金鑰輸入，例如，

```
{
  "user": "<ssh-user>",
  "ssh": {
    "authorized-keys": [
      "<ssh-authorized-key>"
    ]
  }
}
```

- <base64-encoded-bootstrap-container-userdata>： Bottlerocket 引導容器組態的 base64 編碼內容。如需其組態的詳細資訊，請參閱 Bottlerocket 引導容器文件。引導容器負責在執行個體上建立 IAM Roles Anywhere 主機憑證和憑證私有金鑰檔案。然後， 會使用這些登入資料aws_signing_helper來取得臨時登入資料，以便與您的 Amazon EKS 叢集進行身分驗證。傳遞到引導容器的使用者資料採用命令調用的形式，接受 做為您先前建立之憑證和私有金鑰的內容輸入：

```
eks-hybrid-iam-ra-setup --certificate=<certificate> --key=<private-key>
```

步驟 2：使用使用者資料佈建 Bottlerocket vSphere VM

建構 TOML 檔案之後，請在 vSphere VM 建立期間將其做為使用者資料傳遞。請記住，必須在第一次開啟 VM 電源之前設定使用者資料。因此，您需要在建立執行個體時提供它，或者如果您想要提前建立 VM，則 VM 必須處於 poweredOff 狀態，直到您為其設定使用者資料為止。例如，如果使用 govc CLI：

第一次建立 VM

```
govc vm.create \
  -on=true \
  -c=2 \
  -m=4096 \
  -net.adapter=<network-adapter> \
```

```
  -net=<network-name> \
  -e guestinfo.userdata.encoding="base64" \
  -e guestinfo.userdata="$(base64 -w0 settings.toml)" \
  -template=<template-name> \
  <vm-name>
```

更新現有 VM 的使用者資料

```
govc vm.create \
    -on=false \
    -c=2 \
    -m=4096 \
    -net.adapter=<network-adapter> \
    -net=<network-name> \
    -template=<template-name> \
    <vm-name>

govc vm.change
    -vm <vm-name> \
    -e guestinfo.userdata="$(base64 -w0 settings.toml)" \
    -e guestinfo.userdata.encoding="base64"

govc vm.power -on <vm-name>
```

在上述區段中，`-e guestinfo.userdata.encoding="base64"`選項指定使用者資料為 base64 編碼。`-e guestinfo.userdata` 選項會將`settings.toml`檔案的 base64 編碼內容做為使用者資料傳遞至 Bottlerocket 執行個體。將預留位置取代為您的特定值，例如 Bottlerocket OVA 範本和聯網詳細資訊。

步驟 3：驗證混合節點連線

Bottlerocket 執行個體啟動後，它會嘗試加入您的 Amazon EKS 叢集。您可以透過導覽至叢集的運算索引標籤或執行下列命令，在 Amazon EKS 主控台中驗證連線：

```
kubectl get nodes
```

> **Important**
>
> 您的節點將具有 狀態 `Not Ready`，這是由於混合節點上缺少執行的 CNI 所致。如果您的節點未加入叢集，請參閱 the section called “故障診斷”。

步驟 4：設定混合節點的 CNI

若要讓您的混合節點準備好執行應用程式，請繼續執行 上的步驟the section called “設定 CNI”。

升級叢集的混合節點

升級混合節點的指引類似於在 Amazon EC2 中執行的自我管理 Amazon EKS 節點。我們建議您在目標 Kubernetes 版本上建立新的混合節點、將現有應用程式正常遷移到新 Kubernetes 版本上的混合節點，以及從叢集中移除舊 Kubernetes 版本上的混合節點。開始升級之前，請務必檢閱升級的 Amazon EKS 最佳實務。對於具有雲端節點的 Amazon EKS 叢集，Amazon EKS 混合節點具有相同的 eks/latest/userguide/kubernetes-versions.html【Kubernetes 版本支援，type="documentation"】，包括標準和延伸支援。

Amazon EKS 混合節點遵循與上游 Kubernetes 相同的節點版本扭曲政策。Amazon EKS 混合節點不能位於比 Amazon EKS 控制平面更新的版本上，混合節點最多可以比 Amazon EKS 控制平面次要版本舊三個 Kubernetes 次要版本。

如果您沒有為切換遷移升級策略在目標 Kubernetes 版本上建立新的混合節點的備用容量，您也可以使用 Amazon EKS 混合節點 CLI (`nodeadm`) 來升級混合節點的 Kubernetes 版本。

> **Important**
>
> 如果您要使用 就地升級混合節點`nodeadm`，則在舊版 Kubernetes 元件關閉並安裝和啟動新 Kubernetes 版本元件的程序中，節點會有停機時間。

先決條件

升級之前，請確定您已完成下列先決條件。

- 混合節點升級的目標 Kubernetes 版本必須等於或小於 Amazon EKS 控制平面版本。
- 如果您遵循切換遷移升級策略，則在目標 Kubernetes 版本上安裝的新混合節點必須符合the section called “先決條件”需求。這包括在 Amazon EKS 叢集建立期間傳遞的遠端節點網路 CIDR 內擁有 IP 地址。
- 對於切換遷移和就地升級，混合節點必須能夠存取所需的網域，以提取混合節點相依性的新版本。
- 您必須在用來與 Amazon EKS Kubernetes API 端點互動的本機機器或執行個體上安裝 kubectl。
- 您的 CNI 版本必須支援您要升級到的 Kubernetes 版本。如果沒有，請在升級混合節點之前升級 CNI 版本。如需詳細資訊，請參閱the section called “設定 CNI”。

切換遷移 （藍綠） 升級

切換遷移升級是指使用目標 Kubernetes 版本在新主機上建立新的混合節點、將現有應用程式正常遷移到目標 Kubernetes 版本上的新混合節點，以及從您的叢集中移除舊 Kubernetes 版本上的混合節點的程序。此策略也稱為藍綠遷移。

1. 依照the section called “連接混合節點”步驟將新主機連接為混合節點。執行 `nodeadm install`命令時，請使用您的目標 Kubernetes 版本。
2. 啟用目標 Kubernetes 版本上的新混合節點與舊 Kubernetes 版本上的混合節點之間的通訊。此組態可讓您在將工作負載遷移至目標 Kubernetes 版本上的混合節點時，讓 Pod 彼此通訊。
3. 確認目標 Kubernetes 版本上的混合節點已成功加入叢集，且狀態為就緒。
4. 使用下列命令，將您要移除的每個節點標記為不可排程。如此一來，新 Pod 就不會排程或重新排程在您要取代的節點上。如需詳細資訊，請參閱 Kubernetes 文件中的 kubectl 封鎖。`NODE_NAME` 將取代為舊 Kubernetes 版本的混合節點名稱。

   ```
   kubectl cordon NODE_NAME
   ```

   您可以使用下列程式碼片段來識別並繫結特定 Kubernetes 版本 （在此情況下為 `1.28`) 的所有節點。

   ```
   K8S_VERSION=1.28
   for node in $(kubectl get nodes -o json | jq --arg K8S_VERSION
    "$K8S_VERSION" -r '.items[] | select(.status.nodeInfo.kubeletVersion |
    match("\($K8S_VERSION)")).metadata.name')
   do
       echo "Cordoning $node"
       kubectl cordon $node
   done
   ```

5. 如果您目前的部署在混合節點上執行少於兩個 CoreDNS 複本，請將部署擴展到至少兩個複本。我們建議您在混合節點上執行至少兩個 CoreDNS 複本，以在正常操作期間提供彈性。

   ```
   kubectl scale deployments/coredns --replicas=2 -n kube-system
   ```

6. 使用下列命令，耗盡您要從叢集中移除的舊 Kubernetes 版本上的每個混合節點。如需耗盡節點的詳細資訊，請參閱 Kubernetes 文件中的安全地耗盡節點。`NODE_NAME` 將 取代為舊 Kubernetes 版本的混合節點名稱。

```
kubectl drain NODE_NAME --ignore-daemonsets --delete-emptydir-data
```

您可以使用下列程式碼片段來識別和耗盡特定 Kubernetes 版本 （在此情況下為 1.28) 的所有節點。

```
K8S_VERSION=1.28
for node in $(kubectl get nodes -o json | jq --arg K8S_VERSION
 "$K8S_VERSION" -r '.items[] | select(.status.nodeInfo.kubeletVersion |
 match("\($K8S_VERSION)")).metadata.name')
do
    echo "Draining $node"
    kubectl drain $node --ignore-daemonsets --delete-emptydir-data
done
```

7. 您可以使用 從主機nodeadm停止和移除混合節點成品。您必須nodeadm使用具有根/sudo 權限的使用者執行 。根據預設，如果節點上剩餘 Pod，nodeadm uninstall則 不會繼續。如需更多資訊，請參閱the section called “混合節點 nodeadm”。

```
nodeadm uninstall
```

8. 在混合節點成品停止並解除安裝的情況下，從您的叢集中移除節點資源。

```
kubectl delete node node-name
```

您可以使用下列程式碼片段來識別和刪除特定 Kubernetes 版本 （在此情況下為 1.28) 的所有節點。

```
K8S_VERSION=1.28
for node in $(kubectl get nodes -o json | jq --arg K8S_VERSION
 "$K8S_VERSION" -r '.items[] | select(.status.nodeInfo.kubeletVersion |
 match("\($K8S_VERSION)")).metadata.name')
do
    echo "Deleting $node"
    kubectl delete node $node
done
```

9. 視您選擇的 CNI 而定，在執行上述步驟後，您的混合節點上可能會有剩餘的成品。如需詳細資訊，請參閱the section called “設定 CNI”。

就地升級

就地升級程序是指使用 nodeadm upgrade來升級混合節點的 Kubernetes 版本，而無需使用新的實體或虛擬主機和切換遷移策略。程序nodeadm upgrade會關閉在混合節點上執行的現有較舊 Kubernetes 元件、解除安裝現有的較舊 Kubernetes 元件、安裝新的目標 Kubernetes 元件，以及啟動新的目標 Kubernetes 元件。強烈建議您一次升級一個節點，將對混合節點上執行之應用程式的影響降至最低。此程序的持續時間取決於您的網路頻寬和延遲。

1. 使用以下命令將您要升級的節點標記為不可排程。這樣新 Pod 就不會在您升級的節點上排程或重新排程。如需詳細資訊，請參閱 Kubernetes 文件中的 kubectl 封鎖。NODE_NAME 以您要升級的混合節點名稱取代

```
kubectl cordon NODE_NAME
```

2. 使用以下命令耗盡您要升級的節點。如需耗盡節點的詳細資訊，請參閱 Kubernetes 文件中的安全地耗盡節點。NODE_NAME 以您要升級的混合節點名稱取代 。

```
kubectl drain NODE_NAME --ignore-daemonsets --delete-emptydir-data
```

3. 在您升級的混合節點nodeadm upgrade上執行 。您必須nodeadm使用具有根/sudo 權限的使用者執行 。節點的名稱會透過 AWS SSM 和 IAM Roles Anywhere AWS 憑證提供者的升級來保留。您無法在升級程序期間變更登入資料提供者。如需 的組態值the section called “混合節點 nodeadm”，請參閱 nodeConfig.yaml。K8S_VERSION 將 取代為您升級的目標 Kubernetes 版本。

```
nodeadm upgrade K8S_VERSION -c file://nodeConfig.yaml
```

4. 若要允許升級後在節點上排程 Pod，請輸入下列內容。NODE_NAME 將 取代為節點的名稱。

```
kubectl uncordon NODE_NAME
```

5. 觀察混合節點的狀態，並等待節點關閉，然後重新啟動具有就緒狀態的新 Kubernetes 版本。

```
kubectl get nodes -o wide -w
```

混合節點的修補程式安全性更新

本主題說明針對混合節點上執行的特定套件和相依性，執行就地修補安全性更新的程序。最佳實務是建議您定期更新混合節點，以接收 CVEs和安全性修補程式。

如需升級 Kubernetes 版本的步驟，請參閱 the section called “升級混合節點”。

可能需要安全修補的軟體範例之一是 containerd。

Containerd

containerd 是 EKS 混合節點的標準 Kubernetes 容器執行時間和核心相依性，用於管理容器生命週期，包括提取映像和管理容器執行。在混合節點containerd上，您可以透過 nodeadm CLI 或手動安裝 。會根據節點的作業系統，containerd從作業系統分佈套件或 Docker 套件nodeadm安裝 。

當發佈 中的 CVE containerd時，您有下列選項可以升級到混合節點containerd上的修補版本 。

步驟 1：檢查修補程式是否已發佈至套件管理員

您可以參考對應的安全公告，檢查 containerd CVE 修補程式是否已發佈至每個個別的作業系統套件管理員：

- Amazon Linux 2023
- RHEL
- Ubuntu 20.04
- Ubuntu 22.04
- Ubuntu 24.04

如果您使用 Docker 儲存庫做為 的來源containerd，您可以檢查 Docker 安全公告，以識別 Docker 儲存庫中修補版本的可用性。

步驟 2：選擇安裝修補程式的方法

有三種方法可在節點上就地修補和安裝安全升級。您可以使用哪種方法取決於修補程式是否可從套件管理員中的作業系統取得：

1. 使用發佈到套件管理員nodeadm upgrade的修補程式安裝修補程式，請參閱步驟 2 a。
2. 使用套件管理員直接安裝修補程式，請參閱步驟 2 b。
3. 安裝未在套件管理員中發佈的自訂修補程式。請注意，對於 的自訂修補程式有特殊考量containerd，步驟 2 c。

步驟 2 a：使用 修補 **nodeadm upgrade**

確認 containerd CVE 修補程式已發佈至作業系統或 Docker 儲存庫 (Apt 或 RPM) 之後，您可以使用 nodeadm upgrade命令升級至最新版本的 containerd。由於這不是 Kubernetes 版本升級，您必須將目前的 Kubernetes 版本傳遞至nodeadm升級命令。

```
nodeadm upgrade K8S_VERSION --config-source file:///root/nodeConfig.yaml
```

步驟 2 b：使用作業系統套件管理員修補

或者，您也可以透過個別的套件管理員進行更新，並使用它來升級containerd套件，如下所示。

Amazon Linux 2023

```
sudo yum update -y
sudo yum install -y containerd
```

RHEL

```
sudo yum install -y yum-utils
sudo yum-config-manager --add-repo https://download.docker.com/linux/rhel/docker-
ce.repo
sudo yum update -y
sudo yum install -y containerd
```

Ubuntu

```
sudo mkdir -p /etc/apt/keyrings
sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o /etc/apt/keyrings/
docker.asc
echo \
  "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc]
 https://download.docker.com/linux/ubuntu \
  $(. /etc/os-release && echo "${UBUNTU_CODENAME:-$VERSION_CODENAME}") stable" | \
  sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
sudo apt update -y
sudo apt install -y --only-upgrade containerd.io
```

步驟 2 c：CVE **Containerd** 修補程式未發佈於套件管理員

如果修補containerd版本只能透過其他方式使用，而不是在套件管理員中，例如在 GitHub 版本中，則您可以從containerd官方 GitHub 網站安裝 。

1. 如果機器已將叢集加入為混合節點，則需要執行 nodeadm uninstall命令。
2. 安裝官方containerd二進位檔。您可以在 GitHub 上使用官方安裝步驟。
3. 執行 nodeadm install命令，並將 --containerd-source 引數設為 none，這會略過containerd透過 安裝nodeadm。對於節點正在執行的任何作業系統，您可以在containerd來源none中使用 的值。

```
nodeadm install K8S_VERSION --credential-provider CREDS_PROVIDER --containerd-source
 none
```

移除混合節點

本主題說明如何從 Amazon EKS 叢集中刪除混合節點。您必須使用您選擇的 Kubernetes 相容工具刪除混合節點，例如 kubectl。從 Amazon EKS 叢集移除節點物件時，混合節點的費用會停止。如需混合節點定價的詳細資訊，請參閱 Amazon EKS 定價。

> **Important**
>
> 移除節點會對節點上執行的工作負載造成干擾。刪除混合節點之前，建議您先耗盡節點，將 Pod 移至另一個作用中節點。如需耗盡節點的詳細資訊，請參閱 Kubernetes 文件中的安全地耗盡節點。

從您用來與 Amazon EKS 叢集的 Kubernetes API 端點互動的本機機器或執行個體執行以下 kubectl 步驟。如果您使用的是特定kubeconfig檔案，請使用 --kubeconfig旗標。

步驟 1：列出您的節點

```
kubectl get nodes
```

步驟 2：耗盡節點

如需 kubectl drain命令的詳細資訊，請參閱 Kubernetes 文件中的 kubectl 耗盡。

```
kubectl drain --ignore-daemonsets <node-name>
```

步驟 3：停止並解除安裝混合節點成品

您可以使用 Amazon EKS 混合節點 CLI (nodeadm) 從主機停止和移除混合節點成品。您必須nodeadm使用具有根/sudo 權限的使用者執行 。根據預設，如果節點上剩餘 Pod，則 nodeadm uninstall不會繼續。如果您使用 AWS Systems Manager (SSM) 做為登入資料提供者，nodeadm uninstall命令會將主機取消註冊為 AWS SSM 受管執行個體。如需詳細資訊，請參閱the section called “混合節點 nodeadm”。

```
nodeadm uninstall
```

步驟 4：從叢集刪除節點

在混合節點成品停止並解除安裝的情況下，從您的叢集中移除節點資源。

```
kubectl delete node <node-name>
```

步驟 5：檢查是否有剩餘的成品

視您選擇的 CNI 而定，在執行上述步驟後，您的混合節點上可能會有剩餘的成品。如需詳細資訊，請參閱「the section called “設定 CNI”」。

設定混合節點的 CNI、附加元件和 Webhook

為混合節點設定叢集後，混合節點需要容器網路界面 (CNI)、附加元件、Webhook 和可能代理設定的組態。如需與混合節點相容的 EKS 和社群附加元件的完整清單，請參閱 the section called “設定附加元件”。

EKS 叢集洞察 EKS 包含 EKS 混合節點設定中組態錯誤的洞察檢查，這可能會影響叢集或工作負載的功能。如需叢集洞見的詳細資訊，請參閱 the section called “叢集洞察”。

下列列出針對混合節點具有不同組態的附加元件和其他元件：

- 容器聯網界面 (CNI)：支援 Cilium 和 Calico 的核心功能，可與混合節點搭配使用。您可以使用您選擇的工具來管理混合節點上的 CNI，例如 Helm。 AWS VPC CNI 無法與混合節點搭配使用。如需詳細資訊，請參閱the section called “設定 CNI”。
- CoreDNS 和 **kube-proxy** ：CoreDNS 和 kube-proxy會在混合節點加入 EKS 叢集時自動安裝。這些附加元件可在叢集建立後做為 EKS 附加元件進行管理。
- 輸入和負載平衡：您可以將 AWS Load Balancer控制器和 Application Load Balancer (ALB) 或 Network Load Balancer (NLB) 與目標類型搭配使用，ip用於與 AWS Direct Connect 或 AWS Site-

to-Site VPN 連接的混合節點上的工作負載。或者，您可以將您選擇的輸入控制器或負載平衡器用於將流量保留在您的內部部署環境本機的應用程式流量。

- 指標：您可以使用 Amazon Managed Service for Prometheus (AMP) 無代理程式抓取器、 AWS Distro for Open Telemetry (ADOT)，以及具有混合節點的 Amazon CloudWatch 可觀測性代理程式。若要將 AMP 無代理程式抓取器用於混合節點上的 Pod 指標，您的 Pod 必須可從您用於 EKS 叢集的 VPC 存取。
- 日誌：您可以為啟用混合節點的叢集啟用 EKS 控制平面日誌記錄。您可以使用 ADOT EKS 附加元件和 Amazon CloudWatch 可觀測性代理程式 EKS 附加元件進行混合節點和 Pod 記錄。
- Pod 身分和 IRSA：您可以將 EKS Pod 身分和服務帳戶 (IRSA) 的 IAM 角色與在混合節點上執行的應用程式搭配使用，以便為與其他 AWS 服務在混合節點上執行的 Pod 啟用精細存取。
- Webhook：如果您正在執行 Webhook，請參閱 the section called “設定 Webhook” ，了解如果您無法使內部部署 Pod 網路可路由，可選擇在雲端節點上執行 Webhook 的考量事項和步驟。
- Proxy：如果您在內部部署環境中為離開資料中心或邊緣環境的流量使用代理伺服器，您可以將混合節點和叢集設定為使用代理伺服器。如需詳細資訊，請參閱the section called “設定代理”。

主題

- 設定混合節點的 CNI
- 設定混合節點的附加元件
- 設定混合節點的 Webhook
- 設定混合節點的代理

設定混合節點的 CNI

Cilium 和 Calico 支援做為 Amazon EKS 混合節點的容器網路界面 (CNIs)。您必須為混合節點安裝 CNI，才能準備好為工作負載提供服務。在 CNI 執行`Not Ready`之前，混合節點會顯示 狀態。您可以使用您選擇的工具來管理這些 CNIs，例如 Helm。Amazon VPC CNI 與混合節點不相容，且 VPC CNI 已針對`eks.amazonaws.com/compute-type: hybrid`標籤設定抗親和性。

CNI 版本相容性

對於 Amazon EKS 中支援的每個 Kubernetes 版本，EKS 混合節點`v1.16.x`支援並建議使用 Cilium 版本。

Amazon EKS 支援的每個 Kubernetes 版本都`v3.29.x`支援 Calico 版本，並建議用於 EKS 混合節點。

請參閱 Amazon EKS 支援的 Kubernetes 版本支援。EKS 混合節點具有與具有雲端節點的 Amazon EKS 叢集相同的 Kubernetes 版本支援。

支援的功能

AWS 為 Cilium 和 Calico 的下列功能提供技術支援，以便與混合節點搭配使用。如果您計劃在 AWS 支援範圍之外使用功能，我們建議您取得外掛程式的商業支援，或擁有內部專業知識來排除故障，並為 CNI 外掛程式專案提供修正。

功能	Cilium	Calico
Kubernetes 網路一致性	是	是
控制平面到節點的連線	是	是
控制平面到 Pod 的連線	是	是
生命週期管理	安裝、升級、刪除	安裝、升級、刪除
網路模式	VXLAN	VXLAN
IP 地址管理 (IPAM)	叢集範圍 (Cilium IPAM)	Calico IPAM
IP 系列	IPv4	IPv4
BGP	是（鈰控制平面）	是

Cilium 考量事項

- 根據預設，Cilium 設定為在覆蓋/通道模式下執行，並以 VXLAN 做為封裝方法。此模式對基礎實體網路的需求最少。
- 根據預設，Cilium 會遮罩所有 Pod 流量的來源 IP 地址，並將叢集移至節點的 IP 地址。這可讓您使用混合節點執行 Cilium，無論叢集上是否已設定遠端 Pod 網路。如果您停用偽裝，則您的 Pod CIDRs 必須可在內部部署網路上路由，而且您必須使用遠端 Pod 網路設定 Amazon EKS 叢集。
- 如果您在混合節點上執行 Webhook，您的 Pod CIDRs 必須在內部部署網路上可路由，而且您必須使用遠端 Pod 網路設定 Amazon EKS 叢集。如果您的 Pod CIDRs 在內部部署網路上無法路由，建議您在相同叢集的雲端節點上執行 Webhook。如需詳細資訊，請參閱設定混合節點的 Webhook。

- 讓 Pod CIDR 在內部部署網路上可路由的常見方法是使用 BGP 公告 Pod 地址。若要搭配 Cilium 使用 BGP，您必須在 Helm 組態bgpControlPlane.enabled: true中設定 。如需 Cilium BGP 支援的詳細資訊，請參閱 Cilium 文件中的 Cilium BGP 控制平面。
- Cilium 中的預設 IP 地址管理 (IPAM) 稱為 Cluster Scope，Cilium 運算子會根據使用者設定的 Pod CIDRs 為每個節點配置 IP 地址。Pod CIDRs 是以 clusterPoolIPv4PodCIDRList Helm 值設定，這應該符合您為 Amazon EKS 叢集設定的遠端 Pod 網路 CIDRs。Cilium 會將區段從 配置clusterPoolIPv4PodCIDRList給每個節點。每個節點區段的大小是以 clusterPoolIPv4MaskSize Helm 值設定。如需 clusterPoolIPv4PodCIDRList和 的詳細資訊clusterPoolIPv4MaskSize，請參閱 Cilium 文件中的展開叢集集區。

在混合節點上安裝 Cilium

1. 請確定您已在命令列環境中安裝 Helm CLI。如需安裝說明，請參閱 設定 Helm。
2. 安裝 Cilium Helm 儲存庫。

```
helm repo add cilium https://helm.cilium.io/
```

3. 建立名為 的 YAML 檔案cilium-values.yaml。下列範例透過設定eks.amazonaws.com/compute-type: hybrid標籤的親和性，將 Cilium 設定為僅在混合節點上執行。
   - 如果您使用遠端 Pod 網路設定 Amazon EKS 叢集，請為您的 設定相同的 Pod CIDRsclusterPoolIPv4PodCIDRList。例如 10.100.0.0/24。在覆蓋/通道模式下執行 CNI 時，您的內部部署 Pod CIDR 不得與內部部署節點 CIDR 重疊。
   - clusterPoolIPv4MaskSize 根據每個節點所需的 Pod 來設定 。例如，25對於每個節點 128 個 Pod 的 /25 區段大小。
   - 在叢集上部署 Cilium clusterPoolIPv4MaskSize之後，您不應該變更 clusterPoolIPv4PodCIDRList或 ，請參閱 Cilium 文件中的展開叢集集區。
   - 如需 Cilium 的 Helm 值完整清單，請參閱 Cilium 文件中的 Helm 參考。

```
affinity:
  nodeAffinity:
    requiredDuringSchedulingIgnoredDuringExecution:
      nodeSelectorTerms:
      - matchExpressions:
        - key: eks.amazonaws.com/compute-type
          operator: In
          values:
          - hybrid
```

```
ipam:
  mode: cluster-pool
  operator:
    clusterPoolIPv4MaskSize: 25
    clusterPoolIPv4PodCIDRList:
    - POD_CIDR
loadBalancer:
  serviceTopology: true
operator:
  affinity:
    nodeAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        nodeSelectorTerms:
        - matchExpressions:
          - key: eks.amazonaws.com/compute-type
            operator: In
            values:
              - hybrid
  unmanagedPodWatcher:
    restart: false
envoy:
  enabled: false
```

4. 在叢集上安裝 Cilium。
   - CILIUM_VERSION 將 取代為 Cilium 版本 （ 例如 v1.17.0)。建議您為 Cilium 次要版本執行最新的修補程式版本。您可以在 Cilium 文件的穩定版本區段中找到指定次要 Cilium 版本的最新修補程式版本。
   - 如果您要為部署啟用 BGP，請在下列命令中新增 --set bgpControlPlane.enabled=true旗標。
   - 如果您使用特定的 kubeconfig 檔案，請使用 --kubeconfig旗標搭配 Helm 安裝命令。

```
helm install cilium cilium/cilium \
    --version CILIUM_VERSION \
    --namespace kube-system \
    --values cilium-values.yaml
```

5. 您可以使用下列命令確認您的 Cilium 安裝成功。您應該會看到cilium-operator部署和在每個混合節點cilium-agent上執行的 。此外，您的混合節點現在應具有狀態 Ready。如需有關如何為 Cilium 設定 BGP 的資訊，請繼續下一個步驟。

```
kubectl get pods -n kube-system
```

```
NAME                              READY   STATUS    RESTARTS   AGE
cilium-jjjn8                      1/1     Running   0          11m
cilium-operator-d4f4d7fcb-sc5xn   1/1     Running   0          11m
```

```
kubectl get nodes
```

```
NAME                   STATUS   ROLES    AGE   VERSION
mi-04a2cf999b7112233   Ready    <none>   19m   v1.31.0-eks-a737599
```

6. 若要搭配 Cilium 使用 BGP 來透過內部部署網路公告您的 Pod 地址，您必須已搭配 安裝 Ciliumbgp`ControlPlane.enabled: true`。如果您為現有的 Cilium 部署啟用 BGP，如果先前未啟用 BGP，則必須重新啟動 Cilium 運算子來套用 BGP 組態。您可以在 Helm 值`true`中`operator.rollOutPods`將 設定為 ，以在 Helm 安裝/升級程序中重新啟動 Cilium 運算子。

   若要在 Cilium 中設定 BGP，請先使用 `cilium-bgp-cluster.yaml` `CiliumBGPClusterConfig`定義建立名為 的檔案。您可能需要向您的網路管理員取得下列資訊。

   - 針對執行 Cilium 的節點`localASN`，使用 ASN 設定 。
   - `peerASN` 使用現場部署路由器的 ASN 設定 。
   - `peerAddress` 使用執行 Cilium 的每個節點將對等的現場部署路由器 IP 來設定 。

   ```
   apiVersion: cilium.io/v2alpha1
   kind: CiliumBGPClusterConfig
   metadata:
     name: cilium-bgp
   spec:
     nodeSelector:
       matchExpressions:
       - key: eks.amazonaws.com/compute-type
         operator: In
         values:
         - hybrid
     bgpInstances:
     - name: "rack0"
       localASN: NODES_ASN
       peers:
       - name: "onprem-router"
         peerASN: ONPREM_ROUTER_ASN
         peerAddress: ONPREM_ROUTER_IP
   ```

```
      peerConfigRef:
        name: "cilium-peer"
```

7. 將 Cilium BGP 叢集組態套用至您的叢集。

```
kubectl apply -f cilium-bgp-cluster.yaml
```

8. CiliumBGPPeerConfig 資源會定義 BGP 對等組態。多個對等可以共用相同的組態，並提供常見CiliumBGPPeerConfig資源的參考。建立名為 的檔案cilium-bgp-peer.yaml，為您的內部部署網路設定對等組態。如需組態選項的完整清單，請參閱 Cilium 文件中的 BGP 對等組態。

```
apiVersion: cilium.io/v2alpha1
kind: CiliumBGPPeerConfig
metadata:
  name: cilium-peer
spec:
  timers:
    holdTimeSeconds: 30
    keepAliveTimeSeconds: 10
  gracefulRestart:
    enabled: true
    restartTimeSeconds: 120
  families:
    - afi: ipv4
      safi: unicast
      advertisements:
        matchLabels:
          advertise: "bgp"
```

9. 將 Cilium BGP 對等組態套用至叢集。

```
kubectl apply -f cilium-bgp-peer.yaml
```

10. CiliumBGPAdvertisement 資源用於定義與其相關聯的各種廣告類型和屬性。建立名為 的檔案，cilium-bgp-advertisement.yaml並使用 設定來設定 CiliumBGPAdvertisement 資源。

```
apiVersion: cilium.io/v2alpha1
kind: CiliumBGPAdvertisement
metadata:
  name: bgp-advertisements
  labels:
```

```
    advertise: bgp
spec:
  advertisements:
    - advertisementType: "PodCIDR"
    - advertisementType: "Service"
      service:
        addresses:
          - ClusterIP
          - ExternalIP
          - LoadBalancerIP
```

11將 Cilium BGP 公告組態套用至您的叢集。

```
kubectl apply -f cilium-bgp-advertisement.yaml
```

您可以使用 `cilium bgp peers`命令，確認 BGP 對等互連與 Cilium CLI 搭配運作。您應該會在環境的輸出中看到正確的值，工作階段狀態為 `established`。如需故障診斷的詳細資訊，請參閱 Cilium 文件中的故障診斷和操作指南。

升級混合節點上的 Cilium

升級 Cilium 部署之前，請仔細檢閱 Cilium 升級文件和升級備註，以了解目標 Cilium 版本中的變更。

1. 請確定您已在命令列環境中安裝 `helm` CLI。如需安裝說明，請參閱 Helm 文件。
2. 安裝 Cilium Helm 儲存庫。

```
helm repo add cilium https://helm.cilium.io/
```

3. 執行 Cilium 升級預檢。`CILIUM_VERSION` 將 取代為目標 Cilium 版本。建議您為 Cilium 次要版本執行最新的修補程式版本。您可以在 Cilium 文件的穩定版本區段中找到指定次要 Cilium 版本的最新修補程式版本。

```
helm install cilium-preflight cilium/cilium --version CILIUM_VERSION \
  --namespace=kube-system \
  --set preflight.enabled=true \
  --set agent=false \
  --set operator.enabled=false
```

4. 套用 之後`cilium-preflight.yaml`，請確定 Pod `READY` 的數量與執行的 Cilium Pod 數量相同。

```
kubectl get ds -n kube-system | sed -n '1p;/cilium/p'
```

```
NAME                      DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE
 SELECTOR   AGE
cilium                    2         2         2       2            2           <none>
          1h20m
cilium-pre-flight-check   2         2         2       2            2           <none>
          7m15s
```

5. 一旦 READY Pod 的數量相等，請確定 Cilium 預檢部署也標示為 READY 1/1。如果顯示 READY 0/1，請參閱 CNP 驗證區段，並在繼續升級之前解決部署的問題。

```
kubectl get deployment -n kube-system cilium-pre-flight-check -w
```

```
NAME                      READY   UP-TO-DATE   AVAILABLE   AGE
cilium-pre-flight-check   1/1     1            0           12s
```

6. 刪除預檢

```
helm uninstall cilium-preflight --namespace kube-system
```

7. 在正常叢集操作期間，所有 Cilium 元件都應執行相同的版本。下列步驟說明如何將所有元件從一個穩定版本升級至較新穩定版本。從一個次要版本升級至另一個次要版本時，建議您先升級至現有 Cilium 次要版本的最新修補程式版本。若要將中斷降至最低，請將 `upgradeCompatibility`選項設定為您在此叢集中安裝的初始 Cilium 版本。

   在執行 helm 升級命令之前，請將部署的值保留在 中，`cilium-values.yaml`或為您的設定使用`--set`命令列選項。升級操作會覆寫 Cilium ConfigMap，因此請務必在升級時傳遞您的組態值。如果您使用的是 BGP，我們建議您使用`--set bgpControlPlane=true`命令列選項，而不是在值檔案中提供此資訊。

```
helm upgrade cilium cilium/cilium --version CILIUM_VERSION \
  --namespace kube-system \
  --set upgradeCompatibility=1.X \
  -f cilium-values.yaml
```

8. （選用）如果您因為問題而需要復原升級，請執行下列命令。

```
helm history cilium --namespace kube-system
```

```
helm rollback cilium [REVISION] --namespace kube-system
```

從混合節點刪除 Cilium

1. 執行下列命令，從叢集解除安裝所有 Cilium 元件。請注意，解除安裝 CNI 可能會影響節點和 Pod 的運作狀態，不應在生產叢集上執行。

   ```
   helm uninstall cilium --namespace kube-system
   ```

   從叢集移除 CNI 時，預設不會移除 Cilium 設定的介面和路由，如需詳細資訊，請參閱 GitHub 問題。
2. 若要清除磁碟上組態檔案和資源，如果您使用標準組態目錄，您可以移除 GitHub 上 Cilium 儲存庫中的cni-uninstall.sh指令碼所示的檔案。
3. 若要從叢集中移除 Cilium 自訂資源定義 (CRDs)，您可以執行下列命令。

   ```
   kubectl get crds -oname | grep "cilium" | xargs kubectl delete
   ```

Calico 考量事項

- 我們建議您使用 VXLAN 作為封裝方法，在覆蓋/通道模式下執行 Calico。此模式對基礎實體網路的需求最少。如需不同 Calico 聯網模式的詳細資訊，請參閱 Calico 文件中的判斷最佳聯網選項。
- 我們建議您執行 Calico，並將 `natOutgoing`設定為 `true`。將 `natOutgoing`設為 時`true`，所有離開叢集之 Pod 流量的來源 IP 地址會轉譯為節點的 IP 地址。這可讓您使用 Amazon EKS 叢集執行 Calico，無論叢集上是否已設定遠端 Pod 網路。如果您停用 `natOutgoing`，則您的 Pod CIDRs 必須在內部部署網路上可路由，而且您必須使用遠端 Pod 網路設定 Amazon EKS 叢集。
- 如果您在混合節點上執行 Webhook，您的 Pod CIDRs 必須在內部部署網路上可路由，而且您必須使用遠端 Pod 網路設定 Amazon EKS 叢集。如果您的 Pod CIDRs 在內部部署網路上無法路由，建議您在相同叢集的雲端節點上執行 Webhook。如需詳細資訊，請參閱設定混合節點的 Webhook。
- 讓 Pod CIDR 在內部部署網路上可路由的常見方法是使用 BGP 公告 Pod 地址。若要搭配 Calico 使用 BGP，您必須在 Helm 組態`installation.calicoNetwork.bgp: Enabled`中設定 。如需 Calico BGP 支援的詳細資訊，請參閱 Calico 文件中的設定 BGP 對等互連。
- Calico 中的預設 IP 地址管理 (IPAM) 稱為 Calico IPAM，其中`calico-ipam`外掛程式會根據使用者設定的 Pod CIDRs 為每個節點配置 IP 地址。Pod CIDRs 是以 `installation.calicoNetwork.ipPools.cidr` Helm 值設定，這應該符合您為 Amazon EKS 叢集設定的遠端 Pod 網路 CIDRs。Calico 會將區段從 配置`ipPools.cidr`給每個節點。每個節點

區段的大小是以 ipPools.blockSize Helm 值設定。如需使用 Calico 進行 IPAM 的詳細資訊，請參閱 Calico 文件中的 IP 地址管理入門。

在混合節點上安裝 Calico

1. 請確定您已在命令列環境上安裝 helm CLI。如需安裝說明，請參閱 Helm 文件。
2. 安裝 Cilium Helm 儲存庫。

```
helm repo add projectcalico https://docs.tigera.io/calico/charts
```

3. 建立名為 的 YAML 檔案calico-values.yaml。下列範例會設定eks.amazonaws.com/compute-type: hybrid標籤的親和性，將所有 Calico 元件設定為僅在混合節點上執行。
   - POD_CIDR 將 取代為 Pod 的 CIDR 範圍。如果您使用遠端 Pod 網路設定 Amazon EKS 叢集，POD_CIDR您為 Calico 指定的 應與遠端 Pod 網路相同。例如 10.100.0.0/24。在覆蓋/通道模式下執行 CNI 時，您的內部部署 Pod CIDR 不得與內部部署節點 CIDR 重疊。
   - CIDR_SIZE 將 取代為您要配置給每個節點的 CIDR 區段大小。例如，25對於每個節點 128 個 Pod 地址的 /25 區段大小。如需 CIDR blockSize和變更 的詳細資訊blockSize，請參閱 Calico 文件中的變更 IP 集區區塊大小。
   - 在下面的範例中， natOutgoing 已啟用且bgp已停用。根據您的目標組態修改這些值。

```
installation:
  enabled: true
  cni:
    type: Calico
    ipam:
      type: Calico
  calicoNetwork:
    bgp: Disabled
    ipPools:
    - cidr: POD_CIDR
      blockSize: CIDR_SIZE
      encapsulation: VXLAN
      natOutgoing: Enabled
      nodeSelector: eks.amazonaws.com/compute-type == "hybrid"
  controlPlaneReplicas: 1
  controlPlaneNodeSelector:
    eks.amazonaws.com/compute-type: hybrid
  calicoNodeDaemonSet:
    spec:
```

```
    template:
      spec:
        nodeSelector:
          eks.amazonaws.com/compute-type: hybrid
csiNodeDriverDaemonSet:
  spec:
    template:
      spec:
        nodeSelector:
          eks.amazonaws.com/compute-type: hybrid
calicoKubeControllersDeployment:
  spec:
    template:
      spec:
        nodeSelector:
          eks.amazonaws.com/compute-type: hybrid
typhaDeployment:
  spec:
    template:
      spec:
        nodeSelector:
          eks.amazonaws.com/compute-type: hybrid
```

4. 在叢集上安裝 Calico。
   - CALICO_VERSION 將 取代為 Calico 版本 （ 例如 v3.29.0)，請參閱 Calico 版本以尋找 Calico 次要版本的最新修補程式版本。我們建議您執行 Calico 次要版本的最新修補程式版本。
   - 如果您使用的是特定kubeconfig檔案，請使用 --kubeconfig旗標。

```
helm install calico projectcalico/tigera-operator \
    --version CALICO_VERSION \
    --namespace kube-system \
    -f calico-values.yaml
```

5. 您可以使用下列命令確認您的 Calico 安裝成功。您應該會看到tigera-operator部署、在每個混合節點上執行的calico-node代理程式、 csi-node-driver、 calico-apiserver和 calico-kube-controllers 部署。此外，您的混合節點現在應具有狀態 Ready。如果您使用的是 natOutgoing: Disabled，則在您向內部部署網路公告 Pod 地址之前，所有 Calico 元件都無法成功啟動。如需如何為 Calico 設定 BGP 的資訊，請繼續下一個步驟。

```
kubectl get pods -A
```

```
NAMESPACE          NAME                                       READY   STATUS
 RESTARTS   AGE
calico-apiserver   calico-apiserver-6c77bb6d46-2n8mq          1/1     Running   0
      69s
calico-system      calico-kube-controllers-7c5f8556b5-7h267   1/1     Running   0
      68s
calico-system      calico-node-s5nnk                          1/1     Running   0
      68s
calico-system      calico-typha-6487cc9d8c-wc5jm              1/1     Running   0
      69s
calico-system      csi-node-driver-cv42d                      2/2     Running   0
      68s
kube-system        coredns-7bb495d866-2lc9v                   1/1     Running   0
      6m27s
kube-system        coredns-7bb495d866-2t8ln                   1/1     Running   0
      157m
kube-system        kube-proxy-lxzxh                           1/1     Running   0
      18m
kube-system        tigera-operator-f8bc97d4c-28b4d            1/1     Running   0
      90s
```

```
kubectl get nodes
```

```
NAME                  STATUS   ROLES    AGE    VERSION
mi-0c6ec2f6f79176565  Ready    <none>   5h13m  v1.31.0-eks-a737599
```

6. 如果您安裝的 Calico 沒有 BGP，請略過此步驟。若要設定 BGP，calico-bgp.yaml請使用BGPPeer組態和 建立名為 的檔案BGPConfiguration。請務必區分 BGPPeer和 BGPConfiguration。BGPPeer 是啟用 BGP 的路由器或遠端資源，Calico 叢集中的節點將與其對等。BGPPeer 組態asNumber中的 類似於 Cilium 設定 peerASN 。BGPConfiguration 會套用至每個 Calico 節點，而 asNumber 的 BGPConfiguration 相當於 Cilium 設定 localASN。將以下範例中LOCAL_ASN的 ONPREM_ROUTER_IPONPREM_ROUTER_ASN、 和 取代為內部部署環境的值，您可能必須從網路管理員取得這些值。此keepOriginalNextHop: true設定用於確保每個節點僅公告其擁有的 Pod 網路 CIDR。

```
apiVersion: projectcalico.org/v3
kind: BGPPeer
metadata:
  name: calico-hybrid-nodes
```

```
spec:
  peerIP: ONPREM_ROUTER_IP
  asNumber: ONPREM_ROUTER_ASN
  keepOriginalNextHop: true
---
apiVersion: projectcalico.org/v3
kind: BGPConfiguration
metadata:
  name: default
spec:
  nodeToNodeMeshEnabled: false
  asNumber: LOCAL_ASN
```

7. 將 檔案套用至您的叢集。

```
kubectl apply -f calico-bgp.yaml
```

8. 使用下列命令確認 Calico Pod 正在執行。

```
kubectl get pods -n calico-system -w
```

```
NAMESPACE          NAME                                       READY   STATUS
 RESTARTS       AGE
calico-apiserver   calico-apiserver-598bf99b6c-2vltk          1/1     Running   0
          3h24m
calico-system      calico-kube-controllers-75f84bbfd6-zwmnx   1/1     Running   31
 (59m ago)   3h20m
calico-system      calico-node-9b2pg                          1/1     Running   0
          5h17m
calico-system      calico-typha-7d55c76584-kxtnq              1/1     Running   0
          5h18m
calico-system      csi-node-driver-dmnmm                      2/2     Running   0
          5h18m
kube-system        coredns-7bb495d866-dtn4z                   1/1     Running   0
          6h23m
kube-system        coredns-7bb495d866-mk7j4                   1/1     Running   0
          6h19m
kube-system        kube-proxy-vms28                           1/1     Running   0
          6h12m
kube-system        tigera-operator-55f9d9d565-jj9bg           1/1     Running   0
          73m
```

如果您在這些步驟期間遇到問題，請參閱 Calico 文件中的疑難排解指引。

升級混合節點上的 Calico

升級 Calico 部署之前，請仔細檢閱 Calico 升級文件和版本備註，以了解目標 Calico 版本中的變更。升級步驟會根據您是否使用 Helm、Calico 運算子和資料存放區的類型而有所不同。下列步驟假設使用 Helm。

1. 下載您要升級之 Calico 版本的運算子資訊清單。CALICO_VERSION 以您要升級到的版本取代 ，例如 v3.29.0。請務必v在 major.minor.patch 前面加上 。

```
kubectl apply --server-side --force-conflicts \
    -f https://raw.githubusercontent.com/projectcalico/calico/CALICO_VERSION/
manifests/operator-crds.yaml
```

2. 執行 helm upgrade以升級您的 Calico 部署。CALICO_VERSION 以您要升級到的版本取代 ，例如 v3.29.0。從您用來安裝 Calico 的組態值建立 calico-values.yaml 檔案。

```
helm upgrade calico projectcalico/tigera-operator \
    --version CALICO_VERSION \
    --namespace kube-system \
    -f calico-values.yaml
```

從混合節點刪除 Calico

1. 執行下列命令，從您的叢集解除安裝 Calico 元件。請注意，解除安裝 CNI 可能會影響節點和 Pod 的運作狀態，不應在生產叢集上執行。如果您在以下命令中kube-system變更命名空間以外的命名空間中安裝 Calico。

```
helm uninstall calico --namespace kube-system
```

   請注意，從叢集移除 CNI 時，預設不會移除 Calico 設定的介面和路由。
2. 若要清除磁碟上組態檔案和資源，請從 /opt/cni和 /etc/cni目錄中移除 Calico 檔案。
3. 若要從叢集中移除 Calico CRDs，請執行下列命令。

```
kubectl get crds -oname | grep "calico" | xargs kubectl delete
```

```
kubectl get crds -oname | grep "tigera" | xargs kubectl delete
```

設定混合節點的附加元件

此頁面說明在 Amazon EKS 混合節點上執行 AWS 附加元件和社群附加元件的考量。若要進一步了解 Amazon EKS 附加元件，以及從叢集建立、升級和移除附加元件的程序，請參閱 the section called "Amazon EKS 附加元件"。除非此頁面另有說明，否則對於具有混合節點的 Amazon EKS 叢集，建立、升級和移除 Amazon EKS 附加元件的程序與在 AWS Cloud 中執行節點的 Amazon EKS 叢集相同。只有此頁面中包含的附加元件已驗證與 Amazon EKS 混合節點的相容性。

下列 AWS 附加元件與 Amazon EKS 混合節點相容。

AWS 附加元件	相容附加元件版本
kube-proxy	v1.25.14-eksbuild.2 及更高版本
CoreDNS	v1.9.3-eksbuild.7 及更高版本
AWS Distro for OpenTelemetry (ADOT)	v0.102.1-eksbuild.2 及更高版本
CloudWatch 可觀測性代理程式	v2.2.1-eksbuild.1 及更高版本
EKS Pod 身分識別代理程式	• v1.3.3-eksbuild.1 及更高版本， Bottlerocket 除外 • Bottlerocket 的 v1.3.7-eksbuild.2 及更高版本
節點監控代理程式	v1.2.0-eksbuild.1 及更高版本
CSI 快照控制器	v8.1.0-eksbuild.1 及更高版本
AWS Kubernetes 專用私有 CA 連接器	v1.6.0-eksbuild.1 及更高版本

下列社群附加元件與 Amazon EKS 混合節點相容。若要進一步了解社群附加元件，請參閱 the section called "社群附加元件"。

社群附加元件	相容附加元件版本
Kubernetes 指標伺服器	v0.7.2-eksbuild.1 及更高版本
cert-manager	v1.17.2-eksbuild.1 及更高版本

社群附加元件	相容附加元件版本
Prometheus Node Exporter	v1.9.1-eksbuild.2 及更高版本
kube-state-metrics	v2.15.0-eksbuild.4 及更高版本

除了上表中的 Amazon EKS 附加元件之外，Amazon Managed Service for Prometheus Collector 和適用於應用程式輸入 (HTTP) 和負載平衡 (TCP/UDP) 的AWS Load Balancer控制器也與混合節點相容。

有些 AWS 附加元件和社群附加元件與 Amazon EKS 混合節點不相容。這些附加元件的最新版本具有套用至混合節點的預設`eks.amazonaws.com/compute-type: hybrid`標籤的反親和性規則。這可防止它們在部署在叢集中的混合節點上執行。如果您有在 AWS Cloud 中同時執行混合節點和節點的叢集，您可以將叢集中的這些附加元件部署到在 AWS Cloud 中執行的節點。Amazon VPC CNI 與混合節點不相容，並支援 Cilium 和 Calico 做為 Amazon EKS 混合節點的容器網路界面 (CNIs)。如需詳細資訊，請參閱the section called “設定 CNI”。

AWS 附加元件

以下各節說明在混合節點上執行相容 AWS 附加元件與其他 Amazon EKS 運算類型之間的差異。

kube-proxy 和 CoreDNS

當您使用 AWS API 和 AWS SDKs 建立 EKS 叢集時，EKS 預設會將 kube-proxy 和 CoreDNS 安裝為自我管理附加元件，包括來自 AWS CLI 的 。您可以在建立叢集後，使用 Amazon EKS 附加元件覆寫這些附加元件。如需 the section called “kube-proxy”和 的詳細資訊，請參閱 EKS 文件the section called “CoreDNS”。如果您在 AWS Cloud 中同時執行混合模式叢集與混合節點和節點，建議您在混合節點上至少有一個 CoreDNS 複本，並在 AWS Cloud 中的節點上至少有一個 CoreDNS 複本。如需組態步驟the section called “設定 CoreDNS 複本”，請參閱 。

CloudWatch 可觀測性代理程式

CloudWatch 可觀測性代理程式運算子使用 Webhook。如果您在混合節點上執行 運算子，您的內部部署 Pod CIDR 必須在內部部署網路上可路由，而且您必須使用遠端 Pod 網路設定 EKS 叢集。如需詳細資訊，請參閱設定混合節點的 Webhook。

節點層級指標不適用於混合節點，因為 CloudWatch Container Insights 取決於節點層級指標的執行個體中繼資料服務 (IMDS) 可用性。叢集、工作負載、Pod 和容器層級指標可用於混合節點。

安裝附加元件後，請依照使用 Amazon CloudWatch Observability 安裝 CloudWatch 代理 Amazon CloudWatch程式中所述的步驟，必須先更新附加元件資訊清單，代理程式才能在混合節點上成功執行。編輯叢集上的 amazoncloudwatchagents 資源以新增RUN_WITH_IRSA環境變數，如下所示。

```
kubectl edit amazoncloudwatchagents -n amazon-cloudwatch cloudwatch-agent
```

```
apiVersion: v1
items:
- apiVersion: cloudwatch.aws.amazon.com/v1alpha1
  kind: AmazonCloudWatchAgent
  metadata:
    ...
    name: cloudwatch-agent
    namespace: amazon-cloudwatch
    ...
  spec:
    ...
    env:
    - name: RUN_WITH_IRSA # <-- Add this
      value: "True" # <-- Add this
    - name: K8S_NODE_NAME
      valueFrom:
        fieldRef:
          fieldPath: spec.nodeName
          ...
```

適用於混合節點的 Amazon Managed Prometheus 受管收集器

Amazon Managed Service for Prometheus (AMP) 受管收集器包含一個抓取器，可從 Amazon EKS 叢集中的資源探索和收集指標。AMP 會為您管理抓取器，無需自行管理任何執行個體、代理程式或抓取器。

您可以使用 AMP 受管收集器，而不需要混合節點特有的任何其他組態。不過，混合節點上的應用程式指標端點必須可從 VPC 連線，包括從 VPC 到遠端 Pod 網路 CIDRs路由，以及內部部署防火牆中開啟的連接埠。此外，您的叢集必須具有私有叢集端點存取權。

請遵循《Amazon Managed Service for Prometheus 使用者指南》中的使用 AWS 受管收集器中的步驟。

AWS Distro for OpenTelemetry (ADOT)

您可以使用 AWS Distro for OpenTelemetry (ADOT) 附加元件，從混合節點上執行的應用程式收集指標、日誌和追蹤資料。ADOT 使用許可 Webhook 來變更和驗證收集器自訂資源請求。如果您在混合節點上執行 ADOT 運算子，您的內部部署 Pod CIDR 必須在內部部署網路上可路由，而且您必須使用遠端 Pod 網路設定 EKS 叢集。如需詳細資訊，請參閱設定混合節點的 Webhook。

請遵循 AWS Distro for OpenTelemetry 文件中的使用 EKS 附加元件的 Distro for OpenTelemetry 入門中的步驟。 AWS OpenTelemetry

AWS Load Balancer控制器

您可以使用AWS Load Balancer控制器和 Application Load Balancer (ALB) 或 Network Load Balancer (NLB) 搭配與 AWS Direct Connect 或 AWS Site-to-Site VPN 連接之混合節點上的工作負載目標類型 ip。與 ALB 或 NLB 搭配使用的 IP 目標 （必須是可路由的） AWS。The AWS Load Balancer 控制器也使用 Webhook。如果您在混合節點上執行 AWS Load Balancer控制器運算子，您的內部部署 Pod CIDR 必須可在內部部署網路上路由，而且您必須使用遠端 Pod 網路設定 EKS 叢集。如需詳細資訊，請參閱設定混合節點的 Webhook。

若要安裝 AWS Load Balancer控制器，請遵循 the section called “使用 Helm 安裝”或 中的步驟the section called “使用資訊清單安裝”。

對於使用 ALB 的輸入，您必須指定以下註釋。如需說明，請參閱 the section called “應用程式負載平衡”。

```
alb.ingress.kubernetes.io/target-type: ip
```

若要使用 NLB 進行負載平衡，您必須指定下列註釋。如需說明，請參閱 the section called “網路負載平衡”。

```
service.beta.kubernetes.io/aws-load-balancer-type: "external"
service.beta.kubernetes.io/aws-load-balancer-nlb-target-type: "ip"
```

EKS Pod 身分識別代理程式

> **Note**
>
> 若要在執行 Bottlerocket 的混合節點上成功部署 EKS Pod Identity Agent 附加元件，請確定您的 Bottlerocket 版本至少為 v1.39.0。在混合節點環境中，舊版 Bottlerocket 不支援 Pod Identity Agent。

原始 Amazon EKS Pod Identity Agent DaemonSet 依賴節點上 EC2 IMDS 的可用性來取得所需的 AWS 登入資料。由於 IMDS 不適用於混合節點，因此從 1.3.3-eksbuild.1 版開始，Pod Identity Agent 附加元件可選擇地部署掛載所需登入資料的 DaemonSet。執行 Bottlerocket 的混合節點需要不同的方法來掛載憑證，並從 1.3.7-eksbuild.2 版開始，Pod Identity Agent 附加元件可選擇部署專門以 Bottlerocket 混合節點為目標的 DaemonSet。下列各節說明啟用選用 DaemonSets 的程序。

Ubuntu/RHEL/AL2023

1. 若要在 Ubuntu/RHEL/Al2023 混合節點上使用 Pod Identity 代理程式，請在組態的混合區段`enableCredentialsFile: true`中設定 `nodeadm` ，如下所示：

   ```
   apiVersion: node.eks.aws/v1alpha1
   kind: NodeConfig
   spec:
       hybrid:
           enableCredentialsFile: true # <-- Add this
   ```

   這將`nodeadm`設定 在 下的節點上建立要設定的登入資料檔案`/eks-hybrid/.aws/credentials`，以供 `eks-pod-identity-agent` Pod 使用。此登入資料檔案將包含將定期重新整理的臨時 AWS 登入資料。
2. 更新每個節點上的組態後，請使用 `nodeadm` 執行下列`nodeadm init`命令`nodeConfig.yaml`，將混合節點加入 Amazon EKS 叢集。如果您的節點先前已加入叢集，仍請再次執行 `init`命令。

   ```
   nodeadm init -c file://nodeConfig.yaml
   ```
3. 使用 CLI 或 ，`eks-pod-identity-agent`在啟用混合節點支援的情況下安裝 AWS AWS Management Console。
   a. AWS CLI：從您用來管理叢集的機器中，執行下列命令來安裝 `eks-pod-identity-agent` ，並啟用混合節點的支援。使用您叢集的名稱取代 `my-cluster`。

      ```
      aws eks create-addon \
          --cluster-name my-cluster \
          --addon-name eks-pod-identity-agent \
          --configuration-values '{"daemonsets":{"hybrid":{"create": true}}}'
      ```

   b. AWS Management Console：如果您透過 AWS 主控台安裝 Pod Identity Agent 附加元件，請將下列項目新增至選用組態，以部署以混合節點為目標的 DaemonSet。

      ```
      {"daemonsets":{"hybrid":{"create": true}}}
      ```

Bottlerocket

1. 若要在 Bottlerocket 混合節點上使用 Pod Identity 代理程式，請將 `--enable-credentials-file=true`旗標新增至用於 Bottlerocket 引導容器使用者資料的命令，如中所述the section called “使用 Bottlerocket 連接混合節點”。

   a. 如果您使用的是 SSM 登入資料提供者，您的命令應該如下所示：

   ```
   eks-hybrid-ssm-setup --activation-id=<activation-id> --activation-
   code=<activation-code> --region=<region> --enable-credentials-file=true
   ```

   b. 如果您使用的是 IAM Roles Anywhere 登入資料提供者，您的命令應該如下所示：

   ```
   eks-hybrid-iam-ra-setup --certificate=<certificate> --key=<private-key> --enable-
   credentials-file=true
   ```

   這會將引導指令碼設定為在 下的節點上建立登入資料檔案`/var/eks-hybrid/.aws/credentials`，以供 `eks-pod-identity-agent` Pod 使用。此登入資料檔案將包含將定期重新整理的臨時 AWS 登入資料。

2. 使用 CLI 或 ，`eks-pod-identity-agent`在已啟用 Bottlerocket AWS 混合節點的支援下安裝 AWS Management Console。

   a. AWS CLI：從您用來管理叢集的機器中，執行下列命令來安裝 `eks-pod-identity-agent`，並啟用 Bottlerocket 混合節點的支援。使用您叢集的名稱取代 `my-cluster`。

   ```
   aws eks create-addon \
       --cluster-name my-cluster \
       --addon-name eks-pod-identity-agent \
       --configuration-values '{"daemonsets":{"hybrid-bottlerocket":{"create":
    true}}}'
   ```

   b. AWS Management Console：如果您透過 AWS 主控台安裝 Pod Identity Agent 附加元件，請將下列項目新增至選用組態，以部署以 Bottlerocket 混合節點為目標的 DaemonSet。

   ```
   {"daemonsets":{"hybrid-bottlerocket":{"create": true}}}
   ```

CSI 快照控制器

從版本 開始`v8.1.0-eksbuild.2`，CSI 快照控制器附加元件會套用混合節點的軟性反親和性規則，偏好控制器在與 Amazon EKS 控制平面相同的 AWS 區域中於 EC2 `deployment`上執行。在`deployment`與 Amazon EKS 控制平面相同的 AWS 區域中聯合放置 可改善延遲。

社群附加元件

以下各節說明在混合節點上執行相容社群附加元件與其他 Amazon EKS 運算類型之間的差異。

Kubernetes 指標伺服器

控制平面需要達到指標伺服器的 Pod IP （如果啟用 hostNetwork，則為節點 IP)。因此，除非您在 hostNetwork 模式下執行 Metrics Server，否則您必須在建立 Amazon EKS 叢集時設定遠端 Pod 網路，而且必須將 Pod IP 地址設為可路由。使用 CNI 實作邊界閘道協定 (BGP) 是讓您的 Pod IP 地址可路由的常見方式之一。

cert-manager

`cert-manager` 使用 Webhook。如果您在混合節點`cert-manager`上執行 ，您的內部部署 Pod CIDR 必須在內部部署網路上可路由，而且您必須使用遠端 Pod 網路設定 EKS 叢集。如需詳細資訊，請參閱設定混合節點的 Webhook。

設定混合節點的 Webhook

此頁面詳細說明使用混合節點執行 Webhook 的考量事項。Webhook 用於 Kubernetes 應用程式和開放原始碼專案，例如 AWS Load Balancer控制器和 CloudWatch 可觀測性代理程式，以在執行時間執行變動和驗證功能。

可路由 Pod 網路

如果您能夠在內部部署網路上讓內部部署 Pod CIDR 可路由，則可以在混合節點上執行 Webhook。您可以使用數種技術，讓您的內部部署 Pod CIDR 可在內部部署網路上路由，包括邊界閘道協定 (BGP)、靜態路由或其他自訂路由解決方案。BGP 是建議的解決方案，因為它比需要自訂或手動路由組態的替代解決方案更具可擴展性且更容易管理。 AWS 支援 Cilium 和 Calico 的 BGP 功能來公告 Pod CIDRs，如需詳細資訊，請參閱 the section called “可路由遠端 Pod CIDRs” the section called “設定 CNI”和 。

無法路由的 Pod 網路

如果您無法在內部部署網路上讓內部部署 Pod CIDR 可路由，且需要執行 Webhook，建議您在與混合節點相同的 EKS 叢集中的雲端節點上執行所有 Webhook。

混合模式叢集的考量事項

混合模式叢集定義為 EKS 叢集，其具有在 AWS Cloud 中執行的混合節點和節點。執行混合模式叢集時，請考慮下列建議：

- 在 AWS 雲端的節點上執行 VPC CNI，在混合節點上執行 Cilium 或 Calico。在 AWS 雲端節點上執行 AWS 時， 不支援 Cilium 和 Calico。
- 設定 Webhook 在 AWS 雲端的節點上執行。如需如何設定 AWS 和 社群附加元件的 Webhook，the section called “設定附加元件的 Webhook”請參閱 。
- 如果您的應用程式需要在 AWS Cloud 節點上執行的 Pod 直接與在混合節點上執行的 Pod 通訊 (「東西通訊」)，而且您在 AWS Cloud 節點上使用 VPC CNI，在混合節點上使用 Cilium 或 Calico，則您的內部部署 Pod CIDR 必須可在內部部署網路上路由。
- 在 AWS 雲端的節點上執行至少一個 CoreDNS 複本，並在混合節點上執行至少一個 CoreDNS 複本。
- 設定服務流量分佈，將服務流量保持在其來源區域的本機。如需服務流量分佈的詳細資訊，請參閱 the section called “設定服務流量分佈”。
- 如果您將 AWS Application Load Balancer (ALB) 或 Network Load Balancer (NLB) 用於在混合節點上執行的工作負載流量，則與 ALB 或 NLB 搭配使用的 IP 目標必須是可路由的 AWS。
- Metrics Server 附加元件需要從 EKS 控制平面連線至 Metrics Server Pod IP 地址。如果您在混合節點上執行 Metrics Server 附加元件，則您的內部部署 Pod CIDR 必須可在內部部署網路上路由。
- 若要使用 Amazon Managed Service for Prometheus (AMP) 受管收集器收集混合節點的指標，您的內部部署 Pod CIDR 必須可在內部部署網路上路由。或者，您可以將 AMP 受管收集器用於在 AWS 雲端中執行的 EKS 控制平面指標和資源，以及 AWS Distro for OpenTelemetry (ADOT) 附加元件，以收集混合節點的指標。

設定混合模式叢集

若要檢視叢集上執行的變動和驗證 Webhook，您可以在叢集的 EKS 主控台的資源面板中檢視延伸模組資源類型，也可以使用下列命令。EKS 也會在叢集可觀測性儀表板中報告 Webhook 指標，the section called “可觀測性儀表板”如需詳細資訊，請參閱 。

```
kubectl get mutatingwebhookconfigurations
```

```
kubectl get validatingwebhookconfigurations
```

設定服務流量分佈

執行混合模式叢集時，建議您使用 Service Traffic Distribution 將服務流量保留在其來源區域的本機。服務流量分佈（適用於 Kubernetes 版本 1.31 及更新版本 EKS) 是拓撲感知路由的建議解決方案，因為它更可預測。透過服務流量分佈，區域中運作狀態良好的端點將接收該區域的所有流量。透過拓撲感知路由，每個服務必須符合該區域中的數個條件，才能套用自訂路由，否則會將流量平均路由至所有端點。

如果您使用 Cilium 做為 CNI，您必須執行 CNI，並將 `enable-service-topology`設定為 `true`以啟用服務流量分佈。您可以使用 Helm 安裝旗標傳遞此組態，`--set loadBalancer.serviceTopology=true`也可以使用 Cilium CLI 命令 更新現有的安裝`cilium config set enable-service-topology true`。在每個節點上執行的 Cilium 代理程式必須在更新現有安裝的組態之後重新啟動。

下節顯示如何設定 CoreDNS 服務的服務流量分佈的範例，建議您為叢集中的所有服務啟用相同的 ，以避免意外的跨環境流量。

設定 CoreDNS 複本

執行混合模式叢集時，我們建議您在混合節點上至少有一個 CoreDNS 複本，以及在 AWS 雲端節點上至少有一個 CoreDNS 複本。

1. 為每個混合節點新增拓撲區域標籤，例如 `topology.kubernetes.io/zone: onprem`。或者，您可以透過在`nodeadm`組態中指定標籤，在 `nodeadm init`階段設定標籤，請參閱 the section called “用於自訂 kubelet 的節點組態（選用）”。請注意，在 AWS Cloud 中執行的節點會自動取得套用至它們的拓撲區域標籤，其對應至節點的可用區域 (AZ)。

   ```
   kubectl label node hybrid-node-name topology.kubernetes.io/zone=zone
   ```

2. 使用拓撲區域金鑰`podAntiAffinity`新增至 CoreDNS 部署。或者，您可以使用 EKS 附加元件在安裝期間設定 CoreDNS 部署。

   ```
   kubectl edit deployment coredns -n kube-system
   ```

   ```
   spec:
     template:
       spec:
         affinity:
           ...
   ```

```
        podAntiAffinity:
          preferredDuringSchedulingIgnoredDuringExecution:
          - podAffinityTerm:
              labelSelector:
                matchExpressions:
                - key: k8s-app
                  operator: In
                  values:
                  - kube-dns
              topologyKey: kubernetes.io/hostname
            weight: 100
          - podAffinityTerm:
              labelSelector:
                matchExpressions:
                - key: k8s-app
                  operator: In
                  values:
                  - kube-dns
              topologyKey: topology.kubernetes.io/zone
            weight: 50
      ...
```

3. 將 設定trafficDistribution: PreferClose新增至kube-dns服務組態，以啟用服務流量分佈。

```
kubectl patch svc kube-dns -n kube-system --type=merge -p '{
  "spec": {
    "trafficDistribution": "PreferClose"
  }
}'
```

4. 您可以檢視服務的端點配量，確認已啟用kube-dns服務流量分佈。您的端點配量必須顯示拓撲區域標籤hints的 ，以確認服務流量分佈已啟用。如果您沒有看到每個端點地址hints的 ，則不會啟用服務流量分佈。

```
kubectl get endpointslice -A | grep "kube-dns"
```

```
kubectl get endpointslice [.replaceable]`kube-dns-<id>`  -n kube-system -o yaml
```

```
addressType: IPv4
apiVersion: discovery.k8s.io/v1
```

```
endpoints:
- addresses:
  - <your-hybrid-node-pod-ip>
  hints:
    forZones:
    - name: onprem
  nodeName: <your-hybrid-node-name>
  zone: onprem
- addresses:
  - <your-cloud-node-pod-ip>
  hints:
    forZones:
    - name: us-west-2a
  nodeName: <your-cloud-node-name>
  zone: us-west-2a
```

設定附加元件的 Webhook

下列附加元件使用 Webhook，並支援與混合節點搭配使用。

- AWS Load Balancer控制器
- CloudWatch 可觀測性代理程式
- AWS Distro for OpenTelemetry (ADOT)
- `cert-manager`

請參閱下列各節，以設定這些附加元件所使用的 Webhook 在 AWS Cloud 中的節點上執行。

AWS Load Balancer控制器

若要在混合模式叢集設定中使用 AWS Load Balancer控制器，您必須在 AWS 雲端的節點上執行控制器。若要這樣做，請將以下內容新增至 Helm 值組態，或使用 EKS 附加元件組態指定值。

```
affinity:
  nodeAffinity:
    requiredDuringSchedulingIgnoredDuringExecution:
      nodeSelectorTerms:
      - matchExpressions:
        - key: eks.amazonaws.com/compute-type
          operator: NotIn
          values:
```

```
          - hybrid
```

CloudWatch 可觀測性代理程式

CloudWatch 可觀測性代理程式附加元件具有使用 Webhook 的 Kubernetes Operator。若要在混合模式叢集設定中對 AWS Cloud 中的節點執行運算子，請編輯 CloudWatch Observability Agent Operator 組態。您無法在使用 Helm 和 EKS 附加元件安裝期間設定運算子親和性（請參閱 Container-roadmap 問題 #2431）。

```
kubectl edit -n amazon-cloudwatch deployment amazon-cloudwatch-observability-
controller-manager
```

```
spec:
  ...
  template:
    ...
    spec:
      affinity:
        nodeAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
            nodeSelectorTerms:
            - matchExpressions:
              - key: eks.amazonaws.com/compute-type
                operator: NotIn
                values:
                - hybrid
```

AWS Distro for OpenTelemetry (ADOT)

AWS Distro for OpenTelemetry (ADOT) 附加元件具有使用 Webhook 的 Kubernetes Operator。若要在混合模式叢集設定中於 AWS 雲端節點上執行運算子，請將下列項目新增至 Helm 值組態，或使用 EKS 附加元件組態指定值。

```
affinity:
  nodeAffinity:
    requiredDuringSchedulingIgnoredDuringExecution:
      nodeSelectorTerms:
      - matchExpressions:
        - key: eks.amazonaws.com/compute-type
          operator: NotIn
          values:
```

```
          - hybrid
```

如果您的 Pod CIDR 無法在內部部署網路上路由，則 ADOT 收集器必須在混合節點上執行，才能從混合節點中抓取指標，以及在混合節點上執行的工作負載。若要這樣做，請編輯自訂資源定義 (CRD)。

```
kubectl -n opentelemetry-operator-system edit opentelemetrycollectors.opentelemetry.io
 adot-col-prom-metrics
```

```
spec:
  affinity:
    nodeAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        nodeSelectorTerms:
        - matchExpressions:
          - key: eks.amazonaws.com/compute-type
            operator: In
            values:
            - hybrid
```

您可以將以下項目新增至 ADOT 收集器 CRD 組態`scrape_configs`中的`relabel_configs`每個項目，將 ADOT 收集器設定為僅從混合節點和混合節點上執行的資源抓取指標。

```
relabel_configs:
  - action: keep
    regex: hybrid
    source_labels:
    - __meta_kubernetes_node_label_eks_amazonaws_com_compute_type
```

ADOT 附加元件需要`cert-manager`為 ADOT 運算子 Webhook 所使用的 TLS 憑證安裝 。 `cert-manager`也會執行 Webhook，而且您可以使用下列 Helm 值組態，將其設定為在 AWS Cloud 中的節點上執行。

```
affinity:
  nodeAffinity:
    requiredDuringSchedulingIgnoredDuringExecution:
      nodeSelectorTerms:
      - matchExpressions:
        - key: eks.amazonaws.com/compute-type
          operator: NotIn
          values:
```

```
          - hybrid
webhook:
  affinity:
    nodeAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        nodeSelectorTerms:
        - matchExpressions:
          - key: eks.amazonaws.com/compute-type
            operator: NotIn
            values:
            - hybrid
cainjector:
  affinity:
    nodeAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        nodeSelectorTerms:
        - matchExpressions:
          - key: eks.amazonaws.com/compute-type
            operator: NotIn
            values:
            - hybrid
startupapicheck:
  affinity:
    nodeAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        nodeSelectorTerms:
        - matchExpressions:
          - key: eks.amazonaws.com/compute-type
            operator: NotIn
            values:
            - hybrid
```

cert-manager

附加cert-manager元件會執行 Webhook，您可以使用下列 Helm 值組態，將其設定為在 AWS Cloud 中的節點上執行。

```
affinity:
  nodeAffinity:
    requiredDuringSchedulingIgnoredDuringExecution:
      nodeSelectorTerms:
      - matchExpressions:
        - key: eks.amazonaws.com/compute-type
```

```
          operator: NotIn
          values:
          - hybrid
webhook:
  affinity:
    nodeAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        nodeSelectorTerms:
        - matchExpressions:
          - key: eks.amazonaws.com/compute-type
            operator: NotIn
            values:
            - hybrid
cainjector:
  affinity:
    nodeAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        nodeSelectorTerms:
        - matchExpressions:
          - key: eks.amazonaws.com/compute-type
            operator: NotIn
            values:
            - hybrid
startupapicheck:
  affinity:
    nodeAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        nodeSelectorTerms:
        - matchExpressions:
          - key: eks.amazonaws.com/compute-type
            operator: NotIn
            values:
            - hybrid
```

設定混合節點的代理

如果您在內部部署環境中為離開資料中心或邊緣環境的流量使用代理伺服器，則需要單獨設定節點和叢集以使用您的代理伺服器。

叢集

在叢集上，您需要設定 kube-proxy以使用您的代理伺服器。您必須在建立 Amazon EKS 叢集kube-proxy後設定 。

節點

在節點上，您必須設定作業系統、kubelet、 和 Amazon SSM containerd代理程式，以使用您的代理伺服器。您可以在作業系統映像的建置程序期間或在每個混合節點nodeadm init上執行之前進行這些變更。

節點層級組態

您必須在作業系統映像中或在每個混合節點nodeadm init上執行之前套用下列組態。

containerd 代理組態

containerd 是 Kubernetes 的預設容器管理執行期。如果您使用代理進行網際網路存取，則必須設定 ，containerd才能提取 Kubernetes 和 Amazon EKS 所需的容器映像。

在http-proxy.conf/etc/systemd/system/containerd.service.d目錄中名為 的每個混合節點上建立檔案，內容如下。將 proxy-domain和 port 取代為您環境的值。

```
[Service]
Environment="HTTP_PROXY=http://proxy-domain:port"
Environment="HTTPS_PROXY=http://proxy-domain:port"
Environment="NO_PROXY=localhost"
```

containerd 來自使用者資料的 組態

需要為此檔案建立 containerd.service.d目錄。您需要重新載入系統化，才能在不重新啟動的情況下取得組態檔案。在 AL2023 中，當您的指令碼執行時，服務可能已經在執行，因此您也需要重新啟動它。

```
mkdir -p /etc/systemd/system/containerd.service.d
echo '[Service]' > /etc/systemd/system/containerd.service.d/http-proxy.conf
echo 'Environment="HTTP_PROXY=http://proxy-domain:port"' >> /etc/systemd/system/
containerd.service.d/http-proxy.conf
echo 'Environment="HTTPS_PROXY=http://proxy-domain:port"' >> /etc/systemd/system/
containerd.service.d/http-proxy.conf
echo 'Environment="NO_PROXY=localhost"' >> /etc/systemd/system/containerd.service.d/
http-proxy.conf
systemctl daemon-reload
systemctl restart containerd
```

kubelet 代理組態

kubelet 是在每個 Kubernetes 節點上執行的 Kubernetes 節點代理程式，負責管理其上執行的節點和 Pod。如果您在內部部署環境中使用代理，則必須設定 ，kubelet以便它可以與您 Amazon EKS 叢集的公有或私有端點通訊。

在http-proxy.conf/etc/systemd/system/kubelet.service.d/目錄中名為 的每個混合節點上建立檔案，內容如下。將 proxy-domain和 port 取代為您環境的值。

```
[Service]
Environment="HTTP_PROXY=http://proxy-domain:port"
Environment="HTTPS_PROXY=http://proxy-domain:port"
Environment="NO_PROXY=localhost"
```

kubelet 來自使用者資料的 組態

必須為此檔案建立 kubelet.service.d目錄。您需要重新載入系統化，才能在不重新啟動的情況下取得組態檔案。在 AL2023 中，當您的指令碼執行時，服務可能已經在執行，因此您也需要重新啟動它。

```
mkdir -p /etc/systemd/system/kubelet.service.d
echo '[Service]' > /etc/systemd/system/kubelet.service.d/http-proxy.conf
echo 'Environment="HTTP_PROXY=http://proxy-domain:port"' >> /etc/systemd/system/
kubelet.service.d/http-proxy.conf
echo 'Environment="HTTPS_PROXY=http://proxy-domain:port"' >> /etc/systemd/system/
kubelet.service.d/http-proxy.conf
echo 'Environment="NO_PROXY=localhost"' >> /etc/systemd/system/kubelet.service.d/http-
proxy.conf
systemctl daemon-reload
systemctl restart kubelet
```

ssm 代理組態

ssm 是可用於初始化混合節點的登入資料提供者之一。 ssm 負責使用 驗證 AWS 和產生 使用的臨時登入資料kubelet。如果您在內部部署環境中使用代理，並使用 ssm做為節點上的登入資料提供者，則必須設定 ，ssm以便其可以與 Amazon SSM 服務端點通訊。

根據作業系統，在以下路徑http-proxy.conf中在每個稱為 的混合節點上建立檔案

- Ubuntu - /etc/systemd/system/snap.amazon-ssm-agent.amazon-ssm-agent.service.d/http-proxy.conf

- Amazon Linux 2023 和 Red Hat Enterprise Linux - `/etc/systemd/system/amazon-ssm-agent.service.d/http-proxy.conf`

使用下列內容填入檔案。將 `proxy-domain`和 `port` 取代為您環境的值。

```
[Service]
Environment="HTTP_PROXY=http://proxy-domain:port"
Environment="HTTPS_PROXY=http://proxy-domain:port"
Environment="NO_PROXY=localhost"
```

`ssm` 來自使用者資料的 組態

必須為此檔案建立`ssm`系統化服務檔案目錄。目錄路徑取決於節點上使用的作業系統。

- Ubuntu - `/etc/systemd/system/snap.amazon-ssm-agent.amazon-ssm-agent.service.d`
- Amazon Linux 2023 和 Red Hat Enterprise Linux - `/etc/systemd/system/amazon-ssm-agent.service.d`

根據節點上使用的作業系統，取代以下重新啟動命令中的系統化服務名稱

- Ubuntu - `snap.amazon-ssm-agent.amazon-ssm-agent`
- Amazon Linux 2023 和 Red Hat Enterprise Linux - `amazon-ssm-agent`

```
mkdir -p systemd-service-file-directory
echo '[Service]' > [.replaceable]#systemd-service-file-directory/http-proxy.conf
echo 'Environment="HTTP_PROXY=http://[.replaceable]#proxy-domain:port"' >> systemd-
service-file-directory/http-proxy.conf
echo 'Environment="HTTPS_PROXY=http://[.replaceable]#proxy-domain:port"' >>
 [.replaceable]#systemd-service-file-directory/http-proxy.conf
echo 'Environment="NO_PROXY=localhost"' >> [.replaceable]#systemd-service-file-
directory/http-proxy.conf
systemctl daemon-reload
systemctl restart [.replaceable]#systemd-service-name
```

作業系統代理組態

如果您使用代理進行網際網路存取，則必須將作業系統設定為能夠從作業系統的套件管理員提取混合節點相依性。

Ubuntu

1. 將 snap設定為搭配下列命令使用您的代理：

```
sudo snap set system proxy.https=http://proxy-domain:port
sudo snap set system proxy.http=http://proxy-domain:port
```

2. 若要啟用 的代理apt，apt.conf請在 /etc/apt/目錄中建立名為 的檔案。將代理網域和連接埠取代為您環境的值。

```
Acquire::http::Proxy "http://proxy-domain:port";
Acquire::https::Proxy "http://proxy-domain:port";
```

Amazon Linux 2023

1. 設定 dnf 以使用您的代理。為您的環境建立/etc/dnf/dnf.conf具有代理網域和連接埠值的檔案。

```
proxy=http://proxy-domain:port
```

Red Hat Enterprise Linux

1. 設定 yum 以使用您的代理。為您的環境建立/etc/yum.conf具有代理網域和連接埠值的檔案。

```
proxy=http://proxy-domain:port
```

IAM Roles Anywhere 代理組態

IAM Roles Anywhere 登入資料提供者服務負責在使用 IAM Roles Anywhere 搭配 enableCredentialsFile旗標時重新整理登入資料（請參閱 the section called “EKS Pod 身分識別代理程式”）。如果您在內部部署環境中使用代理，則必須設定服務，以便它可以與 IAM Roles Anywhere 端點通訊。

使用下列內容http-proxy.conf在 /etc/systemd/system/aws_signing_helper_update.service.d/目錄中建立名為 的檔案。將 proxy-domain和 port 取代為您環境的值。

```
[Service]
Environment="HTTP_PROXY=http://proxy-domain:port"
Environment="HTTPS_PROXY=http://proxy-domain:port"
Environment="NO_PROXY=localhost"
```

全叢集組態

建立 Amazon EKS 叢集之後，以及在每個混合節點nodeadm init上執行之前，必須套用本節中的組態。

kube-proxy 代理組態

當您的混合節點加入叢集時，Amazon EKS 會自動在每個混合節點kube-proxy上安裝 做為 DaemonSet。 kube-proxy會在由 Amazon EKS 叢集上的 Pod 支援的 服務之間啟用路由。若要設定每個主機， kube-proxy 需要 Amazon EKS 叢集端點的 DNS 解析。

1. 使用下列命令編輯 kube-proxy DaemonSet

   ```
   kubectl -n kube-system edit ds kube-proxy
   ```

   這將在設定的編輯器上開啟 kube-proxy DaemonSet 定義。

2. 新增 HTTP_PROXY和 的環境變數HTTPS_PROXY。請注意，NODE_NAME環境變數應該已存在於您的組態中。port 將 proxy-domain和 取代為您環境的值。

   ```
   containers:
     - command:
       - kube-proxy
       - --v=2
       - --config=/var/lib/kube-proxy-config/config - --hostname-override=$(NODE_NAME)
       env:
       - name: HTTP_PROXY
         value: http://proxy-domain:port
       - name: HTTPS_PROXY
         value: http://proxy-domain:port
       - name: NODE_NAME
         valueFrom:
   ```

```
        fieldRef:
          apiVersion: v1
          fieldPath: spec.nodeName
```

混合節點的概念

使用 Amazon EKS 混合節點，您可以將內部部署或邊緣環境中執行的實體或虛擬機器加入 AWS 雲端中執行的 Amazon EKS 叢集。這種方法帶來許多好處，但也為熟悉在單一網路環境中執行 Kubernetes 叢集的使用者引進了新的聯網概念和架構。

以下章節深入探討 EKS 混合節點的 Kubernetes 和聯網概念，並詳細說明流量如何流經混合架構。這些區段要求您熟悉基本的 Kubernetes 網路知識，例如 Pod、節點、服務、Kubernetes 控制平面、kubelet 和 kube-proxy 的概念。

我們建議您依序閱讀這些頁面，從 the section called “網路概念”開始，接著是 the section called “Kubernetes 概念”，最後是 the section called “流量流程”。

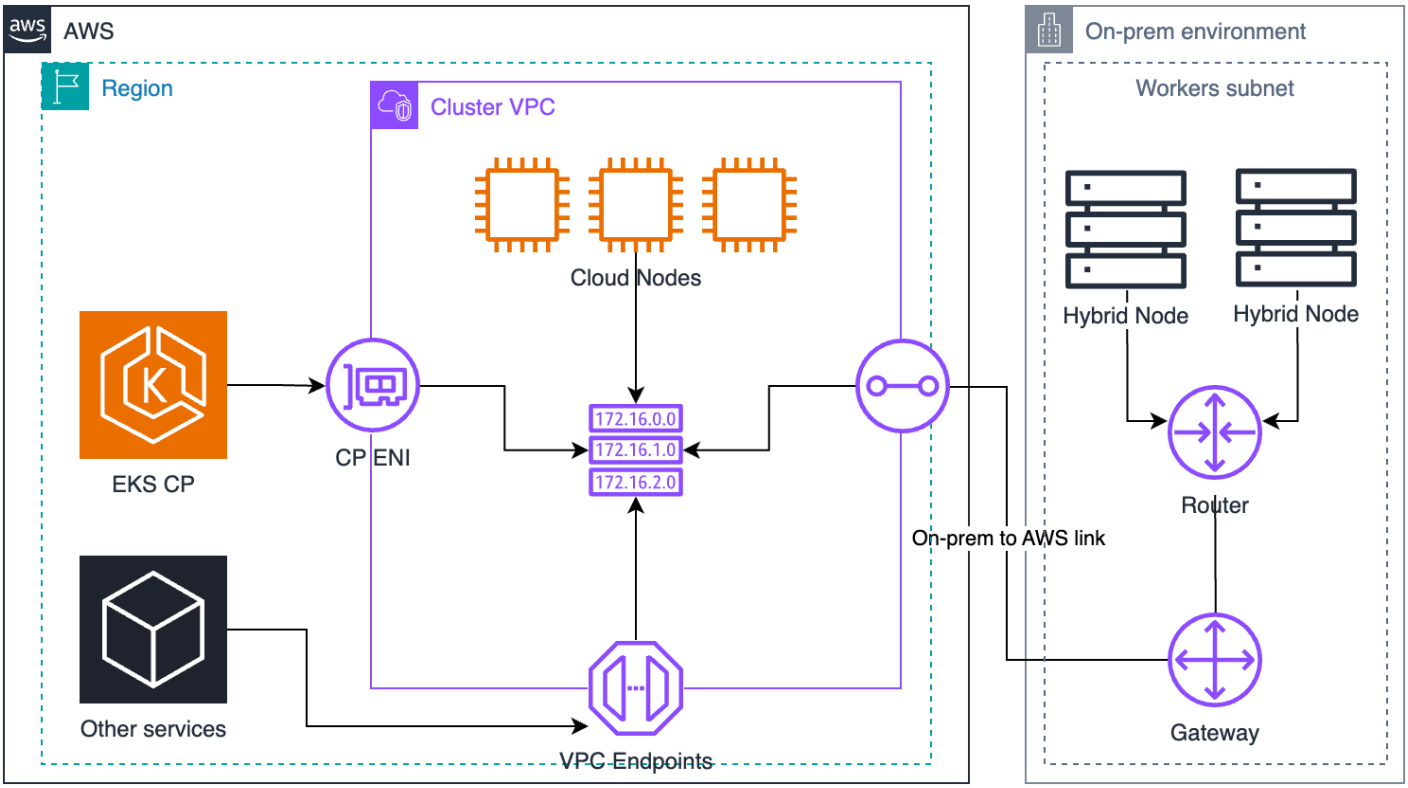
主題

- 混合節點的網路概念
- 混合節點的 Kubernetes 概念
- 混合節點的網路流量流程

混合節點的網路概念

本節詳細說明設計 EKS 混合節點的網路拓撲時，必須考量的核心聯網概念和限制條件。

EKS 混合節點的網路概念



VPC 做為網路中樞

通過您 VPC 的雲端邊界路由的所有流量。這包括在 中執行的 EKS 控制平面或 Pod AWS 與在其上執行的混合節點或 Pod 之間的流量。您可以將叢集的 VPC 視為混合節點與叢集其餘部分之間的網路中樞。此架構可讓您完全控制流量及其路由，但也可讓您負責正確設定 VPC 的路由、安全群組和防火牆。

EKS 控制平面到 VPC

EKS 控制平面會將彈性網路界面 (ENIs) 連接至您的 VPC。這些 ENIs 會處理進出 EKS API 伺服器的流量。您可以在設定叢集時控制 EKS 控制平面 ENIs的位置，因為 EKS 會將 ENIs 連接到您在叢集建立期間傳遞的子網路。

EKS 會將安全群組與 EKS 連接到子網路的 ENIs 建立關聯。這些安全群組允許透過 ENIs 往返 EKS 控制平面的流量。這對 EKS 混合節點很重要，因為您必須允許從混合節點和在其上執行的 Pod 到 EKS 控制平面 ENIs流量。

遠端節點網路

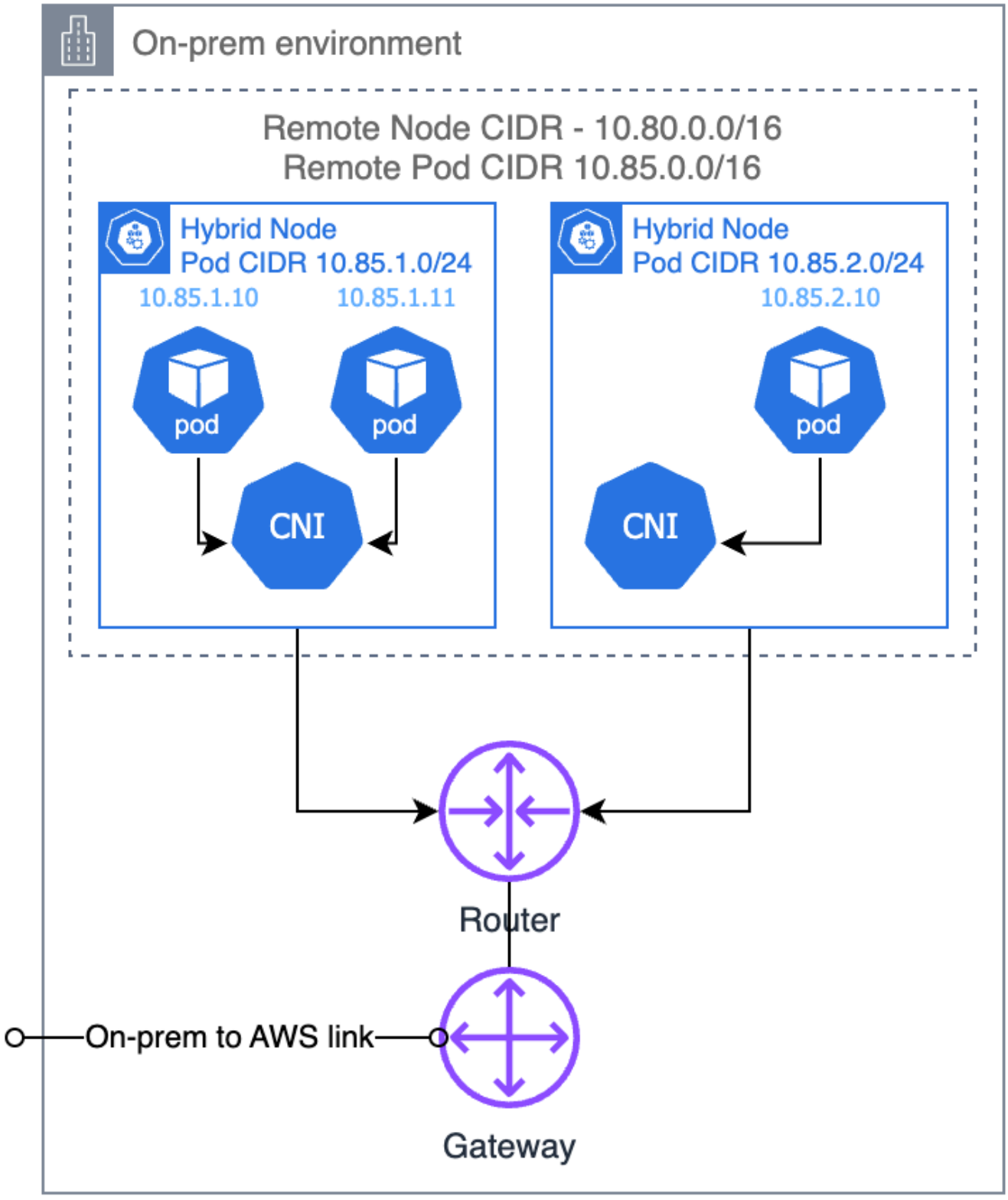
遠端節點網路，特別是遠端節點 CIDRs，是指派給您用作混合節點之機器的 IPs 範圍。當您佈建混合節點時，它們位於您的內部部署資料中心或節點，這是與 EKS 控制平面和 VPC 不同的網路網域。每個混合節點都有來自遠端節點 CIDR 的 IP 地址或地址，該 CIDR 與您 VPC 中的子網路不同。

您可以使用這些遠端節點 CIDRs 設定 EKS 叢集，讓 EKS 知道透過叢集 VPC 路由所有以混合節點 IPs 為目標的流量，例如請求到 kubelet API。

遠端 Pod 網路

遠端 Pod 網路是指派給在混合節點上執行之 Pod 的 IPs 範圍。一般而言，您可以使用這些範圍設定 CNI，CNI 的 IP 地址管理 (IPAM) 功能會負責將這些範圍的配量指派給每個混合節點。當您建立 Pod 時，CNI 會從配置給已排程 Pod 之節點的配量，將 IP 指派給 Pod。

您可以使用這些遠端 Pod CIDRs 設定 EKS 叢集，因此 EKS 控制平面知道透過叢集的 VPC 路由所有以混合節點上執行之 Pod 為目標的流量，例如與 Webhook 的通訊。


On-prem environment
Remote Node CIDR - 10.80.0.0/16
Remote Pod CIDR 10.85.0.0/16
Hybrid Node
Pod CIDR 10.85.1.0/24
10.85.1.10
10.85.1.11
pod
pod
CNI
Hybrid Node
Pod CIDR 10.85.2.0/24
10.85.2.10
pod
CNI
Router
On-prem to AWS link
Gateway

VPC 的內部部署

您用於混合節點的內部部署網路必須路由到您用於 EKS 叢集的 VPC。有數個Network-to-Amazon VPC 連線選項可用於將內部部署網路連線至 VPC。您也可以使用自己的 VPN 解決方案。

請務必在 VPC 和內部部署網路的 AWS 雲端端正確設定路由，以便兩個網路透過兩個網路的連線路由正確的流量。

在 VPC 中，所有流向遠端節點和遠端 Pod 網路的流量都必須透過連線路由至內部部署網路 （稱為「閘道」)。如果您的部分子網路具有不同的路由表，您必須使用混合節點的路由和在其上執行的 Pod 來設定每個路由表。對於連接 EKS 控制平面 ENIs子網路，以及包含必須與混合節點通訊之 EC2 節點或 Pod 的子網路，這是如此。

在內部部署網路中，您必須設定網路，以允許進出 EKS 叢集 VPC 的流量，以及混合節點 AWS 所需的其他服務。EKS 叢集的流量會以雙向周遊閘道。

網路限制條件

完全路由的網路

主要限制條件是 EKS 控制平面和所有節點、雲端或混合節點需要形成完全路由的網路。這表示所有節點都必須能夠透過 IP 地址在第三層互相連接。

由於 EKS 控制平面和雲端節點位於平面網路 (VPC) 中，因此它們可以互相連接。不過，混合節點位於不同的網路網域中。因此，您需要在 VPC 和內部部署網路中設定額外的路由，以在混合節點和叢集的其餘部分之間路由流量。如果混合節點可從彼此和 VPC 連接，您的混合節點可以位於單一平面網路或多個分段網路中。

可路由遠端 Pod CIDRs

若要讓 EKS 控制平面與在混合節點上執行的 Pod 通訊 （例如 Webhook 或 Metrics Server)，或讓在雲端節點上執行的 Pod 與在混合節點上執行的 Pod 通訊 （工作負載東西通訊），您的遠端 Pod CIDR 必須可從 VPC 路由。這表示 VPC 必須能夠透過閘道將流量路由至內部部署網路的 Pod CIDRs，而且您的內部部署網路必須能夠將 Pod 的流量路由至正確的節點。

請務必注意 VPC 和內部部署中 Pod 路由要求之間的差異。VPC 只需要知道任何流向遠端 Pod 的流量都應通過閘道。如果您只有一個遠端 Pod CIDR，則只需要一個路由。

此要求適用於您內部部署網路中的所有躍點，直到與您的混合節點位於相同子網路的本機路由器為止。這是唯一需要注意指派給每個節點的 Pod CIDR 配量的路由器，確保特定 Pod 的流量交付到已排程 Pod 的節點。

您可以選擇將這些內部部署 Pod CIDRs 的路由從本機內部部署路由器傳播到 VPC 路由表，但並非必要。如果您的內部部署 Pod CIDRs 經常變更，且您的 VPC 路由表需要更新以反映不斷變化的 Pod CIDRs，我們建議您將內部部署 Pod CIDRs 傳播到 VPC 路由表，但這並不常見。

請注意，設定內部部署 Pod CIDRs 可路由的限制是選用的。如果您不需要在混合節點上執行 Webhook，或讓雲端節點上的 Pod 與混合節點上的 Pod 通訊，則不需要為內部部署網路上的 Pod CIDRs 設定路由。

為什麼內部部署 Pod CIDRs 需要使用混合節點進行路由？

將 EKS 與雲端節點的 VPC CNI 搭配使用時，VPC CNI 會將 IPs 直接從 VPC 指派給 Pod。這表示不需要任何特殊路由，因為雲端 Pod 和 EKS 控制平面都可以直接到達 Pod IPs。

執行內部部署（以及雲端中的其他 CNIs) 時，Pod 通常會在隔離的覆蓋網路中執行，而 CNI 會負責在 Pod 之間交付流量。這通常透過封裝完成：CNI 會將 pod-to-pod 流量轉換為node-to-node流量，並在兩端負責封裝和解封裝。如此一來，節點和路由器上就不需要額外的組態。

與混合節點的聯網是唯一的，因為它提供兩種拓撲的組合 - EKS 控制平面和雲端節點（使用 VPC CNI) 預期包含節點和 Pod 的平坦網路，而在混合節點上執行的 Pod 透過使用 VXLAN 進行封裝（預設為 Cilium) 位於覆蓋網路中。在混合節點上執行的 Pod 可以到達在雲端節點上執行的 EKS 控制平面和 Pod，前提是內部部署網路可以路由到 VPC。不過，如果沒有內部部署網路上 Pod CIDRs的路由，如果網路不知道如何連接覆蓋網路並路由到正確的節點，則最終會捨棄任何返回內部部署 Pod IP 的流量。

混合節點的 Kubernetes 概念

此頁面詳細說明 EKS 混合節點系統架構基礎的關鍵 Kubernetes 概念。

VPC 中的 EKS 控制平面

EKS 控制平面 ENIs 的 IPs 會存放在 `default` 命名空間的 `kubernetes` `Endpoints` 物件中。當 EKS 建立新的 ENIs或移除較舊的 ENI 時，EKS 會更新此物件，讓 IPs 清單永遠是up-to-date。

您可以透過 `kubernetes`服務使用這些端點，也可以在 `default` 命名空間中使用這些端點。此服務的`ClusterIP`類型一律會獲指派叢集服務 CIDR 的第一個 IP。例如，對於服務 CIDR `172.16.0.0/16`，服務 IP 將為 `172.16.0.1`。

一般而言，這是 Pod（無論在雲端或混合節點中執行）存取 EKS Kubernetes API 伺服器的方式。Pod 使用服務 IP 做為目的地 IP，其會轉譯為其中一個 EKS 控制平面 ENIs 的實際 IPs。主要例外是 `kube-proxy`，因為它會設定轉譯。

EKS API 伺服器端點

kubernetes 服務 IP 不是存取 EKS API 伺服器的唯一方法。當您建立叢集時，EKS 也會建立 Route53 DNS 名稱。這是呼叫 EKS DescribeCluster API 動作時 EKS 叢集endpoint的欄位。

```
{
    "cluster": {
        "endpoint": "https://xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx.gr7.us-
west-2.eks.amazonaws.com",
        "name": "my-cluster",
        "status": "ACTIVE"
    }
}
```

在公有端點存取或公有和私有端點存取叢集中，您的混合節點預設會將此 DNS 名稱解析為公有 IP，可透過網際網路路由。在私有端點存取叢集中，DNS 名稱會解析為 EKS 控制平面 ENIs 的私有 IPs。

這是 kubelet和 kube-proxy存取 Kubernetes API 伺服器的方式。如果您希望所有 Kubernetes 叢集流量流經 VPC，您需要在私有存取模式中設定叢集，或修改內部部署 DNS 伺服器，以將 EKS 叢集端點解析為 EKS 控制平面 ENIs 的私有 IPs。

kubelet 端點

kubelet 公開數個 REST 端點，允許系統的其他部分與每個節點互動並收集資訊。在大多數叢集中，通往kubelet伺服器的大多數流量來自控制平面，但某些監控代理程式也可能與其互動。

透過此界面， kubelet會處理各種請求：擷取日誌 (kubectl logs)、在容器 (kubectl exec) 內執行命令，以及連接埠轉送流量 ()kubectl port-forward。這些請求都透過 與基礎容器執行期互動kubelet，對叢集管理員和開發人員來說是無縫的。

此 API 最常見的取用者是 Kubernetes API 伺服器。當您使用上述任何kubectl命令時， 會向 API 伺服器kubectl發出 API 請求，然後呼叫執行 Pod 的節點 kubelet API。這是需要從 EKS 控制平面連線到節點 IP 的主要原因，以及為什麼即使您的 Pod 正在執行，您也無法存取其日誌或exec節點路由設定錯誤。

節點 IPs

當 EKS 控制平面與節點通訊時，會使用Node物件狀態 () 中回報的其中一個地址status.addresses。

使用 EKS 雲端節點時， kubelet 通常會在節點註冊InternalIP期間將 EC2 執行個體的私有 IP 報告為 。然後，Cloud Controller Manager (CCM) 會驗證此 IP，確保其屬於 EC2 執行個體。此外，CCM

通常會將執行個體的公有 IPs (如 ExternalIP) 和 DNS 名稱 (InternalDNS 和 ExternalDNS) 新增至節點狀態。

不過，混合節點沒有 CCM。當您向 EKS 混合節點 CLI (nodeadm) 註冊混合節點時，它會設定 kubelet 以直接報告機器的 IP 處於節點狀態，而不需要 CCM。

```
apiVersion: v1
kind: Node
metadata:
  name: my-node-1
spec:
  providerID: eks-hybrid:///us-west-2/my-cluster/my-node-1
status:
  addresses:
  - address: 10.1.1.236
    type: InternalIP
  - address: my-node-1
    type: Hostname
```

如果您的機器有多個 IPs，則 kubelet 會依照自己的邏輯選取其中一個 IP。您可以使用 --node-ip旗標來控制選取的 IP，您可以在 nodeadm 中傳入組態spec.kubelet.flags。只有Node物件中報告的 IP 需要來自 VPC 的路由。您的機器可以擁有無法從雲端連線的其他 IPs。

kube-proxy

kube-proxy 負責在每個節點的網路層實作服務抽象。它可做為目的地為 Kubernetes Services 之流量的網路代理和負載平衡器。透過持續監看 Kubernetes API 伺服器是否有與服務和端點相關的變更，kube-proxy 動態更新基礎主機的聯網規則，以確保流量正確導向。

在 iptables模式下， kube-proxy會設定數個netfilter鏈來處理服務流量。這些規則組成下列階層：

1. KUBE-SERVICES 鏈：所有服務流量的進入點。它具有符合每個服務ClusterIP和連接埠的規則。
2. KUBE-SVC-XXX 鏈：服務特定的鏈具有每個服務的負載平衡規則。
3. KUBE-SEP-XXX 鏈：端點特定鏈具有實際DNAT規則。

讓我們檢查default命名空間test-server中服務的情況：* 服務 ClusterIP：172.16.31.14* 服務連接埠：80* 後端裝置：10.2.1.39、 10.2.0.110和 10.2.2.254

當我們檢查iptables規則時 (使用 iptables-save 0 grep -A10 KUBE-SERVICES)：

1. 在 KUBE-SERVICES 鏈中，我們找到符合服務的規則：

```
-A KUBE-SERVICES -d 172.16.31.14/32 -p tcp -m comment --comment "default/test-server
 cluster IP" -m tcp --dport 80 -j KUBE-SVC-XYZABC123456
```

- 此規則符合目的地為 172.16.31.14：80 的封包
- 註解指出此規則的用途： `default/test-server cluster IP`
- 相符的封包會跳至KUBE-SVC-XYZABC123456鏈結

2. KUBE-SVC-XYZABC123456 鏈具有以機率為基礎的負載平衡規則：

```
-A KUBE-SVC-XYZABC123456 -m statistic --mode random --probability 0.33333333349 -j
 KUBE-SEP-POD1XYZABC
-A KUBE-SVC-XYZABC123456 -m statistic --mode random --probability 0.50000000000 -j
 KUBE-SEP-POD2XYZABC
-A KUBE-SVC-XYZABC123456 -j KUBE-SEP-POD3XYZABC
```

- 第一個規則：33.3% 的機會跳到 KUBE-SEP-POD1XYZABC
- 第二個規則：剩餘流量 50% 的機會 （總數的 33.3%) 跳至 KUBE-SEP-POD2XYZABC
- 最後一個規則：所有剩餘的流量 （總數的 33.3%) 會跳至 KUBE-SEP-POD3XYZABC

3. 個別 KUBE-SEP-XXX 鏈會執行 DNAT （目的地 NAT)：

```
-A KUBE-SEP-POD1XYZABC -p tcp -m tcp -j DNAT --to-destination 10.2.0.110:80
-A KUBE-SEP-POD2XYZABC -p tcp -m tcp -j DNAT --to-destination 10.2.1.39:80
-A KUBE-SEP-POD3XYZABC -p tcp -m tcp -j DNAT --to-destination 10.2.2.254:80
```

- 這些 DNAT 規則會重寫目的地 IP 和連接埠，以將流量導向特定 Pod。
- 每個規則處理大約 33.3% 的流量，提供 `10.2.0.110`和 之間的平均負載平衡`10.2.1.3910.2.2.254`。

此多層級鏈結結構可讓 透過核心層級封包操作`kube-proxy`有效率地實作服務負載平衡和重新導向，而不需要資料路徑中的代理程序。

對 Kubernetes 操作的影響

節點`kube-proxy`上的中斷可防止該節點正常路由服務流量，導致依賴叢集服務之 Pod 的逾時或連線失敗。當節點第一次註冊時，這可能特別造成干擾。CNI 需要與 Kubernetes API 伺服器通訊以取得資訊，例如節點的 Pod CIDR，才能設定任何 Pod 網路。為此，它使用 `kubernetes` 服務 IP。不過，

如果 kube-proxy 無法啟動或無法設定正確的iptables規則，傳送至kubernetes服務 IP 的請求不會轉譯為 EKS 控制平面 ENIs 的實際 IPs。因此，CNI 將進入當機迴圈，而且任何 Pod 都無法正常執行。

我們知道 Pod 會使用kubernetes服務 IP 來與 Kubernetes API 伺服器通訊，但kube-proxy需要先設定iptables規則才能讓該運作。

如何與 API 伺服器kube-proxy通訊？

kube-proxy 必須設定為使用 Kubernetes API 伺服器的實際 IP 或解析為它們的 DNS 名稱。在 EKS 的情況下，EKS 會將預設值設定為kube-proxy指向 EKS 在您建立叢集時建立的 Route53 DNS 名稱。您可以在 kube-system 命名空間的 kube-proxy ConfigMap 中看到此值。此 ConfigMap 的內容是注入 Pod kube-proxy 的 kubeconfig ，因此請尋找 clusters0.cluster.server 欄位。此值將符合您 EKS 叢集endpoint的欄位 （呼叫 EKS DescribeCluster API 時）。

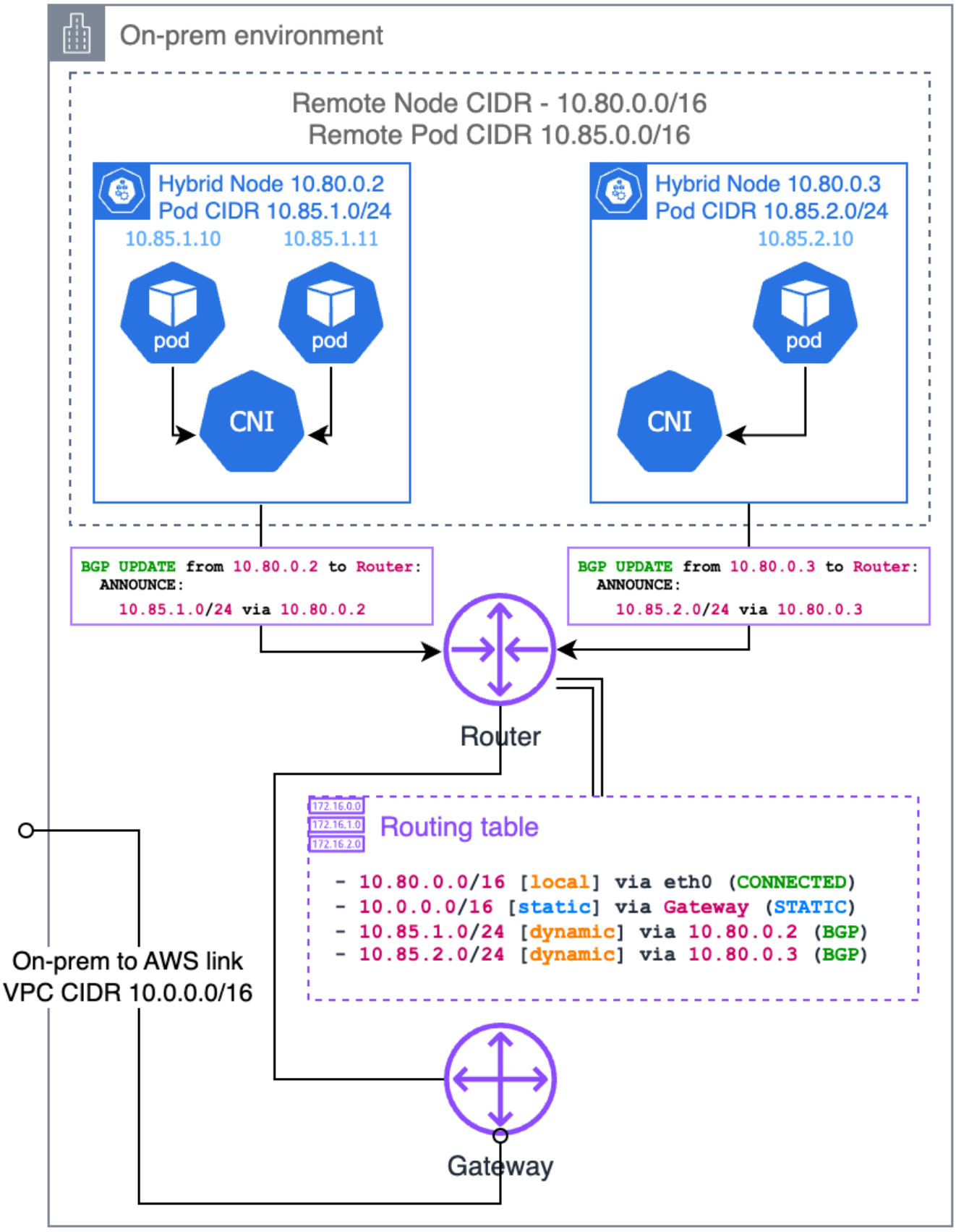
```
apiVersion: v1
data:
  kubeconfig: |-
    kind: Config
    apiVersion: v1
    clusters:
    - cluster:
        certificate-authority: /var/run/secrets/kubernetes.io/serviceaccount/ca.crt
        server: https://xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx.gr7.us-
west-2.eks.amazonaws.com
      name: default
    contexts:
    - context:
        cluster: default
        namespace: default
        user: default
      name: default
    current-context: default
    users:
    - name: default
      user:
        tokenFile: /var/run/secrets/kubernetes.io/serviceaccount/token
kind: ConfigMap
metadata:
  name: kube-proxy
  namespace: kube-system
```

可路由遠端 Pod CIDRs

the section called “網路概念” 此頁面詳細說明在混合節點上執行 Webhook 或讓在雲端節點上執行的 Pod 與在混合節點上執行的 Pod 通訊的需求。關鍵要求是現場部署路由器需要知道哪個節點負責特定的 Pod IP。有幾種方法可以實現這一點，包括邊界閘道協定 (BGP)、靜態路由和地址解析協定 (ARP) 代理。以下各節涵蓋這些內容。

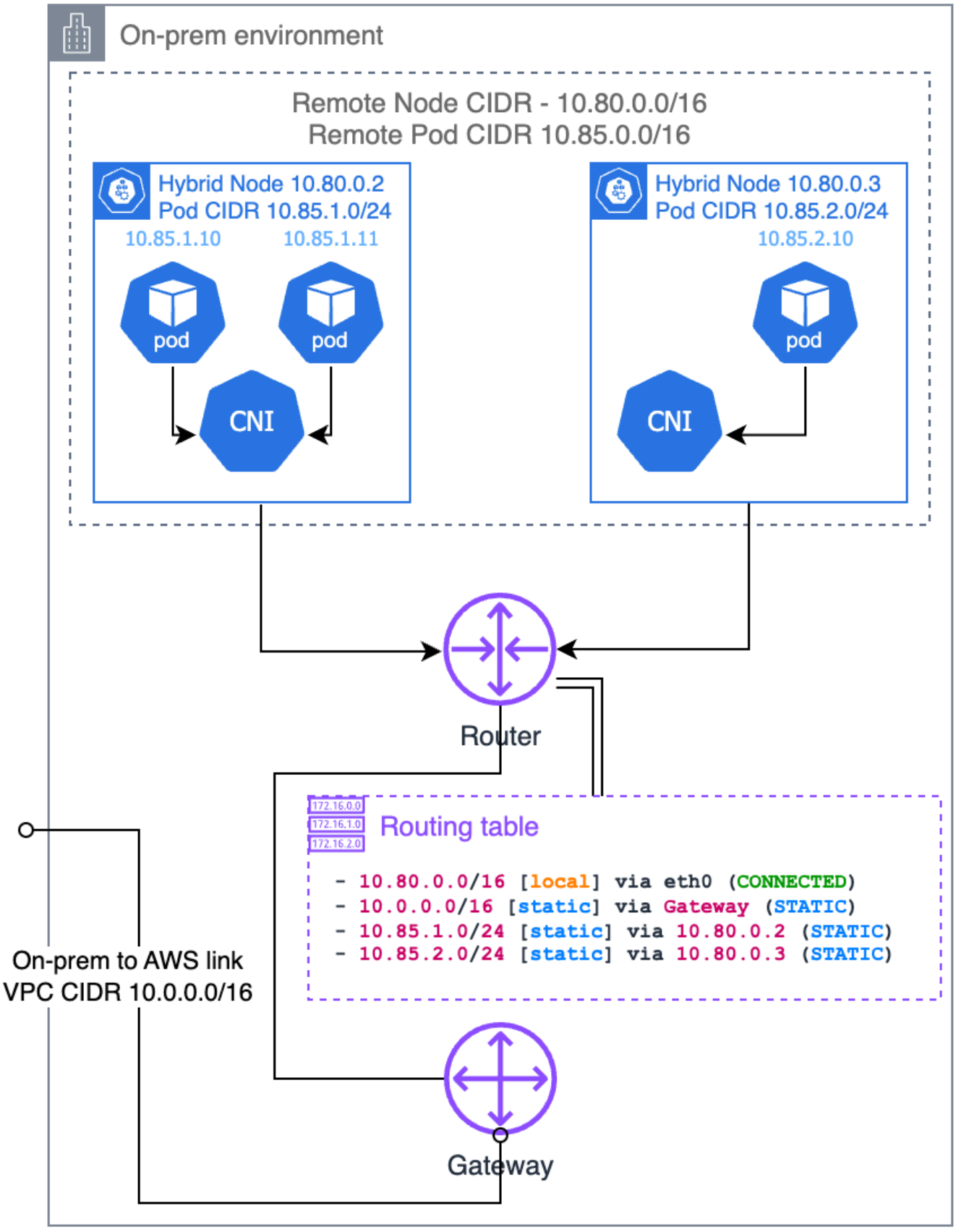
邊界閘道協定 (BGP)

如果您的 CNI 支援它 （例如 Cilium 和 Calico)，您可以使用 CNI 的 BGP 模式，將路由傳播到每個節點的 Pod CIDRs，從節點傳播到本機路由器。使用 CNI 的 BGP 模式時，您的 CNI 充當虛擬路由器，因此您的本機路由器認為 Pod CIDR 屬於不同的子網路，而您的節點是該子網路的閘道。


On-prem environment
Remote Node CIDR - 10.80.0.0/16
Remote Pod CIDR 10.85.0.0/16
Hybrid Node 10.80.0.2
Pod CIDR 10.85.1.0/24
10.85.1.10
10.85.1.11
pod
pod
CNI
Hybrid Node 10.80.0.3
Pod CIDR 10.85.2.0/24
10.85.2.10
pod
CNI
BGP UPDATE from 10.80.0.2 to Router:
ANNOUNCE:
10.85.1.0/24 via 10.80.0.2
BGP UPDATE from 10.80.0.3 to Router:
ANNOUNCE:
10.85.2.0/24 via 10.80.0.3
Router
Routing table
- 10.80.0.0/16 [local] via eth0 (CONNECTED)
- 10.0.0.0/16 [static] via Gateway (STATIC)
- 10.85.1.0/24 [dynamic] via 10.80.0.2 (BGP)
- 10.85.2.0/24 [dynamic] via 10.80.0.3 (BGP)
On-prem to AWS link
VPC CIDR 10.0.0.0/16
Gateway

靜態路由

或者，您可以在本機路由器中設定靜態路由。這是將內部部署 Pod CIDR 路由至 VPC 的最簡單方法，但這也是最容易出錯且難以維護的 Pod CIDR。您需要確保路由始終與現有節點及其指派的 Pod CIDRs up-to-date狀態。如果您的節點數量很小且基礎設施是靜態的，這是可行的選項，不需要在路由器中支援 BGP。如果您選擇這麼做，建議您使用要指派給每個節點的 Pod CIDR 配量來設定 CNI，而不是讓其 IPAM 決定。


On-prem environment
Remote Node CIDR - 10.80.0.0/16
Remote Pod CIDR 10.85.0.0/16
Hybrid Node 10.80.0.2
Pod CIDR 10.85.1.0/24
10.85.1.10
10.85.1.11
pod
pod
CNI
Hybrid Node 10.80.0.3
Pod CIDR 10.85.2.0/24
10.85.2.10
pod
CNI
Router
172.16.0.0
172.16.1.0
172.16.2.0
Routing table
- 10.80.0.0/16 [local] via eth0 (CONNECTED)
- 10.0.0.0/16 [static] via Gateway (STATIC)
- 10.85.1.0/24 [static] via 10.80.0.2 (STATIC)
- 10.85.2.0/24 [static] via 10.80.0.3 (STATIC)
On-prem to AWS link
VPC CIDR 10.0.0.0/16
Gateway

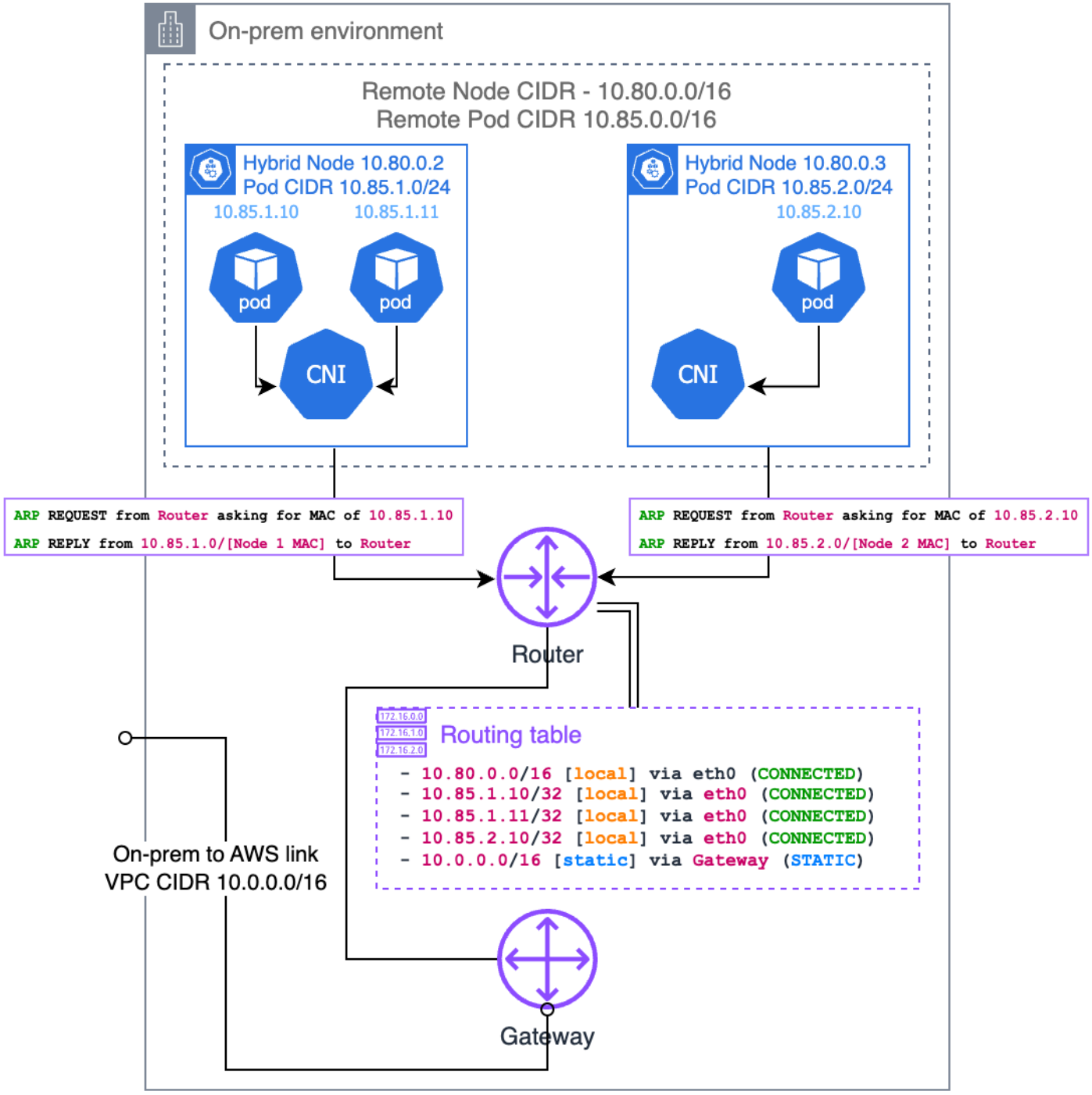
地址解析通訊協定 (ARP) 代理

ARP 代理是讓內部部署 Pod IPs 可路由的另一種方法，當您的混合節點與本機路由器位於相同的第 2 層網路上時，特別有用。啟用 ARP 代理後，節點會回應其託管之 Pod IPs 的 ARP 請求，即使這些 IPs 屬於不同的子網路。

當您本機網路上的裝置嘗試連線到 Pod IP 時，會先傳送 ARP 請求，詢問「誰有此 IP？」。Pod 將以自己的 MAC 地址回應的混合節點託管，表示「我可以處理該 IP 的流量」。這會在本機網路上的裝置與 Pod 之間建立直接路徑，而不需要路由器組態。

若要這麼做，您的 CNI 必須支援代理 ARP 功能。Cilium 內建對代理 ARP 的支援，您可以透過組態啟用。關鍵考量是 Pod CIDR 不得與您環境中的任何其他網路重疊，因為這可能會導致路由衝突。

此方法有幾個優點：* 不需要使用 BGP 設定路由器或維護靜態路由 * 適用於您無法控制路由器組態的環境



Pod-to-Pod 封裝

在內部部署環境中，CNIs 通常會使用封裝通訊協定來建立浮水印網路，可在實體網路之上運作，而不需要重新設定。本節說明此封裝的運作方式。請注意，某些詳細資訊可能會因您使用的 CNI 而有所不同。

封裝會將原始 Pod 網路封包包裝在另一個可透過基礎實體網路路由的網路封包內。這可讓 Pod 跨執行相同 CNI 的節點進行通訊，而不需要實體網路知道如何路由這些 Pod CIDRs。

與 Kubernetes 搭配使用的最常見封裝通訊協定是虛擬可擴展 LAN (VXLAN)，但其他（例如 Geneve) 也會根據您的 CNI 提供。

VXLAN 封裝

VXLAN 在 UDP 封包中封裝第 2 層乙太網路框架。當 Pod 將流量傳送至不同節點上的另一個 Pod 時，CNI 會執行下列動作：

1. CNI 攔截來自 Pod A 的封包
2. CNI 會將原始封包包裝在 VXLAN 標頭中
3. 然後，此包裝封包會透過節點的一般聯網堆疊傳送至目的地節點
4. 目的地節點上的 CNI 會取消包裝封包並將其交付至 Pod B

以下是 VXLAN 封裝期間封包結構發生的情況：

原始 Pod-to-Pod 封包：

```
+-----------------+---------------+-------------+-----------------+
| Ethernet Header | IP Header     | TCP/UDP     | Payload         |
| Src: Pod A MAC  | Src: Pod A IP | Src Port    |                 |
| Dst: Pod B MAC  | Dst: Pod B IP | Dst Port    |                 |
+-----------------+---------------+-------------+-----------------+
```

VXLAN 封裝之後：

```
+-----------------+-------------+--------------+------------
+---------------------------+
| Outer Ethernet  | Outer IP    | Outer UDP    | VXLAN      | Original Pod-to-Pod
 |
| Src: Node A MAC | Src: Node A | Src: Random  | VNI: xx    | Packet (unchanged
 |
| Dst: Node B MAC | Dst: Node B | Dst: 4789    |            | from above)
 |
+-----------------+-------------+--------------+------------
+---------------------------+
```

VXLAN 網路識別符 (VNI) 會區分不同的浮水印網路。

Pod 通訊案例

相同混合節點上的 Pod

當同一混合節點上的 Pod 通訊時，通常不需要封裝。CNI 會設定本機路由，透過節點的內部虛擬介面引導 Pod 之間的流量：

```
Pod A -> veth0 -> node's bridge/routing table -> veth1 -> Pod B
```

封包永遠不會離開節點，也不需要封裝。

不同混合節點上的 Pod

不同混合節點上 Pod 之間的通訊需要封裝：

```
Pod A -> CNI -> [VXLAN encapsulation] -> Node A network -> router or gateway -> Node B
 network -> [VXLAN decapsulation] -> CNI -> Pod B
```

這可讓 Pod 流量周遊實體網路基礎設施，而不需要實體網路來了解 Pod IP 路由。

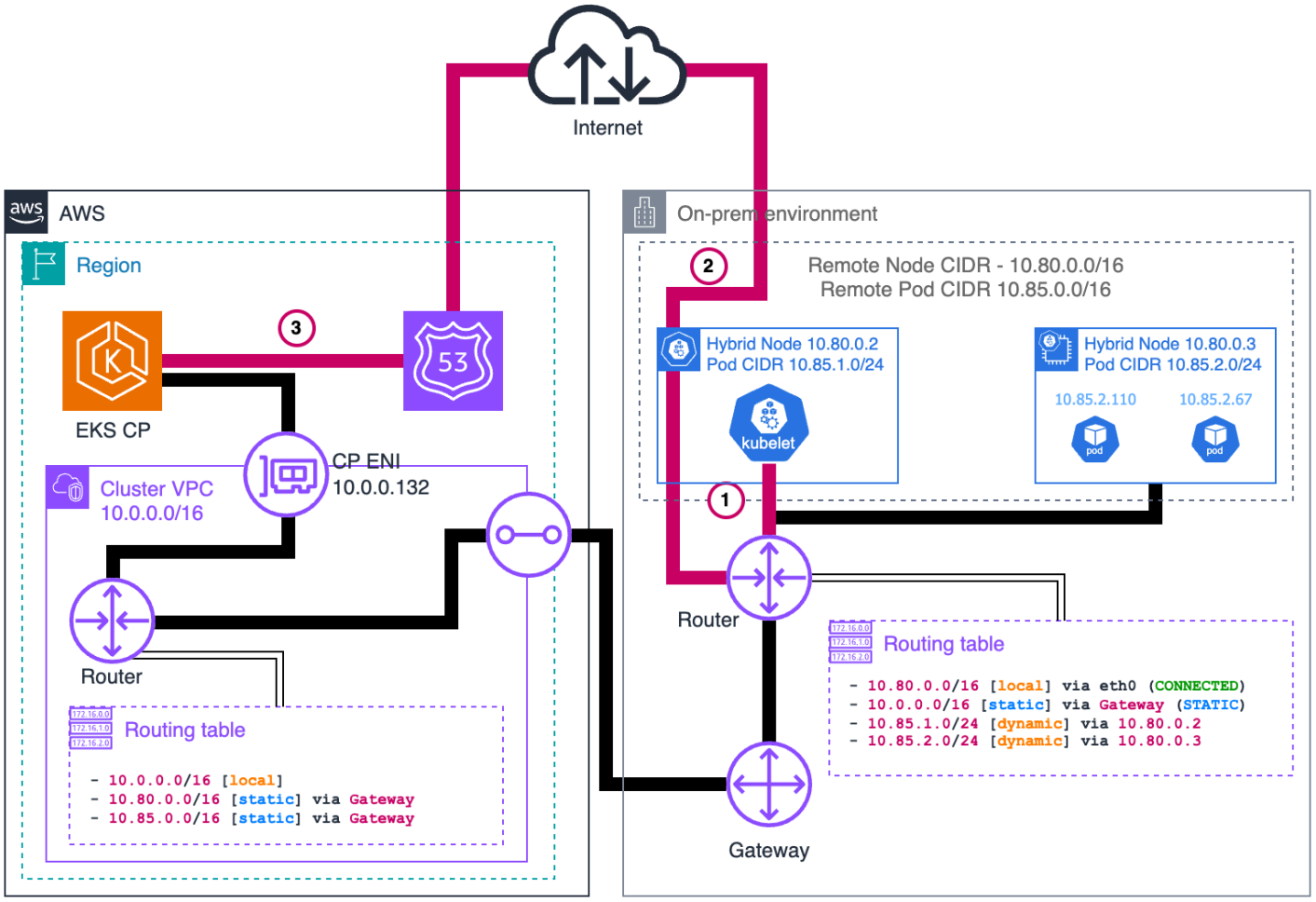
混合節點的網路流量流程

此頁面詳細說明 EKS 混合節點的網路流量流程，圖表顯示不同流量類型的end-to-end網路路徑。

涵蓋下列流量流程：

- the section called “混合節點kubelet到 EKS 控制平面”
- the section called “EKS 控制平面到混合節點 (kubelet 伺服器）”
- the section called “在混合節點上執行的 Pod 到 EKS 控制平面”
- the section called “在混合節點上執行的 EKS 控制平面到 Pod (webhooks)”
- the section called “在混合節點上執行Pod-to-Pod ”
- the section called “雲端節點上的 Pod 到混合節點上的 Pod （東西流量）”

混合節點**kubelet**到 EKS 控制平面



請求

1**kubelet**. 啟動請求

當混合節點kubelet上的 需要與 EKS 控制平面通訊時 （例如，報告節點狀態或取得 Pod 規格），它會使用節點註冊期間提供的 kubeconfig 檔案。這kubeconfig具有 API 伺服器端點 URL (Route53 DNS 名稱），而不是直接 IP 地址。

會執行端點的 kubelet DNS 查詢 （例如， https://xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx.gr7.us-west-2.eks.amazonaws.com ）。在公有存取叢集中，這會解析為屬於執行中 EKS 服務的公有 IP 地址 （例如 54.239.118.52) AWS。kubelet 然後， 會為此端點建立安全的 HTTPS 請求。初始封包如下所示：

```
+--------------------+---------------------+-----------------+
| IP Header          | TCP Header          | Payload         |
```

```
| Src: 10.80.0.2     | Src: 52390 (random) |                 |
| Dst: 54.239.118.52 | Dst: 443            |                 |
+--------------------+---------------------+-----------------+
```

2. 本機路由器路由

由於目的地 IP 是公有 IP 地址，不屬於本機網路，因此 kubelet會將此封包傳送至其預設閘道（本機內部部署路由器）。路由器會檢查目的地 IP，並判斷其為公有 IP 地址。

對於公有流量，路由器通常會將封包轉送至處理傳出流量的網際網路閘道或邊界路由器。這在圖表中被省略，並取決於現場部署網路的設定方式。封包會周遊您的內部部署網路基礎設施，最終到達網際網路服務供應商的網路。

3. 交付至 EKS 控制平面

封包會通過公有網際網路和傳輸網路，直到到達 AWS網路。 AWS的網路會將封包路由到適當區域中的 EKS 服務端點。當封包到達 EKS 服務時，它會轉送到叢集的實際 EKS 控制平面。

透過公有網際網路的路由與我們在其他流量流程中看到的私有 VPC 路由路徑不同。主要差別在於，使用公有存取模式時，從內部部署 kubelet（雖然不是從 Pod) 到 EKS 控制平面的流量不會通過您的 VPC，而是使用全球網際網路基礎設施。

回應

在 EKS 控制平面處理kubelet請求後，它會傳送回應：

3. EKS 控制平面傳送回應

EKS 控制平面會建立回應封包。此封包具有公有 IP 做為來源，而混合節點的 IP 做為目的地：

```
+--------------------+---------------------+-----------------+
| IP Header          | TCP Header          | Payload         |
| Src: 54.239.118.52 | Src: 443            |                 |
| Dst: 10.80.0.2     | Dst: 52390          |                 |
+--------------------+---------------------+-----------------+
```

2. 網際網路路由

回應封包會透過網際網路傳回，並遵循網際網路服務供應商決定的路由路徑，直到到達您的內部部署網路邊緣路由器為止。

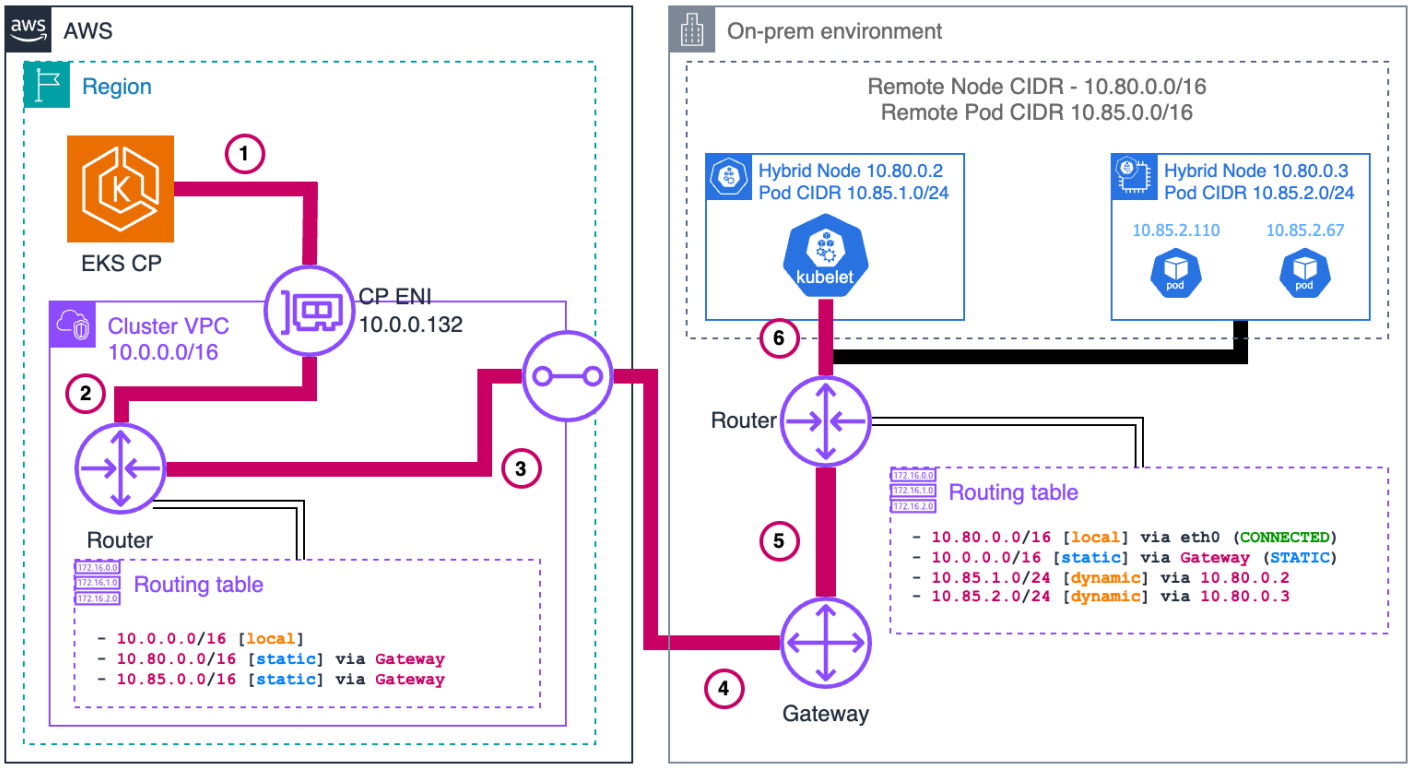
1。本機交付

您的內部部署路由器會收到封包，並將目的地 IP (10.80.0.2) 識別為屬於您的本機網路。它會透過您的本機網路基礎設施轉送封包，直到它到達目標混合節點， 會在其中kubelet接收和處理回應。

混合節點**kube-proxy**到 EKS 控制平面

如果您啟用叢集的公有端點存取，傳回流量會使用公有網際網路。此流量源自混合節點kube-proxy上的 到 EKS 控制平面，並遵循與從 kubelet到 EKS 控制平面的流量相同的路徑。

EKS 控制平面到混合節點 (**kubelet** 伺服器)



請求

1。EKS Kubernetes API 伺服器啟動請求

EKS Kubernetes API 伺服器會從節點物件的狀態擷取節點的 IP 地址 (10.80.0.2)。然後，它會在 VPC 中透過其 ENI 路由此請求，因為目的地 IP 屬於設定的遠端節點 CIDR (10.80.0.0/16)。初始封包如下所示：

```
+-----------------+---------------------+-----------------+
| IP Header       | TCP Header          | Payload         |
| Src: 10.0.0.132 | Src: 67493 (random) |                 |
| Dst: 10.80.0.2  | Dst: 10250          |                 |
```

```
+-----------------+---------------------+-----------------+
```

2. VPC 網路處理

封包離開 ENI 並進入 VPC 聯網層，並在其中導向子網路的閘道以進行進一步路由。

3. VPC 路由表查詢

包含 EKS 控制平面 ENI 之子網路的 VPC 路由表具有遠端節點 CIDR 的特定路由（圖表中的第二個路由）。根據此路由規則，封包會導向至 VPC-to-onprem閘道。

4. 跨邊界傳輸

閘道會透過您建立的連線（例如 Direct Connect 或 VPN)，將封包跨雲端邊界傳輸到您的內部部署網路。

5. 內部部署網路接收

封包會送達本機內部部署路由器，以處理混合節點所在子網路的流量。

6. 最終交付

本機路由器會識別目的地 IP (`10.80.0.2`) 地址屬於其直接連線的網路，並將封包直接轉送至 `kubelet`接收和處理請求的目標混合節點。

回應

在混合節點`kubelet`處理請求之後，它會以相反的相同路徑傳回回應：

```
+-----------------+---------------------+-----------------+
| IP Header       | TCP Header          | Payload         |
| Src: 10.80.0.2  | Src: 10250          |                 |
| Dst: 10.0.0.132 | Dst: 67493          |                 |
+-----------------+---------------------+-----------------+
```

6. **`kubelet`** 傳送回應

混合節點 (`10.80.0.2`) `kubelet`上的 會建立以原始來源 IP 做為目的地的回應封包。目的地不屬於本機網路，因此其會傳送到主機的預設閘道，也就是本機路由器。

5. 本機路由器路由

路由器會判斷目的地 IP (`10.0.0.132`) 屬於 `10.0.0.0/16`，其具有指向所連接閘道的路由 AWS。

4. 跨邊界傳回

封包會傳回相同的內部部署至 VPC 連線 （ 例如 Direct Connect 或 VPN)，以反向跨越雲端邊界。

3. VPC 路由

當封包送達 VPC 時，路由表會識別目的地 IP 是否屬於 VPC CIDR。封包會在 VPC 中路由。

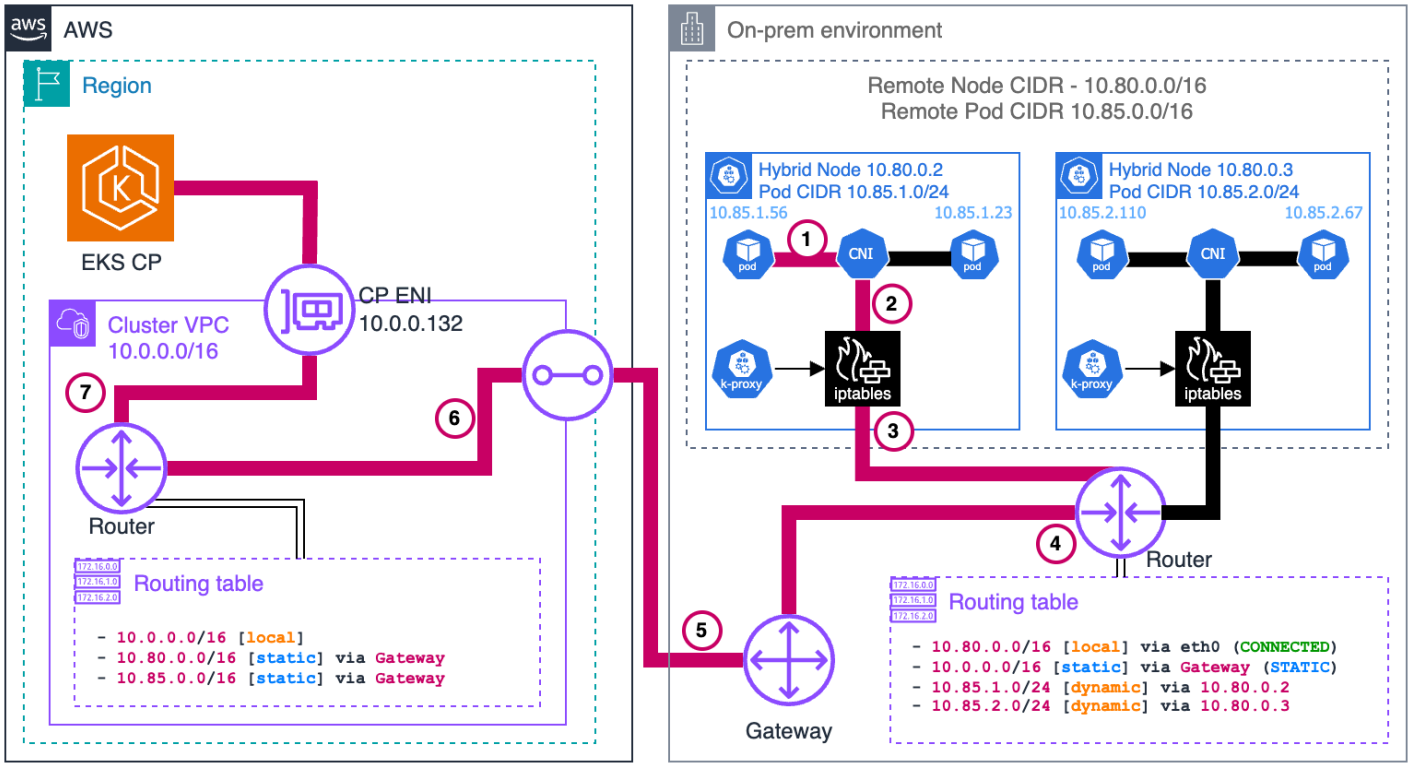
2. VPC 網路交付

VPC 網路層會使用 EKS 控制平面 ENI () 將封包轉送至子網路`10.0.0.132`。

1。ENI 接收

封包到達連接到 Kubernetes API 伺服器的 EKS 控制平面 ENI，完成往返。

在混合節點上執行的 Pod 到 EKS 控制平面



不使用 CNI NAT

請求

Pod 通常會透過 `kubernetes`服務與 Kubernetes API 伺服器通訊。服務 IP 是叢集服務 CIDR 的第一個 IP。此慣例允許在 CoreDNS 可用於連接 API 伺服器之前需要執行的 Pod，例如 CNI。請

求以服務 IP 做為目的地離開 Pod。例如，如果服務 CIDR 為 172.16.0.0/16，則服務 IP 將為 172.16.0.1。

1。Pod 啟動請求

Pod 會從隨機來源連接埠傳送請求至 API 伺服器連接埠 (443172.16.0.1) 上的kubernetes服務 IP ()。封包如下所示：

```
+-----------------+---------------------+-----------------+
| IP Header       | TCP Header          | Payload         |
| Src: 10.85.1.56 | Src: 67493 (random) |                 |
| Dst: 172.16.0.1 | Dst: 443            |                 |
+-----------------+---------------------+-----------------+
```

2. CNI 處理

CNI 偵測到目的地 IP 不屬於其管理的任何 Pod CIDR。由於傳出 NAT 已停用，CNI 會將封包傳遞至主機網路堆疊，而不進行修改。

3. 節點網路處理

封包會進入節點的網路堆疊，其中netfilter掛鉤會觸發 kube-proxy 設定的iptables規則。下列順序適用數個規則：

1. 封包會先命中KUBE-SERVICES鏈結，其中包含符合每個服務 ClusterIP 和連接埠的規則。
2. 比對規則會跳至 kubernetes服務的KUBE-SVC-XXX鏈結 （ 目的地為 的封包172.16.0.1:443)，其中包含負載平衡規則。
3. 負載平衡規則會隨機選取控制平面 ENI IPs (10.0.0.132 或 10.0.1.23) 的其中一個KUBE-SEP-XXX鏈。
4. 選取的KUBE-SEP-XXX鏈結具有將目的地 IP 從服務 IP 變更為所選 IP 的實際規則。這稱為目的地網路地址轉譯 (DNAT)。

套用這些規則後，假設選取的 EKS 控制平面 ENI 的 IP 為 10.0.0.132，封包如下所示：

```
+-----------------+---------------------+-----------------+
| IP Header       | TCP Header          | Payload         |
| Src: 10.85.1.56 | Src: 67493 (random) |                 |
| Dst: 10.0.0.132 | Dst: 443            |                 |
+-----------------+---------------------+-----------------+
```

節點會將封包轉送至其預設閘道，因為目的地 IP 不在本機網路中。

4. 本機路由器路由

本機路由器會判斷目的地 IP (10.0.0.132) 屬於 VPC CIDR (10.0.0.0/16)，並將其轉送至連線的閘道 AWS。

5. 跨邊界傳輸

封包會透過您已建立的連線 （例如 Direct Connect 或 VPN)，跨雲端邊界傳輸到 VPC。

6. VPC 網路交付

VPC 聯網層會將封包路由至 EKS 控制平面 ENI (10.0.0.132) 所在的正確子網路。

7. ENI 接收

封包到達連接到 Kubernetes API 伺服器的 EKS 控制平面 ENI。

回應

EKS 控制平面處理請求後，會傳送回應回 Pod：

7. API 伺服器傳送回應

EKS Kubernetes API 伺服器會使用原始來源 IP 做為目的地來建立回應封包。封包如下所示：

```
+-----------------+---------------------+-----------------+
| IP Header       | TCP Header          | Payload         |
| Src: 10.0.0.132 | Src: 443            |                 |
| Dst: 10.85.1.56 | Dst: 67493          |                 |
+-----------------+---------------------+-----------------+
```

由於目的地 IP 屬於設定的遠端 Pod CIDR (10.85.0.0/16)，它會透過其在 VPC 中的 ENI 傳送它，並將子網路的路由器作為下一個躍點。

6. VPC 路由

VPC 路由表包含遠端 Pod CIDR (10.85.0.0/16) 的項目，會將此流量導向 VPC-to-onprem閘道。

5. 跨邊界傳輸

閘道會透過您建立的連線 （例如 Direct Connect 或 VPN)，將封包跨雲端邊界傳輸到您的內部部署網路。

4. 現場部署網路接收

封包會送達本機內部部署路由器。

3. 交付至節點

路由器的資料表具有 項目10.85.1.0/24，將 10.80.0.2做為下一個躍點，將封包交付至我們的節點。

2. 節點網路處理

當封包由節點的網路堆疊處理時， conntrack(的一部分netfilter) 會比對封包與 Pod 最初建立的連線。由於最初套用了 DNAT， 透過將來源 IP 從 EKS 控制平面 ENI 的 IP 重寫到kubernetes服務 IP 來conntrack反轉 DNAT：

```
+-----------------+---------------------+-----------------+
| IP Header       | TCP Header          | Payload         |
| Src: 172.16.0.1 | Src: 443            |                 |
| Dst: 10.85.1.56 | Dst: 67493          |                 |
+-----------------+---------------------+-----------------+
```

1。CNI 處理

CNI 會識別目的地 IP 屬於其網路中的 Pod，並將封包交付至正確的 Pod 網路命名空間。

此流程展示為什麼遠端 Pod CIDRs 必須能夠從 VPC 正確路由到託管每個 Pod 的特定節點 - 整個傳回路徑取決於 Pod IPs 在雲端和內部部署網路之間的適當路由。

使用 CNI NAT

此流程與沒有 CNI NAT 的流程非常相似，但有一個關鍵差異：CNI 會將來源 NAT (SNAT) 套用至封包，然後再將其傳送至節點的網路堆疊。這會將封包的來源 IP 變更為節點的 IP，允許封包路由回節點，而不需要額外的路由組態。

請求

1。Pod 啟動請求

Pod 會從隨機來源連接埠將請求傳送至 EKS Kubernetes API 伺服器連接埠 (443172.16.0.1) 上的kubernetes服務 IP ()。封包如下所示：

```
+-----------------+---------------------+-----------------+
| IP Header       | TCP Header          | Payload         |
```

```
| Src: 10.85.1.56 | Src: 67493 (random) |                 |
| Dst: 172.16.0.1 | Dst: 443            |                 |
+-----------------+---------------------+-----------------+
```

2. CNI 處理

CNI 偵測到目的地 IP 不屬於其管理的任何 Pod CIDR。由於已啟用傳出 NAT，CNI 會將 SNAT 套用至封包，將來源 IP 變更為節點的 IP，然後再將其傳遞至節點的網路堆疊：

```
+-----------------+---------------------+-----------------+
| IP Header       | TCP Header          | Payload         |
| Src: 10.80.0.2  | Src: 67493 (random) |                 |
| Dst: 172.16.0.1 | Dst: 443            |                 |
+-----------------+---------------------+-----------------+
```

注意：為了清楚起見，範例中`iptables`顯示為個別區塊的 CNI 和 ，但實際上，有些 CNIs 可能會使用 `iptables` 來套用 NAT。

3. 節點網路處理

在這裡，由 設定的`iptables`規則`kube-proxy`的行為與上一個範例中的行為相同，將封包負載平衡到其中一個 EKS 控制平面 ENIs。封包現在如下所示：

```
+-----------------+---------------------+-----------------+
| IP Header       | TCP Header          | Payload         |
| Src: 10.80.0.2  | Src: 67493 (random) |                 |
| Dst: 10.0.0.132 | Dst: 443            |                 |
+-----------------+---------------------+-----------------+
```

節點會將封包轉送至其預設閘道，因為目的地 IP 不在本機網路中。

4. 本機路由器路由

本機路由器會判斷目的地 IP (`10.0.0.132`) 屬於 VPC CIDR (`10.0.0.0/16`)，並將其轉送至連線的閘道 AWS。

5. 跨邊界傳輸

封包會透過您已建立的連線 （例如 Direct Connect 或 VPN)，跨雲端邊界傳輸到 VPC。

6. VPC 網路交付

VPC 聯網層會將封包路由至 EKS 控制平面 ENI (`10.0.0.132`) 所在的正確子網路。

7. ENI 接收

封包到達連接到 Kubernetes API 伺服器的 EKS 控制平面 ENI。

回應

EKS 控制平面處理請求後，會傳送回應回 Pod：

7. API 伺服器傳送回應

EKS Kubernetes API 伺服器會使用原始來源 IP 做為目的地來建立回應封包。封包如下所示：

```
+-----------------+---------------------+-----------------+
| IP Header       | TCP Header          | Payload         |
| Src: 10.0.0.132 | Src: 443            |                 |
| Dst: 10.80.0.2  | Dst: 67493          |                 |
+-----------------+---------------------+-----------------+
```

由於目的地 IP 屬於設定的遠端節點 CIDR (`10.80.0.0/16`)，因此它會透過其在 VPC 中的 ENI 傳送它，並將子網路的路由器作為下一個躍點。

6. VPC 路由

VPC 路由表包含遠端節點 CIDR (`10.80.0.0/16`) 的項目，會將此流量導向 VPC-to-onprem閘道。

5. 跨邊界傳輸

閘道會透過您建立的連線 （ 例如 Direct Connect 或 VPN)，將封包跨雲端邊界傳輸到您的內部部署網路。

4. 現場部署網路接收

封包會送達本機內部部署路由器。

3. 交付至節點

本機路由器會識別目的地 IP (`10.80.0.2`) 地址屬於其直接連線的網路，並將封包直接轉送至目標混合節點。

2. 節點網路處理

當封包由節點的網路堆疊處理時， `conntrack`(的一部分`netfilter`) 會將封包與 Pod 最初建立的連線相符，而且自最初套用 DNAT 以來，它會透過將來源 IP 從 EKS 控制平面 ENI 的 IP 重寫至`kubernetes`服務 IP 來反轉此項目：

```
+-----------------+---------------------+-----------------+
| IP Header       | TCP Header          | Payload         |
| Src: 172.16.0.1 | Src: 443            |                 |
| Dst: 10.80.0.2  | Dst: 67493          |                 |
+-----------------+---------------------+-----------------+
```

1。CNI 處理

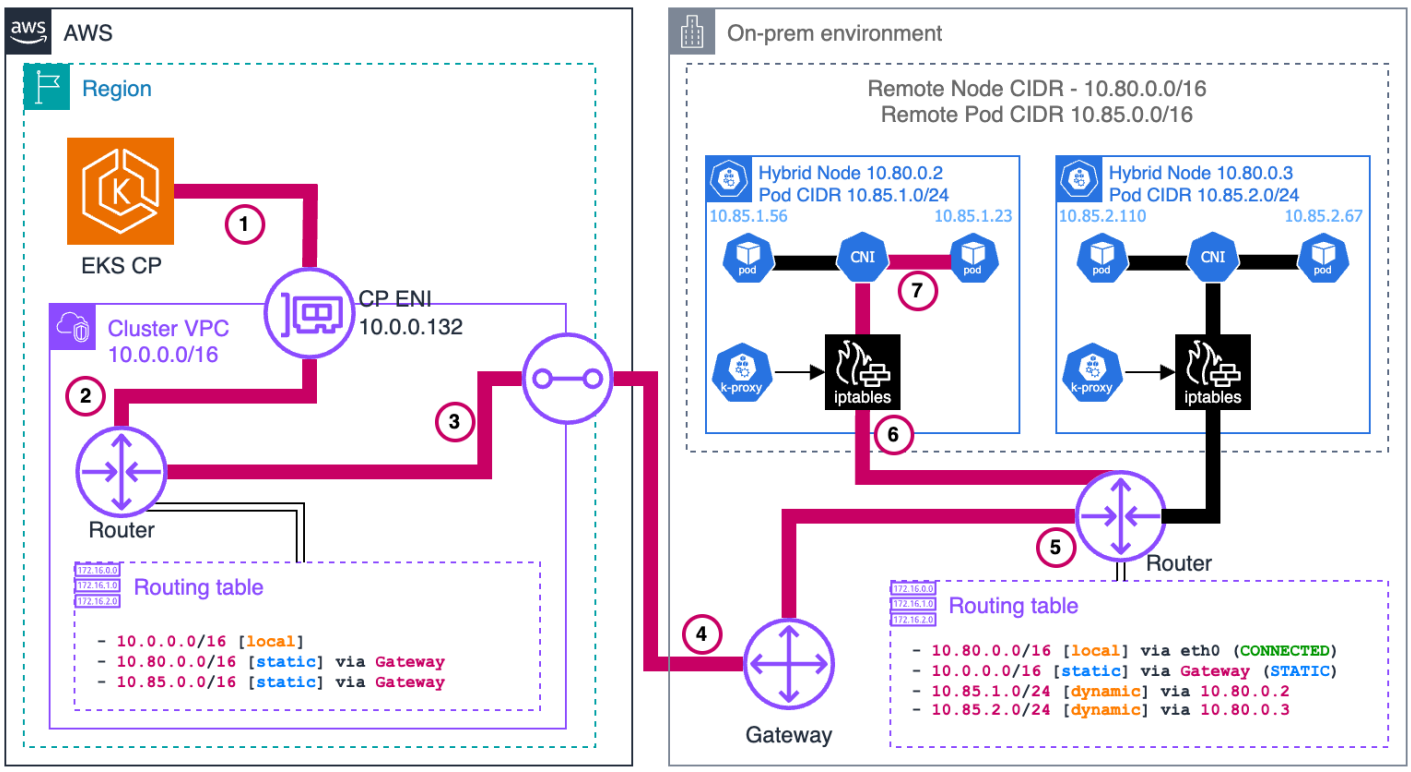
CNI 會識別此封包屬於先前已套用 SNAT 的連線。它會反轉 SNAT，將目的地 IP 變更回 Pod 的 IP：

```
+-----------------+---------------------+-----------------+
| IP Header       | TCP Header          | Payload         |
| Src: 172.16.0.1 | Src: 443            |                 |
| Dst: 10.85.1.56 | Dst: 67493          |                 |
+-----------------+---------------------+-----------------+
```

CNI 偵測到目的地 IP 屬於其網路中的 Pod，並將封包交付至正確的 Pod 網路命名空間。

此流程示範 CNI NAT-ing 如何透過允許封包路由回節點來簡化組態，而不需要額外的 Pod CIDRs 路由。

在混合節點上執行的 EKS 控制平面到 Pod (webhooks)



此流量模式最常見於 Webhook，其中 EKS 控制平面需要直接啟動與混合節點上 Pod 中執行之 Webhook 伺服器的連線。範例包括驗證和變更 API 伺服器在資源驗證或變動程序期間呼叫的許可 Webhook。

請求

1。EKS Kubernetes API 伺服器啟動請求

在叢集中設定 Webhook 並觸發相關的 API 操作時，EKS Kubernetes API 伺服器需要直接連線至 Webhook 伺服器 Pod。API 伺服器會先從與 Webhook 相關聯的服務或端點資源中查詢 Pod 的 IP 地址。

假設 Webhook Pod 在具有 IP 的混合節點上執行`10.85.1.23`，EKS Kubernetes API 伺服器會建立對 Webhook 端點的 HTTPS 請求。初始封包會透過 VPC 中的 EKS 控制平面 ENI 傳送，因為目的地 IP `10.85.1.23`屬於設定的遠端 Pod CIDR (`10.85.0.0/16`)。封包如下所示：

```
+-----------------+---------------------+-----------------+
| IP Header       | TCP Header          | Payload         |
| Src: 10.0.0.132 | Src: 41892 (random) |                 |
| Dst: 10.85.1.23 | Dst: 8443           |                 |
+-----------------+---------------------+-----------------+
```

2. VPC 網路處理

封包離開 EKS 控制平面 ENI 並進入 VPC 聯網層，並將子網路的路由器作為下一個躍點。

3. VPC 路由表查詢

包含 EKS 控制平面 ENI 之子網路的 VPC 路由表包含遠端 Pod CIDR () 的特定路由`10.85.0.0/16`。此路由規則會將封包導向 VPC-to-onprem（例如，適用於 Direct Connect 的 Virtual Private Gateway 或 VPN 連線）：

```
Destination     Target
10.0.0.0/16     local
10.85.0.0/16    vgw-id (VPC-to-onprem gateway)
```

4. 跨邊界傳輸

閘道會透過您建立的連線（例如 Direct Connect 或 VPN)，將封包跨雲端邊界傳輸到您的內部部署網路。封包會在周遊此連線時維護其原始來源和目的地 IP 地址。

5. 現場部署網路接收

封包會送達本機內部部署路由器。路由器會參考其路由表，以判斷如何到達 10.85.1.23 地址。若要這麼做，您的內部部署網路必須具有將流量導向適當混合節點的 Pod CIDRs 路由。

在此情況下，路由器的路由表包含一個項目，指出可透過具有 IP 的混合節點來連接`10.85.1.0/24`子網路`10.80.0.2`：

```
Destination     Next Hop
10.85.1.0/24    10.80.0.2
```

6. 交付至節點

根據路由表項目，路由器會將封包轉送到混合節點 (`10.80.0.2`)。當封包到達節點時，它看起來與 EKS Kubernetes API 伺服器傳送它時相同，目的地 IP 仍然是 Pod 的 IP。

7. CNI 處理

節點的網路堆疊會收到封包，並看到目的地 IP 不是節點自己的 IP，將其傳遞給 CNI 進行處理。CNI 會識別目的地 IP 屬於在此節點本機上執行的 Pod，並透過適當的虛擬介面將封包轉送至正確的 Pod：

```
Original packet -> node routing -> CNI -> Pod's network namespace
```

Pod 中的 Webhook 伺服器會收到請求並加以處理。

回應

在 Webhook Pod 處理請求後，它會以相反的相同路徑傳回回應：

7. Pod 傳送回應

Webhook Pod 會使用自己的 IP 做為來源建立回應封包，並將原始請求者 (EKS 控制平面 ENI) 做為目的地：

```
+-----------------+---------------------+-----------------+
| IP Header       | TCP Header          | Payload         |
| Src: 10.85.1.23 | Src: 8443           |                 |
| Dst: 10.0.0.132 | Dst: 41892          |                 |
+-----------------+---------------------+-----------------+
```

CNI 會識別此封包會移至外部網路 （而非本機 Pod)，並將封包傳遞至保留原始來源 IP 的節點網路堆疊。

6. 節點網路處理

節點會判斷目的地 IP (10.0.0.132) 不在本機網路中，並將封包轉送至其預設閘道 （本機路由器）。

5. 本機路由器路由

本機路由器會參考其路由表，並判斷目的地 IP (10.0.0.132) 屬於 VPC CIDR (10.0.0.0/16)。它會將封包轉送到連接到 的閘道 AWS。

4. 跨邊界傳輸

封包會傳回相同的內部部署至 VPC 連線，以相反方向跨越雲端邊界。

3. VPC 路由

當封包抵達 VPC 時，路由表會識別目的地 IP 是否屬於 VPC 內的子網路。封包會在 VPC 中相應地路由。

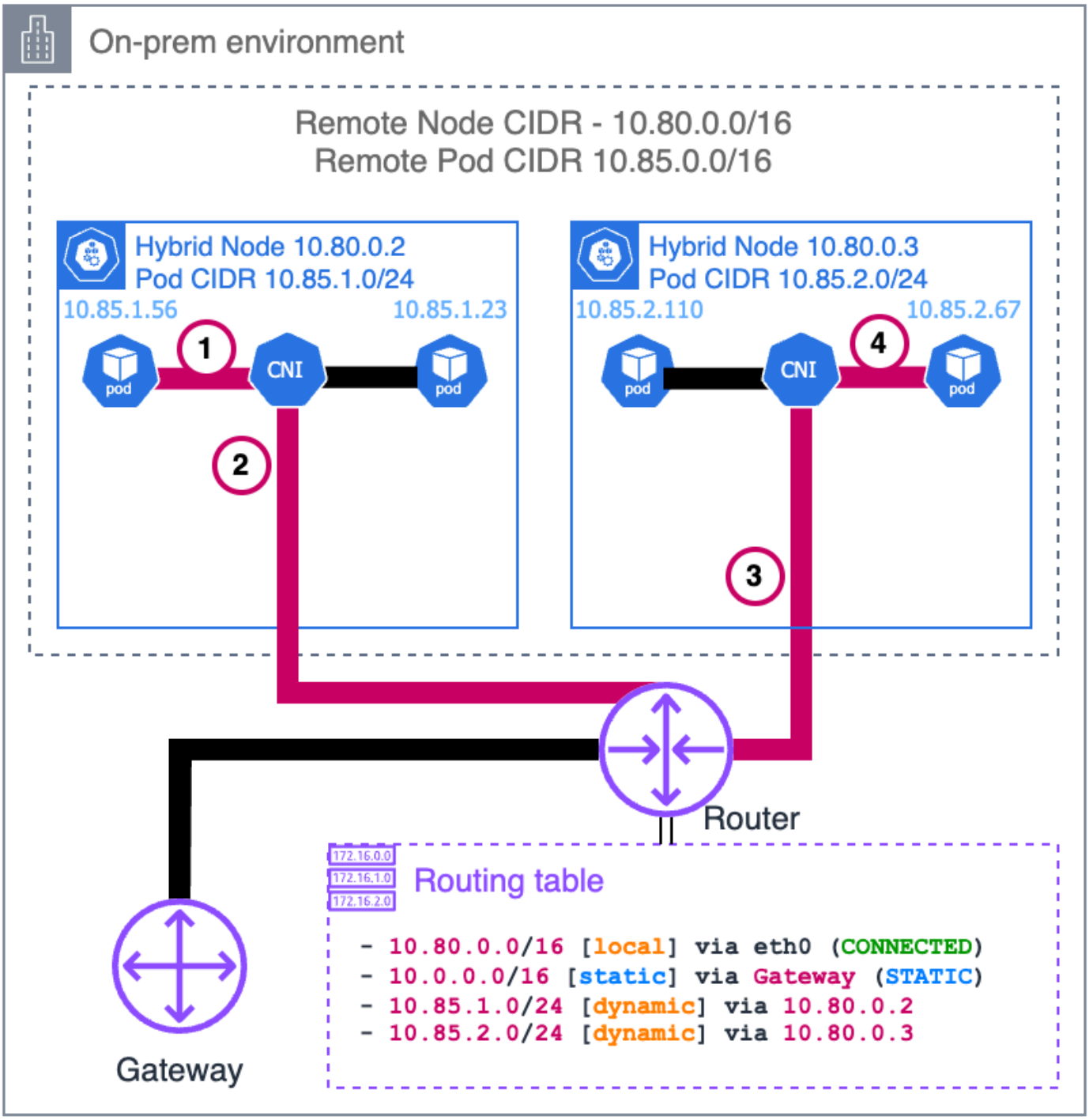
2. 和 1. EKS 控制平面 ENI 接收

封包到達連接到 EKS Kubernetes API 伺服器的 ENI，完成往返。API 伺服器會收到 Webhook 回應，並繼續根據此回應處理原始 API 請求。

此流量流程示範為什麼必須正確設定和路由遠端 Pod CIDRs：

- VPC 必須具有指向內部部署閘道之遠端 Pod CIDRs的路由
- 您的內部部署網路必須具有 Pod CIDRs 的路由，將流量導向託管這些 Pod 的特定節點
- 如果沒有此路由組態，將無法從 EKS 控制平面存取在混合節點上 Pod 中執行的 Webhook 和其他類似服務。

在混合節點上執行Pod-to-Pod



本節說明在不同混合節點上執行的 Pod 如何互相通訊。此範例假設您的 CNI 使用 VXLAN 進行封裝，這在 Cilium 或 Calico 等 CNIs 中很常見。其他封裝通訊協定的整體程序類似，例如 Geneve 或 IP-in-IP。

請求

1。Pod A 啟動通訊

節點 1 上的 Pod A (10.85.1.56) 想要將流量傳送至節點 2 上的 Pod B (10.85.2.67)。初始封包如下所示：

```
+------------------+-----------------+-------------+-----------------+
| Ethernet Header  | IP Header       | TCP/UDP     | Payload         |
| Src: Pod A MAC   | Src: 10.85.1.56 | Src: 43721  |                 |
| Dst: Gateway MAC | Dst: 10.85.2.67 | Dst: 8080   |                 |
+------------------+-----------------+-------------+-----------------+
```

2. CNI 攔截和處理封包

當 Pod A 的封包離開其網路命名空間時，CNI 會攔截它。CNI 會參考其路由表，並判斷： - 目的地 IP (10.85.2.67) 屬於 Pod CIDR - 此 IP 不在本機節點上，但屬於節點 2 (10.80.0.3) - 封包需要使用 VXLAN 封裝。

封裝的決策至關重要，因為基礎實體網路不知道如何直接路由 Pod CIDRs，它只知道如何在節點 IPs之間路由流量。

CNI 會將整個原始封包封裝在 VXLAN 框架內。這有效地建立具有新標頭的「封包內」：

```
+-----------------+---------------+-------------+-----------
+--------------------------+
| Outer Ethernet  | Outer IP      | Outer UDP   | VXLAN     | Original Pod-to-Pod
    |
| Src: Node1 MAC  | Src: 10.80.0.2 | Src: Random  | VNI: 42    | Packet (unchanged
    |
| Dst: Router MAC | Dst: 10.80.0.3 | Dst: 8472    |            | from above)
    |
+-----------------+---------------+-------------+-----------
+--------------------------+
```

有關此封裝的要點：- 外部封包從節點 1 (10.80.0.2) 定址到節點 2 (10.80.0.3) - UDP 連接埠8472是預設使用的 VXLAN 連接埠 Cilium - VXLAN 網路識別符 (VNI) 識別此封包所屬的覆蓋網路 - 整個原始封包（將 Pod A 的 IP 作為來源，將 Pod B 的 IP 作為目的地） 在內部保持不變

封裝的封包現在會進入節點 1 的一般網路堆疊，並以與任何其他封包相同的方式處理：

1. 節點網路處理：節點 1 的網路堆疊會根據其目的地路由封包 (10.80.0.3)

2. 本機網路交付：
   - 如果兩個節點位於相同的第 2 層網路上，封包會直接傳送到節點 2
   - 如果封包位於不同的子網路上，則會先將封包轉送至本機路由器
3. 路由器處理：路由器會根據其路由表轉送封包，並將其交付至節點 2

3. 接收節點處理

當封裝的封包送達節點 2 () 時`10.80.0.3`：

1. 節點的網路堆疊會接收並識別為 VXLAN 封包 (UDP 連接埠 4789)
2. 封包會傳遞至 CNI 的 VXLAN 介面進行處理

4. VXLAN 解壓縮

節點 2 上的 CNI 會處理 VXLAN 封包：

1. 它會去除外部標頭（乙太網路、IP、UDP 和 VXLAN)
2. 它會擷取原始內部封包
3. 封包現在恢復為原始格式：

```
+------------------+-----------------+-------------+-----------------+
| Ethernet Header  | IP Header       | TCP/UDP     | Payload         |
| Src: Pod A MAC   | Src: 10.85.1.56 | Src: 43721  |                 |
| Dst: Gateway MAC | Dst: 10.85.2.67 | Dst: 8080   |                 |
+------------------+-----------------+-------------+-----------------+
```

節點 2 上的 CNI 會檢查目的地 IP (`10.85.2.67`) 和：

1. 識別此 IP 屬於本機 Pod
2. 透過適當的虛擬介面路由封包
3. 將封包交付至 Pod B 的網路命名空間

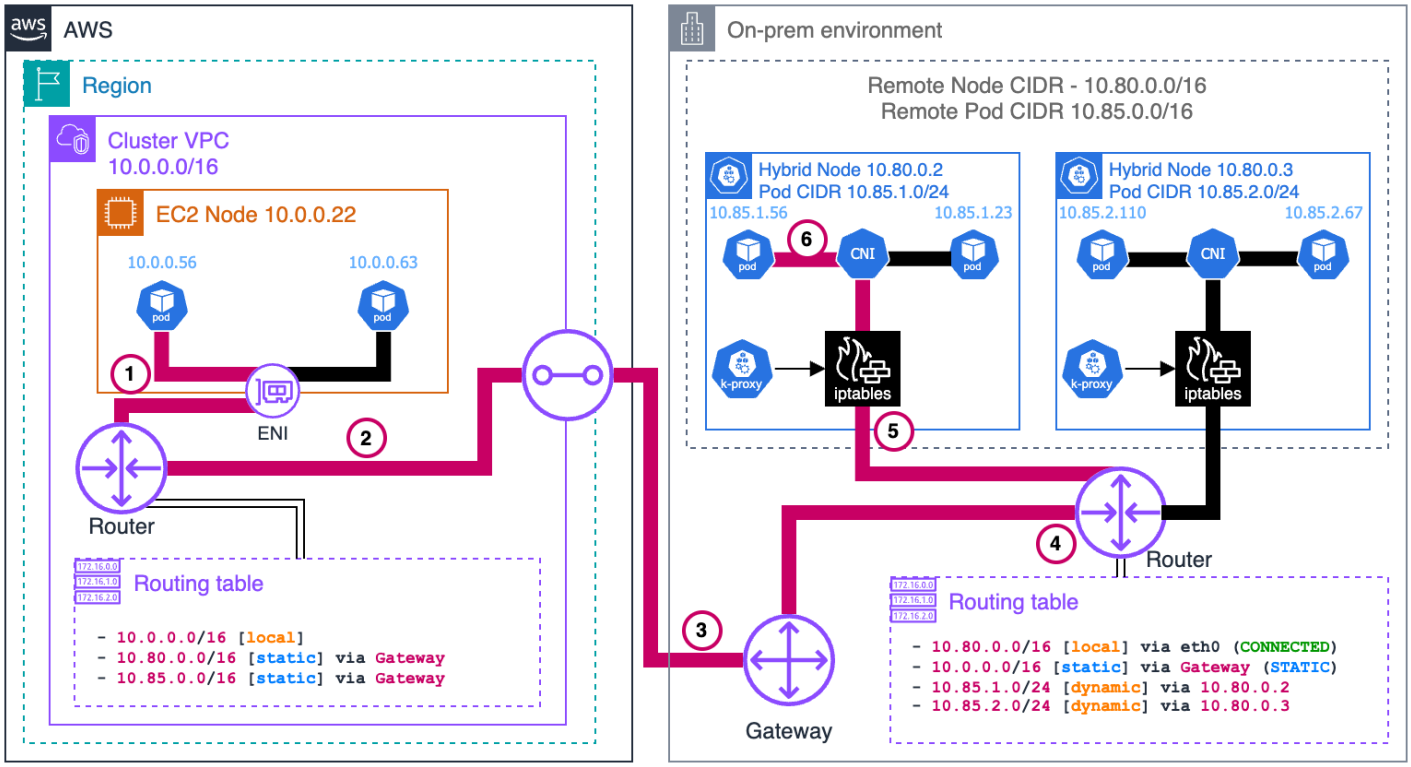
回應

當 Pod B 回應 Pod A 時，整個程序會反向發生：

4. Pod B 將封包傳送至 Pod A (`10.85.1.56`)

5. 節點 2 的 CNI 使用 VXLAN 封裝，將目的地設定為節點 1 (10.80.0.2)
6. 封裝的封包會傳送到節點 1
7. 節點 1 的 CNI 將其解封裝，並交付原始回應給 Pod A

雲端節點上的 Pod 到混合節點上的 Pod （東西流量）



請求

1。Pod A 啟動通訊

EC2 節點上的 Pod A (10.0.0.56) 想要將流量傳送至混合節點上的 Pod B (10.85.1.56)。初始封包如下所示：

```
+-----------------+---------------------+-----------------+
| IP Header       | TCP Header          | Payload         |
| Src: 10.0.0.56  | Src: 52390 (random) |                 |
| Dst: 10.85.1.56 | Dst: 8080           |                 |
+-----------------+---------------------+-----------------+
```

使用 VPC CNI 時，Pod A 具有來自 VPC CIDR 的 IP，並直接連接到 EC2 執行個體上的 ENI。Pod 的網路命名空間已連線至 VPC 網路，因此封包會直接進入 VPC 路由基礎設施。

2. VPC 路由

VPC 路由表包含遠端 Pod CIDR (10.85.0.0/16) 的特定路由，會將此流量導向 VPC-to-onprem閘道：

```
Destination     Target
10.0.0.0/16     local
10.85.0.0/16    vgw-id (VPC-to-onprem gateway)
```

根據此路由規則，封包會導向至連線至內部部署網路的閘道。

3. 跨邊界傳輸

閘道會透過您建立的連線 （ 例如 Direct Connect 或 VPN)，將封包跨雲端邊界傳輸到您的內部部署網路。在此傳輸過程中，封包會維護其原始來源和目的地 IP 地址。

4. 現場部署網路接收

封包會送達本機內部部署路由器。路由器會參考其路由表，以判斷到達 10.85.1.56 地址的下一個跳轉。您的內部部署路由器必須具有將流量導向適當混合節點的 Pod CIDRs 路由。

路由器的資料表有一個項目，指出子網路可透過具有 IP 的混合節點10.85.1.0/24來連接10.80.0.2：

```
Destination     Next Hop
10.85.1.0/24    10.80.0.2
```

5. 節點網路處理

路由器會將封包轉送至混合節點 (10.80.0.2)。當封包到達節點時，它仍然具有 Pod A 的 IP 作為來源，Pod B 的 IP 作為目的地。

6. CNI 處理

節點的網路堆疊會收到封包，並看到目的地 IP 不是自己的 IP 會將其傳遞給 CNI 進行處理。CNI 會識別目的地 IP 屬於在此節點本機上執行的 Pod，並透過適當的虛擬介面將封包轉送至正確的 Pod：

```
Original packet -> node routing -> CNI -> Pod B's network namespace
```

Pod B 會接收封包並視需要處理。

回應

6. Pod B 傳送回應

Pod B 會以自己的 IP 做為來源建立回應封包，而 Pod A 的 IP 做為目的地：

```
+-----------------+---------------------+-----------------+
| IP Header       | TCP Header          | Payload         |
| Src: 10.85.1.56 | Src: 8080           |                 |
| Dst: 10.0.0.56  | Dst: 52390          |                 |
+-----------------+---------------------+-----------------+
```

CNI 會識別此封包目的地為外部網路，並將其傳遞至節點的網路堆疊。

5. 節點網路處理

節點會判斷目的地 IP (`10.0.0.56`) 不屬於本機網路，並將封包轉送至其預設閘道 （本機路由器）。

4. 本機路由器路由

本機路由器會參考其路由表，並判斷目的地 IP (`10.0.0.56`) 屬於 VPC CIDR (`10.0.0.0/16`)。它會將封包轉送到連接到 的閘道 AWS。

3. 跨邊界傳輸

封包會傳回相同的內部部署至 VPC 連線，並反向跨越雲端邊界。

2. VPC 路由

當封包送達 VPC 時，路由系統會識別目的地 IP 是否屬於 VPC 內的子網路。封包會透過 VPC 網路路由至託管 Pod A 的 EC2 執行個體。

1。Pod A 接收回應

封包抵達 EC2 執行個體，並透過其連接的 ENI 直接交付至 Pod A。由於 VPC CNI 不會針對 VPC 中的 Pod 使用浮水印聯網，因此不需要額外的解封裝 - 封包會以其原始標頭完整送達。

此東西流量流程示範了為什麼遠端 Pod CIDRs 必須正確設定並從兩個方向路由：

- VPC 必須具有指向內部部署閘道之遠端 Pod CIDRs 的路由
- 您的內部部署網路必須具有 Pod CIDRs 的路由，將流量導向託管這些 Pod 的特定節點。

混合節點nodeadm參考

Amazon EKS 混合節點 CLI (nodeadm) 可簡化混合節點元件的安裝、組態、註冊和解除安裝。您可以在作業系統映像nodeadm中包含 以自動化混合節點引導，the section called “準備作業系統”如需詳細資訊，請參閱 。

混合節點的nodeadm版本與用於將 Amazon EC2 執行個體引導為 Amazon EKS 叢集中節點的nodeadm版本不同。遵循適當nodeadm版本的文件和參考。本文件頁面適用於混合節點nodeadm版本。

混合節點的原始程式碼nodeadm會發佈在 https://github.com/aws/eks-hybrid GitHub 儲存庫中。

> **Important**
>
> 您必須nodeadm使用具有根/sudo 權限的使用者執行 。

下載 nodeadm

的混合節點版本nodeadm託管在 Amazon CloudFront 前端的 Amazon S3 中。 Amazon CloudFront 若要在每個現場部署主機nodeadm上安裝 ，您可以從現場部署主機執行下列命令。

對於 x86_64 主機

```
curl -OL 'https://hybrid-assets.eks.amazonaws.com/releases/latest/bin/linux/amd64/
nodeadm'
```

對於 ARM 主機

```
curl -OL 'https://hybrid-assets.eks.amazonaws.com/releases/latest/bin/linux/arm64/
nodeadm'
```

將可執行檔許可新增至每個主機上下載的二進位檔。

```
chmod +x nodeadm
```

nodeadm install

nodeadm install 命令用於安裝執行混合節點並將其加入 Amazon EKS 叢集所需的成品和相依性。nodeadm install 命令可以個別在每個混合節點上執行，也可以在映像建置管道期間執行，以在作業系統映像中預先安裝混合節點相依性。

用途

```
nodeadm install [KUBERNETES_VERSION] [flags]
```

位置引數

（必要）KUBERNETES_VERSION 要安裝的 EKS Kubernetes major.minor 版本，例如 1.32

Flags

名稱	必要	描述
-p, --credential-provider	TRUE	要安裝的登入資料提供者。支援的值為 iam-ra 和 ssm。如需詳細資訊，請參閱the section called “準備登入資料”。
-s, --containerd-source	FALSE	的來源containerd 。nodeadm支援containerd 從 OS distro、Docker 套件和略過安裝containerd 進行安裝。 Values (數值) distro - 這是預設值。nodeadm將安裝節點作業系統分佈的containerd 套件。 distro 不是 Red Hat Enterprise Linux (RHEL) 作業系統支援的值。

名稱	必要	描述
		docker - nodeadm將安裝由 Docker 建置和分佈的containerd 套件。docker 不是 Amazon Linux 2023 支援的值 none - nodeadm 不會安裝containerd 套件。您必須先手動安裝 containerd ，才能執行 nodeadm init。
-r, --region	FALSE	指定下載成品 AWS 的區域，例如 SSM Agent。預設為 us-west-2 。
-t, --timeout	FALSE	安裝命令持續時間上限。輸入遵循持續時間格式。例如 1h23m。安裝命令的預設下載逾時設定為 20 分鐘。
-h, --help	FALSE	顯示說明訊息，其中包含可用的旗標、子命令和位置值參數。

範例

1.32 使用 AWS Systems Manager (SSM) 做為登入資料提供者來安裝 Kubernetes 版本

```
nodeadm install 1.32 --credential-provider ssm
```

1.32 使用 AWS Systems Manager (SSM) 作為登入資料提供者安裝 Kubernetes 版本，Docker 作為容器化來源，下載逾時為 20 分鐘。

```
nodeadm install 1.32 --credential-provider ssm --containerd-source docker --timeout 20m
```

1.32 使用 AWS IAM Roles Anywhere 做為登入資料提供者來安裝 Kubernetes 版本

```
nodeadm install 1.32 --credential-provider iam-ra
```

nodeadm config check

nodeadm config check 命令會檢查提供的節點組態是否有錯誤。此命令可用來驗證和驗證混合節點組態檔案的正確性。

用途

```
nodeadm config check [flags]
```

Flags

名稱	必要	描述
-c, --config-source	TRUE	nodeadm 組態的來源。對於混合節點，輸入應該遵循具有檔案配置的 URI。
-h, --help	FALSE	顯示說明訊息，其中包含可用的旗標、子命令和位置值參數。

範例

```
nodeadm config check -c file://nodeConfig.yaml
```

nodeadm init

nodeadm init 命令會啟動混合節點並與設定的 Amazon EKS 叢集連線。the section called “IAM Roles Anywhere 的節點組態” 如需如何設定 nodeConfig.yaml 檔案的詳細資訊，請參閱 the section called “SSM 混合啟用的節點組態”或 。

用途

```
nodeadm init [flags]
```

Flags

名稱	必要	描述
-c, --config-source	TRUE	nodeadm 組態來源。對於混合節點，輸入應該遵循具有檔案配置的 URI。
-s, --skip	FALSE	init 要略過的階段。除非有助於修正問題，否則不建議略過任何階段。 Values (數值) install-validation 略過檢查上述安裝命令是否成功執行。 cni-validation 如果節點上已啟用防火牆，則略過檢查是否已開啟 Cilium 或 Calico CNI 的 VXLAN 連接埠 node-ip-validation 略過檢查節點 IP 是否位於遠端節點網路中的 CIDR 內
-h, --help	FALSE	顯示說明訊息，其中包含可用的旗標、子命令和位置值參數。

範例

```
nodeadm init -c file://nodeConfig.yaml
```

nodeadm upgrade

nodeadm upgrade 命令會將所有已安裝的成品升級到最新版本，並引導節點來設定升級的成品並加入 EKS 叢集 AWS。升級是對節點上執行之工作負載的破壞性命令。請先將工作負載移至另一個節點，再執行升級。

用途

```
nodeadm upgrade [KUBERNETES_VERSION] [flags]
```

位置引數

（必要）KUBERNETES_VERSION 要安裝的 EKS Kubernetes major.minor 版本，例如 1.32

Flags

名稱	必要	描述
-c, --config-source	TRUE	nodeadm 組態來源。對於混合節點，輸入應該遵循具有檔案配置的 URI。
-t, --timeout	FALSE	下載成品的逾時。輸入遵循持續時間格式。例如 1h23m。升級命令的預設下載逾時設定為 10 分鐘。
-s, --skip	FALSE	要略過的升級階段。除非有助於修正問題，否則不建議略過任何階段。 Values (數值) pod-validation 會略過檢查節點上是否沒有執行所有 Pod，但協助程式集和靜態 Pod 除外。 node-validation 略過檢查是否已封鎖節點。 init-validation 在執行升級之前，會略過檢查節點是否已成功初始化。

名稱	必要	描述
-h, --help	FALSE	顯示說明訊息，其中包含可用的旗標、子命令和位置值參數。

範例

```
nodeadm upgrade 1.32 -c file://nodeConfig.yaml
```

```
nodeadm upgrade 1.32 -c file://nodeConfig.yaml --timeout 20m
```

nodeadm uninstall

nodeadm uninstall 命令會停止並移除 期間nodeadm安裝的成品nodeadm install，包括 kubelet 和 containerd。請注意，解除安裝命令不會耗盡或刪除叢集中的混合節點。您必須分別執行耗盡和刪除操作，the section called “刪除混合節點”如需詳細資訊，請參閱 。根據預設，如果節點上剩餘 Pod，nodeadm uninstall則 不會繼續。同樣地， nodeadm uninstall 不會移除您在叢集上執行之其他 Kubernetes 附加元件的 CNI 相依性或相依性。若要從主機完全移除 CNI 安裝，請參閱 中的指示the section called “設定 CNI”。如果您使用 AWS SSM 混合啟用做為內部部署憑證提供者，nodeadm uninstall命令會將您的主機取消註冊為 AWS SSM 受管執行個體。

用途

```
nodeadm uninstall [flags]
```

Flags

名稱	必要	描述
-s, --skip	FALSE	要略過的解除安裝階段。除非有助於修正問題，否則不建議略過任何階段。 Values (數值) pod-validation 會略過檢查節點上是否沒有執行所

名稱	必要	描述
		有 Pod，但協助程式集和靜態 Pod 除外。 `node-validation` 略過檢查是否已封鎖節點。 `init-validation` 在執行解除安裝之前，會略過檢查節點是否已成功初始化。

`-h,`
 	
`--help`
	FALSE	顯示說明訊息，其中包含可用的旗標、子命令和位置值參數。

名稱	必要	描述
`-f`, `--force`	FALSE	強制從 Kubernetes 和 CNI 元件刪除可能包含剩餘檔案的其他目錄。 警告 這將刪除預設 Kubernetes 和 CNI 目錄中的所有內容 (`/var/lib/cni` 、 `/etc/cni/net.d` 等) 。如果您將自己的資料存放在這些位置，請勿使用此旗標。 從 nodeadm 開始`v1.0.9`，`./nodeadm uninstall --skip node-validation,pod-validation --force` 命令不會再刪除`/var/lib/kubelet` 目錄。這是因為它可能包含 Pod 磁碟區和磁碟區子路徑目錄，這些目錄有時包含掛載的節點檔案系統。 安全處理秘訣 - 刪除掛載路徑可能會導致意外刪除實際掛載的節點檔案系統。在手動刪除`/var/lib/kubelet` 目錄之前，請仔細檢查所有作用中的掛載和卸載磁碟區，以避免資料遺失。

範例

```
nodeadm uninstall
```

```
nodeadm uninstall --skip node-validation,pod-validation
```

nodeadm debug

nodeadm debug 命令可用來疑難排解運作狀態不佳或設定錯誤的混合節點。它會驗證下列需求已就位。

- 節點具有必要 AWS APIs的網路存取權，以取得登入資料，
- 節點可以取得所設定混合節點 IAM 角色的 AWS 登入資料，
- 節點具有 EKS Kubernetes API 端點的網路存取權，以及 EKS Kubernetes API 端點憑證的有效性，
- 節點能夠向 EKS 叢集進行身分驗證，其在叢集中的身分有效，而且節點可以透過為 EKS 叢集設定的 VPC 存取 EKS 叢集。

如果發現錯誤，命令的輸出會建議疑難排解步驟。某些驗證步驟會顯示子程序。如果這些失敗，輸出會顯示在驗證錯誤下的 stderr 區段中。

用途

```
nodeadm debug [flags]
```

Flags

名稱	必要	描述
-c, --config-source	TRUE	nodeadm 組態來源。對於混合節點，輸入應該遵循具有檔案配置的 URI。
--no-color	FALSE	停用顏色輸出。適用於自動化。
-h, --help	FALSE	顯示說明訊息，其中包含可用的旗標、子命令和位置值參數。

範例

```
nodeadm debug -c file://nodeConfig.yaml
```

Nodeadm 檔案位置

nodeadm 安裝

執行 時`nodeadm install`，會設定下列檔案和檔案位置。

Artifact	路徑
IAM Roles Anywhere CLI	/usr/local/bin/aws_signing_helper
Kubelet 二進位	/usr/bin/kubelet
Kubectl 二進位	usr/local/bin/kubectl
ECR 登入資料提供者	/etc/eks/image-credential-provider/ecr-credential-provider
AWS IAM 驗證器	/usr/local/bin/aws-iam-authenticator
SSM 設定 CLI	/opt/ssm/ssm-setup-cli
SSM Agent	在 Ubuntu 上 - /snap/amazon-ssm-agent/current/amazon-ssm-agent 在 RHEL 和 AL2023 - /usr/bin/amazon-ssm-agent 上
Containerd	在 Ubuntu 和 AL2023 上 - /usr/bin/containerd 在 RHEL - /bin/containerd 上
Iptables	在 Ubuntu 和 AL2023 上 - /usr/sbin/iptables 在 RHEL - /sbin/iptables 上
CNI 外掛程式	/opt/cni/bin

Artifact	路徑
已安裝成品追蹤器	/opt/nodeadm/tracker

nodeadm init

執行 時`nodeadm init`，會設定下列檔案和檔案位置。

名稱	路徑
Kubelet kubeconfig	/var/lib/kubelet/kubeconfig
Kubelet 組態	/etc/kubernetes/kubelet/config.json
Kubelet 系統化單位	/etc/systemd/system/kubelet.service
映像登入資料提供者組態	/etc/eks/image-credential-provider/config.json
Kubelet env 檔案	/etc/eks/kubelet/environment
Kubelet 憑證	/etc/kubernetes/pki/ca.crt
Containerd 組態	/etc/containerd/config.toml
Containerd 核心模組組態	/etc/modules-load.d/contianerd.conf
AWS 組態檔案	/etc/aws/hybrid/config
AWS 登入資料檔案 （如果啟用登入資料檔案）	/eks-hybrid/.aws/credentials
AWS 簽署協助程式系統單位	/etc/systemd/system/aws_signing_helper_update.service
Sysctl conf 檔案	/etc/sysctl.d/99-nodeadm.conf
Ca-certificates	/etc/ssl/certs/ca-certificates.crt
Gpg 金鑰檔案	/etc/apt/keyrings/docker.asc
Docker 儲存庫來源檔案	/etc/apt/sources.list.d/docker.list

SSM 混合啟用的節點組態

以下是針對混合節點登入資料使用 AWS SSM 混合啟用nodeConfig.yaml的範例。

```
apiVersion: node.eks.aws/v1alpha1
kind: NodeConfig
spec:
  cluster:
    name:              # Name of the EKS cluster
    region:            # AWS Region where the EKS cluster resides
  hybrid:
    ssm:
      activationCode: # SSM hybrid activation code
      activationId:   # SSM hybrid activation id
```

IAM Roles Anywhere 的節點組態

以下是混合節點登入nodeConfig.yaml資料 AWS IAM Roles Anywhere 的範例。

使用 AWS IAM Roles Anywhere 做為內部部署憑證提供者時，nodeName您在nodeadm組態中使用的 必須符合您為混合節點 IAM 角色設定範圍的許可。例如，如果您的混合節點 IAM 角色許可僅允許 AWS IAM Roles Anywhere 在角色工作階段名稱等於主機憑證的 CN 時擔任角色，則nodeadm組態nodeName中的 必須與您憑證的 CN 相同。您使用nodeName的 長度不能超過 64 個字元。如需詳細資訊，請參閱the section called “準備登入資料”。

```
apiVersion: node.eks.aws/v1alpha1
kind: NodeConfig
spec:
  cluster:
    name:               # Name of the EKS cluster
    region:             # AWS Region where the EKS cluster resides
  hybrid:
    iamRolesAnywhere:
      nodeName:         # Name of the node
      trustAnchorArn:   # ARN of the IAM Roles Anywhere trust anchor
      profileArn:       # ARN of the IAM Roles Anywhere profile
      roleArn:          # ARN of the Hybrid Nodes IAM role
      certificatePath: # Path to the certificate file to authenticate with the IAM
 Roles Anywhere trust anchor
      privateKeyPath:  # Path to the private key file for the certificate
```

用於自訂 kubelet 的節點組態 （選用）

您可以在組態中傳遞 kubelet nodeadm組態和旗標。請參閱以下範例，了解如何新增額外的節點標籤abc.amazonaws.com/test-label，以及將 shutdownGracePeriod設定為 30 秒的組態。

```
apiVersion: node.eks.aws/v1alpha1
kind: NodeConfig
spec:
  cluster:
    name:             # Name of the EKS cluster
    region:           # AWS Region where the EKS cluster resides
  kubelet:
    config:           # Map of kubelet config and values
       shutdownGracePeriod: 30s
    flags:            # List of kubelet flags
       - --node-labels=abc.company.com/test-label=true
  hybrid:
    ssm:
      activationCode: # SSM hybrid activation code
      activationId:   # SSM hybrid activation id
```

用於自訂 containerd 的節點組態 （選用）

您可以在組態中傳遞自訂容器nodeadm組態。的容器化組態nodeadm接受內嵌 TOML。請參閱以下範例，了解如何設定容器化以停用刪除容器化內容存放區中的未封裝映像層。

```
apiVersion: node.eks.aws/v1alpha1
kind: NodeConfig
spec:
  cluster:
    name:             # Name of the EKS cluster
    region:           # AWS Region where the EKS cluster resides
  containerd:
    config: |         # Inline TOML containerd additional configuration
       [plugins."io.containerd.grpc.v1.cri".containerd]
       discard_unpacked_layers = false
  hybrid:
    ssm:
      activationCode: # SSM hybrid activation code
      activationId:   # SSM hybrid activation id
```

您也可以使用容器組態來啟用 SELinux 支援。在 containerd 上啟用 SELinux 時，請確保在節點上排程的 Pod 已啟用適當的 securityContext 和 seLinuxOptions。如需設定安全內容的詳細資訊，請參閱 Kubernetes 文件。

> **Note**
>
> Red Hat Enterprise Linux (RHEL) 8 和 RHEL 9 預設已啟用 SELinux，並在主機上設定為嚴格。Amazon Linux 2023 預設已啟用 SELinux，並設定為寬鬆模式。在主機上將 SELinux 設定為寬鬆模式時，在容器化 上啟用它不會封鎖請求，但會根據主機上的 SELinux 組態進行記錄。

```
apiVersion: node.eks.aws/v1alpha1
kind: NodeConfig
spec:
  cluster:
    name:             # Name of the EKS cluster
    region:           # AWS Region where the EKS cluster resides
  containerd:
    config: |         # Inline TOML containerd additional configuration
       [plugins."io.containerd.grpc.v1.cri"]
       enable_selinux = true
  hybrid:
    ssm:
      activationCode: # SSM hybrid activation code
      activationId:   # SSM hybrid activation id
```

對混合節點進行故障診斷

本主題涵蓋您在使用 Amazon EKS 混合節點時可能看到的一些常見錯誤，以及如何修正這些錯誤。如需其他疑難排解資訊，請參閱 AWS re：Post 上的 故障診斷Amazon EKS 知識中心標籤。https://repost.aws/tags/knowledge-center/TA4IvCeWI1TE66q4jEj4Z9zg/amazon-elastic-kubernetes-service如果您無法解決問題，請聯絡 AWS Support。

節點故障診斷 **nodeadm debug** 您可以從混合節點執行 nodeadm debug命令，以驗證是否符合聯網和憑證需求。如需 nodeadm debug命令的詳細資訊，請參閱 the section called “混合節點 nodeadm”。

使用叢集洞察偵測混合節點的問題 Amazon EKS 叢集洞察包括洞察檢查，以偵測叢集中 EKS 混合節點組態的常見問題。您可以從 AWS Management Console、 AWS CLI 和 AWS SDKs 檢視所有洞見檢查的結果。如需叢集洞見的詳細資訊，請參閱 the section called “叢集洞察”。

安裝混合節點疑難排解

下列疑難排解主題與使用 nodeadm install命令在主機上安裝混合節點相依性相關。

nodeadm 命令失敗 **must run as root**

nodeadm install 命令必須使用主機上具有根或sudo權限的使用者來執行。如果您nodeadm install使用沒有根或sudo權限的使用者執行 ，您會在nodeadm輸出中看到下列錯誤。

```
"msg":"Command failed","error":"must run as root"
```

無法連線至相依性

nodeadm install 命令會安裝混合節點所需的相依性。混合節點相依性包括 containerd、kubectl、 kubelet和 AWS SSM 或 AWS IAM Roles Anywhere 元件。您必須擁有執行所在位置的存取權nodeadm install，才能下載這些相依性。如需您必須能夠存取之位置清單的詳細資訊，請參閱 the section called “準備聯網”。如果您沒有存取權，您會在nodeadm install輸出中看到類似以下的錯誤。

```
"msg":"Command failed","error":"failed reading file from url: ...: max retries achieved
 for http request"
```

無法更新套件管理員

nodeadm install 命令會在安裝混合節點相依性dnf update之前執行 apt update 或 yum update或 。如果此步驟未成功，您可能會看到類似以下的錯誤。若要修復，您可以在執行 dnf update 之前執行 apt update yum update或 nodeadm install，也可以嘗試重新執行 nodeadm install。

```
failed to run update using package manager
```

超過逾時或內容截止日期

執行 時nodeadm install，如果您在安裝程序的各個階段看到錯誤，指出超過逾時或內容截止日期，則連線速度可能會很慢，導致無法在達到逾時之前安裝混合節點相依性。若要解決這些問題，您可以嘗試使用 中的 --timeout旗標nodeadm來延長下載相依性的逾時持續時間。

```
nodeadm install K8S_VERSION --credential-provider CREDS_PROVIDER --timeout 20m0s
```

連接混合節點疑難排解

本節中的疑難排解主題與使用 nodeadm init命令將混合節點連線至 EKS 叢集的程序相關。

操作錯誤或不支援的配置

執行 時nodeadm init，如果您看到與 operation error或 相關的錯誤unsupported scheme，請檢查您的 nodeConfig.yaml，以確保其格式正確並傳遞給 nodeadm。如需 的格式和選項的詳細資訊nodeConfig.yaml，請參閱 the section called “混合節點 nodeadm”。

```
"msg":"Command failed","error":"operation error ec2imds: GetRegion, request canceled,
 context deadline exceeded"
```

混合節點 IAM 角色缺少**eks:DescribeCluster**動作的許可

執行 時nodeadm init， nodeadm 會嘗試呼叫 Describe Cluster 來收集 EKS 叢集的相關資訊。如果您的混合節點 IAM 角色沒有 eks:DescribeCluster動作的許可。如需混合節點 IAM 角色所需許可的詳細資訊，請參閱 the section called “準備登入資料”。

```
"msg":"Command failed","error":"operation error EKS: DescribeCluster, https response
 error StatusCode: 403 ... AccessDeniedException"
```

不在遠端節點網路 CIDR 中的節點 IP

執行 時nodeadm init，如果節點的 IP 地址不在指定的遠端節點網路 CIDRs內，您可能會遇到錯誤。錯誤看起來與下列範例類似：

```
node IP 10.18.0.1 is not in any of the remote network CIDR blocks [10.0.0.0/16
 192.168.0.0/16]
```

此範例顯示 IP 10.18.0.1 的節點，嘗試加入具有遠端網路 CIDRs 10.0.0.0/16 和 192.168.0.0/16 的叢集。發生錯誤，因為 10.18.0.1 不在任一範圍內。

確認您已正確設定 RemoteNodeNetworks以包含所有節點 IP 地址。如需聯網組態的詳細資訊，請參閱 the section called “準備聯網”。

- 在叢集所在的區域中執行下列命令，以檢查您的RemoteNodeNetwork組態。確認輸出中列出的 CIDR 區塊包含節點的 IP 範圍，並且與錯誤訊息中列出的 CIDR 區塊相同。如果它們不相符，請確認您 中的叢集名稱和區域nodeConfig.yaml符合您預期的叢集。

```
aws eks describe-cluster --name CLUSTER_NAME --region REGION_NAME --query
 cluster.remoteNetworkConfig.remoteNodeNetworks
```

- 確認您正在使用預期的節點：
  - 檢查其主機名稱和 IP 地址是否符合您要向叢集註冊的 IP 地址，以確認您在正確的節點上。
  - 確認此節點位於正確的內部部署網路中 (CIDR 範圍在叢集設定RemoteNodeNetwork期間註冊為的節點）。

如果您的節點 IP 仍然不符合預期，請檢查下列項目：

- 如果您使用的是 IAM Roles Anywhere， kubelet 會在 IAM Roles Anywhere 上執行 DNS 查詢，nodeName並在可用時使用與節點名稱相關聯的 IP。如果您維護節點的 DNS 項目，請確認這些項目指向遠端節點網路 CIDRs內的 IPs。
- 如果您的節點有多個網路介面， kubelet可能會選取遠端節點網路 CIDRs 外部 IP 地址的介面做為預設。若要使用不同的界面，請使用 中的 --node-ipkubelet旗標指定其 IP 地址nodeConfig.yaml。如需詳細資訊，請參閱the section called “混合節點 nodeadm”。您可以在節點上執行下列命令，以檢視節點的網路介面及其 IP 地址：

```
ip addr show
```

混合節點不會出現在 EKS 叢集中

如果您執行 nodeadm init且已完成，但您的混合節點未出現在叢集中，混合節點和 EKS 控制平面之間的網路連線可能會出現問題，您可能未設定必要的安全群組許可，或者您可能沒有混合節點 IAM 角色與 Kubernetes 角色型存取控制 (RBAC) 的必要映射。您可以使用下列命令檢查 kubelet和kubelet日誌的狀態，以啟動偵錯程序。從無法加入叢集的混合節點執行下列命令。

```
systemctl status kubelet
```

```
journalctl -u kubelet -f
```

無法與叢集通訊

如果您的混合節點無法與叢集控制平面通訊，您可能會看到類似以下的日誌。

```
"Failed to ensure lease exists, will retry" err="Get ..."
```

```
"Unable to register node with API server" err="Post ..."
```

```
Failed to contact API server when waiting for CSINode publishing ... dial tcp <ip
 address>: i/o timeout
```

如果您看到這些訊息，請檢查下列項目，以確保其符合 中詳述的混合節點需求the section called “準備聯網”。

- 確認傳遞給 EKS 叢集的 VPC 具有您內部部署節點的 Transit Gateway (TGW) 或 Virtual Private Gateway (VGW) 路由，以及選擇性的 Pod CIDRs。
- 確認您的 EKS 叢集有額外的安全群組，其中包含內部部署節點 CIDRs 的傳入規則，以及選用的 Pod CIDRs。
- 確認您的現場部署路由器已設定為允許進出 EKS 控制平面的流量。

未授權

如果您的混合節點能夠與 EKS 控制平面通訊，但無法註冊，您可能會看到類似以下的日誌。請注意，以下日誌訊息中的主要差異是`Unauthorized`錯誤。這表示節點無法執行其任務，因為它沒有必要的許可。

```
"Failed to ensure lease exists, will retry" err="Unauthorized"
```

```
"Unable to register node with API server" err="Unauthorized"
```

```
Failed to contact API server when waiting for CSINode publishing: Unauthorized
```

如果您看到這些訊息，請檢查下列項目，以確保其符合 the section called “準備登入資料”和 中的混合節點需求詳細資訊the section called “準備叢集存取”。

- 確認混合節點的身分符合您預期的混合節點 IAM 角色。這可以透過`sudo aws sts get-caller-identity`從混合節點執行來完成。

- 確認您的混合節點 IAM 角色具有必要的許可。
- 確認您的叢集中有混合節點 IAM 角色的 EKS 存取項目，或確認您的 aws-auth ConfigMap 有混合節點 IAM 角色的項目。如果您使用的是 EKS 存取項目，請確認混合節點 IAM 角色的存取項目具有 HYBRID_LINUX存取類型。如果您使用的是 aws-auth ConfigMap，請確認混合節點 IAM 角色的項目符合 中詳述的要求和格式the section called “準備叢集存取”。

向 EKS 叢集註冊但顯示狀態的混合節點 **Not Ready**

如果您的混合節點已成功向 EKS 叢集註冊，但混合節點顯示狀態 Not Ready，則要檢查的第一件事是容器聯網界面 (CNI) 狀態。如果您尚未安裝 CNI，則預期混合節點的狀態為 Not Ready。安裝 CNI 並成功執行後，節點會更新為狀態 Ready。如果您嘗試安裝 CNI，但未成功執行，請參閱此頁面the section called “混合節點 CNI 疑難排解”的 。

憑證簽署請求 (CSRs) 停滯待定

將混合節點連接到 EKS 叢集後，如果您看到混合節點有待定CSRs，您的混合節點不符合自動核准的要求。如果混合節點CSRs CSRs 是由具有eks.amazonaws.com/compute-type: hybrid標籤的節點建立，且 CSR 具有下列主體別名 (SANs)，則會自動核准和發出混合節點的 CSR：至少有一個 DNS SAN 等於節點名稱，且 IP SANs屬於遠端節點網路 CIDRs。

混合設定檔已存在

如果您變更nodeadm組態並嘗試使用新組態重新註冊節點，您可能會看到錯誤，指出混合設定檔已存在，但其內容已變更。執行 nodeadm uninstall後接 nodeadm install和 ，而不是nodeadm init在組態變更之間執行 nodeadm init。這可確保正確清除組態中的變更。

```
"msg":"Command failed","error":"hybrid profile already exists at /etc/aws/hybrid/config
 but its contents do not align with the expected configuration"
```

混合節點無法解析私有 API

執行 後nodeadm init，如果您在kubelet日誌中看到錯誤，顯示由於有 而無法聯絡 EKS Kubernetes API 伺服器no such host，您可能需要變更內部部署網路或主機層級中 EKS Kubernetes API 端點的 DNS 項目。請參閱 AWS Route53 文件中的轉送傳入 DNS 查詢到您的 VPC。

```
Failed to contact API server when waiting for CSINode publishing: Get ... no such host
```

無法在 EKS 主控台中檢視混合節點

如果您已註冊混合節點，但無法在 EKS 主控台中檢視它們，請檢查您用來檢視主控台的 IAM 主體許可。您使用的 IAM 主體必須具有特定的最低 IAM 和 Kubernetes 許可，才能在 主控台中檢視資源。如需詳細資訊，請參閱the section called “存取叢集資源”。

執行混合節點疑難排解

如果您的混合節點已向 EKS 叢集註冊、狀態為 Ready，然後轉換至狀態 Not Ready，則可能會有各種問題導致運作狀態不佳，例如節點缺少足夠 CPU、記憶體或可用磁碟空間的資源，或節點與 EKS 控制平面中斷連線。您可以使用下列步驟對節點進行故障診斷，如果無法解決問題，請聯絡 AWS Support。

nodeadm debug 從混合節點執行 ，以驗證是否符合聯網和登入資料需求。如需 nodeadm debug命令的詳細資訊，請參閱 the section called “混合節點 nodeadm”。

取得節點狀態

```
kubectl get nodes -o wide
```

檢查節點條件和事件

```
kubectl describe node NODE_NAME
```

取得 Pod 狀態

```
kubectl get pods -A -o wide
```

檢查 Pod 條件和事件

```
kubectl describe pod POD_NAME
```

檢查 Pod 日誌

```
kubectl logs POD_NAME
```

檢查**kubectl**日誌

```
systemctl status kubelet
```

```
journalctl -u kubelet -f
```

Pod 即時性探查失敗或 Webhook 無法運作

如果混合節點上執行的應用程式、附加元件或 Webhook 未正確啟動，您可能會遇到網路問題，封鎖與 Pod 的通訊。若要讓 EKS 控制平面聯絡在混合節點上執行的 Webhook，您必須使用遠端 Pod 網路設定 EKS 叢集，並將 VPC 路由表中的現場部署 Pod CIDR 路由，目標為 Transit Gateway (TGW)、虛擬私有閘道 (VPW) 或您用來連接 VPC 與現場部署網路的其他閘道。如需混合節點聯網需求的詳細資訊，請參閱 the section called “準備聯網”。此外，您還必須在內部部署防火牆中允許此流量，並確保您的路由器可以正確路由到您的 Pod。the section called “設定 Webhook” 如需在混合節點上執行 Webhook 需求的詳細資訊，請參閱 。

此案例的常見 Pod 日誌訊息如下所示，其中 ip-address 是 Kubernetes 服務的叢集 IP。

```
dial tcp <ip-address>:443: connect: no route to host
```

`kubectl logs` 或 `kubectl exec` 命令無法運作

如果 `kubectl logs`或 `kubectl exec`命令在其他`kubectl`命令成功時逾時，問題可能與遠端網路組態有關。這些命令會透過叢集連線至節點上的`kubelet`端點。如需更多資訊，請參閱the section called “`kubelet` 端點”。

確認您的節點 IPs 和 Pod IPs位於為您的叢集設定的遠端節點網路和遠端 Pod 網路 CIDRs內。使用以下命令來檢查 IP 指派。

```
kubectl get nodes -o wide
```

```
kubectl get pods -A -o wide
```

將這些 IPs與您設定的遠端網路 CIDRs進行比較，以確保路由正確。如需網路組態需求，請參閱the section called “準備聯網”。

混合節點 CNI 疑難排解

如果您在一開始使用混合節點啟動 Cilium 或 Calico 時遇到問題，通常是由於混合節點或混合節點上執行的 CNI Pod 與 EKS 控制平面之間的聯網問題。請確定您的環境符合準備混合節點聯網的要求。將問題分成幾個部分很有用。

EKS 叢集組態

RemoteNodeNetwork 和 RemotePodNetwork 組態是否正確？

VPC 組態

VPC 路由表中是否有目標為 Transit Gateway 或 Virtual Private Gateway 的 RemoteNodeNetwork 和 RemotePodNetwork 路由？

安全群組組態

RemoteNodeNetwork 和 RemotePodNetwork 是否有傳入和傳出規則？

現場部署網路

是否有進出 EKS 控制平面以及混合節點和混合節點上執行之 Pod 的路由和存取權？

CNI 組態

如果使用浮水印網路，CNI 的 IP 集區組態是否與使用 Webhook 時為 EKS 叢集設定的 RemotePodNetwork 相符？

混合節點具有**Ready**未安裝 CNI 的狀態

如果您的混合節點顯示狀態 Ready，但您尚未在叢集上安裝 CNI，則混合節點上可能有舊的 CNI 成品。根據預設，當您使用 Helm 等工具解除安裝 Cilium 和 Calico 時，不會從實體或虛擬機器中移除磁碟上資源。此外，這些 CNIs 的自訂資源定義 (CRDs) 仍可能存在於舊安裝的叢集上。如需詳細資訊，請參閱 的 Delete Cilium and Delete Calico 章節the section called “設定 CNI”。

Cilium 疑難排解

如果您在混合節點上執行 Cilium 時遇到問題，請參閱 Cilium 文件中的疑難排解步驟。以下各節涵蓋在混合節點上部署 Cilium 時可能唯一的問題。

Cilium 未啟動

如果每個混合節點上執行的 Cilium 代理程式未啟動，請檢查 Cilium 代理程式 Pod 的日誌是否有錯誤。Cilium 代理程式需要連線到 EKS Kubernetes API 端點才能啟動。如果未正確設定此連線，則 Cilium 代理程式啟動會失敗。在此情況下，您會在 Cilium 代理程式 Pod 日誌中看到類似以下內容的日誌訊息。

```
msg="Unable to contact k8s api-server"
```

```
level=fatal msg="failed to start: Get \"https://<k8s-cluster-ip>:443/api/v1/namespaces/
kube-system\": dial tcp <k8s-cluster-ip>:443: i/o timeout"
```

Cilium 代理程式會在主機網路上執行。您的 EKS 叢集必須使用 設定RemoteNodeNetwork才能進行 Cilium 連線。確認您的 EKS 叢集有額外的安全群組，其中包含 的傳入規則RemoteNodeNetwork、您的 VPC 中有 的路由RemoteNodeNetwork，以及您的內部部署網路已正確設定，以允許連線至 EKS 控制平面。

如果 Cilium 運算子正在執行，且部分 Cilium 代理程式正在執行，但並非全部，請確認您有可用的 Pod IPs 可配置給叢集中的所有節點。您可以在 Cilium 組態clusterPoolIPv4PodCIDRList中使用叢集集區 IPAM 搭配 時，設定可配置的 Pod CIDRs 大小。每個節點 CIDR 大小是使用 Cilium 組態中的 clusterPoolIPv4MaskSize設定進行設定。如需詳細資訊，請參閱 Cilium 文件中的展開叢集集區。

Cilium BGP 無法運作

如果您使用 Cilium BGP 控制平面向內部部署網路公告您的 Pod 或服務地址，您可以使用下列 Cilium CLI 命令來檢查 BGP 是否向資源公告路由。如需安裝 Cilium CLI 的步驟，請參閱 Cilium 文件中的安裝 Cilium CLI。

```
cilium bgp peers
```

```
cilium bgp routes
```

如果您使用 Cilium BGP 公告類型為 的 服務 IPsLoadBalancer，則您的 CiliumLoadBalancerIPPool和 服務必須具有相同的標籤，這應該用於 的選擇器 中CiliumBGPAdvertisement。以下所示為範例。請注意，如果您使用 Cilium BGP 公告類型為 LoadBalancer 的服務 IPs，BGP 路由可能會在 Cilium 代理程式重新啟動期間中斷。如需詳細資訊，請參閱 Cilium 文件中的失敗案例。

服務

```
kind: Service
apiVersion: v1
metadata:
```

```
  name: guestbook
  labels:
    app: guestbook
spec:
  ports:
  - port: 3000
    targetPort: http-server
  selector:
    app: guestbook
  type: LoadBalancer
```

CiliumLoadBalancerIPPool

```
apiVersion: cilium.io/v2alpha1
kind: CiliumLoadBalancerIPPool
metadata:
  name: guestbook-pool
  labels:
    app: guestbook
spec:
  blocks:
  - cidr: <CIDR to advertise>
  serviceSelector:
    matchExpressions:
      - { key: app, operator: In, values: [ guestbook ] }
```

CiliumBGPAdvertisement

```
apiVersion: cilium.io/v2alpha1
kind: CiliumBGPAdvertisement
metadata:
  name: bgp-advertisements-guestbook
  labels:
    advertise: bgp
spec:
  advertisements:
    - advertisementType: "Service"
      service:
        addresses:
          - ExternalIP
          - LoadBalancerIP
      selector:
        matchExpressions:
```

```
          - { key: app, operator: In, values: [ guestbook ] }
```

Calico 疑難排解

如果您在混合節點上執行 Calico 時遇到問題，請參閱 Calico 文件中的疑難排解步驟。以下各節涵蓋在混合節點上部署 Calico 時可能唯一的問題。

下表摘要說明 Calico 元件，以及它們是否預設在節點或 Pod 網路上執行。如果您將 Calico 設定為使用 NAT 進行傳出 Pod 流量，則必須將內部部署網路設定為將流量路由到內部部署節點 CIDR，而且您的 VPC 路由表必須使用內部部署節點 CIDR 的路由，並將傳輸閘道 (TGW) 或虛擬私有閘道 (VGW) 作為目標。如果您未將 Calico 設定為使用 NAT 進行傳出 Pod 流量，則必須將內部部署網路設定為將流量路由到內部部署 Pod CIDR，而且您的 VPC 路由表必須使用內部部署 Pod CIDR 的路由，並將傳輸閘道 (TGW) 或虛擬私有閘道 (VGW) 作為目標。

元件	網路
Calico API 伺服器	節點
Kubernetes 的 Calico 控制器	Pod
Calico 節點代理程式	節點
Calico `typha`	節點
Calico CSI 節點驅動程式	Pod
Calico 運算子	節點

Calico 資源已排程或在封鎖節點上執行

根據預設，未以 DaemonSet 執行的 Calico 資源具有彈性的容錯能力，可在尚未準備好排程或執行 Pod 的封鎖節點上排程這些資源。您可以將您的運算子安裝變更為包含下列項目，以限制非 DaemonSet Calico 資源的容錯。

```
installation:
  ...
  controlPlaneTolerations:
  - effect: NoExecute
    key: node.kubernetes.io/unreachable
    operator: Exists
```

```
    tolerationSeconds: 300
  - effect: NoExecute
    key: node.kubernetes.io/not-ready
    operator: Exists
    tolerationSeconds: 300
  calicoKubeControllersDeployment:
    spec:
      template:
        spec:
          tolerations:
          - effect: NoExecute
            key: node.kubernetes.io/unreachable
            operator: Exists
            tolerationSeconds: 300
          - effect: NoExecute
            key: node.kubernetes.io/not-ready
            operator: Exists
            tolerationSeconds: 300
  typhaDeployment:
    spec:
      template:
        spec:
          tolerations:
          - effect: NoExecute
            key: node.kubernetes.io/unreachable
            operator: Exists
            tolerationSeconds: 300
          - effect: NoExecute
            key: node.kubernetes.io/not-ready
            operator: Exists
            tolerationSeconds: 300
```

登入資料疑難排解

對於 AWS SSM 混合啟用和 AWS IAM Roles Anywhere，您可以從混合節點執行下列命令，驗證混合節點 IAM 角色的登入資料是否已在您的混合節點上正確設定。確認節點名稱和混合節點 IAM 角色名稱符合您的期望。

```
sudo aws sts get-caller-identity
```

```
{
    "UserId": "ABCDEFGHIJKLM12345678910:<node-name>",
```

```
    "Account": "<aws-account-id>",
    "Arn": "arn:aws: sts::<aws-account-id>:assumed-role/<hybrid-nodes-iam-role/<node-
name>"
}
```

AWS Systems Manager (SSM) 疑難排解

如果您針對混合節點登入資料使用 AWS SSM 混合啟用，請注意 安裝在混合節點上的下列 SSM 目錄和成品nodeadm。如需 SSM 代理程式的詳細資訊，請參閱 AWS Systems Manager 使用者指南中的使用 SSM 代理程式。

描述	位置
SSM 代理程式	Ubuntu - /snap/amazon-ssm-agent/current/amazon-ssm-agent RHEL 和 AL2023 - /usr/bin/amazon-ssm-agent
SSM 代理程式日誌	/var/log/amazon/ssm
AWS 登入資料	/root/.aws/credentials
SSM 設定 CLI	/opt/ssm/ssm-setup-cli

重新啟動 SSM 代理程式

重新啟動 SSM 代理程式可以解決某些問題。您可以使用以下命令來重新啟動它。

AL2023 和其他作業系統

```
systemctl restart amazon-ssm-agent
```

Ubuntu

```
systemctl restart snap.amazon-ssm-agent.amazon-ssm-agent
```

檢查 SSM 端點的連線

確認您可以從混合節點連線至 SSM 端點。如需 SSM 端點的清單，請參閱 AWS Systems Manager 端點和配額。將以下命令us-west-2中的 取代為 AWS SSM 混合啟用 AWS 的區域。

```
ping ssm.us-west-2.amazonaws.com
```

檢視已註冊 SSM 執行個體的連線狀態

您可以使用下列 CLI 命令，檢查已向 SSM AWS 混合啟用註冊之執行個體的連線狀態。以執行個體的機器 ID 取代機器 ID。

```
aws ssm get-connection-status --target mi-012345678abcdefgh
```

SSM 設定 CLI 檢查總和不相符

執行時，nodeadm install如果您看到ssm-setup-cli檢查總和不相符的問題，您應該確認主機上沒有較舊的現有 SSM 安裝。如果您的主機上有較舊的 SSM 安裝，請將其移除並重新執行nodeadm install以解決問題。

```
Failed to perform agent-installation/on-prem registration: error while verifying
 installed ssm-setup-cli checksum: checksum mismatch with latest ssm-setup-cli.
```

SSM **InvalidActivation**

如果您看到向 AWS SSM 註冊執行個體時發生錯誤，請確認 activationId中的 activationCode、 region和 nodeConfig.yaml 正確無誤。EKS 叢集 AWS 的區域必須符合 SSM 混合啟用的區域。如果這些值設定錯誤，您可能會看到類似以下的錯誤。

```
ERROR Registration failed due to error registering the instance with AWS SSM.
 InvalidActivation
```

SSM **ExpiredTokenException**：請求中包含的安全字符已過期

如果 SSM 代理程式無法重新整理登入資料，您可能會看到 ExpiredTokenException。在此案例中，如果您能夠從混合節點連線至 SSM 端點，您可能需要重新啟動 SSM 代理程式來強制重新整理登入資料。

```
"msg":"Command failed","error":"operation error SSM: DescribeInstanceInformation, https
 response error StatusCode: 400, RequestID: eee03a9e-f7cc-470a-9647-73d47e4cf0be, api
 error ExpiredTokenException: The security token included in the request is expired"
```

執行 register machine 命令時發生 SSM 錯誤

如果您看到向 SSM 註冊機器時發生錯誤，您可能需要重新執行 nodeadm install ，以確保所有 SSM 相依性都已正確安裝。

```
"error":"running register machine command: , error: fork/exec /opt/aws/ssm-setup-cli:
 no such file or directory"
```

SSM **ActivationExpired**

執行 時nodeadm init，如果您因為啟用過期而看到向 SSM 註冊執行個體的錯誤，您需要建立新的 SSM 混合啟用、nodeConfig.yaml使用新 SSM 混合啟用activationId的 activationCode和更新 ，並重新執行 nodeadm init。

```
"msg":"Command failed","error":"SSM activation expired. Please use a valid activation"
```

```
ERROR Registration failed due to error registering the instance with AWS SSM.
 ActivationExpired
```

SSM 無法重新整理快取的登入資料

如果您看到重新整理快取登入資料失敗，您的主機上可能已刪除/root/.aws/credentials檔案。首先檢查您的 SSM 混合啟用，並確保其處於作用中狀態，且您的混合節點已正確設定為使用啟用。檢查位於 的 SSM 代理程式日誌，/var/log/amazon/ssm並在 SSM 端解決問題後重新執行nodeadm init命令。

```
"Command failed","error":"operation error SSM: DescribeInstanceInformation, get
 identity: get credentials: failed to refresh cached credentials"
```

清除 SSM

若要從主機移除 SSM 代理程式，您可以執行下列命令。

```
dnf remove -y amazon-ssm-agent
sudo apt remove --purge amazon-ssm-agent
snap remove amazon-ssm-agent
rm -rf /var/lib/amazon/ssm/Vault/Store/RegistrationKey
```

AWS IAM Roles Anywhere 疑難排解

如果您將 AWS IAM Roles Anywhere 用於混合節點憑證，請注意 安裝在混合節點上的下列目錄和成品nodeadm。如需故障診斷 IAM Roles Anywhere 的詳細資訊，請參閱《IAM Roles Anywhere AWS 使用者指南》中的故障診斷 IAM Roles Anywhere 身分和存取權。 AWS

描述	位置
IAM Roles Anywhere CLI	/usr/local/bin/aws_signing_helper
預設憑證位置和名稱	/etc/iam/pki/server.pem
預設金鑰位置和名稱	/etc/iam/pki/server.key

IAM Roles Anywhere 無法重新整理快取的登入資料

如果您看到重新整理快取登入資料失敗，請檢閱 的內容，/etc/aws/hybrid/config並確認nodeadm組態中的 IAM Roles Anywhere 已正確設定。確認 /etc/iam/pki存在。每個節點都必須有唯一的憑證和金鑰。根據預設，使用 IAM Roles Anywhere 做為登入資料提供者時， nodeadm 會將 /etc/iam/pki/server.pem 用於憑證位置和名稱，將 /etc/iam/pki/server.key 用於私有金鑰。您可能需要先建立目錄，再將憑證和金鑰放入具有 的目錄中sudo mkdir -p /etc/iam/pki。您可以使用以下命令來驗證憑證的內容。

```
openssl x509 -text -noout -in server.pem
```

```
open /etc/iam/pki/server.pem: no such file or directory
could not parse PEM data
Command failed {"error": "... get identity: get credentials: failed to refresh cached
 credentials, process provider error: error in credential_process: exit status 1"}
```

IAM Roles Anywhere 未獲授權執行 **sts:AssumeRole**

在 kubelet日誌中，如果您在使用 IAM Roles Anywhere 時看到 sts:AssumeRole操作的存取遭拒問題，請檢查混合節點 IAM 角色的信任政策，以確認允許 IAM Roles Anywhere 服務主體擔任混合節點 IAM 角色。此外，請確認您的混合節點 IAM 角色信任政策中已正確設定信任錨點 ARN，且您的混合節點 IAM 角色已新增至您的 IAM Roles Anywhere 設定檔。

```
could not get token: AccessDenied: User: ... is not authorized to perform:
 sts:AssumeRole on resource: ...
```

IAM Roles Anywhere 未獲授權設定 **roleSessionName**

在kubelet日誌中，如果您看到設定 的存取遭拒問題roleSessionName，請確認您已將 IAM Roles Anywhere 設定檔acceptRoleSessionName設為 true。

```
AccessDeniedException: Not authorized to set roleSessionName
```

作業系統疑難排解

RHEL

權利或訂閱管理員註冊失敗

如果您正在執行 nodeadm install，且由於權利註冊問題而無法安裝混合節點相依性，請確定您已在主機上正確設定 Red Hat 使用者名稱和密碼。

```
This system is not registered with an entitlement server
```

Ubuntu

找不到 GLIBC

如果您將 Ubuntu 用於作業系統，並將 IAM Roles Anywhere 用於具有混合節點的登入資料提供者，並發現 GLIBC 的問題，您可以手動安裝該相依性來解決問題。

```
GLIBC_2.32 not found (required by /usr/local/bin/aws_signing_helper)
```

執行下列命令來安裝相依性：

```
ldd --version
sudo apt update && apt install libc6
sudo apt install glibc-source
```

Bottlerocket

如果您已啟用 Bottlerocket 管理容器，則可以使用 SSH 存取它，以使用更高的權限進行進階偵錯和故障診斷。下列各節包含需要在 Bottlerocket 主機內容上執行的命令。進入管理員容器後，您可以執行sheltie以取得 Bottlerocket 主機中的完整根 shell。

```
sheltie
```

您也可以從管理員容器 Shell 的下列區段中執行命令，方法是在每個命令前面加上 sudo chroot /.bottlerocket/rootfs。

```
sudo chroot /.bottlerocket/rootfs <command>
```

使用 logdog 進行日誌收集

Bottlerocket 提供logdog公用程式，可有效收集日誌和系統資訊，以供故障診斷之用。

```
logdog
```

logdog 公用程式會從 Bottlerocket 主機上的不同位置收集日誌，並將其合併為 tarball。根據預設，tarball 會在 建立/var/log/support/bottlerocket-logs.tar.gz，並且可從 的主機容器存取/.bottlerocket/support/bottlerocket-logs.tar.gz。

使用 journalctl 存取系統日誌

您可以檢查各種系統服務的狀態，例如 kubelet、 等containerd，並使用下列命令檢視其日誌。-f 旗標會即時跟隨日誌。

若要檢查kubelet服務狀態和擷取kubelet日誌，您可以執行：

```
systemctl status kubelet
journalctl -u kubelet -f
```

若要檢查containerd服務狀態並擷取協同運作containerd執行個體的日誌，您可以執行：

```
systemctl status containerd
journalctl -u containerd -f
```

若要檢查host-containerd服務狀態並擷取主機containerd執行個體的日誌，您可以執行：

```
systemctl status host-containerd
journalctl -u host-containerd -f
```

若要擷取引導容器和主機容器的日誌，您可以執行：

```
journalctl _COMM=host-ctr -f
```

為您的叢集使用應用程式資料儲存

您可以搭配 Amazon EKS 使用一系列 AWS 儲存服務，以滿足應用程式的儲存需求。透過容器儲存介面 (CSI) 驅動程式的 AWS支援廣度，您可以輕鬆使用 Amazon EBS、Amazon S3、Amazon EFS、Amazon FSX 和 Amazon File Cache，以滿足在 Amazon EKS 上執行之應用程式的儲存需求。

本章涵蓋 Amazon EKS 叢集的儲存選項。

主題

- 搭配 Amazon EBS 使用 Kubernetes 磁碟區儲存
- 搭配 Amazon EFS 使用彈性檔案系統儲存
- 搭配 Amazon FSx for Lustre 使用高效能應用程式儲存
- 搭配 FSx for NetApp ONTAP 使用高效能應用程式儲存
- 搭配 Amazon FSx for OpenZFS 使用資料儲存
- 使用 Amazon File Cache 將延遲降至最低
- 使用適用於 Amazon S3 CSI 驅動程式的掛載點存取 Amazon S3 物件
- 啟用 CSI 磁碟區的快照功能

搭配 Amazon EBS 使用 Kubernetes 磁碟區儲存

> **Note**
>
> 新功能：Amazon EKS Auto Mode 會自動執行區塊儲存的例行任務。了解如何 the section called “部署具狀態工作負載”。

Amazon Elastic Block Store (Amazon EBS) 容器儲存介面 (CSI) 驅動程式會管理 Amazon EBS 磁碟區的生命週期，做為您建立之 Kubernetes 磁碟區的儲存體。Amazon EBS CSI 驅動程式會為這些類型的 Kubernetes 磁碟區建立 Amazon EBS 磁碟區：一般暫時性磁碟區和持久性磁碟區。

考量事項

- 您不需要在 EKS Auto Mode 叢集上安裝 Amazon EBS CSI 控制器。
- 您無法將 Amazon EBS 磁碟區掛載至 Fargate Pod。

- 您可以在 Fargate 節點上執行 Amazon EBS CSI 控制器，但是 Amazon EBS CSI 節點 DaemonSet 僅能在 Amazon EC2 執行個體上執行。
- Amazon EBS 磁碟區和 Amazon EBS CSI 驅動程式與 Amazon EKS 混合節點不相容。
- 將提供最新附加元件版本和一個先前版本的支援。最新版本中發現的錯誤或漏洞將回溯到新次要版本中的先前版本。
- EKS Auto Mode 需要使用儲存類別ebs.csi.eks.amazonaws.com做為佈建器。標準 Amazon EBS CSI 驅動程式 (ebs.csi.aws.com) 會分別管理自己的磁碟區。若要搭配 EKS Auto Mode 使用現有磁碟區，請使用磁碟區快照將它們遷移至使用 Auto Mode 佈建器的儲存類別。

> **Important**
>
> 若要使用 Amazon EBS CSI 驅動程式的快照功能，您必須先安裝 CSI 快照控制器。如需詳細資訊，請參閱the section called “CSI 快照控制器”。

先決條件

- 現有的叢集。若要查看所需的平台版本，請執行下列命令。

```
aws eks describe-addon-versions --addon-name aws-ebs-csi-driver
```

- EBS CSI 驅動程式需要 AWS IAM 許可。
  - AWS 建議使用 EKS Pod 身分。如需詳細資訊，請參閱the section called “設定 EKS Pod 身分識別的概觀”。
  - 如需服務帳戶的 IAM 角色相關資訊，請參閱 the section called “IAM OIDC 供應商”。

步驟 1：建立 IAM 角色

Amazon EBS CSI 外掛程式需要 IAM 許可，才能代表您呼叫 AWS APIs。如果您不執行這些步驟，嘗試安裝附加元件並執行 kubectl describe pvc 會顯示 failed to provision volume with StorageClass並顯示could not create volume in EC2: UnauthorizedOperation錯誤。如需詳細資訊，請參閱 GitHub 上的設定驅動程式許可。

> **Note**
>
> 除非您封鎖對 IMDS 的存取，否則 Pod 將可存取指派給 IAM 角色的許可。如需詳細資訊，請參閱the section called “最佳實務”。

下列程序說明如何建立 IAM 角色，並將 AWS 受管政策連接至該角色。若要實作此程序，您可以使用下列其中一個工具：

- the section called “eksctl”
- the section called “AWS Management Console”
- the section called “AWS CLI”

> **Note**
>
> 本程序中的特定步驟是針對將驅動程式用作 Amazon EKS 附加元件而編寫。需要執行不同的步驟，才能將驅動程式用作自我管理附加元件。如需詳細資訊，請參閱在 GitHub 上設定驅動程式許可。

eksctl

1. 建立 IAM 角色並連接政策。 AWS 維護 AWS 受管政策，或者您可以建立自己的自訂政策。您可以使用下列命令建立 IAM 角色並連接 AWS 受管政策。使用您叢集的名稱取代 *my-cluster*。命令會部署建立 IAM 角色並將 IAM 政策連接至其中的 AWS CloudFormation 堆疊。

```
eksctl create iamserviceaccount \
        --name ebs-csi-controller-sa \
        --namespace kube-system \
        --cluster my-cluster \
        --role-name AmazonEKS_EBS_CSI_DriverRole \
        --role-only \
        --attach-policy-arn arn:aws: iam::aws:policy/service-role/
AmazonEBSCSIDriverPolicy \
        --approve
```

2. 如果您不使用自訂 KMS 金鑰，可以略過此步驟。如果您在 Amazon EBS 磁碟區上使用一個 進行加密，請視需要自訂 IAM 角色。例如，請執行以下操作：

a. 複製以下程式碼並貼到新 kms-key-for-encryption-on-ebs.json 檔案中。將 *custom-key-arn* 取代為自訂 KMS 金鑰 ARN。

```
{
      "Version": "2012-10-17",
      "Statement": [
        {
          "Effect": "Allow",
          "Action": [
            "kms:CreateGrant",
            "kms:ListGrants",
            "kms:RevokeGrant"
          ],
          "Resource": ["custom-key-arn"],
          "Condition": {
            "Bool": {
              "kms:GrantIsForAWSResource": "true"
            }
          }
        },
        {
          "Effect": "Allow",
          "Action": [
            "kms:Encrypt",
            "kms:Decrypt",
            "kms:ReEncrypt*",
            "kms:GenerateDataKey*",
            "kms:DescribeKey"
          ],
          "Resource": ["custom-key-arn"]
        }
      ]
    }
```

b. 建立政策。您可以將 *KMS_Key_For_Encryption_On_EBS_Policy* 變更為不同的名稱。但是，如果您這樣做，請務必在稍後的步驟中也進行變更。

```
aws iam create-policy \
      --policy-name KMS_Key_For_Encryption_On_EBS_Policy \
      --policy-document file://kms-key-for-encryption-on-ebs.json
```

c. 使用以下命令將 IAM 政策連接至角色。使用您的帳戶 ID 取代 *111122223333*。

```
aws iam attach-role-policy \
      --policy-arn arn:aws: iam::111122223333:policy/
KMS_Key_For_Encryption_On_EBS_Policy \
      --role-name AmazonEKS_EBS_CSI_DriverRole
```

AWS Management Console

1. 前往 https://console.aws.amazon.com/iam/ 開啟 IAM 主控台。
2. 在左側導覽窗格中，選擇 Roles (角色)。
3. 在 Roles (角色) 頁面上，選擇 Create role (建立角色)。
4. 在 Select trusted entity (選取信任的實體) 頁面上，執行以下作業：
   a. 在 Trusted entity type (信任的實體類型) 區段中，選擇 Web identity (Web 身分)。
   b. 對於 Identity provider (身分提供者)，為您的叢集選擇 OpenID Connect provider URL (OpenID Connect 供應商 URL)(如 Amazon EKS Overview (概觀) 下所示)。
   c. 針對 Audience (對象)，選擇 `sts.amazonaws.com`。
   d. 選擇下一步。
5. 在 Add permissions (新增許可) 頁面上，執行以下作業：
   a. 在 Filter policies (篩選政策) 方塊中，輸入 `AmazonEBSCSIDriverPolicy`。
   b. 勾選在搜尋中傳回的 `AmazonEBSCSIDriverPolicy` 左側的核取方塊。
   c. 選擇下一步。
6. 在 Name, review, and create (命名、檢閱和建立) 頁面上，執行以下作業：
   a. 針對角色名稱，輸入角色的唯一名稱，例如 *`AmazonEKS_EBS_CSI_DriverRole`*。
   b. 藉由連接標籤做為鍵值對，在新增標籤 (選用) 下將中繼資料新增至角色。如需有關在 IAM 中使用標籤的詳細資訊，請參閱《IAM 使用者指南》中的標記 IAM 資源。
   c. 選擇建立角色。
7. 建立角色之後，在主控台中選擇角色，以開啟角色進行編輯。
8. 選擇 Trust Relationships (信任關係) 標籤，然後選擇 Edit Trust Relationship (編輯信任政策)。
9. 查找與下列行相似的行：

```
"oidc.eks.region-code.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE:aud":
 "sts.amazonaws.com"
```

在上一行的末尾新增一個逗號，然後在上一行之後新增下一行。將 *region-code* 取代為您的叢集所在的 AWS 區域。使用叢集的 OIDC 提供者 ID 取代 *EXAMPLED539D4633E53DE1B71EXAMPLE*。

```
"oidc.eks.region-code.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE:sub":
 "system:serviceaccount:kube-system:ebs-csi-controller-sa"
```

10選擇 Update policy (更新政策) 以完成操作。

11如果在 Amazon EBS 磁碟區上使用自訂 KMS 金鑰進行加密，請視需要自訂 IAM 角色。例如，請執行以下操作：

a. 在左側導覽窗格中選擇 Policies (政策)。

b. 在 Policies (政策) 頁面上，選擇 Create a policy (建立政策)。

c. 在建立政策頁面上，選擇 JSON 標籤。

d. 將下列程式碼複製並貼到編輯器中，以自訂 KMS 金鑰 ARN 取代 *custom-key-arn*。

```
{
      "Version": "2012-10-17",
      "Statement": [
        {
          "Effect": "Allow",
          "Action": [
            "kms:CreateGrant",
            "kms:ListGrants",
            "kms:RevokeGrant"
          ],
          "Resource": ["custom-key-arn"],
          "Condition": {
            "Bool": {
              "kms:GrantIsForAWSResource": "true"
            }
          }
        },
        {
          "Effect": "Allow",
          "Action": [
            "kms:Encrypt",
            "kms:Decrypt",
            "kms:ReEncrypt*",
            "kms:GenerateDataKey*",
            "kms:DescribeKey"
```

```
            ],
            "Resource": ["custom-key-arn"]
        }
      ]
    }
```

e. 選擇下一步：標籤。

f. 在 Add tags (Optional) (新增標籤 (選用)) 頁面上，選擇 Next: Review (下一步：檢閱)。

g. 針對名稱，輸入政策的唯一名稱 （ 例如 *KMS_Key_For_Encryption_On_EBS_Policy*)。

h. 選擇建立政策。

i. 在左側導覽窗格中，選擇 Roles (角色)。

j. 在主控台中選擇 ***AmazonEKS_EBS_CSI_DriverRole*** 以開啟它進行編輯。

k. 在新增許可下拉式清單中，選擇連接政策。

l. 在篩選政策方塊中，輸入 *KMS_Key_For_Encryption_On_EBS_Policy*。

m. 選取搜尋中傳回的 *KMS_Key_For_Encryption_On_EBS_Policy* 左側的核取方塊。

n. 選擇連接政策。

AWS CLI

1. 檢視叢集的 OIDC 提供者 URL。使用您的叢集名稱取代 *my-cluster*。如果來自命令的輸出是 None，則請檢閱先決條件。

```
aws eks describe-cluster --name my-cluster --query "cluster.identity.oidc.issuer" --
output text
```

範例輸出如下。

```
https://oidc.eks.region-code.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE
```

2. 建立 IAM 角色，授予其 AssumeRoleWithWebIdentity 動作。

a. 將下列內容複製到名為 的檔案aws-ebs-csi-driver-trust-policy.json。使用您的帳戶 ID 取代 *111122223333*。將 *EXAMPLED539D4633E53DE1B71EXAMPLE* 和 *region-code* 取代為上一個步驟中傳回的值。

```
{
      "Version": "2012-10-17",
      "Statement": [
```

```
        {
          "Effect": "Allow",
          "Principal": {
            "Federated": "arn:aws: iam::111122223333:oidc-provider/
oidc.eks.region-code.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE"
          },
          "Action": "sts:AssumeRoleWithWebIdentity",
          "Condition": {
            "StringEquals": {
              "oidc.eks.region-code.amazonaws.com/id/
EXAMPLED539D4633E53DE1B71EXAMPLE:aud": "sts.amazonaws.com",
              "oidc.eks.region-code.amazonaws.com/id/
EXAMPLED539D4633E53DE1B71EXAMPLE:sub": "system:serviceaccount:kube-system:ebs-csi-
controller-sa"
            }
          }
        }
      ]
    }
```

b. 建立角色。您可以將 *AmazonEKS_EBS_CSI_DriverRole* 變更為不同的名稱。若要變更名稱，請務必在稍後的步驟中進行變更。

```
aws iam create-role \
      --role-name AmazonEKS_EBS_CSI_DriverRole \
      --assume-role-policy-document file://"aws-ebs-csi-driver-trust-policy.json"
```

3. 連接政策。 會 AWS 維護 AWS 受管政策，或者您可以建立自己的自訂政策。使用下列命令將 AWS 受管政策連接至角色。

```
aws iam attach-role-policy \
      --policy-arn arn:aws: iam::aws:policy/service-role/AmazonEBSCSIDriverPolicy \
      --role-name AmazonEKS_EBS_CSI_DriverRole
```

4. 如果在 Amazon EBS 磁碟區上使用自訂 KMS 金鑰進行加密，請視需要自訂 IAM 角色。例如，請執行以下操作：

a. 複製以下程式碼並貼到新 kms-key-for-encryption-on-ebs.json 檔案中。將 *custom-key-arn* 取代為自訂 KMS 金鑰 ARN。

```
{
      "Version": "2012-10-17",
      "Statement": [
```

```
        {
          "Effect": "Allow",
          "Action": [
            "kms:CreateGrant",
            "kms:ListGrants",
            "kms:RevokeGrant"
          ],
          "Resource": ["custom-key-arn"],
          "Condition": {
            "Bool": {
              "kms:GrantIsForAWSResource": "true"
            }
          }
        },
        {
          "Effect": "Allow",
          "Action": [
            "kms:Encrypt",
            "kms:Decrypt",
            "kms:ReEncrypt*",
            "kms:GenerateDataKey*",
            "kms:DescribeKey"
          ],
          "Resource": ["custom-key-arn"]
        }
      ]
    }
```

b. 建立政策。您可以將 *KMS_Key_For_Encryption_On_EBS_Policy* 變更為不同的名稱。但是，如果您這樣做，請務必在稍後的步驟中也進行變更。

```
aws iam create-policy \
      --policy-name KMS_Key_For_Encryption_On_EBS_Policy \
      --policy-document file://kms-key-for-encryption-on-ebs.json
```

c. 使用以下命令將 IAM 政策連接至角色。使用您的帳戶 ID 取代 *111122223333*。

```
aws iam attach-role-policy \
      --policy-arn arn:aws: iam::111122223333:policy/
KMS_Key_For_Encryption_On_EBS_Policy \
      --role-name AmazonEKS_EBS_CSI_DriverRole
```

現在您已建立 Amazon EBS CSI 驅動程式 IAM 角色，您可以繼續下一節。當您使用此 IAM 角色部署附加元件時，它會建立並設定為使用名為 的服務帳戶ebs-csi-controller-sa。服務帳戶繫結至clusterrole獲指派所需 Kubernetes 許可的 Kubernetes。

步驟 2：取得 Amazon EBS CSI 驅動程式

我們建議您透過 Amazon EKS 附加元件安裝 Amazon EBS CSI 驅動程式，以提高安全性並減少工作量。若要將 Amazon EKS 附加元件新增至叢集，請參閱 the section called “建立附加元件”。如需附加元件的詳細資訊，請參閱 the section called “Amazon EKS 附加元件”。

> **Important**
>
> 將 Amazon EBS 驅動程式新增為 Amazon EKS 附加元件之前，請確認叢集上沒有安裝自我管理版本的驅動程式。若是如此，請參閱在 GitHub 上解除安裝自我管理的 Amazon EBS CSI 驅動程式。

或者，如果您想要自我管理安裝 Amazon EBS CSI 驅動程式，請參閱 GitHub 上的安裝。

步驟 3：部署範例應用程式

您可以部署各種範例應用程式，再視需要進行修改。如需詳細資訊，請參閱 GitHub 上的 Kubernetes 範例。

搭配 Amazon EFS 使用彈性檔案系統儲存

Amazon Elastic File System (Amazon EFS) 提供無伺服器、完全彈性的檔案儲存功能，讓您無需佈建或管理儲存容量和效能，即可分享檔案資料。Amazon EFS 容器儲存界面 (CSI) 驅動程式提供 CSI 界面，允許在 上執行的 Kubernetes 叢集 AWS 管理 Amazon EFS 檔案系統的生命週期。本主題旨在說明如何將 Amazon EFS CSI 驅動程式部署到您的 Amazon EKS 叢集。

考量事項

- Amazon EFS CSI 驅動程式與 Windows 型容器映像不相容。
- 您無法將動態佈建用於具有 Fargate 節點的持久性磁碟區，但您可以使用靜態佈建。
- 動態佈建需要 1.2 或更新版本的驅動程式。您可以在任何支援的 Amazon EKS 叢集版本 （ 請參閱 eks/latest/userguide/kubernetes-versions.html【Amazon EKS 支援的版本，type="documentation"】）上使用1.1驅動程式版本，將靜態佈建用於持久性磁碟區。

- 此驅動程式的 1.3.2 版或更新版本支援 Arm64 架構，包括以 Amazon EC2 Graviton 為基礎的執行個體。
- 此驅動程式的 1.4.2 版或更新版本支援使用 FIPS 掛載檔案系統。
- 記下 Amazon EFS 的資源配額。例如，您可以為每個 Amazon EFS 檔案系統建立 1000 個存取點的配額。如需詳細資訊，請參閱無法變更的 Amazon EFS 資源配額。
- 從 2.0.0 版開始，此驅動程式會從使用 切換至 `stunnel` `efs-proxy` 以進行 TLS 連線。使用 `efs-proxy` 時，它會開啟相當於 1 的執行緒數目，加上其執行節點的核心數目。
- Amazon EFS CSI 驅動程式與 Amazon EKS 混合節點不相容。

先決條件

- Amazon EFS CSI 驅動程式需要 AWS Identity and Access Management (IAM) 許可。
  - AWS 建議使用 EKS Pod 身分。如需詳細資訊，請參閱the section called “設定 EKS Pod 身分識別的概觀”。
  - 如需服務帳戶 IAM 角色和為您的叢集設定 IAM OpenID Connect (OIDC) 供應商的相關資訊，請參閱 the section called “IAM OIDC 供應商”。
- 在您的裝置或 AWS CloudShell 上安裝和設定的 AWS 命令列界面 (AWS CLI) 版本 `1.27.160` `2.12.3`或更新版本。若要檢查您目前的版本，請使用 `aws --version | cut -d / -f2 | cut -d ' ' -f1`。適用於 macOS 的 `yum`、 `apt-get`或 Homebrew 等套件管理員通常是最新版本 CLI AWS 後面的幾個版本。若要安裝最新版本，請參閱《 AWS 命令列界面使用者指南》中的使用 aws 設定安裝 和快速組態。 https://docs.aws.amazon.com/cli/latest/userguide/cli-configure-quickstart.html#cli-configure-quickstart-config安裝在 AWS CloudShell 中的 AWS CLI 版本也可能是最新版本後面的幾個版本。若要更新它，請參閱《CloudShell AWS 使用者指南》中的將 CLI 安裝到您的主目錄。 AWS CloudShell
- `kubectl` 命令列工具安裝在您的裝置或 AWS CloudShell 上。版本可以與叢集的 Kubernetes 版本相同，也可以比叢集的 Kubernetes 版本更早或更晚一個次要版本。例如，如果您的叢集版本為 `1.29`，則可以搭配使用 `kubectl` `1.28`、`1.29` 或 `1.30` 版。若要安裝或升級 `kubectl`，請參閱 the section called “設定 kubectl 和 eksctl”。

> **Note**
>
> 在 Fargate 上執行的 Pod 會自動掛載 Amazon EFS 檔案系統，而不需要手動驅動程式安裝步驟。

步驟 1：建立 IAM 角色

Amazon EFS CSI 驅動程式需要 IAM 許可才能與檔案系統互動。建立 IAM 角色，並將所需的 AWS 受管政策連接到該角色。若要實作此程序，您可以使用下列其中一個工具：

- the section called “eksctl”
- the section called “AWS Management Console”
- the section called “AWS CLI”

> **Note**
>
> 本程序中的特定步驟是針對將驅動程式用作 Amazon EKS 附加元件而編寫。如需自我管理安裝的詳細資訊，請參閱在 GitHub 上設定驅動程式許可。

eksctl

如果使用 Pod 身分

執行下列命令來建立與 的 IAM 角色和 Pod Identity 關聯eksctl。使用您的叢集名稱取代 my-cluster。您也可以AmazonEKS_EFS_CSI_DriverRole將 取代為不同的名稱。

```
export cluster_name=my-cluster
export role_name=AmazonEKS_EFS_CSI_DriverRole
eksctl create podidentityassociation \
    --service-account-name efs-csi-controller-sa \
    --namespace kube-system \
    --cluster $cluster_name \
    --role-name $role_name \
    --permission-policy-arns arn:aws: iam::aws:policy/service-role/
AmazonEFSCSIDriverPolicy
```

如果針對服務帳戶使用 IAM 角色

執行下列命令以使用 建立 IAM 角色eksctl。使用您的叢集名稱取代 my-cluster。您也可以AmazonEKS_EFS_CSI_DriverRole將 取代為不同的名稱。

```
export cluster_name=my-cluster
export role_name=AmazonEKS_EFS_CSI_DriverRole
eksctl create iamserviceaccount \
```

```
    --name efs-csi-controller-sa \
    --namespace kube-system \
    --cluster $cluster_name \
    --role-name $role_name \
    --role-only \
    --attach-policy-arn arn:aws: iam::aws:policy/service-role/AmazonEFSCSIDriverPolicy
 \
    --approve
TRUST_POLICY=$(aws iam get-role --output json --role-name $role_name --query
 'Role.AssumeRolePolicyDocument' | \
    sed -e 's/efs-csi-controller-sa/efs-csi-*/' -e 's/StringEquals/StringLike/')
aws iam update-assume-role-policy --role-name $role_name --policy-document
 "$TRUST_POLICY"
```

AWS Management Console

執行下列動作以使用 建立 IAM 角色 AWS Management Console。

1. 前往 https://console.aws.amazon.com/iam/ 開啟 IAM 主控台。
2. 在左側導覽窗格中，選擇 Roles (角色)。
3. 在 Roles (角色) 頁面上，選擇 Create role (建立角色)。
4. 在 Select trusted entity (選取信任的實體) 頁面上，執行以下作業：
   a. 如果使用 EKS Pod 身分：
      i. 在信任的實體類型區段中，選擇 AWS 服務。
      ii. 在服務或使用案例下拉式清單中，選擇 EKS。
      iii. 在使用案例區段中，選擇 EKS - Pod 身分。
      iv. 選擇下一步。
   b. 如果針對服務帳戶使用 IAM 角色：
      i. 在 Trusted entity type (信任的實體類型) 區段中，選擇 Web identity (Web 身分)。
      ii. 對於 Identity provider (身分提供者)，為您的叢集選擇 OpenID Connect provider URL (OpenID Connect 供應商 URL)(如 Amazon EKS Overview (概觀) 下所示)。
      iii. 針對 Audience (對象)，選擇 `sts.amazonaws.com`。
      iv. 選擇下一步。
5. 在 Add permissions (新增許可) 頁面上，執行以下作業：
   a. 在 Filter policies (篩選政策) 方塊中，輸入 `AmazonEFSCSIDriverPolicy`。
   b. 勾選在搜尋中傳回的 `AmazonEFSCSIDriverPolicy` 左側的核取方塊。

c. 選擇下一步。

6. 在 Name, review, and create (命名、檢閱和建立) 頁面上，執行以下作業：

a. 針對 Role name (角色名稱)，為您的角色輸入唯一名稱 (例如 `AmazonEKS_EFS_CSI_DriverRole`)。

b. 藉由連接標籤做為鍵值對，在新增標籤 (選用) 下將中繼資料新增至角色。如需有關在 IAM 中使用標籤的詳細資訊，請參閱《IAM 使用者指南》中的標記 IAM 資源。

c. 選擇建立角色。

7. 建立角色之後：

a. 如果使用 EKS Pod 身分：

i. 開啟 Amazon EKS 主控台。

ii. 在左側導覽窗格中，選取叢集，然後選取您要為其設定 EKS Pod 身分關聯的叢集名稱。

iii. 選擇存取索引標籤。

iv. 在 Pod 身分關聯中，選擇建立。

v. 選擇 IAM 角色下拉式清單，然後選取新建立的角色。

vi. 選擇 Kubernetes 命名空間欄位並輸入 `kube-system`。

vii. 選擇 Kubernetes 服務帳戶欄位並輸入 `efs-csi-controller-sa`。

viii 選擇建立。

ix. 如需建立 Pod Identity 關聯的詳細資訊，請參閱 the section called “建立 Pod Identity 關聯 AWS （主控台）”。

b. 如果針對服務帳戶使用 IAM 角色：

i. 選擇角色以開啟角色進行編輯。

ii. 選擇 Trust Relationships (信任關係) 標籤，然後選擇 Edit Trust Relationship (編輯信任政策)。

iii. 查找與下列行相似的行：

```
"oidc.eks.region-code.amazonaws.com/id/<EXAMPLED539D4633E53DE1B71EXAMPLE>:aud":
 "sts.amazonaws.com"
```

在前一行上方加入下列行。`<region-code>` 將 取代為叢集所在的 AWS 區域。`<EXAMPLED539D4633E53DE1B71EXAMPLE>` 使用叢集的 OIDC 提供者 ID 取代 。

```
"oidc.eks.<region-code>.amazonaws.com/id/
<EXAMPLED539D4633E53DE1B71EXAMPLE>:sub": "system:serviceaccount:kube-system:efs-
csi-*",
```

iv. 將 Condition 運算子從 "StringEquals" 修改為 "StringLike"。

v. 選擇 Update policy (更新政策) 以完成操作。

AWS CLI

執行下列命令以使用 CLI 建立 IAM AWS 角色。

如果使用 Pod 身分

1. 建立將 AssumeRole和 TagSession動作授予pods.eks.amazonaws.com服務的 IAM 角色。

   a. 將下列內容複製到名為 aws-efs-csi-driver-trust-policy-pod-identity.json 的檔案。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowEksAuthToAssumeRoleForPodIdentity",
            "Effect": "Allow",
            "Principal": {
                "Service": "pods.eks.amazonaws.com"
            },
            "Action": [
                "sts:AssumeRole",
                "sts:TagSession"
            ]
        }
    ]
}
```

   b. 建立角色。使用您的叢集名稱取代 my-cluster。您也可以AmazonEKS_EFS_CSI_DriverRole將 取代為不同的名稱。

```
export cluster_name=my-cluster
export role_name=AmazonEKS_EFS_CSI_DriverRole
aws iam create-role \
  --role-name $role_name \
  --assume-role-policy-document file://"aws-efs-csi-driver-trust-policy-pod-
identity.json"
```

2. 使用下列命令將所需的 AWS 受管政策連接至角色。

```
aws iam attach-role-policy \
  --policy-arn arn:aws: iam::aws:policy/service-role/AmazonEFSCSIDriverPolicy \
  --role-name $role_name
```

3. 執行下列命令來建立 Pod Identity 關聯。 `arn:aws: iam::<111122223333>:role/my-role` 將 取代為先前步驟中建立的角色。

```
aws eks create-pod-identity-association --cluster-name $cluster_name --role-arn {arn-aws}iam::<111122223333>:role/my-role --namespace kube-system --service-account efs-csi-controller-sa
```

4. 如需建立 Pod Identity 關聯的詳細資訊，請參閱 the section called “建立 Pod Identity 關聯AWS（主控台）”。

如果針對服務帳戶使用 IAM 角色

1. 檢視叢集的 OIDC 提供者 URL。使用您的叢集名稱取代 `my-cluster`。您也可以`AmazonEKS_EFS_CSI_DriverRole`將 取代為不同的名稱。

```
export cluster_name=my-cluster
export role_name=AmazonEKS_EFS_CSI_DriverRole
aws eks describe-cluster --name $cluster_name --query "cluster.identity.oidc.issuer"
 --output text
```

   範例輸出如下。

```
https://oidc.eks.<region-code>.amazonaws.com/id/<EXAMPLED539D4633E53DE1B71EXAMPLE>
```

   如果來自命令的輸出是 None，則請檢閱先決條件。

2. 建立授予 `AssumeRoleWithWebIdentity` 動作許可的 IAM 角色。

   a. 將下列內容複製到名為 `aws-efs-csi-driver-trust-policy.json` 的檔案。使用您的帳戶 ID 取代 <111122223333>。使用前一個步驟傳回的值取代 <EXAMPLED539D4633E53DE1B71EXAMPLE> 和 <region-code>。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
```

```
      "Effect": "Allow",
      "Principal": {
        "Federated": "arn:aws: iam::<111122223333>:oidc-provider/oidc.eks.<region-
code>.amazonaws.com/id/<EXAMPLED539D4633E53DE1B71EXAMPLE>"
      },
      "Action": "sts:AssumeRoleWithWebIdentity",
      "Condition": {
        "StringLike": {
          "oidc.eks.region-code.amazonaws.com/id/
<EXAMPLED539D4633E53DE1B71EXAMPLE>:sub": "system:serviceaccount:kube-system:efs-
csi-*",
          "oidc.eks.region-code.amazonaws.com/id/
<EXAMPLED539D4633E53DE1B71EXAMPLE>:aud": "sts.amazonaws.com"
        }
      }
    }
  ]
}
```

b. 建立角色。

```
aws iam create-role \
  --role-name $role_name \
  --assume-role-policy-document file://"aws-efs-csi-driver-trust-policy.json"
```

3. 使用下列命令將所需的 AWS 受管政策連接至角色。

```
aws iam attach-role-policy \
  --policy-arn arn:aws: iam::aws:policy/service-role/AmazonEFSCSIDriverPolicy \
  --role-name $role_name
```

步驟 2：取得 Amazon EFS CSI 驅動程式

我們建議您透過 Amazon EKS 附加元件安裝 Amazon EFS CSI 驅動程式。若要將 Amazon EKS 附加元件新增至叢集，請參閱 the section called “建立附加元件”。如需附加元件的詳細資訊，請參閱 the section called “Amazon EKS 附加元件”。如果您無法使用 Amazon EKS 附加元件，建議您提交問題，說明為何無法前往容器藍圖 GitHub 儲存庫。

> **Important**
>
> 將 Amazon EFS 驅動程式新增為 Amazon EKS 附加元件之前，請確認叢集上沒有安裝自我管理版本的驅動程式。若是如此，請參閱在 GitHub 上解除安裝 Amazon EFS CSI 驅動程式。GitHub

或者，如果您想要自我管理安裝 Amazon EFS CSI 驅動程式，請參閱 GitHub 上的安裝。

步驟 3：建立 Amazon EFS 檔案系統

如需建立 Amazon EFS 檔案系統，請參閱 GitHub 上的為 Amazon EKS 建立 Amazon EFS 檔案系統。

步驟 4：部署範例應用程式

您可以部署各種範例應用程式，再視需要進行修改。如需詳細資訊，請參閱 GitHub 上的範例。

搭配 Amazon FSx for Lustre 使用高效能應用程式儲存

Amazon FSx for Lustre 容器儲存介面 (CSI) 驅動程式提供 CSI 介面，允許 Amazon EKS 叢集管理 Amazon FSx for Lustre 檔案系統的生命週期。如需詳細資訊，請參閱 Amazon FSx for Lustre 使用者指南。

如需如何將 Amazon FSx for Lustre CSI 驅動程式部署至 Amazon EKS 叢集並驗證其是否正常運作的詳細資訊，請參閱 the section called “部署驅動程式”。

部署 FSx for Lustre 驅動程式

本主題說明如何將 FSx for Lustre CSI 驅動程式部署到您的 Amazon EKS 叢集，並驗證其是否有效。我們建議您使用最新版的驅動程式。如需可用版本，請參閱 GitHub 上的 CSI 規格相容性對照表。

> **Note**
>
> Fargate 或 Amazon EKS 混合節點不支援驅動程式。

如需可用參數的詳細說明，以及示範驅動程式功能的完整範例，請參閱 GitHub 上的 FSx for Lustre Container Storage Interface (CSI) 驅動程式專案。

先決條件

- 現有的叢集。
- Amazon FSx CSI 驅動程式 EKS 附加元件需要 EKS Pod Identity 代理程式進行身分驗證。如果沒有此元件，附加元件將會失敗並出現錯誤 `Amazon EKS Pod Identity agent is not installed in the cluster`，導致磁碟區無法運作。在部署 FSx CSI 驅動程式附加元件之前或之後安裝 Pod Identity 代理程式。如需詳細資訊，請參閱the section called “設定 代理程式”。
- 裝置或 AWS CloudShell 上安裝和設定的 AWS 命令列界面 (AWS CLI) 版本 `1.27.160` `2.12.3`或更新版本。若要檢查您目前的版本，請使用 `aws --version | cut -d / -f2 | cut -d ' ' -f1`。適用於 macOS 的 `yum`、 `apt-get`或 Homebrew 等套件管理員通常是最新版本 CLI AWS 後面的幾個版本。若要安裝最新版本，請參閱《 AWS 命令列界面使用者指南》中的使用 aws 設定安裝 和快速組態。 https://docs.aws.amazon.com/cli/latest/userguide/cli-configure-quickstart.html#cli-configure-quickstart-config安裝在 AWS CloudShell 中的 AWS CLI 版本也可能是最新版本後面的幾個版本。若要更新它，請參閱《CloudShell AWS 使用者指南》中的將 CLI 安裝到您的主目錄。AWS CloudShell
- 裝置或 AWS CloudShell 上安裝的`eksctl`命令列工具版本 `0.210.0`或更新版本。如需有關安裝或更新 `eksctl` 的指示，請參閱 `eksctl` 文件中的安裝一節。
- `kubectl` 命令列工具安裝在您的裝置或 AWS CloudShell 上。版本可以與叢集的 Kubernetes 版本相同，也可以比叢集的 Kubernetes 版本更早或更晚一個次要版本。例如，如果您的叢集版本為 `1.29`，則可以搭配使用 `kubectl` `1.28`、`1.29` 或 `1.30` 版。若要安裝或升級 `kubectl`，請參閱 the section called “設定 kubectl 和 eksctl”。

步驟 1：建立 IAM 角色

Amazon FSx CSI 外掛程式需要 IAM 許可，才能代表您呼叫 AWS APIs。

> **Note**
>
> 除非您封鎖對 IMDS 的存取，否則 Pod 將可存取指派給 IAM 角色的許可。如需詳細資訊，請參閱the section called “最佳實務”。

下列程序說明如何建立 IAM 角色，並將 AWS 受管政策連接至該角色。

1. 建立 IAM 角色，並使用下列命令連接 AWS 受管政策。`my-cluster` 將 取代為您要使用的叢集名稱。命令會部署建立 IAM 角色並將 IAM 政策連接至其中的 AWS CloudFormation 堆疊。

```
eksctl create iamserviceaccount \
    --name fsx-csi-controller-sa \
    --namespace kube-system \
    --cluster my-cluster \
    --role-name AmazonEKS_FSx_CSI_DriverRole \
    --role-only \
    --attach-policy-arn arn:aws: iam::aws:policy/AmazonFSxFullAccess \
    --approve
```

建立服務帳戶時，您會看到幾行輸出。輸出的最後一行類似於下列內容。

```
[#]  1 task: {
    2 sequential sub-tasks: {
        create IAM role for serviceaccount "kube-system/fsx-csi-controller-sa",
        create serviceaccount "kube-system/fsx-csi-controller-sa",
    } }
[#]  building iamserviceaccount stack "eksctl-my-cluster-addon-iamserviceaccount-
kube-system-fsx-csi-controller-sa"
[#]  deploying stack "eksctl-my-cluster-addon-iamserviceaccount-kube-system-fsx-csi-
controller-sa"
[#]  waiting for CloudFormation stack "eksctl-my-cluster-addon-iamserviceaccount-
kube-system-fsx-csi-controller-sa"
[#]  created serviceaccount "kube-system/fsx-csi-controller-sa"
```

請注意已部署的 AWS CloudFormation 堆疊名稱。在先前的範例輸出中，堆疊被命名為 `eksctl-my-cluster-addon-iamserviceaccount-kube-system-fsx-csi-controller-sa`。

現在您已建立 Amazon FSx CSI 驅動程式 IAM 角色，您可以繼續下一節。當您使用此 IAM 角色部署附加元件時，它會建立並設定為使用名為 的服務帳戶`fsx-csi-controller-sa`。服務帳戶繫結至`clusterrole`獲指派所需 Kubernetes 許可的 Kubernetes。

步驟 2：安裝 Amazon FSx CSI 驅動程式

我們建議您透過 Amazon EKS 附加元件安裝 Amazon FSx CSI 驅動程式，以提高安全性並減少工作量。若要將 Amazon EKS 附加元件新增至叢集，請參閱 the section called “建立附加元件”。如需附加元件的詳細資訊，請參閱 the section called “Amazon EKS 附加元件”。

> **Important**
>
> 叢集中預先存在的 Amazon FSx CSI 驅動程式安裝可能會導致附加元件安裝失敗。當您嘗試在存在非 EKS FSx CSI 驅動程式的情況下安裝 Amazon EKS 附加元件版本時，安裝會因為資源衝突而失敗。在安裝期間使用 `OVERWRITE`旗標來解決此問題。
>
> ```
> aws eks create-addon --addon-name aws-fsx-csi-driver --cluster-name my-cluster
>  --resolve-conflicts OVERWRITE
> ```

或者，如果您想要自我管理安裝 Amazon FSx CSI 驅動程式，請參閱 GitHub 上的安裝。

步驟 3：部署儲存類別、持久性磁碟區宣告和範例應用程式

此程序使用 FSx for Lustre 容器儲存介面 (CSI) 驅動程式 GitHub 儲存庫，來耗用動態佈建的 FSx for Lustre 磁碟區。

1. 記下叢集的安全群組。您可以在聯網區段 AWS Management Console 下的 中或使用下列 CLI AWS 命令來查看。`my-cluster` 將 取代為您要使用的叢集名稱。

   ```
   aws eks describe-cluster --name my-cluster --query
    cluster.resourcesVpcConfig.clusterSecurityGroupId
   ```

2. 根據《Amazon FSx for Lustre 使用者指南》中的 Amazon VPC 安全群組中顯示的標準，為您的 Amazon FSx 檔案系統建立安全群組。對於 VPC，選擇叢集的 VPC，如聯網區段所示。對於「與 Lustre 用戶端關聯的安全群組」，請使用您的叢集安全群組。您可以單獨保留傳出規則，以允許 All traffic (所有流量)。
3. 執行以下命令，下載儲存類別清單檔案。

   ```
   curl -O https://raw.githubusercontent.com/kubernetes-sigs/aws-fsx-csi-driver/master/
   examples/kubernetes/dynamic_provisioning/specs/storageclass.yaml
   ```

4. 編輯 `storageclass.yaml` 檔案的參數區段。將每個範例值取代為您自己的值。

   ```
   parameters:
     subnetId: subnet-0eabfaa81fb22bcaf
     securityGroupIds: sg-068000ccf82dfba88
     deploymentType: PERSISTENT_1
     automaticBackupRetentionDays: "1"
   ```

```
  dailyAutomaticBackupStartTime: "00:00"
  copyTagsToBackups: "true"
  perUnitStorageThroughput: "200"
  dataCompressionType: "NONE"
  weeklyMaintenanceStartTime: "7:09:00"
  fileSystemTypeVersion: "2.12"
```

- **subnetId** – Amazon FSx for Lustre 檔案系統應建立所在的子網路 ID。並非所有可用區域都支援 Amazon FSx for Lustre。開啟位於 https://console.aws.amazon.com/fsx/ 的 Amazon FSx for Lustre 主控台，確認您要使用的子網路位於支援的可用區域中。子網路可以包含您的節點，也可以是不同的子網路或 VPC：
  - 您可以在運算區段下選取節點群組， AWS Management Console 以檢查 中的節點子網路。
  - 如果您指定的子網路與您擁有節點的子網路不同，則必須連接 VPCs，而且您必須確保在安全群組中開啟必要的連接埠。
- **securityGroupIds** – 您為檔案系統建立的安全群組 ID。
- **deploymentType**（選用） – 檔案系統部署類型。有效值為 SCRATCH_1、SCRATCH_2、PERSISTENT_1、PERSISTENT_2。如需部署類型的詳細資訊，請參閱建立您的 Amazon FSx for Lustre 檔案系統。
- 其他參數 (選用)：如需有關其他參數的詳細資訊，請參閱 GitHub 上的編輯 StorageClass。

5. 建立儲存類別清單檔案。

```
kubectl apply -f storageclass.yaml
```

範例輸出如下。

```
storageclass.storage.k8s.io/fsx-sc created
```

6. 下載持續性磁碟區宣告清單檔案。

```
curl -O https://raw.githubusercontent.com/kubernetes-sigs/aws-fsx-csi-driver/master/
examples/kubernetes/dynamic_provisioning/specs/claim.yaml
```

7. (選用) 編輯 claim.yaml 檔案。根據您的儲存要求和您在上一個步驟中選取的 deploymentType，將 1200Gi 變更為下列某一個增量值。

```
storage: 1200Gi
```

- SCRATCH_2 和 PERSISTENT：1.2 TiB 或 2.4 TiB，超過 2.4 TiB 則以 2.4 TiB 為單位遞增。

- SCRATCH_1：1.2 TiB、2.4 TiB 或 3.6 TiB，超過 3.6 TiB 則以 3.6 TiB 為單位遞增。

8. 建立持續性磁碟區宣告。

```
kubectl apply -f claim.yaml
```

範例輸出如下。

```
persistentvolumeclaim/fsx-claim created
```

9. 確認已佈建檔案系統。

```
kubectl describe pvc
```

範例輸出如下。

```
Name:          fsx-claim
Namespace:     default
StorageClass:  fsx-sc
Status:        Bound
[...]
```

> **Note**
>
> Status 可能會顯示為 Pending 約 5-10 分鐘，然後變更為 Bound。在 Status為 之前，請勿繼續下一個步驟Bound。如果 Status 顯示 Pending 超過 10 分鐘，請使用 Events 中的警告訊息作為解決任何問題的參考。

10部署範例應用程式。

```
kubectl apply -f https://raw.githubusercontent.com/kubernetes-sigs/aws-fsx-csi-driver/master/examples/kubernetes/dynamic_provisioning/specs/pod.yaml
```

11確認範例應用程式正在執行。

```
kubectl get pods
```

範例輸出如下。

```
NAME      READY   STATUS              RESTARTS   AGE
fsx-app   1/1     Running             0          8s
```

12驗證應用程式是否正確掛載了檔案系統。

```
kubectl exec -ti fsx-app -- df -h
```

範例輸出如下。

```
Filesystem                  Size  Used Avail Use% Mounted on
overlay                      80G  4.0G   77G   5% /
tmpfs                        64M     0   64M   0% /dev
tmpfs                       3.8G     0  3.8G   0% /sys/fs/cgroup
192.0.2.0@tcp:/abcdef01     1.1T  7.8M  1.1T   1% /data
/dev/nvme0n1p1               80G  4.0G   77G   5% /etc/hosts
shm                          64M     0   64M   0% /dev/shm
tmpfs                       6.9G   12K  6.9G   1% /run/secrets/kubernetes.io/
serviceaccount
tmpfs                       3.8G     0  3.8G   0% /proc/acpi
tmpfs                       3.8G     0  3.8G   0% /sys/firmware
```

13確認範例應用程式是否已將資料寫入 FSx for Lustre 檔案系統。

```
kubectl exec -it fsx-app -- ls /data
```

範例輸出如下。

```
out.txt
```

此範例輸出顯示範例應用程式成功地將 `out.txt` 檔案寫入檔案系統。

> **Note**
>
> 刪除叢集之前，請確認刪除 FSx for Lustre 檔案系統。如需詳細資訊，請參閱 《FSx for Lustre 使用者指南》中的清除資源。

FSx for Lustre 的效能調校

搭配 Amazon EKS 使用 FSx for Lustre 時，您可以在節點初始化期間套用 Lustre 調校，以最佳化效能。建議的方法是使用啟動範本使用者資料，以確保所有節點的組態一致。

這些調校包括：

- 網路和 RPC 最佳化
- Lustre 模組管理
- LRU（鎖定資源單位）調校
- 用戶端快取控制設定
- OST 和 MDC 的 RPC 控制

如需實作這些效能調校的詳細說明：

- 如需最佳化標準節點（非 EFA) 的效能，請參閱 the section called “Optimize（非 EBA)” 以取得可新增至啟動範本使用者資料的完整指令碼。
- 如需最佳化已啟用 EFA 節點的效能，請參閱 the section called “最佳化 (EFA)”。

最佳化節點上的 Amazon FSx for Lustre 效能 (EFA)

本主題說明如何使用 Amazon EKS 和 Amazon FSx for Lustre 設定 Elastic Fabric Adapter (EFA) 調校。

> **Note**
>
> - 如需建立和部署 FSx for Lustre CSI 驅動程式的詳細資訊，請參閱 the section called “部署驅動程式”。
> - 如需在沒有 EFA 的情況下最佳化標準節點，請參閱 the section called “Optimize（非 EBA)”。

步驟 1. 建立 EKS 叢集

使用提供的組態檔案建立叢集：

```
# Create cluster using efa-cluster.yaml
eksctl create cluster -f efa-cluster.yaml
```

範例efa-cluster.yaml：

```
#efa-cluster.yaml

apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: csi-fsx
  region: us-east-1
  version: "1.30"

iam:
  withOIDC: true

availabilityZones: ["us-east-1a", "us-east-1d"]

managedNodeGroups:
  - name: my-efa-ng
    instanceType: c6gn.16xlarge
    minSize: 1
    desiredCapacity: 1
    maxSize: 1
    availabilityZones: ["us-east-1b"]
    volumeSize: 300
    privateNetworking: true
    amiFamily: Ubuntu2204
    efaEnabled: true
    preBootstrapCommands:
      - |
        #!/bin/bash
        eth_intf="$(/sbin/ip -br -4 a sh | grep $(hostname -i)/ | awk '{print $1}')"
        efa_version=$(modinfo efa | awk '/^version:/ {print $2}' | sed 's/[^0-9.]//g')
        min_efa_version="2.12.1"

        if [[ "$(printf '%s\n' "$min_efa_version" "$efa_version" | sort -V | head -
n1)" != "$min_efa_version" ]]; then
            sudo curl -O https://efa-installer.amazonaws.com/aws-efa-
installer-1.37.0.tar.gz
            tar -xf aws-efa-installer-1.37.0.tar.gz && cd aws-efa-installer
```

```
            echo "Installing EFA driver"
            sudo apt-get update && apt-get upgrade -y
            sudo apt install -y pciutils environment-modules libnl-3-dev libnl-
route-3-200 libnl-route-3-dev dkms
            sudo ./efa_installer.sh -y
            modinfo efa
        else
            echo "Using EFA driver version $efa_version"
        fi

        echo "Installing Lustre client"
        sudo wget -O - https://fsx-lustre-client-repo-public-keys.s3.amazonaws.com/fsx-
ubuntu-public-key.asc | gpg --dearmor | sudo tee /usr/share/keyrings/fsx-ubuntu-public-
key.gpg > /dev/null
        sudo echo "deb [signed-by=/usr/share/keyrings/fsx-ubuntu-public-key.gpg]
 https://fsx-lustre-client-repo.s3.amazonaws.com/ubuntu jammy main" > /etc/apt/
sources.list.d/fsxlustreclientrepo.list
        sudo apt update | tail
        sudo apt install -y lustre-client-modules-$(uname -r) amazon-ec2-utils | tail
        modinfo lustre

        echo "Loading Lustre/EFA modules..."
        sudo /sbin/modprobe lnet
        sudo /sbin/modprobe kefalnd ipif_name="$eth_intf"
        sudo /sbin/modprobe ksocklnd
        sudo lnetctl lnet configure

        echo "Configuring TCP interface..."
        sudo lnetctl net del --net tcp 2> /dev/null
        sudo lnetctl net add --net tcp --if $eth_intf

        # For P5 instance type which supports 32 network cards,
        # by default add 8 EFA interfaces selecting every 4th device (1 per PCI bus)
        echo "Configuring EFA interface(s)..."
        instance_type="$(ec2-metadata --instance-type | awk '{ print $2 }')"
        num_efa_devices="$(ls -1 /sys/class/infiniband | wc -l)"
        echo "Found $num_efa_devices available EFA device(s)"

        if [[ "$instance_type" == "p5.48xlarge" || "$instance_type" ==
 "p5e.48xlarge" ]]; then
           for intf in $(ls -1 /sys/class/infiniband | awk 'NR % 4 == 1'); do
               sudo lnetctl net add --net efa --if $intf --peer-credits 32
          done
        else
```

```
        # Other instances: Configure 2 EFA interfaces by default if the instance
 supports multiple network cards,
        # or 1 interface for single network card instances
        # Can be modified to add more interfaces if instance type supports it
            sudo lnetctl net add --net efa --if $(ls -1 /sys/class/infiniband | head -
n1) --peer-credits 32
            if [[ $num_efa_devices -gt 1 ]]; then
               sudo lnetctl net add --net efa --if $(ls -1 /sys/class/infiniband | tail
 -n1) --peer-credits 32
            fi
        fi

        echo "Setting discovery and UDSP rule"
        sudo lnetctl set discovery 1
        sudo lnetctl udsp add --src efa --priority 0
        sudo /sbin/modprobe lustre

        sudo lnetctl net show
        echo "Added $(sudo lnetctl net show | grep -c '@efa') EFA interface(s)"
```

步驟 2. 建立節點群組

建立已啟用 EFA 的節點群組：

```
# Create node group using efa-ng.yaml
eksctl create nodegroup -f efa-ng.yaml
```

> **Important**
>
> === 在 區段中調整您環境的這些值# 5. Mount FSx filesystem。

```
FSX_DNS="<your-fsx-filesystem-dns>" # Needs to be adjusted.
MOUNT_NAME="<your-mount-name>" # Needs to be adjusted.
MOUNT_POINT="</your/mount/point>" # Needs to be adjusted.
```

===

範例`efa-ng.yaml`：

```
apiVersion: eksctl.io/v1alpha5
```

```
kind: ClusterConfig

metadata:
  name: final-efa
  region: us-east-1

managedNodeGroups:
  - name: ng-1
    instanceType: c6gn.16xlarge
    minSize: 1
    desiredCapacity: 1
    maxSize: 1
    availabilityZones: ["us-east-1a"]
    volumeSize: 300
    privateNetworking: true
    amiFamily: Ubuntu2204
    efaEnabled: true
    preBootstrapCommands:
      - |
        #!/bin/bash
        exec 1> >(logger -s -t $(basename $0)) 2>&1


 ############################################################################################
        #                                        Configuration Section
         #

 ############################################################################################

        # File System Configuration
        FSX_DNS="<your-fsx-filesystem-dns>" # Needs to be adjusted.
        MOUNT_NAME="<your-mount-name>" # Needs to be adjusted.
        MOUNT_POINT="</your/mount/point>" # Needs to be adjusted.

        # Lustre Tuning Parameters
        LUSTRE_LRU_MAX_AGE=600000
        LUSTRE_MAX_CACHED_MB=64
        LUSTRE_OST_MAX_RPC=32
        LUSTRE_MDC_MAX_RPC=64
        LUSTRE_MDC_MOD_RPC=50

        # File paths
        FUNCTIONS_SCRIPT="/usr/local/bin/lustre_functions.sh"
        TUNINGS_SCRIPT="/usr/local/bin/apply_lustre_tunings.sh"
```

```
        SERVICE_FILE="/etc/systemd/system/lustre-tunings.service"

        #EFA
        MIN_EFA_VERSION="2.12.1"

        # Function to check if a command was successful
        check_success() {
            if [ $? -eq 0 ]; then
                echo "SUCCESS: $1"
            else
                echo "FAILED: $1"
                return 1
            fi
        }

        echo "********Starting FSx for Lustre configuration********"

        # 1. Install Lustre client
        if grep -q '^ID=ubuntu' /etc/os-release; then
            echo "Detected Ubuntu, proceeding with Lustre setup..."
            # Add Lustre repository
            sudo wget -O - https://fsx-lustre-client-repo-public-keys.s3.amazonaws.com/
fsx-ubuntu-public-key.asc | sudo gpg --dearmor | sudo tee /usr/share/keyrings/fsx-
ubuntu-public-key.gpg > /dev/null

            echo "deb [signed-by=/usr/share/keyrings/fsx-ubuntu-public-key.gpg]
 https://fsx-lustre-client-repo.s3.amazonaws.com/ubuntu jammy main" | sudo tee /etc/
apt/sources.list.d/fsxlustreclientrepo.list

            sudo apt-get update
            sudo apt-get install -y lustre-client-modules-$(uname -r)
            sudo apt-get install -y lustre-client
        else
            echo "Not Ubuntu, exiting"
            exit 1
        fi

        check_success "Install Lustre client"

        # Ensure Lustre tools are in the PATH
        export PATH=$PATH:/usr/sbin

        # 2. Apply network and RPC tunings
        echo "********Applying network and RPC tunings********"
```

```
        if ! grep -q "options ptlrpc ptlrpcd_per_cpt_max" /etc/modprobe.d/
modprobe.conf; then
            echo "options ptlrpc ptlrpcd_per_cpt_max=64" | sudo tee -a /etc/modprobe.d/
modprobe.conf
            check_success "Set ptlrpcd_per_cpt_max"
        else
            echo "ptlrpcd_per_cpt_max already set in modprobe.conf"
        fi

        if ! grep -q "options ksocklnd credits" /etc/modprobe.d/modprobe.conf; then
            echo "options ksocklnd credits=2560" | sudo tee -a /etc/modprobe.d/
modprobe.conf
            check_success "Set ksocklnd credits"
        else
            echo "ksocklnd credits already set in modprobe.conf"
        fi

        # 3. Load Lustre modules
        manage_lustre_modules() {
            echo "Checking for existing Lustre modules..."
            if lsmod | grep -q lustre; then
                echo "Existing Lustre modules found."

                # Check for mounted Lustre filesystems
                echo "Checking for mounted Lustre filesystems..."
                if mount | grep -q "type lustre"; then
                    echo "Found mounted Lustre filesystems. Attempting to unmount..."
                    mounted_fs=$(mount | grep "type lustre" | awk '{print $3}')
                    for fs in $mounted_fs; do
                        echo "Unmounting $fs"
                        sudo umount $fs
                        check_success "Unmount filesystem $fs"
                    done
                else
                    echo "No Lustre filesystems mounted."
                fi

                # After unmounting, try to remove modules
                echo "Attempting to remove Lustre modules..."
                sudo lustre_rmmod
                if [ $? -eq 0 ]; then
                    echo "SUCCESS: Removed existing Lustre modules"
                else
```

```
                    echo "WARNING: Could not remove Lustre modules. They may still be
 in use."
                    echo "Please check for any remaining Lustre processes or mounts."
                    return 1
                fi
            else
                echo "No existing Lustre modules found."
            fi

            echo "Loading Lustre modules..."
            sudo modprobe lustre
            check_success "Load Lustre modules" || exit 1

            echo "Checking loaded Lustre modules:"
            lsmod | grep lustre
        }

        # Managing Lustre kernel modules
        echo "********Managing Lustre kernel modules********"
        manage_lustre_modules

        # 4. Initializing Lustre networking
        echo "********Initializing Lustre networking********"
        sudo lctl network up
        check_success "Initialize Lustre networking" || exit 1

        # 4.5 EFA Setup and Configuration
        setup_efa() {
            echo "********Starting EFA Setup********"

            # Get interface and version information
            eth_intf="$(/sbin/ip -br -4 a sh | grep $(hostname -i)/ | awk '{print
 $1}')"
            efa_version=$(modinfo efa | awk '/^version:/ {print $2}' | sed 's/[^0-9.]//
g')
            min_efa_version=$MIN_EFA_VERSION

            # Install or verify EFA driver
            if [[ "$(printf '%s\n' "$min_efa_version" "$efa_version" | sort -V | head -
n1)" != "$min_efa_version" ]]; then
                echo "Installing EFA driver..."
                sudo curl -O https://efa-installer.amazonaws.com/aws-efa-
installer-1.37.0.tar.gz
                tar -xf aws-efa-installer-1.37.0.tar.gz && cd aws-efa-installer
```

```
                # Install dependencies
                sudo apt-get update && apt-get upgrade -y
                sudo apt install -y pciutils environment-modules libnl-3-dev libnl-
route-3-200 libnl-route-3-dev dkms

                # Install EFA
                sudo ./efa_installer.sh -y
                modinfo efa
            else
                echo "Using existing EFA driver version $efa_version"
            fi
        }

        configure_efa_network() {
            echo "********Configuring EFA Network********"

            # Load required modules
            echo "Loading network modules..."
            sudo /sbin/modprobe lnet
            sudo /sbin/modprobe kefalnd ipif_name="$eth_intf"
            sudo /sbin/modprobe ksocklnd

            # Initialize LNet
            echo "Initializing LNet..."
            sudo lnetctl lnet configure

            # Configure TCP interface
            echo "Configuring TCP interface..."
            sudo lnetctl net del --net tcp 2> /dev/null
            sudo lnetctl net add --net tcp --if $eth_intf

            # For P5 instance type which supports 32 network cards,
            # by default add 8 EFA interfaces selecting every 4th device (1 per PCI
 bus)
            echo "Configuring EFA interface(s)..."
            instance_type="$(ec2-metadata --instance-type | awk '{ print $2 }')"
            num_efa_devices="$(ls -1 /sys/class/infiniband | wc -l)"
            echo "Found $num_efa_devices available EFA device(s)"

            if [[ "$instance_type" == "p5.48xlarge" || "$instance_type" ==
 "p5e.48xlarge" ]]; then
                # P5 instance configuration
                for intf in $(ls -1 /sys/class/infiniband | awk 'NR % 4 == 1'); do
```

```
                    sudo lnetctl net add --net efa --if $intf --peer-credits 32
                done
            else
                # Standard configuration
                # Other instances: Configure 2 EFA interfaces by default if the
instance supports multiple network cards,
                # or 1 interface for single network card instances
                # Can be modified to add more interfaces if instance type supports it
                sudo lnetctl net add --net efa --if $(ls -1 /sys/class/infiniband |
head -n1) --peer-credits 32
                if [[ $num_efa_devices -gt 1 ]]; then
                    sudo lnetctl net add --net efa --if $(ls -1 /sys/class/infiniband |
tail -n1) --peer-credits 32
                fi
            fi

            # Configure discovery and UDSP
            echo "Setting up discovery and UDSP rules..."
            sudo lnetctl set discovery 1
            sudo lnetctl udsp add --src efa --priority 0
            sudo /sbin/modprobe lustre

            # Verify configuration
            echo "Verifying EFA network configuration..."
            sudo lnetctl net show
            echo "Added $(sudo lnetctl net show | grep -c '@efa') EFA interface(s)"
        }

        # Main execution
        setup_efa
        configure_efa_network

        # 5. Mount FSx filesystem
        if [ ! -z "$FSX_DNS" ] && [ ! -z "$MOUNT_NAME" ]; then
            echo "********Creating mount point********"
            sudo mkdir -p $MOUNT_POINT
            check_success "Create mount point"

            echo "Mounting FSx filesystem..."
            sudo mount -t lustre ${FSX_DNS}@tcp:/${MOUNT_NAME} ${MOUNT_POINT}
            check_success "Mount FSx filesystem"
        else
            echo "Skipping FSx mount as DNS or mount name is not provided"
        fi
```

```
# 6. Applying Lustre performance tunings
echo "********Applying Lustre performance tunings********"

# Get number of CPUs
NUM_CPUS=$(nproc)

# Calculate LRU size (100 * number of CPUs)
LRU_SIZE=$((100 * NUM_CPUS))

#Apply LRU tunings
echo "Apply LRU tunings"
sudo lctl set_param ldlm.namespaces.*.lru_max_age=${LUSTRE_LRU_MAX_AGE}
check_success "Set lru_max_age"
sudo lctl set_param ldlm.namespaces.*.lru_size=$LRU_SIZE
check_success "Set lru_size"

# Client Cache Control
sudo lctl set_param llite.*.max_cached_mb=${LUSTRE_MAX_CACHED_MB}
check_success "Set max_cached_mb"

# RPC Controls
sudo lctl set_param osc.*OST*.max_rpcs_in_flight=${LUSTRE_OST_MAX_RPC}
check_success "Set OST max_rpcs_in_flight"

sudo lctl set_param mdc.*.max_rpcs_in_flight=${LUSTRE_MDC_MAX_RPC}
check_success "Set MDC max_rpcs_in_flight"

sudo lctl set_param mdc.*.max_mod_rpcs_in_flight=${LUSTRE_MDC_MOD_RPC}
check_success "Set MDC max_mod_rpcs_in_flight"

# 7. Verify all tunings
echo "********Verifying all tunings********"

# Function to verify parameter value
verify_param() {
    local param=$1
    local expected=$2
    local actual=$3

    if [ "$actual" == "$expected" ]; then
        echo "SUCCESS: $param is correctly set to $expected"
    else
        echo "WARNING: $param is set to $actual (expected $expected)"
```

```
            fi
        }

        echo "Verifying all parameters:"

        # LRU tunings
        actual_lru_max_age=$(lctl get_param -n ldlm.namespaces.*.lru_max_age | head -1)
        verify_param "lru_max_age" "600000" "$actual_lru_max_age"

        actual_lru_size=$(lctl get_param -n ldlm.namespaces.*.lru_size | head -1)
        verify_param "lru_size" "$LRU_SIZE" "$actual_lru_size"

        # Client Cache
        actual_max_cached_mb=$(lctl get_param -n llite.*.max_cached_mb | grep
"max_cached_mb:" | awk '{print $2}')
        verify_param "max_cached_mb" "64" "$actual_max_cached_mb"

        # RPC Controls
        actual_ost_rpcs=$(lctl get_param -n osc.*OST*.max_rpcs_in_flight | head -1)
        verify_param "OST max_rpcs_in_flight" "32" "$actual_ost_rpcs"

        actual_mdc_rpcs=$(lctl get_param -n mdc.*.max_rpcs_in_flight | head -1)
        verify_param "MDC max_rpcs_in_flight" "64" "$actual_mdc_rpcs"

        actual_mdc_mod_rpcs=$(lctl get_param -n mdc.*.max_mod_rpcs_in_flight | head -1)
        verify_param "MDC max_mod_rpcs_in_flight" "50" "$actual_mdc_mod_rpcs"

        # Network and RPC configurations from modprobe.conf
        actual_ptlrpc=$(grep "ptlrpc ptlrpcd_per_cpt_max" /etc/modprobe.d/modprobe.conf
| awk '{print $3}')
        verify_param "ptlrpcd_per_cpt_max" "ptlrpcd_per_cpt_max=64" "$actual_ptlrpc"

        actual_ksocklnd=$(grep "ksocklnd credits" /etc/modprobe.d/modprobe.conf | awk
'{print $3}')
        verify_param "ksocklnd credits" "credits=2560" "$actual_ksocklnd"

        # 8. Setup persistence
        setup_persistence() {
            # Create functions file
            cat << EOF > $FUNCTIONS_SCRIPT
        #!/bin/bash

        apply_lustre_tunings() {
            local NUM_CPUS=\$(nproc)
```

```
        local LRU_SIZE=\$((100 * NUM_CPUS))

        echo "Applying Lustre performance tunings..."
        lctl set_param ldlm.namespaces.*.lru_max_age=$LUSTRE_LRU_MAX_AGE
        lctl set_param ldlm.namespaces.*.lru_size=\$LRU_SIZE
        lctl set_param llite.*.max_cached_mb=$LUSTRE_MAX_CACHED_MB
        lctl set_param osc.*OST*.max_rpcs_in_flight=$LUSTRE_OST_MAX_RPC
        lctl set_param mdc.*.max_rpcs_in_flight=$LUSTRE_MDC_MAX_RPC
        lctl set_param mdc.*.max_mod_rpcs_in_flight=$LUSTRE_MDC_MOD_RPC
    }
    EOF

        # Create tuning script
        cat << EOF > $TUNINGS_SCRIPT
    #!/bin/bash
    exec 1> >(logger -s -t \$(basename \$0)) 2>&1

    source $FUNCTIONS_SCRIPT

    # Function to check if Lustre is mounted
    is_lustre_mounted() {
        mount | grep -q "type lustre"
    }

    # Function to mount Lustre
    mount_lustre() {
        echo "Mounting Lustre filesystem..."
        mkdir -p $MOUNT_POINT
        mount -t lustre ${FSX_DNS}@tcp:/${MOUNT_NAME} $MOUNT_POINT
        return \$?
    }

    # Main execution
    # Try to mount if not already mounted
    if ! is_lustre_mounted; then
        echo "Lustre filesystem not mounted, attempting to mount..."
        mount_lustre
    fi

    # Wait for successful mount (up to 5 minutes)
    for i in {1..30}; do
        if is_lustre_mounted; then
            echo "Lustre filesystem mounted, applying tunings..."
            apply_lustre_tunings
```

```
            exit 0
        fi
        echo "Waiting for Lustre filesystem to be mounted... (attempt $i/30)"
        sleep 10
    done

    echo "Timeout waiting for Lustre filesystem mount"
    exit 1
    EOF

    # Create systemd service

    # Create systemd directory if it doesn't exist
    sudo mkdir -p /etc/systemd/system/

        # Create service file directly for Ubuntu
        cat << EOF > $SERVICE_FILE
    [Unit]
    Description=Apply Lustre Performance Tunings
    After=network.target remote-fs.target

    [Service]
    Type=oneshot
    ExecStart=/bin/bash -c 'source $FUNCTIONS_SCRIPT && $TUNINGS_SCRIPT'
    RemainAfterExit=yes

    [Install]
    WantedBy=multi-user.target
    EOF


        # Make scripts executable and enable service
        sudo chmod +x $FUNCTIONS_SCRIPT
        sudo chmod +x $TUNINGS_SCRIPT
        systemctl enable lustre-tunings.service
        systemctl start lustre-tunings.service
    }

    echo "********Setting up persistent tuning********"
    setup_persistence

    echo "FSx for Lustre configuration completed."
```

（選用）步驟 3。驗證 EFA 設定

SSH 至節點：

```
# Get instance ID from EKS console or AWS CLI
ssh -i /path/to/your-key.pem ec2-user@<node-internal-ip>
```

驗證 EFA 組態：

```
sudo lnetctl net show
```

檢查設定日誌：

```
sudo cat /var/log/cloud-init-output.log
```

以下是 的預期輸出範例`lnetctl net show`：

```
net:
    - net type: tcp
      ...
    - net type: efa
      local NI(s):
        - nid: xxx.xxx.xxx.xxx@efa
          status: up
```

部署範例

a. 建立 claim.yaml

```
#claim.yaml

apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: fsx-claim-efa
spec:
  accessModes:
    - ReadWriteMany
  storageClassName: ""
  resources:
    requests:
```

```
      storage: 4800Gi
  volumeName: fsx-pv
```

套用宣告：

```
kubectl apply -f claim.yaml
```

b. 建立 pv.yaml

更新 <replaceable-placeholders>：

```
#pv.yaml

apiVersion: v1
kind: PersistentVolume
metadata:
  name: fsx-pv
spec:
  capacity:
    storage: 4800Gi
  volumeMode: Filesystem
  accessModes:
    - ReadWriteMany
  mountOptions:
    - flock
  persistentVolumeReclaimPolicy: Recycle
  csi:
    driver: fsx.csi.aws.com
    volumeHandle: fs-<1234567890abcdef0>
    volumeAttributes:
      dnsname: fs-<1234567890abcdef0>.fsx.us-east-1.amazonaws.com
      mountname: <abcdef01>
```

套用持久性磁碟區：

```
kubectl apply -f pv.yaml
```

c. 建立 pod.yaml

```
#pod.yaml

apiVersion: v1
```

```
kind: Pod
metadata:
  name: fsx-efa-app
spec:
  containers:
  - name: app
    image: amazonlinux:2
    command: ["/bin/sh"]
    args: ["-c", "while true; do dd if=/dev/urandom bs=100M count=20 > data/test_file;
 sleep 10; done"]
    resources:
      requests:
        vpc.amazonaws.com/efa: 1
      limits:
        vpc.amazonaws.com/efa: 1
    volumeMounts:
    - name: persistent-storage
      mountPath: /data
  volumes:
  - name: persistent-storage
    persistentVolumeClaim:
      claimName: fsx-claim-efa
```

套用 Pod：

```
kubectl apply -f pod.yaml
```

其他驗證命令

驗證 Pod 掛載和寫入檔案系統：

```
kubectl exec -ti fsx-efa-app -- df -h | grep data
# Expected output:
# <192.0.2.0>@tcp:/<abcdef01>  4.5T  1.2G  4.5T   1% /data

kubectl exec -ti fsx-efa-app -- ls /data
# Expected output:
# test_file
```

節點上的 SSH 來驗證流量是否通過 EFA：

```
sudo lnetctl net show -v
```

預期的輸出會顯示具有流量統計資料的 EFA 介面。

相關資訊

- the section called “部署驅動程式”
- the section called “Optimize （ 非 EBA)”
- Amazon FSx for Lustre 效能
- Elastic Fabric Adapter

最佳化節點上的 Amazon FSx for Lustre 效能 （ 非 EBA)

您可以使用啟動範本使用者資料，在節點初始化期間套用調校參數，以最佳化 Amazon FSx for Lustre 效能。

> **Note**
>
> - 如需建立和部署 FSx for Lustre CSI 驅動程式的詳細資訊，請參閱 the section called “部署驅動程式”。如需使用啟用 EFA 的節點最佳化效能，請參閱 the section called “最佳化 (EFA)”。

為什麼要使用啟動範本使用者資料？

- 在節點初始化期間自動套用調校。
- 確保所有節點的組態一致。
- 無需手動節點組態。

範例指令碼概觀

本主題中定義的範例指令碼會執行這些操作：

```
# 1. Install Lustre client
```

- 自動偵測您的作業系統版本：Amazon Linux 2 (AL2) 或 Amazon Linux 2023 (AL2023)。
- 安裝適當的 Lustre 用戶端套件。

 # 2. Apply network and RPC tunings

- ptlrpcd_per_cpt_max=64 用於平行 RPC 處理的集合。
- 設定 ksocklnd credits=2560以最佳化網路緩衝區。

 # 3. Load Lustre modules

- 如果存在，則安全地移除現有的 Lustre 模組。
- 處理現有檔案系統的卸載。
- 載入新的 Lustre 模組。

 # 4. Lustre Network Initialization

- 初始化 Lustre 網路組態。
- 設定必要的網路參數。

 # 5. Mount FSx filesystem

- 您必須調整本節中您環境的值。

 # 6. Apply tunings

- LRU （鎖定資源單位） 調校：
  - lru_max_age=600000
  - lru_size 根據 CPU 計數計算
- 用戶端快取控制： max_cached_mb=64
- RPC 控制項：
  - OST max_rpcs_in_flight=32
  - MDC max_rpcs_in_flight=64
  - MDC max_mod_rpcs_in_flight=50

 # 7. Verify tunings

- 驗證所有套用的調校。

- 報告每個參數的成功或警告。

`# 8. Setup persistence`

- 您也必須在本節中調整環境的值。
- 自動偵測您的作業系統版本 (AL2 或 AL2023)，以決定要套用Systemd的服務。
- 系統啟動。
- Systemd 啟動 `lustre-tunings`服務（由於 `WantedBy=multi-user.target`)。
- 服務執行`apply_lustre_tunings.sh`：
  - 檢查檔案系統是否已掛載。
  - 如果未掛載，則掛載檔案系統。
  - 等待成功掛載（最多五分鐘）。
  - 成功掛載後套用調校參數。
- 設定會保持作用中狀態，直到重新啟動為止。
- 服務會在指令碼完成後結束。
  - Systemd 會將服務標記為「作用中（已結束）」。
- 程序會在下次重新開機時重複。

建立啟動範本

1. 前往 https://console.aws.amazon.com/ec2/ 開啟 Amazon EC2 主控台。
2. 選擇啟動範本。
3. 選擇 Create launch template (建立啟動範本)。
4. 在進階詳細資訊中，找到使用者資料區段。
5. 貼上以下指令碼，並視需要更新任何項目。

> **Important**
>
> 在 區段和 區段`# 5. Mount FSx filesystem`中的 `setup_persistence()`函數`apply_lustre_tunings.sh`中，為您的環境調整這些值`# 8. Setup persistence`：
>
> ```
> FSX_DNS="<your-fsx-filesystem-dns>" # Needs to be adjusted.
> MOUNT_NAME="<your-mount-name>" # Needs to be adjusted.
> ```

```
    MOUNT_POINT="</your/mount/point>" # Needs to be adjusted.
```

```
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary="==MYBOUNDARY=="
--==MYBOUNDARY==
Content-Type: text/x-shellscript; charset="us-ascii"
#!/bin/bash
exec 1> >(logger -s -t $(basename $0)) 2>&1
# Function definitions
check_success() {
    if [ $? -eq 0 ]; then
        echo "SUCCESS: $1"
    else
        echo "FAILED: $1"
        return 1
    fi
}
apply_tunings() {
    local NUM_CPUS=$(nproc)
    local LRU_SIZE=$((100 * NUM_CPUS))
    local params=(
        "ldlm.namespaces.*.lru_max_age=600000"
        "ldlm.namespaces.*.lru_size=$LRU_SIZE"
        "llite.*.max_cached_mb=64"
        "osc.*OST*.max_rpcs_in_flight=32"
        "mdc.*.max_rpcs_in_flight=64"
        "mdc.*.max_mod_rpcs_in_flight=50"
    )
    for param in "${params[@]}"; do
        lctl set_param $param
        check_success "Set ${param%%=*}"
    done
}
verify_param() {
    local param=$1
    local expected=$2
    local actual=$3

    if [ "$actual" == "$expected" ]; then
        echo "SUCCESS: $param is correctly set to $expected"
    else
```

```
        echo "WARNING: $param is set to $actual (expected $expected)"
    fi
}
verify_tunings() {
    local NUM_CPUS=$(nproc)
    local LRU_SIZE=$((100 * NUM_CPUS))
    local params=(
        "ldlm.namespaces.*.lru_max_age:600000"
        "ldlm.namespaces.*.lru_size:$LRU_SIZE"
        "llite.*.max_cached_mb:64"
        "osc.*OST*.max_rpcs_in_flight:32"
        "mdc.*.max_rpcs_in_flight:64"
        "mdc.*.max_mod_rpcs_in_flight:50"
    )
    echo "Verifying all parameters:"
    for param in "${params[@]}"; do
        name="${param%%:*}"
        expected="${param#*:}"
        actual=$(lctl get_param -n $name | head -1)
        verify_param "${name##*.}" "$expected" "$actual"
    done
}
setup_persistence() {
    # Create functions file
    cat << 'EOF' > /usr/local/bin/lustre_functions.sh
#!/bin/bash
apply_lustre_tunings() {
    local NUM_CPUS=$(nproc)
    local LRU_SIZE=$((100 * NUM_CPUS))

    echo "Applying Lustre performance tunings..."
    lctl set_param ldlm.namespaces.*.lru_max_age=600000
    lctl set_param ldlm.namespaces.*.lru_size=$LRU_SIZE
    lctl set_param llite.*.max_cached_mb=64
    lctl set_param osc.*OST*.max_rpcs_in_flight=32
    lctl set_param mdc.*.max_rpcs_in_flight=64
    lctl set_param mdc.*.max_mod_rpcs_in_flight=50
}
EOF
    # Create tuning script
    cat << 'EOF' > /usr/local/bin/apply_lustre_tunings.sh
#!/bin/bash
exec 1> >(logger -s -t $(basename $0)) 2>&1
# Source the functions
```

```
source /usr/local/bin/lustre_functions.sh
# FSx details
FSX_DNS="<your-fsx-filesystem-dns>" # Needs to be adjusted.
MOUNT_NAME="<your-mount-name>" # Needs to be adjusted.
MOUNT_POINT="</your/mount/point>" # Needs to be adjusted.
# Function to check if Lustre is mounted
is_lustre_mounted() {
    mount | grep -q "type lustre"
}
# Function to mount Lustre
mount_lustre() {
    echo "Mounting Lustre filesystem..."
    mkdir -p ${MOUNT_POINT}
    mount -t lustre ${FSX_DNS}@tcp:/${MOUNT_NAME} ${MOUNT_POINT}
    return $?
}
# Main execution
# Try to mount if not already mounted
if ! is_lustre_mounted; then
    echo "Lustre filesystem not mounted, attempting to mount..."
    mount_lustre
fi
# Wait for successful mount (up to 5 minutes)
for i in {1..30}; do
    if is_lustre_mounted; then
        echo "Lustre filesystem mounted, applying tunings..."
        apply_lustre_tunings
        exit 0
    fi
    echo "Waiting for Lustre filesystem to be mounted... (attempt $i/30)"
    sleep 10
done
echo "Timeout waiting for Lustre filesystem mount"
exit 1
EOF
    # Create systemd service
    cat << 'EOF' > /etc/systemd/system/lustre-tunings.service
[Unit]
Description=Apply Lustre Performance Tunings
After=network.target remote-fs.target
StartLimitIntervalSec=0
[Service]
Type=oneshot
ExecStart=/usr/local/bin/apply_lustre_tunings.sh
```

```
RemainAfterExit=yes
Restart=on-failure
RestartSec=30
[Install]
WantedBy=multi-user.target
EOF
    chmod +x /usr/local/bin/lustre_functions.sh
    chmod +x /usr/local/bin/apply_lustre_tunings.sh
    systemctl enable lustre-tunings.service
    systemctl start lustre-tunings.service
}
echo "Starting FSx for Lustre configuration..."
# 1. Install Lustre client
if grep -q 'VERSION="2"' /etc/os-release; then
    amazon-linux-extras install -y lustre
elif grep -q 'VERSION="2023"' /etc/os-release; then
    dnf install -y lustre-client
fi
check_success "Install Lustre client"
# 2. Apply network and RPC tunings
export PATH=$PATH:/usr/sbin
echo "Applying network and RPC tunings..."
if ! grep -q "options ptlrpc ptlrpcd_per_cpt_max" /etc/modprobe.d/modprobe.conf; then
    echo "options ptlrpc ptlrpcd_per_cpt_max=64" | tee -a /etc/modprobe.d/
modprobe.conf
    echo "options ksocklnd credits=2560" | tee -a /etc/modprobe.d/modprobe.conf
fi
# 3. Load Lustre modules
modprobe lustre
check_success "Load Lustre modules" || exit 1
# 4. Lustre Network Initialization
lctl network up
check_success "Initialize Lustre networking" || exit 1
# 5. Mount FSx filesystem
FSX_DNS="<your-fsx-filesystem-dns>" # Needs to be adjusted.
MOUNT_NAME="<your-mount-name>" # Needs to be adjusted.
MOUNT_POINT="</your/mount/point>" # Needs to be adjusted.
if [ ! -z "$FSX_DNS" ] && [ ! -z "$MOUNT_NAME" ]; then
    mkdir -p $MOUNT_POINT
    mount -t lustre ${FSX_DNS}@tcp:/${MOUNT_NAME} ${MOUNT_POINT}
    check_success "Mount FSx filesystem"
fi
# 6. Apply tunings
apply_tunings
```

```
# 7. Verify tunings
verify_tunings
# 8. Setup persistence
setup_persistence
echo "FSx for Lustre configuration completed."
---===MYBOUNDARY==--
```

6. 建立 Amazon EKS 節點群組時，請選取此啟動範本。如需詳細資訊，請參閱the section called “建立”。

相關資訊

- the section called “部署驅動程式”
- the section called “最佳化 (EFA)”
- Amazon FSx for Lustre 效能

搭配 FSx for NetApp ONTAP 使用高效能應用程式儲存

NetApp Trident 使用容器儲存介面 (CSI) 相容驅動程式提供動態儲存協同運作。這允許 Amazon EKS 叢集管理由 Amazon FSx for NetApp ONTAP 檔案系統支援的持久性磁碟區 (PV) 的生命週期。請注意，Amazon FSx for NetApp ONTAP CSI 驅動程式與 Amazon EKS 混合節點不相容。若要開始使用，請參閱 NetApp Trident 文件中的搭配使用 Trident 與 Amazon FSx for NetApp ONTAP。 NetApp

Amazon FSx for NetApp ONTAP 是一種儲存服務，允許您在雲端中啟動和執行全受管的 ONTAP 檔案系統。ONTAP 是 NetApp 的檔案系統技術，可提供廣泛採用的一組資料存取和資料管理功能。FSx for ONTAP 提供內部部署 NetApp 檔案系統的功能、效能和 APIs，具有全受管 AWS 服務的靈活性、可擴展性和簡易性。如需詳細資訊，請參閱《FSx for ONTAP 使用者指南》。

Important

如果您使用 Amazon FSx for NetApp ONTAP 搭配 Amazon EBS CSI 驅動程式來佈建 EBS 磁碟區，則必須指定 不在該`multipath.conf`檔案中使用 EBS 裝置。如需支援的方法，請參閱組態檔案封鎖清單。請見此處範例。

```
defaults {
       user_friendly_names yes
       find_multipaths no
     }
     blacklist {
```

```
        device {
          vendor "NVME"
          product "Amazon Elastic Block Store"
        }
      }
```

搭配 Amazon FSx for OpenZFS 使用資料儲存

Amazon FSx for OpenZFS 是一種全受管檔案儲存服務，可讓您輕鬆地將資料 AWS 從內部部署 ZFS 或其他 Linux 型檔案伺服器移至 。您可以執行此動作，而無需變更應用程式程式碼或資料管理方式。這項服務提供建置在開放原始碼 OpenZFS 檔案系統上的高度可靠、可擴展、高效且功能豐富的檔案儲存。它結合了這些功能與全受管 AWS 服務的靈活性、可擴展性和簡單性。如需詳細資訊，請參閱《Amazon FSx for OpenZFS 使用者指南》。

FSx for OpenZFS 容器儲存介面 (CSI) 驅動程式提供 CSI 介面，允許 Amazon EKS 叢集管理 FSx for OpenZFS 磁碟區的生命週期。請注意，Amazon FSx for OpenZFS CSI 驅動程式與 Amazon EKS 混合節點不相容。若要將 FSx for OpenZFS CSI 驅動程式部署到您的 Amazon EKS 叢集，請參閱 GitHub 上的 aws-fsx-openzfs-csi-driver。

使用 Amazon File Cache 將延遲降至最低

Amazon File Cache 是全受管的高速快取， AWS 用於處理檔案資料，無論資料存放在何處。Amazon File Cache 會在第一次存取資料時自動將資料載入快取，並在不使用資料時釋出資料。如需詳細資訊，請參閱 Amazon File Cache 使用者指南。

Amazon File Cache 容器儲存介面 (CSI) 驅動程式提供 CSI 介面，允許 Amazon EKS 叢集管理 Amazon 檔案快取的生命週期。請注意，Amazon File Cache CSI 驅動程式與 Amazon EKS 混合節點不相容。若要將 Amazon File Cache CSI 驅動程式部署到您的 Amazon EKS 叢集，請參閱 GitHub 上的 aws-file-cache-csi-driver。

使用適用於 Amazon S3 CSI 驅動程式的掛載點存取 Amazon S3 物件

使用適用於 Amazon S3 容器儲存界面 (CSI) 驅動程式的掛載點，您的 Kubernetes 應用程式可以透過檔案系統界面存取 Amazon S3 物件，在不變更任何應用程式程式碼的情況下實現高彙總輸送量。CSI 驅動程式以適用於 Amazon S3 的掛載點為基礎，將 Amazon S3 儲存貯體做為磁碟區，可供 Amazon EKS 和自我管理 Kubernetes 叢集中的容器存取。

考量事項

- Amazon S3 CSI 驅動程式的掛載點目前與 Windows 型容器映像不相容。
- Amazon S3 CSI 驅動程式掛載點目前與 Amazon EKS 混合節點不相容。
- Amazon S3 CSI 驅動程式的掛載點不支援 AWS Fargate。不過，支援在 Amazon EC2 中執行的容器（使用 Amazon EKS 或自訂 Kubernetes 安裝）。
- Amazon S3 CSI 驅動程式的掛載點僅支援靜態佈建。不支援動態佈建或建立新儲存貯體。

> **Note**
>
> 靜態佈建是指使用 `PersistentVolume` 物件`bucketName`中指定為 的現有 Amazon S3 儲存貯`volumeAttributes`體。如需詳細資訊，請參閱 GitHub 上的靜態佈建。

- 使用適用於 Amazon S3 CSI 驅動程式的掛載點掛載的磁碟區不支援所有 POSIX 檔案系統功能。如需檔案系統行為的詳細資訊，請參閱 GitHub 上的 Amazon S3 檔案系統行為掛載點。

如需部署驅動程式的詳細資訊，請參閱 the section called “部署驅動程式”。如需移除驅動程式的詳細資訊，請參閱 the section called “移除驅動程式”。

部署適用於 Amazon S3 驅動程式的掛載點

使用適用於 Amazon S3 容器儲存界面 (CSI) 驅動程式的掛載點，您的 Kubernetes 應用程式可以透過檔案系統界面存取 Amazon S3 物件，在不變更任何應用程式程式碼的情況下實現高彙總輸送量。

此程序將示範如何部署 Amazon S3 CSI Amazon EKS 驅動程式的掛載點。在繼續之前，請檢閱考量事項。

先決條件

- 叢集的現有 AWS Identity and Access Management (IAM) OpenID Connect (OIDC) 提供者。若要判定您是否已經擁有一個，或是要建立一個，請參閱 the section called “IAM OIDC 供應商”。
- 在您的裝置或 AWS CloudShell 上安裝和設定的 AWS CLI 版本 2.12.3 或更新版本。
- `kubectl` 命令列工具安裝在您的裝置或 AWS CloudShell 上。版本可以與叢集的 Kubernetes 版本相同，也可以比叢集的 Kubernetes 版本更早或更晚一個次要版本。例如，如果您的叢集版本為 `1.29`，則可以搭配使用 `kubectl` `1.28`、`1.29` 或 `1.30` 版。若要安裝或升級 `kubectl`，請參閱 the section called “設定 kubectl 和 eksctl”。

步驟 1：建立 IAM 政策

Amazon S3 CSI 驅動程式的掛載點需要 Amazon S3 許可，才能與您的檔案系統互動。此節將介紹如何建立可授予必要許可的 IAM 政策。

下列範例政策遵循掛載點的 IAM 許可建議。或者，您可以使用 AWS 受管政策連結：iam/home？#/policies/arn:aws: iam：：aws：policy/AmazonS3FullAccess $jsonEditor【AmazonS3FullAccess，type="console"】，但此受管政策授予比掛載點所需的許可更多。

如需掛載點建議許可的詳細資訊，請參閱 GitHub 上的掛載點 IAM 許可。

1. 前往 https://console.aws.amazon.com/iam/ 開啟 IAM 主控台。
2. 在左側導覽窗格中選擇 Policies (政策)。
3. 在政策頁面上，選擇建立政策。
4. 對於政策編輯器，選擇 JSON。
5. 在政策編輯器下，複製並貼上以下內容：

> **Important**
>
> 將 `amzn-s3-demo-bucket1` 取代為您自己 Amazon S3 儲存貯體的名稱。

```
{
   "Version": "2012-10-17",
   "Statement": [
        {
            "Sid": "MountpointFullBucketAccess",
            "Effect": "Allow",
            "Action": [
                "s3:ListBucket"
            ],
            "Resource": [
                "arn:aws: s3:::amzn-s3-demo-bucket1"
            ]
        },
        {
            "Sid": "MountpointFullObjectAccess",
            "Effect": "Allow",
            "Action": [
```

```
                "s3:GetObject",
                "s3:PutObject",
                "s3:AbortMultipartUpload",
                "s3:DeleteObject"
            ],
            "Resource": [
                "arn:aws: s3:::amzn-s3-demo-bucket1/*"
            ]
        }
    ]
}
```

Amazon S3 Express One Zone 儲存類別引進的目錄儲存貯體會使用與一般用途儲存貯體不同的身分驗證機制。您應該使用 `s3:*`動作，而不是使用 `s3express:CreateSession`動作。如需目錄儲存貯體的資訊，請參閱《Amazon S3 使用者指南》中的目錄儲存貯體。

以下是您將用於目錄儲存貯體的最低權限政策範例。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "s3express:CreateSession",
            "Resource": "arn:aws: s3express:us-west-2:111122223333:bucket/amzn-s3-
demo-bucket1--usw2-az1--x-s3"
        }
    ]
}
```

6. 選擇下一步。
7. 在檢閱與建立頁面上，為您的政策命名。本範例逐步教學使用的名稱是 `AmazonS3CSIDriverPolicy`。
8. 選擇建立政策。

步驟 2：建立 IAM 角色

Amazon S3 CSI 驅動程式的掛載點需要 Amazon S3 許可，才能與您的檔案系統互動。此節將介紹如何建立 IAM 角色以委派這些許可。若要建立此角色，您可以使用下列其中一個工具：

- the section called “eksctl”

- the section called “AWS Management Console”
- the section called “AWS CLI”

> **Note**
>
> IAM 政策 `AmazonS3CSIDriverPolicy` 於上一節建立。

eksctl

使用 建立 Amazon S3 CSI 驅動程式 IAM 角色的掛載點 **`eksctl`**

若要建立 IAM 角色和 Kubernetes 服務帳戶，請執行下列命令。這些命令也會將 `AmazonS3CSIDriverPolicy` IAM 政策連接至角色、使用 IAM 角色的 Amazon Resource Name (ARN`s3-csi-controller-sa`) 標註 Kubernetes 服務帳戶 ()，並將 Kubernetes 服務帳戶名稱新增至 IAM 角色的信任政策。

```
CLUSTER_NAME=my-cluster
REGION=region-code
ROLE_NAME=AmazonEKS_S3_CSI_DriverRole
POLICY_ARN=AmazonEKS_S3_CSI_DriverRole_ARN
eksctl create iamserviceaccount \
    --name s3-csi-driver-sa \
    --namespace kube-system \
    --cluster $CLUSTER_NAME \
    --attach-policy-arn $POLICY_ARN \
    --approve \
    --role-name $ROLE_NAME \
    --region $REGION \
    --role-only
```

AWS Management Console

1. 前往 https://console.aws.amazon.com/iam/ 開啟 IAM 主控台。
2. 在左側導覽窗格中，選擇 Roles (角色)。
3. 在 Roles (角色) 頁面上，選擇 Create role (建立角色)。
4. 在 Select trusted entity (選取信任的實體) 頁面上，執行以下作業：
   a. 在 Trusted entity type (信任的實體類型) 區段中，選擇 Web identity (Web 身分)。

b. 對於 Identity provider (身分提供者)，為您的叢集選擇 OpenID Connect provider URL (OpenID Connect 供應商 URL)(如 Amazon EKS Overview (概觀) 下所示)。

如果沒有顯示 URLs，請檢閱先決條件。

c. 針對 Audience (對象)，選擇 `sts.amazonaws.com`。

d. 選擇下一步。

5. 在 Add permissions (新增許可) 頁面上，執行以下作業：

a. 在篩選政策方塊中，輸入 AmazonS3CSIDriverPolicy。

> **Note**
>
> 此政策於上一節建立。

b. 勾選在搜尋中傳回的 `AmazonS3CSIDriverPolicy` 結果左側的核取方塊。

c. 選擇下一步。

6. 在 Name, review, and create (命名、檢閱和建立) 頁面上，執行以下作業：

a. 針對角色名稱，輸入角色的唯一名稱，例如 AmazonEKS_S3_CSI_DriverRole。

b. 藉由連接標籤做為鍵值對，在新增標籤 (選用) 下將中繼資料新增至角色。如需有關在 IAM 中使用標籤的詳細資訊，請參閱《IAM 使用者指南》中的標記 IAM 資源。

c. 選擇建立角色。

7. 建立角色之後，在主控台中選擇角色，以開啟角色進行編輯。

8. 選擇 Trust Relationships (信任關係) 標籤，然後選擇 Edit Trust Relationship (編輯信任政策)。

9. 查找與下列相似的行：

```
"oidc.eks.region-code.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE:aud":
 "sts.amazonaws.com"
```

在上一行的末尾新增一個逗號，然後在上一行之後新增下一行。將 *region-code* 取代為您的叢集所在的 AWS 區域。使用叢集的 OIDC 提供者 ID 取代 *EXAMPLED539D4633E53DE1B71EXAMPLE*。

```
"oidc.eks.region-code.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE:sub":
 "system:serviceaccount:kube-system:s3-csi-driver-sa"
```

10確定運算Condition子設定為 "StringEquals"。

11選擇 Update policy (更新政策) 以完成操作。

AWS CLI

1. 檢視叢集的 OIDC 提供者 URL。使用您叢集的名稱取代 *my-cluster*。如果來自命令的輸出是 None，則請檢閱先決條件。

```
aws eks describe-cluster --name my-cluster --query "cluster.identity.oidc.issuer" --
output text
```

範例輸出如下。

```
https://oidc.eks.region-code.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE
```

2. 建立 IAM 角色，將 Kubernetes 服務帳戶授予 AssumeRoleWithWebIdentity 動作。

a. 將下列內容複製到名為 aws-s3-csi-driver-trust-policy.json 的檔案。使用您的帳戶 ID 取代 *111122223333*。將 *EXAMPLED539D4633E53DE1B71EXAMPLE* 和 *region-code* 取代為上一個步驟中傳回的值。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Federated": "arn:aws: iam::111122223333:oidc-provider/oidc.eks.region-
code.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE"
      },
      "Action": "sts:AssumeRoleWithWebIdentity",
      "Condition": {
        "StringEquals": {
          "oidc.eks.region-code.amazonaws.com/id/
EXAMPLED539D4633E53DE1B71EXAMPLE:sub": "system:serviceaccount:kube-system:s3-csi-
driver-sa",
          "oidc.eks.region-code.amazonaws.com/id/
EXAMPLED539D4633E53DE1B71EXAMPLE:aud": "sts.amazonaws.com"
        }
      }
    }
  ]
```

```
}
```

b. 建立角色。您可以將 *AmazonEKS_S3_CSI_DriverRole* 變更為不同的名稱，但如果您這麼做，也請務必在後續步驟中進行變更。

```
aws iam create-role \
  --role-name AmazonEKS_S3_CSI_DriverRole \
  --assume-role-policy-document file://"aws-s3-csi-driver-trust-policy.json"
```

3. 使用以下命令將先前建立的 IAM 政策連接至角色。

```
aws iam attach-role-policy \
  --policy-arn arn:aws: iam::aws:policy/AmazonS3CSIDriverPolicy \
  --role-name AmazonEKS_S3_CSI_DriverRole
```

> **Note**
>
> IAM 政策 AmazonS3CSIDriverPolicy 於上一節建立。

4. 如果您要將驅動程式安裝為 Amazon EKS 附加元件，請略過此步驟。對於驅動程式的自我管理安裝，請建立以您建立之 IAM 角色的 ARN 標註的 Kubernetes 服務帳戶。

a. 將下列內容儲存到名為 mountpoint-s3-service-account.yaml 的檔案中。使用您的帳戶 ID 取代 *111122223333*。

```
---
apiVersion: v1
kind: ServiceAccount
metadata:
  labels:
    app.kubernetes.io/name: aws-mountpoint-s3-csi-driver
  name: mountpoint-s3-csi-controller-sa
  namespace: kube-system
  annotations:
    eks.amazonaws.com/role-arn: arn:aws: iam::111122223333:role/
AmazonEKS_S3_CSI_DriverRole
```

b. 在您的叢集上建立 Kubernetes 服務帳戶。Kubernetes 服務帳戶 (mountpoint-s3-csi-controller-sa) 會標註您建立名為 *AmazonEKS_S3_CSI_DriverRole* 的 IAM 角色。

```
kubectl apply -f mountpoint-s3-service-account.yaml
```

> **Note**
>
> 當您在此程序中部署外掛程式時，其會建立並設定為使用名為 `s3-csi-driver-sa` 的服務帳戶。

步驟 3：安裝 Amazon S3 CSI 驅動程式的掛載點

您可以透過 Amazon EKS 附加元件安裝適用於 Amazon S3 CSI 驅動程式的掛載點。您可以使用下列工具將附加元件新增至您的叢集：

- the section called “eksctl”
- the section called “AWS Management Console”
- the section called “AWS CLI”

或者，您也可以將 Amazon S3 CSI 驅動程式的掛載點安裝為自我管理的安裝。如需執行自我管理安裝的指示，請參閱 GitHub 上的安裝。

從 開始`v1.8.0`，您可以設定可容忍 CSI 驅動程式 Pod 的污點。若要這樣做，請指定一組自訂的污點，以使用 來容忍，`node.tolerations`或使用 來清理所有污點`node.tolerateAllTaints`。如需詳細資訊，請參閱 Kubernetes 文件中的污點和容錯。

eksctl

使用 新增 Amazon S3 CSI 附加元件 **`eksctl`**

執行下列命令。將 *`my-cluster`* 取代為您的叢集名稱、將 *`111122223333`* 取代為您的 帳戶 ID，並將 *`AmazonEKS_S3_CSI_DriverRole`* 取代為先前建立的 IAM 角色名稱。

```
eksctl create addon --name aws-mountpoint-s3-csi-driver --cluster my-cluster \
  --service-account-role-arn arn:aws: iam::111122223333:role/
AmazonEKS_S3_CSI_DriverRole --force
```

如果您移除 *`--force`* 選項，且任何 Amazon EKS 附加元件設定與您現有的設定衝突，則更新 Amazon EKS 附加元件會失敗，而且您會收到錯誤訊息，協助您解決衝突。指定此選項之前，請確定 Amazon EKS 附加元件不會管理您需要管理的設定，因為這些設定會以此選項覆寫。如需有關此設定之其他選項的詳細資訊，請參閱 `eksctl` 文件中的 Addons (附加元件)。如需 Amazon EKS Kubernetes 欄位管理的詳細資訊，請參閱 the section called “您可以自訂的欄位”。

您可以透過eksctl組態檔案自訂 。如需詳細資訊，請參閱 eksctl 文件中的使用組態值。下列範例顯示如何容忍所有污點。

```
# config.yaml
...

addons:
- name: aws-mountpoint-s3-csi-driver
  serviceAccountRoleARN: arn:aws: iam::111122223333:role/AmazonEKS_S3_CSI_DriverRole
  configurationValues: |-
    node:
      tolerateAllTaints: true
```

AWS Management Console

1. 開啟 Amazon EKS 主控台。
2. 在左側導覽窗格中選擇叢集。
3. 選擇您要設定 Amazon S3 CSI 附加元件掛載點的叢集名稱。
4. 選擇附加元件索引標籤。
5. 選擇取得更多附加元件。
6. 在選取附加元件頁面上，執行下列動作：
   a. 在 Amazon EKS-addons 區段中，選取 Amazon S3 CSI 驅動程式掛載點核取方塊。
   b. 選擇下一步。
7. 在設定選取的附加元件設定頁面上，執行以下操作：
   a. 選取您要使用的版本。
   b. 針對選取 IAM 角色，選取您連接 Amazon S3 CSI 驅動程式 IAM 政策掛載點的 IAM 角色名稱。
   c. （選用）展開選用組態設定後，更新衝突解決方法。如果您選取覆寫，則可以使用 Amazon EKS 附加元件設定覆寫現有附加元件的一或多個設定。如果您未啟用此選項，且與現有設定發生衝突，則操作會失敗。您可以使用產生的錯誤訊息對此衝突進行疑難排解。選取此選項之前，請確定 Amazon EKS 附加元件不會管理自我管理所需的設定。
   d. （選用）在展開選用組態設定之後，在組態值欄位中設定容錯。
   e. 選擇下一步。
8. 在檢閱並新增頁面上，選擇建立。附加元件安裝完成後，您會看到已安裝的附加元件。

AWS CLI

使用 CLI 新增 Amazon S3 CSI AWS 附加元件的掛載點

執行下列命令。將 *my-cluster* 取代為您的叢集名稱、將 *111122223333* 取代為您的 帳戶 ID，並將 *AmazonEKS_S3_CSI_DriverRole* 取代為先前建立的角色名稱。

```
aws eks create-addon --cluster-name my-cluster --addon-name aws-mountpoint-s3-csi-
driver \
  --service-account-role-arn arn:aws: iam::111122223333:role/
AmazonEKS_S3_CSI_DriverRole
```

您可以使用 `--configuration-values`旗標自訂 命令。下列替代範例顯示如何容忍所有污點。

```
aws eks create-addon --cluster-name my-cluster --addon-name aws-mountpoint-s3-csi-
driver \
  --service-account-role-arn arn:aws: iam::111122223333:role/
AmazonEKS_S3_CSI_DriverRole \
  --configuration-values '{"node":{"tolerateAllTaints":true}}'
```

步驟 4：設定 Amazon S3 的掛載點

在大多數情況下，您只能使用儲存貯體名稱來設定 Amazon S3 的掛載點。如需設定 Amazon S3 掛載點的說明，請參閱在 GitHub 上設定 Amazon S3 掛載點。

步驟 5：部署範例應用程式

您可以將靜態佈建部署到現有 Amazon S3 儲存貯體上的驅動程式。如需詳細資訊，請參閱 GitHub 上的靜態佈建。

移除 Amazon S3 Amazon EKS 附加元件的掛載點

您有兩個選項可移除 Amazon S3 CSI 驅動程式的掛載點。

- Preserve add-on software on your cluster (在叢集上保留附加元件軟體)：此選項會移除任何設定的 Amazon EKS 管理。其也會移除 Amazon EKS 通知您更新的功能，並在您啟動更新後自動更新 Amazon EKS 附加元件。不過，該選項會保留您叢集上的附加元件軟體。此選項會使附加元件成為自我管理安裝，而不是 Amazon EKS 附加元件。使用此選項時，附加元件不會停機。本程序中的命令使用此選項。

- Remove add-on software entirely from your cluster (從叢集中完全移除附加元件軟體)：只有叢集上沒有資源依賴於附加元件提供的功能時，我們才會建議您將 Amazon EKS 附加元件從叢集中移除。若要執行此選項，請從您在此程序中使用的命令刪除 `--preserve`。

如果附加元件具有與其相關聯的 IAM 帳戶，則不會移除 IAM 帳戶。

您可以使用下列工具來移除 Amazon S3 CSI 附加元件：

- the section called “eksctl”
- the section called “AWS Management Console”
- the section called “AWS CLI”

eksctl

使用 移除 Amazon S3 CSI 附加元件 **`eksctl`**

將 *`my-cluster`* 取代為您的叢集名稱，然後執行下列命令。

```
eksctl delete addon --cluster my-cluster --name aws-mountpoint-s3-csi-driver --preserve
```

AWS Management Console

1. 開啟 Amazon EKS 主控台。
2. 在左側導覽窗格中選擇叢集。
3. 選取要移除 Amazon EBS CSI 附加元件的叢集名稱。
4. 選擇附加元件索引標籤。
5. 選擇 Amazon S3 CSI 驅動程式的掛載點。
6. 選擇移除。
7. 在移除：aws-mountpoint-s3-csi-driver 確認對話方塊中，執行下列動作：
   a. 若希望 Amazon EKS 停止管理附加元件的設定，請選取在叢集上保留。若要在叢集上保留附加元件軟體，請執行此動作。如此一來，您就可以自行管理附加元件的所有設定。
   b. 輸入 `aws-mountpoint-s3-csi-driver`。
   c. 選取 Remove (移除)。

AWS CLI

使用 CLI 移除 Amazon S3 CSI AWS 附加元件

將 *my-cluster* 取代為您的叢集名稱，然後執行下列命令。

```
aws eks delete-addon --cluster-name my-cluster --addon-name aws-mountpoint-s3-csi-
driver --preserve
```

啟用 CSI 磁碟區的快照功能

快照功能允許資料point-in-time副本。若要讓此功能在 Kubernetes 中運作，您需要具備快照支援的 CSI 驅動程式（例如 Amazon EBS CSI 驅動程式）和 CSI 快照控制器。快照控制器可作為 Amazon EKS 受管附加元件或自我管理安裝使用。

以下是使用 CSI 快照控制器的一些注意事項。

- 快照控制器必須與具有快照功能的 CSI 驅動程式一起安裝。如需 Amazon EBS CSI 驅動程式的安裝說明，請參閱 the section called “Amazon EBS”。
- Kubernetes 不支援透過 CSI 遷移提供的磁碟區快照，例如使用`StorageClass`具有佈建器的 Amazon EBS 磁碟區`kubernetes.io/aws-ebs`。必須使用參照 CSI 驅動程式佈建程式 (`ebs.csi.aws.com`) 的 `StorageClass` 來建立磁碟區。
- Amazon EKS Auto Mode 不包含快照控制器。EKS Auto Mode 的儲存功能與快照控制器相容。

我們建議您透過 Amazon EKS 受管附加元件安裝 CSI 快照控制器。此附加元件包含在 Amazon EKS 上建立和管理快照所需的自訂資源定義 (CRDs)。若要將 Amazon EKS 附加元件新增至叢集，請參閱 the section called “建立附加元件”。如需附加元件的詳細資訊，請參閱 the section called “Amazon EKS 附加元件”。

或者，如果您想要自我管理安裝 CSI 快照控制器，請參閱 GitHub `external-snapshotter`上上游 Kubernetes 中的用量。

設定 Amazon EKS 叢集的聯網

您的 Amazon EKS 叢集是在 VPC 中建立。Pod 網路是由 Amazon VPC 容器網路界面 (CNI) 外掛程式為在 AWS 基礎設施上執行的節點提供。如果您在自己的基礎設施上執行節點，請參閱 the section called “設定 CNI”。本章包括以下主題，可讓您進一步了解有關叢集聯網。

主題

- 從管理主控台將現有的 VPC 子網路新增至 Amazon EKS 叢集
- 檢視 VPC 和子網路的 Amazon EKS 聯網需求
- 為您的 Amazon EKS 叢集建立 Amazon VPC
- 檢視叢集的 Amazon EKS 安全群組需求
- 管理 Amazon EKS 叢集的網路附加元件

從管理主控台將現有的 VPC 子網路新增至 Amazon EKS 叢集

1. 在管理主控台中導覽至您的叢集
2. 從聯網索引標籤選取管理 VPC 資源
3. 從子網路下拉式清單中，從叢集的 VPC 選取其他子網路。

若要建立新的 VPC 子網路：

- 檢閱 EKS 子網路需求
- 請參閱《Amazon Virtual Private Cloud 使用者指南》中的建立子網路。

檢視 VPC 和子網路的 Amazon EKS 聯網需求

當您建立叢集時，您可以指定 VPC 和至少兩個位於不同可用區域的子網。本主題提供您與叢集一起使用的 VPC 和子網的 Amazon EKS 特定需求和注意事項的概觀。如果您沒有可搭配 Amazon EKS 使用的 VPC，請參閱 the section called “建立 VPC”。如果您要在 AWS Outposts 上建立本機或延伸叢集，請參閱 the section called “建立 VPC 和子網路”而非本主題。本主題中的內容適用於具有混合節點的 Amazon EKS 叢集。如需混合節點的其他聯網需求，請參閱 the section called “準備聯網”。

VPC 要求和注意事項

當您建立叢集時，您指定的 VPC 必須符合下列要求和注意事項：

- 針對您要建立的叢集、任何節點和其他 Kubernetes 資源，VPC 必須有足夠數目的 IP 地址。如果您想要使用的 VPC 沒有足夠的 IP 地址數量，請嘗試增加可用的 IP 地址數量。

  您可以更新叢集組態來變更叢集使用的子網路和安全群組，藉此完成此操作。您可以從 更新 AWS Management Console、最新版本的 AWS CLI、 AWS CloudFormation 和`eksctl`版本 `v0.164.0-rc.0` 或更新版本。這麼做才能為子網路提供更多可用 IP 地址，以成功升級叢集版本。

  > **Important**
  >
  > 您新增的所有子網路必須與建立叢集時原本提供的 AZ 位於同一組。新的子網路必須滿足所有其他要求，例如，它們必須具有足夠的 IP 地址。
  > 舉例來說，假設您建立叢集並指定四個子網路。依照您指定它們的順序，第一個子網路位於 `us-west-2a` 可用區域中，第二個和第三個子網路位於 `us-west-2b` 可用區域中，而第四個子網路位於 `us-west-2c` 可用區域中。如果要變更子網路，則必須在三個可用區域中各提供至少一個子網路，而且子網路必須與原始子網路位於相同的 VPC 中。

  如果您需要的 IP 地址超過 VPC 中的 CIDR 區塊，您可以將其他無類別域間路由 (CIDR) 區塊與 VPC 產生關聯，以新增其他 CIDR 區塊。您可以在建立叢集的之前或之後，將私有 (RFC 1918) 和公有 (非 RFC 1918) CIDR 區塊與 VPC 關聯。

  您可以在新增新 CIDR 區塊後立即新增使用該區塊的節點。不過，由於控制平面只有在對帳完成後才會辨識新的 CIDR 區塊，因此您與 VPC 相關聯的 CIDR 區塊可能需要最多一小時才能辨識叢集。然後，您可以在新的 CIDR `kubectl cp`區塊中對節點和 Pod `kubectl exec`執行 `kubectl logs`、、 和 `kubectl port-forward`命令 （這些命令使用 `kubelet API`)。此外，如果您有做為 Webhook 後端的 Pod，則必須等待控制平面對帳完成。
- 當您透過 Transit Gateway、VPCs 對等互連或其他聯網組態將 EKS 叢集連線至其他 VPC 時，請避免 IP 地址範圍重疊。當您 EKS 叢集的服務 CIDR 與連線 VPC 的 CIDR 重疊時，就會發生 CIDR 衝突。在這些情況下，ServiceIP 地址優先於具有相同 IP 地址VPCs 中的資源，但流量路由可能會變得無法預測，且應用程式可能無法連線到預期資源。

  為了避免 CIDR 衝突，請確保您的 EKS 服務 CIDR 不會與任何連線的 VPC CIDRs 重疊，並維護所有 CIDR 指派的集中記錄。如果您遇到 CIDR 重疊，您可以使用傳輸閘道搭配共用服務 VPC。如需

詳細資訊，請參閱隔離 VPC 與共享服務和混合網路中的 Amazon EKS VPC 可路由 IP 地址保護模式。此外，請參閱《EKS 最佳實務指南VPCs 和子網路考量頁面跨 VPC 的通訊一節。

- 如果您希望 Kubernetes 將IPv6地址指派給 Pod 和服務，請將 IPv6 CIDR 區塊與您的 VPC 建立關聯。如需詳細資訊，請參閱 Amazon VPC 使用者指南中的建立 IPv6 CIDR 區塊與 VPC 的關聯。您無法將IPv6地址與在混合節點上執行的 Pod 和服務搭配使用，也無法將混合節點與使用 IPv6 IP 地址系列設定的叢集搭配使用。
- VPC 必須支援 DNS 主機名稱和 DNS 解析。否則，節點無法註冊到您的叢集。如需詳細資訊，請參閱《Amazon VPC 使用者指南》中的 VPC 的 DNS 屬性。
- VPC 可能需要使用 AWS PrivateLink 的 VPC 端點。如需詳細資訊，請參閱the section called “子網需求和注意事項”。

如果您使用 Kubernetes `1.14` 或舊版建立叢集，Amazon EKS 會將以下標籤新增至您的 VPC：

金鑰	值
`kubernetes.io/cluster/` *`my-cluster`*	`owned`

此標籤僅供 Amazon EKS 使用。您可以在不影響服務的情況下移除該標籤。它不會與版本 `1.15`或更新版本的叢集搭配使用。

子網需求和注意事項

當您建立叢集時，Amazon EKS 會在您指定的子網中建立 2-4 個彈性網路介面。這些網路介面啟用您的叢集和 VPC 之間的通訊。這些網路介面還啟用了 Kubernetes 的功能，例如 `kubectl exec` 和 `kubectl logs`。每個 Amazon EKS 建立的網路介面都在其說明中包含 `Amazon EKS` *`cluster-name`* 字樣。

Amazon EKS 可以在您建立叢集時指定的任何子網中建立其網路介面。建立叢集後，您可以變更 Amazon EKS 要在其中建立網路介面的子網路。當您更新叢集的 Kubernetes 版本時，Amazon EKS 會刪除其建立的原始網路介面，並建立新的網路介面。這些網路介面可能在與原始網路介面相同的子網中建立，或在與原始網路介面不同的子網中建立。若要控制在其中建立的子網路網路介面，可以將建立叢集時指定的子網路數量限制為僅有兩個子網路，或在建立叢集後更新子網路。

叢集的子網路要求

您建立或更新叢集時指定的子網路必須符合下列要求：

- 每個子網必須至少有六個 IP 地址，以供 Amazon EKS 使用。但是，我們建議至少使用 16 個 IP 地址。
- 子網路必須至少位於兩個不同的可用區域。
- 子網路不能位於 AWS Outposts 或 AWS Wavelength 中。但是，如果您的 VPC 中有它們，則可以部署自我管理的節點和 Kubernetes 資源到這些類型的子網。如需自我管理節點的詳細資訊，請參閱 the section called “自我管理的節點”。
- 子網可以是公有子網，也可以是私有子網。不過，如果可能的話，我們建議您指定私有子網。公有子網路是具有包含網際網路閘道路由之路由表的子網路，而私有子網路則是具有不包含網際網路閘道路由之路由表的子網路。
- 子網路不能位於下列可用區域：

AWS 區域	區域名稱	不允許的可用區域 ID
us-east-1	美國東部 (維吉尼亞北部)	use1-az3
us-west-1	美國西部 (加利佛尼亞北部)	usw1-az2
ca-central-1	加拿大 (中部)	cac1-az3

依元件的 IP 地址系列用量

下表包含 Amazon EKS 每個元件所使用的 IP 地址系列。您可以使用網路位址轉譯 (NAT) 或其他相容性系統，從具有資料表項目「否」值的系列中的來源 IP 地址連接到這些元件。

功能可能會因叢集的 IP 系列 (ipFamily) 設定而有所不同。此設定會變更 Kubernetes 指派給 Services 的 CIDR 區塊所使用的 IP 地址類型。設定值為 IPv4 的叢集稱為 IPv4 叢集，設定值為 IPv6 的叢集稱為 IPv6 叢集。

元件	IPv4 地址	IPv6 地址	雙堆疊地址
EKS API 公有端點	是[1, 3]	是[1, 3]	是[1, 3]
EKS API VPC 端點	是	否	否
EKS 驗證 API 公有端點 (EKS Pod 身分)	是 [1]	是 [1]	是 [1]

元件	IPv4 地址	IPv6 地址	雙堆疊地址
EKS 身分驗證 API VPC 端點 (EKS Pod 身分)	是 [1]	是 [1]	是 [1]
IPv4 Kubernetes 叢集公有端點[2]	是	否	否
IPv4 Kubernetes 叢集私有端點[2]	是	否	否
IPv6 Kubernetes 叢集公有端點[2]	是[1, 4]	是[1, 4]	是[4]
IPv6 Kubernetes 叢集私有端點[2]	是[1, 4]	是[1, 4]	是[4]
Kubernetes 叢集子網路	是[2]	否	是[2]
節點主要 IP 地址	是[2]	否	是[2]
服務 IP 地址的叢集 CIDR 範圍	是[2]	是[2]	否
來自 VPC CNI 的 Pod IP 地址	是[2]	是[2]	否
IRSA OIDC 發行者 URLs	是[1, 3]	是[1, 3]	是[1, 3]

Note

[1] 端點是同時具有 IPv4和 IPv6地址的雙堆疊。您在 外部的應用程式 AWS、叢集的節點，以及叢集內的 Pod 都可以透過 IPv4或 到達此端點IPv6。

[2] 當您建立IPv4叢集時，您可以在叢集的 IP 系列 (ipFamily) 設定IPv6中選擇叢集和叢集，這無法變更。相反地，您必須在建立另一個叢集和遷移工作負載時選擇不同的設定。

[3] 雙堆疊端點於 2024 年 8 月推出。若要搭配 CLI AWS 使用雙堆疊端點，請參閱 SDK AWS SDKs 和工具參考指南中的雙堆疊和 FIPS 端點組態。以下列出新的端點：

EKS API 公有端點

```
eks.region.api.aws
```

IRSA OIDC 發行者 URLs

```
oidc-eks.region.api.aws
```

[4] 雙堆疊叢集端點於 2024 年 10 月推出。EKS 會為在此日期之後建立並在叢集的 IP 系列 (ipFamily) 設定IPv6中選取的新叢集建立下列端點：

EKS 叢集公有/私有端點

```
eks-cluster.region.api.aws
```

節點的子網路要求

您可以將節點和 Kubernetes 資源部署到您建立叢集時指定的相同子網。不過，這不是必要的。這是因為您也可以將節點和 Kubernetes 資源部署到您在建立叢集時未指定的子網路。如果您將節點部署到不同的子網路，Amazon EKS 不會在這些子網路中建立叢集網路介面。您部署節點和 Kubernetes 資源的任何子網必須符合以下要求：

- 子網必須具有足夠的可用 IP 地址，以便將所有節點和 Kubernetes 資源部署至其中。
- 如果您希望 Kubernetes 將IPv6地址指派給 Pod 和服務，則必須有一個與子網路相關聯的 IPv6 CIDR 區塊和一個 IPv4 CIDR 區塊。如需詳細資訊，請參閱《Amazon VPC 使用者指南》中的將 IPv6 CIDR 區塊與您的子網路建立關聯。與子網關聯的路由表必須包含傳送到 IPv4 和 IPv6 地址的路由。如需詳細資訊，請參閱《Amazon VPC 使用者指南》中的 Routes (路由)。Pod 僅指派一個 IPv6 地址。但是，Amazon EKS 為您的叢集和節點建立的網路介面受指派一個 IPv4 和一個 IPv6 地址。
- 如果您需要從網際網路到 Pod 的傳入存取，請確定至少有一個公有子網路，其具有足夠的可用 IP 地址，可供部署負載平衡器和輸入。您可以將負載平衡器部署至公有子網。負載平衡器可以對私有或公有子網路中的 Pod 進行負載平衡。如果可能的話，我們建議您將節點部署至私有子網。
- 如果您計畫將節點部署到公有子網，則該子網必須自動指派 IPv4 公有地址或 IPv6 地址。如果您將節點部署到具有相關聯 IPv6 CIDR 區塊的子網，則私有子網也必須自動指派 IPv6 地址。如果您在 2020 年 3 月 26 日之後使用 Amazon EKS 提供的 AWS CloudFormation 範本來部署 VPC，則會啟

用此設定。如果在此日期之前使用範本部署 VPC，或者使用自己的 VPC，則必須手動啟用此設定。如需範本，請參閱 the section called “建立 VPC”。如需詳細資訊，請參閱《Amazon VPC 使用者指南》中的修改子網路的公有 IPv4 定址屬性和修改子網路的 IPv6 定址屬性。

- 如果您部署節點的子網路是私有子網路，且其路由表不包含網路位址轉譯 (NAT) 裝置 (IPv4) 或輸出限定閘道 (IPv6) 的路由，請使用 AWS PrivateLink將 VPC 端點新增至您的 VPC。節點和 Pod 需要通訊的所有 AWS 服務都需要 VPC 端點。範例包括 Amazon ECR、Elastic Load Balancing、Amazon CloudWatch、 AWS Security Token Service 和 Amazon Simple Storage Service (Amazon S3)。端點必須包含節點所在的子網。並非所有 AWS 服務都支援 VPC 端點。如需詳細資訊，請參閱什麼是 AWS PrivateLink？以及AWS 與 AWS PrivateLink 整合的 服務。有關更多 Amazon EKS 要求的列表，請參閱 the section called “私有叢集”。
- 如果要將負載平衡器部署到子網，則子網必須具有以下標籤：
  - 私有子網路

金鑰	值
kubernetes.io/role/internal-elb	1

  - 公有子網路

金鑰	值
kubernetes.io/role/elb	1

建立版本 1.18和更早版本的 Kubernetes 叢集時，Amazon EKS 會將下列標籤新增至指定的所有子網路。

金鑰	值
kubernetes.io/cluster/ *my-cluster*	shared

現在建立新的 Kubernetes 叢集時，Amazon EKS 不會將標籤新增至子網路。如果該標籤位於先前為之前版本的叢集所使用的子網路上1.19，則當叢集更新為較新版本時，不會自動從子網路中移除該標籤。版本 2.1.1 AWS Load Balancer控制器需要此標籤。如果您使用較新版本的 Load Balancer

Controller，則您無須中斷服務便可移除標籤。如需控制器的詳細資訊，請參閱 the section called “AWS Load Balancer控制器”。

如果您使用 `eksctl`或任何 Amazon EKS AWS CloudFormation VPC 範本來部署 VPC，則適用下列條件：

- 在 2020 年 3 月 26 日或之後：公有 IPv4 地址會由公有子網自動指派給此公有子網中已部署的新節點。
- 2020 年 3 月 26 日之前 – 公有子網路不會自動將公有IPv4地址指派給部署到公有子網路的新節點。

這項變更會影響部署於公有子網中的新節點群組的下列行為：

- 受管節點群組 – 如果節點群組在 2020 年 4 月 22 日或之後部署到公有子網路，則必須為公有子網路啟用公有 IP 地址的自動指派。如需詳細資訊，請參閱修改子網的公有 IPv4 定址屬性。
- Linux、Windows 或 Arm 自我管理節點群組 – 如果節點群組在 2020 年 3 月 26 日或之後部署到公有子網路，則必須為公有子網路啟用公有 IP 地址的自動指派。否則，必須改用公有 IP 地址啟動節點。如需詳細資訊，請參閱修改子網路的公有 IPv4 定址屬性或在執行個體啟動期間指派公有 IPv4 地址。

共用子網路需求和注意事項

您可以使用 VPC 共用，與相同 AWS 組織內的其他 AWS 帳戶共用子網路。您可以在共用子網路中建立 Amazon EKS 叢集，但會有下列考量：

- VPC 子網路的擁有者必須與參與者帳戶共用子網路，該帳戶才能在其中建立 Amazon EKS 叢集。
- 您無法使用 VPC 的預設安全群組啟動資源，因為它屬於擁有者。此外，參與者無法啟動使用由其他參與者或擁有者所擁有之安全群組的資源。
- 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。
- 如果叢集是由參與者建立，則需要考量下列事項：
  - 必須在該帳戶中建立叢集 IAM 角色和節點 IAM 角色。如需詳細資訊，請參閱the section called “叢集 IAM 角色”及the section called “節點 IAM 角色”。
  - 所有節點都必須由相同的參與者建立，包括受管節點群組。

- 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。
- 如果您使用適用於 Kubernetes 的 Amazon VPC CNI 外掛程式的自訂聯網功能，則需要使用擁有者帳戶中列出的可用區域 ID 映射來建立每個 `ENIConfig`。如需詳細資訊，請參閱the section called “自訂聯網”。

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。

為您的 Amazon EKS 叢集建立 Amazon VPC

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。此虛擬網路非常近似於您在自有資料中心內運作的傳統網路。但是，它帶來了使用 Amazon Web 服務的可擴展基礎架構的好處。部署生產 Amazon EKS 叢集前，建議您完整了解 Amazon VPC 服務。如需詳細資訊，請參閱 Amazon VPC 使用者指南。

Amazon EKS 叢集、節點和 Kubernetes 資源已部署到 VPC。如果您要將現有的 VPC 與 Amazon EKS 結合使用，則該 VPC 必須符合 the section called “VPC 和子網要求” 中描述的要求。本主題說明如何使用 Amazon EKS provided AWS CloudFormation 範本建立符合 Amazon EKS 要求的 VPC。部署範本後，您可以檢視範本建立的資源，確切了解其建立的資源，以及這些資源的組態。如果您使用的是混合節點，VPC 必須在內部部署網路的路由表中具有路由。如需混合節點網路需求的詳細資訊，請參閱 the section called “準備聯網”。

先決條件

若要為 Amazon EKS 建立 VPC，您必須擁有建立 Amazon VPC 資源所需的 IAM 許可。這些資源包括 VPC、子網、安全群組、路由表和路由，以及網際網路和 NAT 閘道。如需詳細資訊，請參閱《Amazon VPC 使用者指南》中的使用公有子網路範例政策建立 VPC，以及《服務授權參考》中的動作完整清單。

您可以建立含公有和私有子網路、僅公有子網路或僅私有子網路的 VPC。

公有和私有子網路

此 VPC 具有兩個公用和兩個私有子網路。公有子網路的關聯路由表具有網際網路閘道的路由。不過，私有子網路的路由表沒有網際網路閘道的路由。一個公有子網路和一個私有子網路會部署到相同的可用

區域。其他公有和私有子網路會部署到相同區域中的第二個可用區域 AWS 。我們建議大多數部署使用此選項。

使用此選項，您可以將節點部署到私有子網。此選項可讓 Kubernetes 將負載平衡器部署到公有子網路，這些子網路可以將流量負載平衡到私有子網路中節點上執行的 Pod。公有IPv4地址會自動指派給部署到公有子網路的節點，但公有IPv4地址不會指派給部署到私有子網路的節點。

您可以將 IPv6 地址指派給公有和私有子網路中的節點。私有子網路中的節點可以與叢集和其他 AWS 服務通訊。Pod 可以使用IPv4位址透過 NAT 閘道與網際網路通訊，或使用在每個可用區域中部署IPv6的地址與僅限傳出網際網路閘道通訊。部署的安全群組包含拒絕來自叢集或節點之外的來源的所有傳入流量，但卻允許所有傳出流量的規則。子網路會加上標籤，以便 Kubernetes 可以將負載平衡器部署到它們。

a. 開啟 AWS CloudFormation 主控台。

b. 從導覽列中選取支援 Amazon EKS AWS 的區域。

c. 選擇 Create stack (建立堆疊)、With new resources (standard) (使用新資源 (標準))。

d. 在 Prerequisite - Prepare template (必要條件 - 準備範本) 下，請確定選擇 Template is ready (範本已就緒)，然後在 Specify template (指定範本) 下選取 Amazon S3 URL。

e. 您可以建立僅支援 IPv4 的 VPC，或支援 IPv4 和 IPv6 的 VPC。將下列 URL 之一貼入 Amazon S3 URL 下的文字區域並選擇 Next (下一步)：

- IPv4

```
https://s3.us-west-2.amazonaws.com/amazon-eks/cloudformation/2020-10-29/amazon-eks-vpc-private-subnets.yaml
```

- IPv4 和 IPv6

```
https://s3.us-west-2.amazonaws.com/amazon-eks/cloudformation/2020-10-29/amazon-eks-ipv6-vpc-public-private-subnets.yaml
```

a. 在 Specify stack details (識別堆疊詳細資訊) 頁面上，輸入參數，然後選擇 Next (下一步)。

- 堆疊名稱：為您的 AWS CloudFormation 堆疊選擇堆疊名稱。例如，您可以使用在先前步驟中使用的範本名稱。此名稱僅能使用英數字元 (區分大小寫) 和連字號。它必須以英數字元開頭，且長度不可超過 100 個字元。名稱在您要建立叢集 AWS 的區域和 AWS 帳戶中必須是唯一的。

- VpcBlock：選擇適用於您 VPC 的 IPv4 CIDR 範圍。您部署的每個節點、Pod 和負載平衡器都會獲指派此區塊IPv4的地址。IPv4 預設值為大多數實作提供足夠的 IP 地址，但如果沒有，則可以進行變更。如需詳細資訊，請參閱《Amazon VPC 使用者指南》中的 VPC 和子網路規模調整。您也可以在 VPC 建立後，將其他 CIDR 區塊新增至 VPC。如果您要建立 IPv6 VPC，會自動從 Amazon 的全域單點傳送地址空間為您指派 IPv6 CIDR 範圍。
- PublicSubnet01Block：指定適用於公有子網路 1 的 IPv4 CIDR 區塊。預設值為大多數實作提供足夠的 IP 地址，但如果沒有，則可以進行變更。如果您要建立 IPv6 VPC，則會在範本中為您指定此區塊。
- PublicSubnet02Block：指定適用於公有子網路 2 的 IPv4 CIDR 區塊。預設值為大多數實作提供足夠的 IP 地址，但如果沒有，則可以進行變更。如果您要建立 IPv6 VPC，則會在範本中為您指定此區塊。
- PrivateSubnet01Block：指定適用於私有子網路 1 的 IPv4 CIDR 區塊。預設值為大多數實作提供足夠的 IP 地址，但如果沒有，則可以進行變更。如果您要建立 IPv6 VPC，則會在範本中為您指定此區塊。
- PrivateSubnet02Block：指定適用於私有子網路 2 的 IPv4 CIDR 區塊。預設值為大多數實作提供足夠的 IP 地址，但如果沒有，則可以進行變更。如果您要建立 IPv6 VPC，則會在範本中為您指定此區塊。

b. (選用) 在 Configure stack options (設定堆疊選項) 頁面上，為堆疊資源加上標籤，然後選擇 Next (下一步)。

c. 在 Review (檢閱) 頁面上，選擇 Create stack (建立堆疊)。

d. 堆疊建立後，從主控台將其選取，然後選擇 Outputs (輸出)。

e. 為已建立的 VPC 記錄 VpcId。建立叢集和節點時，您需要此值。

f. 記錄已建立的子網路的 SubnetIds (子網路識別碼)，以及您是否將它們建立為公有或私有子網路的子網路。建立叢集和節點時，您至少需要其中兩個。

g. 如果您建立了 IPv4 VPC，請跳過此步驟。如果建立 IPv6 VPC，則必須為範本建立的公有子網路啟用自動指派 IPv6 地址選項。已為私有子網路啟用該設定。若要啟用此設定，請完成下列步驟：

   i. 在 https://console.aws.amazon.com/vpc/ 開啟 Amazon VPC 主控台。

   ii. 在左側導覽窗格中，選擇 Subnets (子網路)。

   iii. 選取其中一個公有子網路 (***stack-name***/SubnetPublic01 或 ***stack-name***/SubnetPublic02 包含公有文字) ，然後選擇動作、編輯子網路設定。

   iv. 勾選 Enable auto-assign IPv6 address (啟用自動指派 IPv6 地址) 核取方塊，然後選擇 Save (儲存)。

   v. 為您的其他公有子網路再次完成上述步驟。

僅限公有子網路

此 VPC 有三個公有子網路，部署到 AWS 區域中的不同可用區域。所有節點都會自動指派公有 IPv4 地址，並且可以透過網際網路閘道傳送和接收網際網路流量。部署的安全群組會拒絕所有輸入流量，並允許所有輸出流量。子網路會加上標籤，以便 Kubernetes 可以將負載平衡器部署到它們。

a. 開啟 AWS CloudFormation 主控台。

b. 從導覽列中選取支援 Amazon EKS AWS 的區域。

c. 選擇 Create stack (建立堆疊)、With new resources (standard) (使用新資源 (標準))。

d. 在 Prepare template (準備範本) 下，請確定 Template is ready (範本已就緒)，然後在 Template source (範本來源) 下選取 Amazon S3 URL。

e. 將下列 URL 貼入 Amazon S3 URL 下的文字區域並選擇 Next (下一步)：

```
https://s3.us-west-2.amazonaws.com/amazon-eks/cloudformation/2020-10-29/amazon-eks-vpc-sample.yaml
```

a. 在 Specify Details (指定詳細資訊) 頁面上，輸入參數，然後選擇 Next (下一步)。

- 堆疊名稱：為您的 AWS CloudFormation 堆疊選擇堆疊名稱。例如，您可以呼叫 *amazon-eks-vpc-sample*。此名稱僅能使用英數字元 (區分大小寫) 和連字號。它必須以英數字元開頭，且長度不可超過 100 個字元。名稱在您要建立叢集 AWS 的區域和 AWS 帳戶中必須是唯一的。
- VpcBlock：選擇適用於您 VPC 的 CIDR 區塊。您部署的每個節點、Pod 和負載平衡器都會獲指派此區塊IPv4的地址。IPv4 預設值為大多數實作提供足夠的 IP 地址，但如果沒有，則可以進行變更。如需詳細資訊，請參閱《Amazon VPC 使用者指南》中的 VPC 和子網路規模調整。您也可以在 VPC 建立後，將其他 CIDR 區塊新增至 VPC。
- Subnet01Block：指定適用於子網路 1 的 CIDR 區塊。預設值為大多數實作提供足夠的 IP 地址，但如果沒有，則可以進行變更。
- Subnet02Block：指定適用於子網路 2 的 CIDR 區塊。預設值為大多數實作提供足夠的 IP 地址，但如果沒有，則可以進行變更。
- Subnet03Block：指定適用於子網路 3 的 CIDR 區塊。預設值為大多數實作提供足夠的 IP 地址，但如果沒有，則可以進行變更。

b. (選用) 在 Options (選項) 頁面上，為堆疊資源加上標籤。選擇 Next (下一步)。

c. 在 Review (檢閱) 頁面上，選擇 Create (建立)。

d. 堆疊建立後，從主控台將其選取，然後選擇 Outputs (輸出)。

e. 為已建立的 VPC 記錄 VpcId。建立叢集和節點時，您需要此值。

f. 為已建立的子網路記錄 SubnetIds。建立叢集和節點時，您至少需要其中兩個。

g. （選用）您部署至此 VPC 的任何叢集都可以將私有IPv4地址指派給您的 Pod 和服務。如果您想要將叢集部署至此 VPC，以將私有IPv6地址指派給您的 Pod 和服務，請更新 VPC、子網路、路由表和安全群組。如需詳細資訊，請參閱《Amazon VPC 使用者指南》中的將IPv4 中現有的VPC 遷移至IPv6。Amazon EKS 要求您的子網路啟用 Auto-assign IPv6 地址選項。預設為停用。

只有私有子網路

此 VPC 有三個私有子網路，部署到 AWS 區域中的不同可用區域。部署到子網路的資源無法存取網際網路，網際網路也無法存取子網路中的資源。範本會使用 AWS PrivateLink 為節點通常需要存取的數個 AWS 服務建立 VPC 端點。如果您的節點需要傳出網際網路存取權，您可以在建立 VPC 後的每個子網的可用區域中新增公有 NAT 閘道。建立的安全群組會拒絕所有傳入流量，部署到子網中的資源除外。安全群組亦允許所有傳出流量。子網路會加上標籤，以便 Kubernetes 可以對它們部署負載平衡器。如果您要使用此組態建立 VPC，請參閱 the section called “私有叢集” 以了解其他需求和考量事項。

a. 開啟 AWS CloudFormation 主控台。

b. 從導覽列中選取支援 Amazon EKS AWS 的區域。

c. 選擇 Create stack (建立堆疊)、With new resources (standard) (使用新資源 (標準))。

d. 在 Prepare template (準備範本) 下，請確定 Template is ready (範本已就緒)，然後在 Template source (範本來源) 下選取 Amazon S3 URL。

e. 將下列 URL 貼入 Amazon S3 URL 下的文字區域並選擇 Next (下一步)：

```
https://s3.us-west-2.amazonaws.com/amazon-eks/cloudformation/2020-10-29/amazon-eks-fully-private-vpc.yaml
```

a. 在 Specify Details (指定詳細資訊) 頁面上，輸入參數，然後選擇 Next (下一步)。

- 堆疊名稱：為您的 AWS CloudFormation 堆疊選擇堆疊名稱。例如，您可以呼叫 *amazon-eks-fully-private-vpc*。此名稱僅能使用英數字元 (區分大小寫) 和連字號。它必須以英數字元開頭，且長度不可超過 100 個字元。名稱在您要建立叢集 AWS 的區域和 AWS 帳戶中必須是唯一的。
- VpcBlock：選擇適用於您 VPC 的 CIDR 區塊。您部署的每個節點、Pod 和負載平衡器都會獲指派此區塊IPv4的地址。IPv4 預設值為大多數實作提供足夠的 IP 地址，但如果沒有，則可以進行變更。如需詳細資訊，請參閱《Amazon VPC 使用者指南》中的 VPC 和子網路規模調整。您也可以在 VPC 建立後，將其他 CIDR 區塊新增至 VPC。

- PrivateSubnet01Block：指定適用於子網路 1 的 CIDR 區塊。預設值為大多數實作提供足夠的 IP 地址，但如果沒有，則可以進行變更。
- PrivateSubnet02Block：指定適用於子網路 2 的 CIDR 區塊。預設值為大多數實作提供足夠的 IP 地址，但如果沒有，則可以進行變更。
- PrivateSubnet03Block：指定適用於子網路 3 的 CIDR 區塊。預設值為大多數實作提供足夠的 IP 地址，但如果沒有，則可以進行變更。

b. (選用) 在 Options (選項) 頁面上，為堆疊資源加上標籤。選擇 Next (下一步)。

c. 在 Review (檢閱) 頁面上，選擇 Create (建立)。

d. 堆疊建立後，從主控台將其選取，然後選擇 Outputs (輸出)。

e. 為已建立的 VPC 記錄 VpcId。建立叢集和節點時，您需要此值。

f. 為已建立的子網路記錄 SubnetIds。建立叢集和節點時，您至少需要其中兩個。

g. （選用）您部署至此 VPC 的任何叢集都可以將私有IPv4地址指派給您的 Pod 和服務。如果您希望將叢集部署到此 VPC 以指派私有IPv6地址給 Pod 和服務，請更新 VPC、子網路、路由表和安全群組。如需詳細資訊，請參閱《Amazon VPC 使用者指南》中的將IPv4 中現有的VPC 遷移至IPv6。Amazon EKS 要求子網路啟用Auto-assign IPv6地址選項（預設為停用）。

檢視叢集的 Amazon EKS 安全群組需求

本主題說明 Amazon EKS 叢集的安全群組要求。

預設叢集安全群組

當您建立叢集時，Amazon EKS 會建立名為 的安全群組eks-cluster-sg-*my-cluster*-*uniqueID*。安全性群組具有以下預設規則：

規則類型	通訊協定	連接埠	來源	目的地
傳入	全部	全部	自我	
傳出	全部	全部		0.0.0.0/0(IPv4) 或 ：：/0 (IPv6)
傳出	全部	全部		自我（適用於 EFA 流量）

預設安全群組包含傳出規則，允許具有相同安全群組目的地的 Elastic Fabric Adapter (EFA) 流量。這可讓叢集內的 EFA 流量受益於 AI/ML 和高效能運算 (HPC) 工作負載。如需詳細資訊，請參閱《Amazon Elastic Compute Cloud 使用者指南》中的適用於 Amazon EC2 上的 AI/ML 和 HPC 工作負載的 Elastic Fabric Adapter。

> **Important**
>
> 如果您的叢集不需要傳出規則，您可以將其移除。如果將其移除，您仍必須具有限制叢集流量中列出的最低規則。如果您移除傳入規則，則 Amazon EKS 會在叢集更新時重新建立該規則。

Amazon EKS 會將下列標籤新增至安全群組。如果您移除標籤，則 Amazon EKS 會在叢集更新時將其新增回安全群組。

金鑰	值
kubernetes.io/cluster/ *my-cluster*	owned
aws:eks:cluster-name	*my-cluster*
Name	eks-cluster-sg- *my-cluster* -*uniqueid*

Amazon EKS 會自動將此安全群組與以下資源關聯，其資源也會建立：

- 2—4 個彈性網路界面（本文件的其餘部分稱為網路界面），網路界面於叢集建立時建立。
- 您建立的任何受管節點群組中節點的網路界面。

此安全群組的設計目的是允許來自控制平面和 受管節點群組的所有流量彼此之間可以自由流動，以及允許所有的輸出流量流往任何目的地。當您建立叢集時，您可以（選用）指定自己的安全群組。如果您指定自己的安全群組，則 Amazon EKS 還會將您指定的安全群組與為叢集建立的網路界面建立關聯。不過，它不會將它們與您建立的任何節點群組建立關聯。

您可以在叢集的聯網區段 AWS Management Console 下的 中，判斷叢集安全群組的 ID。或者，您可以執行下列 CLI AWS 命令來執行此操作。

```
aws eks describe-cluster --name my-cluster --query
 cluster.resourcesVpcConfig.clusterSecurityGroupId
```

限制叢集流量

如果您需要限制叢集和節點之間的開放連接埠，則可以移除預設傳出規則，並且新增叢集所需的下列最低規則：如果您移除預設傳入規則，Amazon EKS 會在叢集更新時重新建立該規則。

規則類型	通訊協定	連線埠	目的地
傳出	TCP	443	叢集安全群組
傳出	TCP	10250	叢集安全群組
傳出 (DNS)	TCP 和 UDP	53	叢集安全群組

您還必須為以下流量新增規則：

- 您預期節點用於節點間通訊的任何通訊協定和連接埠。
- 需要對外網際網路存取，以便叢集在啟動時存取 Amazon EKS API，進行叢集自我檢查和節點註冊。如果您的節點沒有網際網路存取，請檢閱部署具有有限網際網路存取的私有叢集以取得其他考量。
- 需要虛擬節點存取以從 Amazon ECR 或是從提取映像所需的其他容器登錄檔 API 提取映像，例如 DockerHub。如需詳細資訊，請參閱《 AWS 一般參考》中的 AWS IP 地址範圍。
- 節點對 Amazon S3 進行存取。
- 需要 IPv4 和 IPv6 地址的個別規則。
- 如果您使用的是混合節點，則必須將額外的安全群組新增至叢集，以允許與內部部署節點和 Pod 通訊。如需詳細資訊，請參閱the section called “準備聯網”。

如果您考慮限制規則，建議您先徹底測試所有 Pod，再將已變更的規則套用至生產叢集。

如果您最初使用 Kubernetes 1.14 和 eks.3 平台版本或更舊版本部署叢集，則請考慮下列事項：

- 您可能還擁有控制平面和節點安全群組。建立這些群組時，群組包括之前的表格中列出的受限制的規則。不再需要這些安全群組，可以進行移除。但是，您需要確定您的叢集安全群組包含那些群組包含的規則。

- 如果您直接使用 API 部署叢集，或使用 CLI AWS 或 AWS CloudFormation 等工具來建立叢集，而且未在叢集建立時指定安全群組，則 VPC 的預設安全群組會套用至 Amazon EKS 建立的叢集網路介面。

共用安全群組

Amazon EKS 支援共用安全群組。

- 安全群組 VPC 關聯會將安全群組與相同帳戶和區域中VPCs 建立關聯。
  - 了解如何在 Amazon VPCs 使用者指南中將安全群組與多個 VPC 建立關聯。
- 共用安全群組可讓您與其他 AWS 帳戶共用安全群組。帳戶必須位於相同的 AWS 組織中。
  - 了解如何在 Amazon VPC 使用者指南中與組織共用安全群組。
- 安全群組一律僅限於單一 AWS 區域。

Amazon EKS 的考量事項

- EKS 具有與標準安全群組相同的共用或多 VPC 安全群組需求。

管理 Amazon EKS 叢集的網路附加元件

數個聯網附加元件可用於 Amazon EKS 叢集。

內建附加元件

> **Note**
>
> 當您建立 EKS 叢集時：
>
> - 使用 AWS 主控台：內建附加元件（例如 CoreDNS、kube-proxy 等）會自動安裝為 Amazon EKS 附加元件。您可以透過 AWS 主控台、CLI 或 SDKs 輕鬆設定和更新這些項目。
> - 使用其他方法 (CLI、 SDKs 等）：相同的內建附加元件會安裝為執行為一般 Kubernetes 部署的自我管理版本。這些需要手動組態和更新，因為它們無法透過 AWS 工具進行管理。

建議使用 Amazon EKS 附加元件而非自我管理版本，以簡化附加元件管理，並透過 AWS 服務啟用集中式組態和更新。

適用於 Kubernetes 的 Amazon VPC CNI 外掛程式

此 CNI 附加元件會建立彈性網路介面並將其連接至 Amazon EC2 節點。附加元件也會將私有IPv4或IPv6地址從您的 VPC 指派給每個 Pod 和服務。您的叢集預設會安裝此附加元件。如需詳細資訊，請參閱the section called “Amazon VPC CNI”。如果您使用的是混合節點，則預設仍會安裝 VPC CNI，但無法使用反親和性規則在混合節點上執行。如需混合節點 CNI 選項的詳細資訊，請參閱 the section called “設定 CNI”。

CoreDNS

CoreDNS 是一個靈活、可擴展的 DNS 伺服器，可作為 Kubernetes 集群 DNS。CoreDNS 為叢集中的所有 Pod 提供名稱解析。您的叢集預設會安裝此附加元件。如需詳細資訊，請參閱the section called “CoreDNS”。

kube-proxy

此附加元件會在 Amazon EC2 節點上維護網路規則，並啟用與 Pod 的網路通訊。您的叢集預設會安裝此附加元件。如需詳細資訊，請參閱the section called “kube-proxy”。

選用 AWS 的聯網附加元件

AWS Load Balancer控制器

當您部署類型為 的 Kubernetes 服務物件時loadbalancer，控制器會建立 AWS Network Load Balancer 。當您建立 Kubernetes 輸入物件時，控制器會建立 AWS Application Load Balancer。我們建議您使用此控制器來佈建 Network Load Balancer，而不是使用內建於 Kubernetes 的舊版 Cloud Provider 控制器。如需詳細資訊，請參閱 AWS Load Balancer 控制器說明文件。

AWS Gateway API 控制器

此控制器可讓您使用 Kubernetes 閘道 API 跨多個 Kubernetes 叢集連接服務。控制器使用 Amazon VPC Lattice 服務來連接在 Amazon EC2 執行個體、容器和無伺服器函數上執行的 Kubernetes 服務。 https://docs.aws.amazon.com/vpc-lattice/latest/ug/what-is-vpc-service-network.html如需詳細資訊，請參閱 AWS 閘道 API 控制器文件。

如需附加元件的詳細資訊，請參閱 the section called “Amazon EKS 附加元件”。

使用 Amazon VPC CNI 將 IPs 指派給 Pod

> **Tip**
>
> 使用 Amazon EKS Auto Mode，您不需要安裝或升級聯網附加元件。自動模式包含 Pod 聯網和負載平衡功能。
> 如需詳細資訊，請參閱EKS 自動模式。

適用於 Kubernetes 的 Amazon VPC CNI 外掛程式會部署在 Amazon EKS 叢集中的每個 Amazon EC2 節點上。附加元件會建立彈性網路界面並將其連接到 Amazon EC2 節點。附加元件也會將私有IPv4或IPv6地址從您的 VPC 指派給每個 Pod。

附加元件的版本會隨叢集中的每個 Fargate 節點一起部署，但您不會在 Fargate 節點上更新。其他相容的 CNI 外掛程式可用於 Amazon EKS 叢集，但這是 Amazon EKS 針對在 AWS 基礎設施上執行的節點所支援的唯一 CNI 外掛程式。如需其他相容 CNI 外掛程式的詳細資訊，請參閱 the section called “替代 CNI 外掛程式”。VPC CNI 不支援與混合節點搭配使用。如需混合節點 CNI 選項的詳細資訊，請參閱 the section called “設定 CNI”。

下表列出每個 Kubernetes 版本的最新可用 Amazon EKS 附加元件類型版本。

Amazon VPC CNI 版本

Kubernetes 版本	Amazon EKS 類型的 VPC CNI 版本
1.33	v1.20.0-eksbuild.1
1.32	v1.20.0-eksbuild.1
1.31	v1.20.0-eksbuild.1
1.30	v1.20.0-eksbuild.1
1.29	v1.20.0-eksbuild.1
1.28	v1.20.0-eksbuild.1
1.27	v1.20.0-eksbuild.1

Kubernetes 版本	Amazon EKS 類型的 VPC CNI 版本
1.26	v1.19.6-eksbuild.1

> **Important**
>
> 如果您要自我管理此附加元件，資料表中的版本可能與可用的自我管理版本不同。如需更新此附加元件之自我管理類型的詳細資訊，請參閱 the section called “更新 （ 自我管理 ）”。

> **Important**
>
> 若要升級至 VPC CNI v1.12.0 或更新版本，您必須先升級至 VPC CNI v1.7.0。我們建議您一次更新一個次要版本。

考量事項

以下是使用 功能的考量。

- 版本指定為 `major-version.minor-version.patch-version-eksbuild.build-number`。
- 檢查每個功能的版本相容性。適用於 Kubernetes 的 Amazon VPC CNI 外掛程式每個版本的某些功能都需要特定 Kubernetes 版本。使用不同的 Amazon EKS 功能時，如果需要特定版本的附加元件，則會在功能文件中註明。除非您有執行舊版本的特定理由，否則建議您執行最新版本。

建立 Amazon VPC CNI (Amazon EKS 附加元件)

使用下列步驟建立適用於 Kubernetes EKS 附加元件的 Amazon VPC CNI 外掛程式。

開始之前，請檢閱考量事項。如需詳細資訊，請參閱the section called “考量事項”。

先決條件

以下是適用於 Kubernetes EKS 附加元件的 Amazon VPC CNI 外掛程式的先決條件。

- 現有 Amazon EKS 叢集。若要部署叢集，請參閱 開始使用。

- 叢集的現有 AWS Identity and Access Management (IAM) OpenID Connect (OIDC) 提供者。若要判定您是否已經擁有一個，或是要建立一個，請參閱 the section called “IAM OIDC 供應商”。
- 連接 AmazonEKS_CNI_Policy IAM 政策（如果您的叢集使用 IPv4 系列）或 IPv6 政策（如果您的叢集使用 IPv6 系列）的 IAM 角色。如需 VPC CNI 角色的詳細資訊，請參閱 the section called “為 IRSA 設定”。如需 IPv6 政策的相關資訊，請參閱 the section called “為使用 IPv6 系列的叢集建立 IAM 政策”。

> **Important**
>
> 適用於 Kubernetes 版本的 Amazon VPC CNI 外掛程式`v1.16.0`，以`v1.16.1`實作 CNI 規格版本 `v1.0.0`。如需 CNI `v1.0.0` 規格的詳細資訊，請參閱 GitHub 上的容器網路界面 (CNI) 規格。

程序

完成先決條件後，請使用下列步驟來建立附加元件。

1. 查看叢集上目前安裝了哪些附加元件版本。

```
kubectl describe daemonset aws-node --namespace kube-system | grep amazon-k8s-cni: |
 cut -d : -f 3
```

範例輸出如下。

```
v1.16.4-eksbuild.2
```

2. 查看叢集上安裝的附加元件類型。視您用來建立叢集的工具而定，您的叢集上目前可能沒有安裝 Amazon EKS 附加元件類型。使用您叢集的名稱取代 *my-cluster*。

```
aws eks describe-addon --cluster-name my-cluster --addon-name vpc-cni --query
 addon.addonVersion --output text
```

如果傳回版本號碼，您的叢集上安裝了 Amazon EKS 類型的附加元件，不需要完成此程序中的其餘步驟。如果傳回錯誤，表示叢集上未安裝 Amazon EKS 類型的附加元件。完成此程序的剩餘步驟以安裝該類型。

3. 儲存您目前安裝的附加元件。

```
kubectl get daemonset aws-node -n kube-system -o yaml > aws-k8s-cni-old.yaml
```

4. 使用 CLI AWS 建立附加元件。如果您想要使用 AWS Management Console 或 `eksctl` 建立附加元件，請參閱 the section called “建立附加元件”並`vpc-cni`指定附加元件名稱。將隨後的命令複製到您的裝置。視需要對命令進行下列修改，然後執行修改後的命令。
   - 使用您叢集的名稱取代 *`my-cluster`*。
   - 將 *`v1.20.0-eksbuild.1`* 取代為叢集版本最新版本資料表中列出的最新版本。如需最新版本的資料表，請參閱 the section called “Amazon VPC CNI 版本”。
   - 將 *`111122223333`* 取代為您的帳戶 ID，並將 *`AmazonEKSVPCCNIRole`* 取代為您已建立的現有 IAM 角色名稱。指定角色需要您擁有叢集的 IAM OpenID Connect (OIDC) 提供者。若要判定您的叢集是否已經擁有一個提供者，或是要建立一個提供者，請參閱 the section called “IAM OIDC 供應商”。

   ```
   aws eks create-addon --cluster-name my-cluster --addon-name vpc-cni --addon-version
    v1.20.0-eksbuild.1 \
       --service-account-role-arn arn:aws: iam::111122223333:role/AmazonEKSVPCCNIRole
   ```

   如果您已將自訂設定套用至與 Amazon EKS 附加元件預設設定衝突的目前附加元件，則建立可能會失敗。若建立失敗，您會收到錯誤，其中的訊息有助於您解決問題。或者，您可以將 `--resolve-conflicts OVERWRITE` 新增至上一條命令。這可讓附加元件覆寫任何現有的自訂設定。建立附加元件之後，您可以使用自訂設定進行更新。
5. 確認叢集 Kubernetes 版本的最新版本附加元件已新增至叢集。使用您叢集的名稱取代 *`my-cluster`*。

   ```
   aws eks describe-addon --cluster-name my-cluster --addon-name vpc-cni --query
    addon.addonVersion --output text
   ```

   建立附加元件的動作可能需要幾秒鐘的時間才能完成。

   範例輸出如下。

   ```
   v1.20.0-eksbuild.1
   ```
6. 如果您已對原始附加元件進行自訂設定，則在建立 Amazon EKS 附加元件之前，請使用您在上一個步驟中儲存的組態，以自訂設定更新 EKS 附加元件。請遵循 the section called “更新 (EKS 附加元件) ” 中的步驟。

7. (選用) 安裝 `cni-metrics-helper` 到您的叢集。它湊集彈性網路介面和 IP 地址資訊，匯總叢集層級的指標，並將指標發佈至 Amazon CloudWatch。如需詳細資訊，請參閱 GitHub 上的 cni-metrics-helper。

更新 Amazon VPC CNI (Amazon EKS 附加元件)

更新 Kubernetes 附加元件的 Amazon VPC CNI 外掛程式的 Amazon EKS 類型。如果您尚未將 附加元件的 Amazon EKS 類型新增至叢集，您可以依照 進行安裝the section called “建立”。或者，遵循 更新其他類型的 VPC CNI 安裝the section called “更新 （ 自我管理 ）”。

1. 查看叢集上目前安裝了哪些附加元件版本。使用您的叢集名稱取代 *my-cluster*。

```
aws eks describe-addon --cluster-name my-cluster --addon-name vpc-cni --query
 "addon.addonVersion" --output text
```

範例輸出如下。

```
v1.20.0-eksbuild.1
```

將版本與 中最新版本的資料表進行比較the section called “Amazon VPC CNI 版本”。如果傳回的版本與最新版本資料表中叢集 Kubernetes 版本的版本相同，則您已在叢集上安裝最新版本，不需要完成此程序的其餘部分。如果您收到錯誤，而不是輸出中的版本號碼，則表示叢集上沒有安裝 Amazon EKS 類型的附加元件。您必須先建立附加元件，才能使用此程序進行更新。若要建立 VPC CNI 附加元件的 Amazon EKS 類型，您可以遵循 the section called “建立”。

2. 儲存您目前安裝的附加元件。

```
kubectl get daemonset aws-node -n kube-system -o yaml > aws-k8s-cni-old.yaml
```

3. 使用 CLI AWS 更新您的附加元件。如果您想要使用 AWS Management Console 或 `eksctl` 來更新附加元件，請參閱 the section called “更新 附加元件”。將隨後的命令複製到您的裝置。視需要對命令進行下列修改，然後執行修改後的命令。
   - 使用您叢集的名稱取代 *my-cluster*。
   - 將 *v1.20.0-eksbuild.1* 取代為叢集版本最新版本資料表中列出的最新版本。
   - 將 *111122223333* 取代為您的帳戶 ID，並將 *AmazonEKSVPCCNIRole* 取代為您已建立的現有 IAM 角色名稱。若要為 VPC CNI 建立 IAM 角色，請參閱 the section called “步驟 1：建立 Kubernetes IAM 角色的 Amazon VPC CNI 外掛程式”。指定角色需要您擁有叢集的 IAM OpenID

Connect (OIDC) 提供者。若要判定您的叢集是否已經擁有一個提供者，或是要建立一個提供者，請參閱 the section called “IAM OIDC 供應商”。

- `--resolve-conflicts PRESERVE` 選項會保留附加元件的現有組態值。如果您已設定附加元件設定的自訂值，但未使用此選項，Amazon EKS 會以其預設值覆寫您的值。如果您使用此選項，建議您在更新生產叢集上的附加元件之前，測試非生產叢集上的任何欄位和值變更。如果您將此值變更為 OVERWRITE，則所有設定都會變更為 Amazon EKS 預設值。如果您已為任何設定設定自訂值，這些值可能會以 Amazon EKS 預設值覆寫。如果您將此值變更為 none，Amazon EKS 不會變更任何設定的值，但更新可能會失敗。若更新失敗，您會收到錯誤訊息，以協助您解決衝突。
- 如果您未更新組態設定，`--configuration-values '{`*`"env":{"AWS_VPC_K8S_CNI_EXTERNALSNAT":"true"}`*`}'`請從命令中移除 。如果您要更新組態設定，請將 *`"env"#{"AWS_VPC_K8S_CNI_EXTERNALSNAT"#"true"}`* 取代為您要設定的設定。在此範例中，AWS_VPC_K8S_CNI_EXTERNALSNAT 環境變數設定為 `true`。您指定的值必須對組態結構描述有效。如果您不知道組態結構描述，請執行 `aws eks describe-addon-configuration --addon-name vpc-cni --addon-version` *`v1.20.0-eksbuild.1`* ，將 *`v1.20.0-eksbuild.1`* 取代為您要查看組態的附加元件版本編號。結構描述會在輸出中傳回。如果您有任何現有的自訂組態，並且想要全部移除，同時將所有設定的值設定回 Amazon EKS 預設值，請從命令中移除 *`"env":{"AWS_VPC_K8S_CNI_EXTERNALSNAT":"true"}`* ，這樣就會得到空白的 {}。如需每個設定的說明，請參閱 GitHub 上的 CNI 組態變數。

```
aws eks update-addon --cluster-name my-cluster --addon-name vpc-cni --addon-version
 v1.20.0-eksbuild.1 \
    --service-account-role-arn arn:aws: iam::111122223333:role/AmazonEKSVPCCNIRole
 \
    --resolve-conflicts PRESERVE --configuration-values '{"env":
{"AWS_VPC_K8S_CNI_EXTERNALSNAT":"true"}}'
```

更新動作可能需要幾秒鐘的時間才能完成。

4. 確認附加元件版本已更新。使用您叢集的名稱取代 *`my-cluster`*。

```
aws eks describe-addon --cluster-name my-cluster --addon-name vpc-cni
```

更新動作可能需要幾秒鐘的時間才能完成。

範例輸出如下。

```
{
```

```
    "addon": {
        "addonName": "vpc-cni",
        "clusterName": "my-cluster",
        "status": "ACTIVE",
        "addonVersion": "v1.20.0-eksbuild.1",
        "health": {
            "issues": []
        },
        "addonArn": "arn:aws: eks:region:111122223333:addon/my-cluster/vpc-
cni/74c33d2f-b4dc-8718-56e7-9fdfa65d14a9",
        "createdAt": "2023-04-12T18:25:19.319000+00:00",
        "modifiedAt": "2023-04-12T18:40:28.683000+00:00",
        "serviceAccountRoleArn": "arn:aws: iam::111122223333:role/
AmazonEKSVPCCNIRole",
        "tags": {},
        "configurationValues": "{\"env\":{\"AWS_VPC_K8S_CNI_EXTERNALSNAT\":\"true
\"}}"
    }
}
```

更新 Amazon VPC CNI （自我管理附加元件）

> **Important**
>
> 建議將附加元件的 Amazon EKS 類型新增到叢集，而不是使用附加元件的自我管理類型。如果您不熟悉類型之間的差異，請參閱 the section called “Amazon EKS 附加元件”。如需將 Amazon EKS 附加元件新增至叢集的詳細資訊，請參閱 the section called “建立附加元件”。如果您無法使用 Amazon EKS 附加元件，建議您提交問題，說明為何無法前往容器藍圖 GitHub 儲存庫。

1. 確認您叢集上未安裝 Amazon EKS 類型的附加元件。使用您叢集的名稱取代 *my-cluster*。

```
aws eks describe-addon --cluster-name my-cluster --addon-name vpc-cni --query
 addon.addonVersion --output text
```

如果傳回錯誤訊息，表示叢集上未安裝 Amazon EKS 類型的附加元件。若要自行管理附加元件，請完成此程序中的剩餘步驟來更新附加元件。如果傳回版本編號，則表明已在叢集上安裝附加元件的 Amazon EKS 類型。若要將其更新，請使用 the section called “更新 附加元件” 中的程序，而不是使

用此程序。如果您不熟悉附加元件類型之間的差異，請參閱 the section called “Amazon EKS 附加元件”。

2. 查看叢集上目前安裝了哪些容器映像版本。

```
kubectl describe daemonset aws-node --namespace kube-system | grep amazon-k8s-cni: |
 cut -d : -f 3
```

範例輸出如下。

```
v1.20.0-eksbuild.1
```

您的輸出可能不包含建置編號。

3. 備份目前的設定，以便在更新版本後設定相同的設定。

```
kubectl get daemonset aws-node -n kube-system -o yaml > aws-k8s-cni-old.yaml
```

若要檢閱可用的版本，並熟悉您要更新之版本中的變更，請參閱 GitHub 上的版本。請注意，我們建議更新至最新可用版本資料表中列出的相同 `major`.`minor`.`patch` 版本，即使 GitHub 上提供了更新版本。如需最新的可用版本資料表，請參閱 the section called “Amazon VPC CNI 版本”。資料表中列出的建置版本未在 GitHub 上列出的自我管理版本中指定。透過以下列其中一種選項完成任務來更新您的版本：

- 如果您沒有任何附加元件的自訂設定，請在 GitHub 的`To apply this release:`標題下執行命令，以取得您要更新的版本。
- 如果有自訂設定，則請使用以下命令下載清單檔案。將 *https://raw.githubusercontent.com/aws/amazon-vpc-cni-k8s/v1.20.0/config/master/aws-k8s-cni.yaml* 變更為您要更新之 GitHub 發行版本的 URL。

  ```
  curl -O https://raw.githubusercontent.com/aws/amazon-vpc-cni-k8s/v1.20.0/config/
  master/aws-k8s-cni.yaml
  ```

  如有必要，請使用您在前一步所建立備份中的自訂設定修改清單檔案，然後將修改後的清單檔案套用至叢集。如果您的節點無法存取提取映像的私有 Amazon EKS Amazon ECR 儲存庫（請參閱資訊清單中開頭為 `image:`的行），則您必須下載映像、將其複製到您自己的儲存庫，然後修改資訊清單以從儲存庫提取映像。如需詳細資訊，請參閱the section called “將映像複製到儲存庫”。

```
kubectl apply -f aws-k8s-cni.yaml
```

4. 確認您的叢集上現在已安裝新版本。

```
kubectl describe daemonset aws-node --namespace kube-system | grep amazon-k8s-cni: |
 cut -d : -f 3
```

範例輸出如下。

```
v1.20.0
```

5. (選用) 安裝 `cni-metrics-helper` 到您的叢集。它湊集彈性網路介面和 IP 地址資訊，匯總叢集層級的指標，並將指標發佈至 Amazon CloudWatch。如需詳細資訊，請參閱 GitHub 上的 cni-metrics-helper。

設定 Amazon VPC CNI 外掛程式以使用 IRSA

適用於 Kubernetes 的 Amazon VPC CNI 外掛程式是 Amazon EKS 叢集中 Pod 聯網的聯網外掛程式。外掛程式負責將 VPC IP 地址配置給 Kubernetes Pod，並為每個節點上的 Pod 設定必要的聯網。

> **Note**
>
> Amazon VPC CNI 外掛程式也支援 Amazon EKS Pod 身分。如需詳細資訊，請參閱the section called “指派 IAM 角色”。

外掛程式：

- 需要 AWS Identity and Access Management (IAM) 許可。如果您的叢集使用 `IPv4` 系列，則會在 `AmazonEKS_CNI_Policy` AWS 受管政策中指定許可。如果您的叢集使用 `IPv6` 系列，則必須將許可新增至您建立的 IAM 政策；如需說明，請參閱 the section called “為使用 IPv6 系列的叢集建立 IAM 政策”。您可附加政策至 Amazon EKS 節點 IAM 角色，或附加至單獨 IAM 角色。如需將政策連接至 Amazon EKS 節點 IAM 角色的說明，請參閱 the section called “節點 IAM 角色”。建議如本主題所述，將其指派給不同的角色。
- 建立 並設定為在部署`aws-node`時使用名為 的 Kubernetes 服務帳戶。服務帳戶綁定到一個名為 `aws-node` 的 Kubernetes `clusterrole`，它被指派了所需的 Kubernetes 許可。

> **Note**
>
> 適用於 Kubernetes 的 Amazon VPC CNI 外掛程式的 Pod 可存取指派給 Amazon EKS 節點 IAM 角色的許可，除非您封鎖對 IMDS 的存取。如需詳細資訊，請參閱限制存取指派給工作節點的執行個體設定檔。

- 需要現有的 Amazon EKS 叢集。若要部署叢集，請參閱 開始使用。
- 需要您叢集的現有 AWS Identity and Access Management (IAM) OpenID Connect (OIDC) 供應商。若要判定您是否已經擁有一個，或是要建立一個，請參閱 the section called “IAM OIDC 供應商”。

步驟 1：建立 Kubernetes IAM 角色的 Amazon VPC CNI 外掛程式

1. 確定您叢集的 IP 系列。

   ```
   aws eks describe-cluster --name my-cluster | grep ipFamily
   ```

   範例輸出如下。

   ```
   "ipFamily": "ipv4"
   ```

   輸出可能會傳回 `ipv6`。

2. 建立 IAM 角色。您可以使用 `eksctl`或 `kubectl`和 AWS CLI 來建立 IAM 角色。

   eksctl

   - 建立 IAM 角色，並使用與您叢集 IP 系列相符的命令將 IAM 政策連接至此角色。命令會建立和部署建立 IAM 角色的 AWS CloudFormation 堆疊、連接您指定的政策，並使用所建立 aws-node IAM 角色的 ARN 來註釋現有的 Kubernetes 服務帳戶。

     - `IPv4`

       用您的值取代*`my-cluster`*。

       ```
       eksctl create iamserviceaccount \
           --name aws-node \
           --namespace kube-system \
           --cluster my-cluster \
           --role-name AmazonEKSVPCCNIRole \
           --attach-policy-arn arn:aws: iam::aws:policy/AmazonEKS_CNI_Policy \
       ```

```
    --override-existing-serviceaccounts \
    --approve
```

- IPv6

  用您的值取代*my-cluster*。將 *111122223333* 取代為您的帳戶 ID，以及將 *AmazonEKS_CNI_IPv6_Policy* 取代為您的 IPv6 策略名稱。如果您沒有 IPv6政策，請參閱 the section called “為使用 IPv6 系列的叢集建立 IAM 政策” 來建立政策。若要將 IPv6 與您的叢集一起使用，其必須滿足幾個要求。如需詳細資訊，請參閱the section called “IPv6”。

  ```
  eksctl create iamserviceaccount \
      --name aws-node \
      --namespace kube-system \
      --cluster my-cluster \
      --role-name AmazonEKSVPCCNIRole \
      --attach-policy-arn arn:aws: iam::111122223333:policy/
  AmazonEKS_CNI_IPv6_Policy \
      --override-existing-serviceaccounts \
      --approve
  ```

kubectl 和 AWS CLI

i. 檢視叢集的 OIDC 提供者 URL。

   ```
   aws eks describe-cluster --name my-cluster --query
    "cluster.identity.oidc.issuer" --output text
   ```

   範例輸出如下。

   ```
   https://oidc.eks.region-code.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE
   ```

   如果未傳回任何輸出，則您必須為叢集建立 IAM OIDC 提供商。

ii. 將下列內容複製到名為 *vpc-cni-trust-policy.json* 的檔案。將 *111122223333* 取代為您的帳戶 ID，並將 *EXAMPLED539D4633E53DE1B71EXAMPLE* 取代為上一個步驟中傳回的輸出。將 *region-code* 取代為您的叢集所在的 AWS 區域。

   ```
   {
       "Version": "2012-10-17",
       "Statement": [
           {
   ```

```
            "Effect": "Allow",
            "Principal": {
                "Federated": "arn:aws: iam::111122223333:oidc-provider/
oidc.eks.region-code.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE"
            },
            "Action": "sts:AssumeRoleWithWebIdentity",
            "Condition": {
                "StringEquals": {
                    "oidc.eks.region-code.amazonaws.com/id/
EXAMPLED539D4633E53DE1B71EXAMPLE:aud": "sts.amazonaws.com",
                    "oidc.eks.region-code.amazonaws.com/id/
EXAMPLED539D4633E53DE1B71EXAMPLE:sub": "system:serviceaccount:kube-system:aws-
node"
                }
            }
        }
    ]
}
```

iii. 建立角色。您可以將 *AmazonEKSVPCCNIRole* 取代為您選擇的任何名稱。

```
aws iam create-role \
  --role-name AmazonEKSVPCCNIRole \
  --assume-role-policy-document file://"vpc-cni-trust-policy.json"
```

iv. 將所需的 IAM 政策連接至角色。執行與叢集的 IP 系列相符的命令。

- IPv4

  ```
  aws iam attach-role-policy \
    --policy-arn arn:aws: iam::aws:policy/AmazonEKS_CNI_Policy \
    --role-name AmazonEKSVPCCNIRole
  ```

- IPv6

  將 *111122223333* 取代為您的帳戶 ID，以及將 *AmazonEKS_CNI_IPv6_Policy* 取代為您的 IPv6 策略名稱。如果您沒有 IPv6政策，請參閱 the section called “為使用 IPv6 系列的叢集建立 IAM 政策” 來建立政策。若要將 IPv6 與您的叢集一起使用，其必須滿足幾個要求。如需詳細資訊，請參閱the section called “IPv6”。

  ```
  aws iam attach-role-policy \
    --policy-arn arn:aws: iam::111122223333:policy/AmazonEKS_CNI_IPv6_Policy \
    --role-name AmazonEKSVPCCNIRole
  ```

v. 執行下列命令以使用您之前建立之 IAM 角色的 ARN 來標註 aws-node 服務帳戶。以您自己的值取代*###*。

```
kubectl annotate serviceaccount \
    -n kube-system aws-node \
    eks.amazonaws.com/role-arn=arn:aws: iam::111122223333:role/
AmazonEKSVPCCNIRole
```

3. （選用）設定 Kubernetes 服務帳戶所使用的 AWS Security Token Service 端點類型。如需詳細資訊，請參閱the section called “STS 端點”。

步驟 2：重新部署 Kubernetes Pod 的 Amazon VPC CNI 外掛程式

1. 刪除並重新建立與服務帳戶相關聯的任何現有 Pod，以套用登入資料環境變數。註釋不會套用至目前在沒有註釋的情況下執行的 Pod。下列命令會刪除現有的 aws-node DaemonSet Pod，並使用服務帳戶註釋進行部署。

```
kubectl delete Pods -n kube-system -l k8s-app=aws-node
```

2. 確認 Pod 全部重新啟動。

```
kubectl get pods -n kube-system -l k8s-app=aws-node
```

3. 描述其中一個 Pod，並確認 AWS_WEB_IDENTITY_TOKEN_FILE和 AWS_ROLE_ARN環境變數是否存在。將 *cpjw7* 取代為上一個步驟輸出中傳回的其中一個 Pod 名稱。

```
kubectl describe pod -n kube-system aws-node-cpjw7 | grep 'AWS_ROLE_ARN:\|
AWS_WEB_IDENTITY_TOKEN_FILE:'
```

範例輸出如下。

```
AWS_ROLE_ARN: arn:aws: iam::111122223333:role/AmazonEKSVPCCNIRole
      AWS_WEB_IDENTITY_TOKEN_FILE:  /var/run/secrets/eks.amazonaws.com/
serviceaccount/token
      AWS_ROLE_ARN: arn:aws: iam::111122223333:role/AmazonEKSVPCCNIRole
      AWS_WEB_IDENTITY_TOKEN_FILE:            /var/run/secrets/eks.amazonaws.com/
serviceaccount/token
```

會傳回兩組重複結果，因為 Pod 包含兩個容器。兩個容器具有相同的值。

如果您的 Pod 使用 AWS 區域端點，則上一個輸出也會傳回以下行。

```
AWS_STS_REGIONAL_ENDPOINTS=regional
```

步驟 3：從節點 IAM 角色移除 CNI 政策

如果您的 Amazon EKS 節點 IAM 角色目前已連接 AmazonEKS_CNI_Policy IAM (IPv4) 政策或 IPv6 政策，且您已建立個別的 IAM 角色、改為連接政策，並將其指派給 Kubernetes aws-node 服務帳戶，則建議您使用符合叢集 IP 系列的 AWS CLI 命令，從節點角色中移除政策。將 *AmazonEKSNodeRole* 取代為您的節點角色名稱。

- IPv4

```
aws iam detach-role-policy --role-name AmazonEKSNodeRole --policy-arn arn:aws:
 iam::aws:policy/AmazonEKS_CNI_Policy
```

- IPv6

  將 *111122223333* 取代為您的帳戶 ID，以及將 *AmazonEKS_CNI_IPv6_Policy* 取代為您的 IPv6 策略名稱。

```
aws iam detach-role-policy --role-name AmazonEKSNodeRole --policy-arn arn:aws:
 iam::111122223333:policy/AmazonEKS_CNI_IPv6_Policy
```

為使用 **IPv6** 系列的叢集建立 IAM 政策

如果您建立的叢集使用 IPv6 系列，且叢集已設定適用於 Kubernetes 附加元件的 Amazon VPC CNI 外掛程式版本 1.10.1或更新版本，則您需要建立可指派給 IAM 角色的 IAM 政策。如果您現有的叢集在建立時並未使用 IPv6 系列進行設定，則若要使用 IPv6，您必須建立新的叢集。如需搭配使用 IPv6 與叢集的詳細資訊，請參閱 the section called “IPv6”。

1. 複製下列文字並將它儲存至名為 vpc-cni-ipv6-policy.json 的檔案。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
```

```
            "Action": [
                "ec2:AssignIpv6Addresses",
                "ec2:DescribeInstances",
                "ec2:DescribeTags",
                "ec2:DescribeNetworkInterfaces",
                "ec2:DescribeInstanceTypes"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:CreateTags"
            ],
            "Resource": [
                "arn:aws: ec2:*:*:network-interface/*"
            ]
        }
    ]
}
```

2. 建立 IAM 政策。

```
aws iam create-policy --policy-name AmazonEKS_CNI_IPv6_Policy --policy-document
 file://vpc-cni-ipv6-policy.json
```

了解 VPC CNI 模式和組態

適用於 Kubernetes 的 Amazon VPC CNI 外掛程式提供 Pod 的聯網。使用下表進一步了解可用的聯網功能。

網路功能	進一步了解
設定您的叢集，將 IPv6 地址指派給叢集、Pod 和服務	the section called “IPv6”
使用 Pod 的 IPv4 來源網路地址轉譯	the section called “傳出流量”
限制往返 Pod 的網路流量	the section called “限制流量”
自訂節點中的次要網路介面	the section called “自訂聯網”

網路功能	進一步了解
增加節點的 IP 地址	the section called “增加 IP 地址”
針對 Pod 網路流量使用安全群組	the section called “Pod 的安全群組”
針對 Pod 使用多個網路介面	the section called “多個界面”

了解叢集、Pod 和 服務的 IPv6 地址

適用於：具有 Amazon EC2 執行個體和 Fargate Pod 的 Pod

根據預設，Kubernetes 會將IPv4地址指派給您的 Pod 和服務。您可以設定叢集將IPv4地址指派給 Pod 和服務，而不是將IPv6地址指派給 Pod 和服務。即使 Kubernetes 也不支援雙堆疊 Pod 或服務。因此，您無法同時將 IPv4和 IPv6地址指派給您的 Pod 和服務。

您可以在建立叢集時選擇要用於此叢集的 IP 系列。您無法在建立叢集後變更系列。

如需部署 Amazon EKS IPv6叢集的教學課程，請參閱 the section called “部署”。

以下是使用 功能的考量：

IPv6 功能支援

- 不支援 Windows：不支援 Windows Pod 和服務。
- 需要 Nitro 型 EC2 節點：您只能IPv6搭配 AWS Nitro 型 Amazon EC2 或 Fargate 節點使用 。
- 支援的 EC2 和 Fargate 節點：您可以將 IPv6 the section called “Pod 的安全群組”搭配 Amazon EC2 節點和 Fargate 節點使用。
- 不支援 Outpost：您無法IPv6搭配 使用 AWS Outposts 上的 Amazon EKS。
- 不支援 FSx for Lustre：the section called “Amazon FSx for Lustre”不支援 。
- 不支援自訂聯網：如果您之前使用 the section called “自訂聯網” 協助緩解 IP 地址耗盡，則可以IPv6改用 。您無法搭配 使用自訂聯網IPv6。如果您使用自訂聯絡進行網路隔離，則可能需要繼續為叢集使用自訂聯網和 IPv4 系列。

IP 地址指派

- Kubernetes 服務：Kubernetes 服務只會指派IPv6地址。它們不會指派 IPv4 地址。

- Pod：Pod 會獲指派 IPv6 地址和主機本機 IPv4 地址。主機本機 IPv4 地址是透過使用與 VPC CNI 鏈結的主機本機 CNI 外掛程式來指派，且地址不會報告給 Kubernetes 控制平面。只有在 Pod 需要與其他 Amazon VPC 或網際網路中的外部 IPv4 資源通訊時，才會使用此資源。主機本機 IPv4 地址會 SNATed（透過 VPC CNI) 至工作者節點之主要 ENI 的主要 IPv4 地址。
- Pod 和服務：Pod 和服務只會接收IPv6地址，而不會接收IPv4地址。當 Pod 需要與外部IPv4端點通訊時，它們會在節點本身上使用 NAT。此內建 NAT 功能不需要 DNS64 和 NAT64。對於需要公有網際網路存取的流量，Pod 的流量是轉換為公有 IP 地址的來源網路地址。
- 路由地址：當 Pod 在 VPC 外部通訊時，會保留其原始IPv6地址（不會轉譯為節點IPv6的地址）。此流量會直接透過網際網路閘道或僅輸出網際網路閘道路由。
- 節點：所有節點都會指派 IPv4和 IPv6 地址。
- Fargate Pod：每個 Fargate Pod 都會從 CIDR 接收其部署所在子網路指定的IPv6地址。執行 Fargate Pods 的基礎硬體單位會從指派給部署硬體單位之子網路CIDRs 取得唯一的 IPv4和IPv6地址。

如何**IPv6**搭配 EKS 使用

- 建立新的叢集：您必須建立新的叢集，並指定您要使用該叢集的 IPv6 系列。您無法為您從先前版本更新的叢集啟用 IPv6 系列。如需如何建立新叢集的指示，請參閱考量事項 。
- 使用最新的 VPC CNI：部署 Amazon VPC CNI 版本 1.10.1或更新版本。依預設，將部署此版本或更新版本。部署附加元件之後，您無法先移除叢集中所有節點群組中的所有節點，1.10.1再將 Amazon VPC CNI 附加元件降級至低於 的版本。
- 設定 **IPv6** 的 VPC CNI：如果您使用 Amazon EC2 節點，則必須使用 IP 字首委派和 設定 Amazon VPC CNI 附加元件IPv6。如果您在建立叢集時選擇 IPv6 系列，則附加元件的 1.10.1 版本預設為此設定。自我管理或 Amazon EKS 附加元件都是這種情況。如需 IP 字首委派的詳細資訊，請參閱 the section called “增加 IP 地址”。
- 設定 **IPv4** 和 **IPv6** 地址：當您建立叢集時，您指定的 VPC 和子網路必須具有指派給您指定之 VPC 和子網路的 IPv6 CIDR 區塊。此外，還必須為其指派 IPv4 CIDR 區塊。這是因為，即使您只想使用 IPv6，VPC 仍然需要 IPv4 CIDR 區塊才能正常工作。如需詳細資訊，請參閱 Amazon VPC 使用者指南中的建立 IPv6 CIDR 區塊與 VPC 的關聯。
- 自動指派 IPv6 地址至節點：建立節點時，您必須指定設定為自動指派IPv6地址的子網路。否則，您無法部署節點。根據預設，會停用此設定。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 定址屬性。
- 將路由表設定為使用 **IPv6** ：指派給子網路的路由表必須具有IPv6地址的路由。如需詳細資訊，請參閱 Amazon VPC 使用者指南中的遷移至 IPv6。

- 設定 **IPv6** 的安全群組：您的安全群組必須允許IPv6地址。如需詳細資訊，請參閱 Amazon VPC 使用者指南中的遷移至 IPv6。
- 設定負載平衡器：使用版本 2.3.1或更新版本的 AWS Load Balancer控制器，在 IP 模式下使用 the section called “應用程式負載平衡”或 網路流量將 HTTP 應用程式負載平衡the section called “網路負載平衡”至具有任一負載平衡器的 IPv6 Pod，但不使用執行個體模式。如需詳細資訊，請參閱the section called “ AWS Load Balancer控制器”。
- 新增 **IPv6** IAM 政策：您必須將 IPv6 IAM 政策連接至節點 IAM 或 CNI IAM 角色。在兩者之間，建議您將其連接至 CNI IAM 角色。如需詳細資訊，請參閱the section called “為使用 IPv6 系列的叢集建立 IAM 政策”及the section called “步驟 1：建立 Kubernetes IAM 角色的 Amazon VPC CNI 外掛程式”。
- 評估所有元件：在部署IPv6叢集之前，執行與您整合的應用程式、Amazon EKS 附加元件 AWS 和服務的完整評估。這是為了確保使用 IPv6 時一切都能正常工作。
- 新增**BootstrapArguments**自我管理節點群組：在使用 IPv6 系列的叢集中建立自我管理節點群組時，使用者資料必須BootstrapArguments針對節點啟動時執行的 bootstrap.sh 檔案包含下列項目。將 *your-cidr* 取代為叢集 VPC 的 IPv6 CIDR 範圍。

```
--ip-family ipv6 --service-ipv6-cidr your-cidr
```

如果您不知道叢集IPv6CIDR的範圍，可以使用下列命令來查看它（需要 AWS CLI 版本 2.4.9或更新版本）。

```
aws eks describe-cluster --name my-cluster --query
 cluster.kubernetesNetworkConfig.serviceIpv6Cidr --output text
```

部署 Amazon EKS **IPv6**叢集和受管 Amazon Linux 節點

在本教學課程中，您將部署 IPv6 Amazon VPC、含 IPv6 系列的 Amazon EKS 叢集，以及含有 Amazon EC2 Amazon Linux 節點的受管節點群組。您無法在IPv6叢集中部署 Amazon EC2 Windows 節點。您也可以將 Fargate 節點部署到您的叢集，但為了簡化起見，本主題並未提供這些指示。

先決條件

開始教學課程之前，請先完成下列各項：

安裝並設定建立和管理 Amazon EKS 叢集所需的下列工具和資源。

- 我們建議您熟悉所有設定，並使用符合您需求的設定部署叢集。如需詳細資訊，請參閱 the section called “建立叢集”、 the section called “受管節點群組”和本主題的考量事項。您只可在建立叢集時啟用某些設定。
- kubectl 命令列工具安裝在您的裝置或 AWS CloudShell 上。版本可以與叢集的 Kubernetes 版本相同，也可以比叢集的 Kubernetes 版本更早或更晚一個次要版本。例如，如果您的叢集版本為 1.29，則可以搭配使用 kubectl 1.28、1.29 或 1.30 版。若要安裝或升級 kubectl，請參閱 the section called “設定 kubectl 和 eksctl”。
- 您使用的 IAM 安全主體必須具有使用 Amazon EKS IAM 角色、服務連結角色、 AWS CloudFormation、VPC 和相關資源的許可。如需詳細資訊，請參閱《IAM 使用者指南》中的動作和使用服務連結角色。
- 如果您使用 eksctl，請在電腦上安裝 版本 0.210.0或更新版本。如需有關安裝或更新的指示，請參閱 eksctl 文件中的 Installation 一節。
- 在您的裝置或 AWS CloudShell 上安裝和設定的 AWS 命令列界面 (AWS CLI) 版本 1.27.160 2.12.3或更新版本。若要檢查您目前的版本，請使用 aws --version | cut -d / -f2 | cut -d ' ' -f1。適用於 macOS 的 yum、 apt-get或 Homebrew 等套件管理員通常是最新版本 CLI AWS 後面的幾個版本。若要安裝最新版本，請參閱《 AWS 命令列界面使用者指南》中的使用 aws 設定安裝 和 快速組態。 https://docs.aws.amazon.com/cli/latest/userguide/cli-configure-quickstart.html#cli-configure-quickstart-config安裝在 AWS CloudShell 中的 AWS CLI 版本也可能是最新版本後面的幾個版本。若要更新它，請參閱《CloudShell AWS 使用者指南》中的將 CLI 安裝到您的主目錄。 AWS CloudShell 如果您使用 AWS CloudShell，您可能需要安裝 2.12.3 版或更新版本或 1.27.160 版或更新版本的 AWS CLI，因為安裝在 AWS CloudShell 中的預設 AWS CLI 版本可能是舊版。

您可以使用 eksctl 或 CLI 來部署IPv6叢集。

使用 eksctl 部署 IPv6 叢集

a. 建立 ipv6-cluster.yaml 檔案。將隨後的命令複製到您的裝置。視需要對命令進行下列修改，然後執行修改後的命令：

- 以叢集的名稱取代 *my-cluster*。此名稱僅能使用英數字元 (區分大小寫) 和連字號。它必須以英數字元開頭，且長度不可超過 100 個字元。名稱在您要建立叢集 AWS 的區域和 AWS 帳戶中必須是唯一的。
- 將 *region-code* 取代為 Amazon EKS 支援的任何 AWS 區域。如需 AWS 區域清單，請參閱《 AWS 一般參考指南》中的 Amazon EKS 端點和配額。

- 包含叢集版本的 `version` 值。如需詳細資訊，請參閱 eks/latest/userguide/kubernetes-versions.html【Amazon EKS 支援的版本，type="documentation"】。
- 將 *my-nodegroup* 取代為您的節點群組名稱。節點群組名稱不能超過 63 個字元。它必須以字母或數字開頭，但剩餘字元也可以包含連字符和底線。
- 將 *t3.medium* 取代為任何 AWS Nitro 系統執行個體類型。

```
cat >ipv6-cluster.yaml <<EOF
---
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: my-cluster
  region: region-code
  version: "X.XX"

kubernetesNetworkConfig:
  ipFamily: IPv6

addons:
  - name: vpc-cni
    version: latest
  - name: coredns
    version: latest
  - name: kube-proxy
    version: latest

iam:
  withOIDC: true

managedNodeGroups:
  - name: my-nodegroup
    instanceType: t3.medium
EOF
```

b. 建立叢集。

```
eksctl create cluster -f ipv6-cluster.yaml
```

叢集建立需要幾分鐘的時間。在您看到最後一行輸出之前，請勿繼續，這看起來與下列輸出類似。

```
[...]
[#]  EKS cluster "my-cluster" in "region-code" region is ready
```

c. 確認預設 Pod 已指派IPv6地址。

```
kubectl get pods -n kube-system -o wide
```

範例輸出如下。

```
NAME                       READY   STATUS    RESTARTS   AGE     IP
                    NODE                                              NOMINATED NODE
 READINESS GATES
aws-node-rslts             1/1     Running   1          5m36s
 2600:1f13:b66:8200:11a5:ade0:c590:6ac8   ip-192-168-34-75.region-
code.compute.internal   <none>           <none>
aws-node-t74jh             1/1     Running   0          5m32s
 2600:1f13:b66:8203:4516:2080:8ced:1ca9   ip-192-168-253-70.region-
code.compute.internal  <none>           <none>
coredns-85d5b4454c-cw7w2   1/1     Running   0          56m
 2600:1f13:b66:8203:34e5::                ip-192-168-253-70.region-
code.compute.internal  <none>           <none>
coredns-85d5b4454c-tx6n8   1/1     Running   0          56m
 2600:1f13:b66:8203:34e5::1               ip-192-168-253-70.region-
code.compute.internal  <none>           <none>
kube-proxy-btpbk           1/1     Running   0          5m36s
 2600:1f13:b66:8200:11a5:ade0:c590:6ac8   ip-192-168-34-75.region-
code.compute.internal   <none>           <none>
kube-proxy-jjk2g           1/1     Running   0          5m33s
 2600:1f13:b66:8203:4516:2080:8ced:1ca9   ip-192-168-253-70.region-
code.compute.internal  <none>           <none>
```

d. 確認已為預設服務指派 IPv6 地址。

```
kubectl get services -n kube-system -o wide
```

範例輸出如下。

```
NAME       TYPE        CLUSTER-IP          EXTERNAL-IP   PORT(S)         AGE
 SELECTOR
```

```
kube-dns   ClusterIP   fd30:3087:b6c2::a   <none>        53/UDP,53/TCP   57m   k8s-
app=kube-dns
```

e. （選用）部署範例應用程式或部署AWS Load Balancer控制器和範例應用程式，將 HTTP 應用程式與 the section called “應用程式負載平衡”或 網路流量與 負載平衡the section called “網路負載平衡”至 IPv6 Pod。

f. 完成您為此教學課程建立的叢集和節點之後，您應該清除使用下列命令建立的資源。

```
eksctl delete cluster my-cluster
```

使用 CLI 部署 IPv6 AWS 叢集

Important

- 您必須以同一位使用者的身分完成本程序中的所有步驟。若要檢查目前使用者，請執行以下命令：

  ```
  aws sts get-caller-identity
  ```

- 必須在同一 shell 中完成此程序中的所有步驟。若干步驟使用前面步驟中設定的變數。如果在不同的 Shell 中設定變數值，則使用變數的步驟將無法正常運作。如果您使用 AWS CloudShell 完成下列程序，請記住，如果您在大約 20-30 分鐘內未使用鍵盤或指標與其互動，則 shell 工作階段會結束。正在執行的進程不算作互動。
- 這些指示是針對 Bash shell 編寫的，在其他 shell 中可能需要進行調整。

將程序此步驟中的所有*###*取代為您自己的值。

a. 執行下列命令以設定稍後步驟中使用的某些變數。將 *region-code* 取代為您要部署資源 AWS 的區域。此值可以是 Amazon EKS 支援的任何 AWS 區域。如需 AWS 區域清單，請參閱《 AWS 一般參考指南》中的 Amazon EKS 端點和配額。以叢集的名稱取代 *my-cluster*。此名稱僅能使用英數字元 (區分大小寫) 和連字號。它必須以英數字元開頭，且長度不可超過 100 個字元。名稱在您要建立叢集 AWS 的區域和 AWS 帳戶中必須是唯一的。將 *my-nodegroup* 取代為您的節點群組名稱。節點群組名稱不能超過 63 個字元。它必須以字母或數字開頭，但剩餘字元也可以包含連字符和底線。使用您的帳戶 ID 取代 *111122223333*。

```
export region_code=region-code
```

```
export cluster_name=my-cluster
export nodegroup_name=my-nodegroup
export account_id=111122223333
```

b. 使用符合 Amazon EKS 和 IPv6 要求的公有和私有子網建立 Amazon VPC。

i. 執行下列命令來設定 AWS CloudFormation 堆疊名稱的變數。您可以使用選擇的任何名稱取代 *my-eks-ipv6-vpc*。

```
export vpc_stack_name=my-eks-ipv6-vpc
```

ii. 使用 AWS CloudFormation 範本建立 IPv6 VPC。

```
aws cloudformation create-stack --region $region_code --stack-name $vpc_stack_name
 \
  --template-url https://s3.us-west-2.amazonaws.com/amazon-eks/
cloudformation/2020-10-29/amazon-eks-ipv6-vpc-public-private-subnets.yaml
```

堆疊需要幾分鐘的時間建立。執行下列命令。在命令的輸出為 之前，請勿繼續下一個步驟CREATE_COMPLETE。

```
aws cloudformation describe-stacks --region $region_code --stack-name
 $vpc_stack_name --query Stacks[].StackStatus --output text
```

iii. 檢索已建立的公有子網路的 ID。

```
aws cloudformation describe-stacks --region $region_code --stack-name
 $vpc_stack_name \
    --query='Stacks[].Outputs[?OutputKey==`SubnetsPublic`].OutputValue' --output
 text
```

範例輸出如下。

```
subnet-0a1a56c486EXAMPLE,subnet-099e6ca77aEXAMPLE
```

iv. 為已建立的公有子網啟用自動指派 IPv6 地址選項。

```
aws ec2 modify-subnet-attribute --region $region_code --subnet-id
 subnet-0a1a56c486EXAMPLE --assign-ipv6-address-on-creation
aws ec2 modify-subnet-attribute --region $region_code --subnet-id
 subnet-099e6ca77aEXAMPLE --assign-ipv6-address-on-creation
```

v. 從部署的 AWS CloudFormation 堆疊擷取範本建立的子網路和安全群組名稱，並將其存放在變數中，以供後續步驟使用。

```
security_groups=$(aws cloudformation describe-stacks --region $region_code --
stack-name $vpc_stack_name \
    --query='Stacks[].Outputs[?OutputKey==`SecurityGroups`].OutputValue' --output
 text)

public_subnets=$(aws cloudformation describe-stacks --region $region_code --stack-
name $vpc_stack_name \
    --query='Stacks[].Outputs[?OutputKey==`SubnetsPublic`].OutputValue' --output
 text)

private_subnets=$(aws cloudformation describe-stacks --region $region_code --
stack-name $vpc_stack_name \
    --query='Stacks[].Outputs[?OutputKey==`SubnetsPrivate`].OutputValue' --output
 text)

subnets=${public_subnets},${private_subnets}
```

c. 建立叢集 IAM 角色，並將所需的 Amazon EKS IAM 受管政策連接到該角色。Amazon EKS 管理的 Kubernetes 叢集會代表您呼叫其他 AWS 服務，以管理您與服務搭配使用的資源。

i. 執行下列命令以建立 `eks-cluster-role-trust-policy.json` 檔案。

```
cat >eks-cluster-role-trust-policy.json <<EOF
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "eks.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF
```

ii. 執行下列命令，以設定角色名稱的變數。您可以將 *myAmazonEKSClusterRole* 取代為選擇的任何名稱。

```
export cluster_role_name=myAmazonEKSClusterRole
```

iii. 建立角色。

```
aws iam create-role --role-name $cluster_role_name --assume-role-policy-document
 file://"eks-cluster-role-trust-policy.json"
```

iv. 擷取 IAM 角色的 ARN 並將其儲存在變數中，以便在稍後的步驟中使用。

```
CLUSTER_IAM_ROLE=$(aws iam get-role --role-name $cluster_role_name --
query="Role.Arn" --output text)
```

v. 將必要的 Amazon EKS 受管 IAM 政策連接到角色。

```
aws iam attach-role-policy --policy-arn arn:aws: iam::aws:policy/
AmazonEKSClusterPolicy --role-name $cluster_role_name
```

d. 建立叢集。

```
aws eks create-cluster --region $region_code --name $cluster_name --kubernetes-
version 1.XX \
   --role-arn $CLUSTER_IAM_ROLE --resources-vpc-config subnetIds=
$subnets,securityGroupIds=$security_groups \
   --kubernetes-network-config ipFamily=ipv6
```

i. 注意：您可能會收到錯誤，告知您請求中的其中一個可用區域沒有足夠的容量來建立 Amazon EKS 叢集。如果發生這種情況，錯誤輸出包含的可用區域可支援新的叢集。使用至少兩個位於帳戶的支援可用區域子網路來建立您的叢集。如需詳細資訊，請參閱the section called “容量不足”。

   建立叢集需要幾分鐘才能完成。執行下列命令。在命令的輸出為 之前，請勿繼續下一個步驟ACTIVE。

```
aws eks describe-cluster --region $region_code --name $cluster_name --query
 cluster.status
```

e. 為叢集建立或更新 kubeconfig 檔案，以便能夠與您的叢集通訊。

```
aws eks update-kubeconfig --region $region_code --name $cluster_name
```

根據預設，config 檔案會在 中建立，~/.kube或新叢集的組態會新增至 中的現有config檔案~/.kube。

f. 建立節點 IAM 角色。

i. 執行下列命令以建立 vpc-cni-ipv6-policy.json 檔案。

```
cat >vpc-cni-ipv6-policy.json <<EOF
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ec2:AssignIpv6Addresses",
                "ec2:DescribeInstances",
                "ec2:DescribeTags",
                "ec2:DescribeNetworkInterfaces",
                "ec2:DescribeInstanceTypes"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:CreateTags"
            ],
            "Resource": [
                "arn:aws: ec2:*:*:network-interface/*"
            ]
        }
    ]
}
EOF
```

ii. 建立 IAM 政策。

```
aws iam create-policy --policy-name AmazonEKS_CNI_IPv6_Policy --policy-document
 file://vpc-cni-ipv6-policy.json
```

iii. 執行下列命令以建立 node-role-trust-relationship.json 檔案。

```
cat >node-role-trust-relationship.json <<EOF
```

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "ec2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF
```

iv. 執行下列命令，以設定角色名稱的變數。您可以將 *AmazonEKSNodeRole* 取代為選擇的任何名稱。

```
export node_role_name=AmazonEKSNodeRole
```

v. 建立 IAM 角色。

```
aws iam create-role --role-name $node_role_name --assume-role-policy-document
 file://"node-role-trust-relationship.json"
```

vi. 將 IAM 政策連接至 IAM 角色。

```
aws iam attach-role-policy --policy-arn arn:aws: iam::$account_id:policy/
AmazonEKS_CNI_IPv6_Policy \
    --role-name $node_role_name
```

> **Important**
>
> 為實現本教學的簡單性，將政策連接至此 IAM 角色。但是，在生產叢集中，我們建議將政策連接至單獨的 IAM 角色。如需詳細資訊，請參閱the section called “為 IRSA 設定”。

vii.將兩個所需的 IAM 受管政策連接到 IAM 角色。

```
aws iam attach-role-policy --policy-arn arn:aws: iam::aws:policy/
AmazonEKSWorkerNodePolicy \
  --role-name $node_role_name
```

```
aws iam attach-role-policy --policy-arn arn:aws: iam::aws:policy/
AmazonEC2ContainerRegistryReadOnly \
  --role-name $node_role_name
```

viii擷取 IAM 角色的 ARN 並將其儲存在變數中，以便在稍後的步驟中使用。

```
node_iam_role=$(aws iam get-role --role-name $node_role_name --query="Role.Arn" --
output text)
```

g. 建立受管節點群組。

i. 查看您在上一個步驟中建立的子網路的 ID。

```
echo $subnets
```

範例輸出如下。

```
subnet-0a1a56c486EXAMPLE,subnet-099e6ca77aEXAMPLE,subnet-0377963d69EXAMPLE,subnet-0c05f819
```

ii. 建立節點群組。將 *0a1a56c486EXAMPLE*、*099e6ca77aEXAMPLE*、*0377963d69EXAMPLE* 和 *0c05f819d5EXAMPLE* 取代為上一個步驟輸出中傳回的值。請確保從以下命令中的上一個輸出刪除子網路 ID 之間的逗號。您可以將 *t3.medium* 取代為任何 AWS Nitro 系統執行個體類型。

```
aws eks create-nodegroup --region $region_code --cluster-name $cluster_name --
nodegroup-name $nodegroup_name \
    --subnets subnet-0a1a56c486EXAMPLE subnet-099e6ca77aEXAMPLE
 subnet-0377963d69EXAMPLE subnet-0c05f819d5EXAMPLE \
    --instance-types t3.medium --node-role $node_iam_role
```

此節點群組需要幾分鐘的時間建立。執行下列命令。在傳回的輸出為 之前，請勿繼續進行下一個步驟ACTIVE。

```
aws eks describe-nodegroup --region $region_code --cluster-name $cluster_name --
nodegroup-name $nodegroup_name \
    --query nodegroup.status --output text
```

h. 確認預設 Pod 已在 IP 欄中指派IPv6地址。

```
kubectl get pods -n kube-system -o wide
```

範例輸出如下。

```
NAME                       READY   STATUS    RESTARTS   AGE     IP
                    NODE                                           NOMINATED NODE
 READINESS GATES
aws-node-rslts             1/1     Running   1          5m36s
 2600:1f13:b66:8200:11a5:ade0:c590:6ac8   ip-192-168-34-75.region-
code.compute.internal   <none>           <none>
aws-node-t74jh             1/1     Running   0          5m32s
 2600:1f13:b66:8203:4516:2080:8ced:1ca9   ip-192-168-253-70.region-
code.compute.internal  <none>           <none>
coredns-85d5b4454c-cw7w2   1/1     Running   0          56m
 2600:1f13:b66:8203:34e5::                ip-192-168-253-70.region-
code.compute.internal  <none>           <none>
coredns-85d5b4454c-tx6n8   1/1     Running   0          56m
 2600:1f13:b66:8203:34e5::1               ip-192-168-253-70.region-
code.compute.internal  <none>           <none>
kube-proxy-btpbk           1/1     Running   0          5m36s
 2600:1f13:b66:8200:11a5:ade0:c590:6ac8   ip-192-168-34-75.region-
code.compute.internal   <none>           <none>
kube-proxy-jjk2g           1/1     Running   0          5m33s
 2600:1f13:b66:8203:4516:2080:8ced:1ca9   ip-192-168-253-70.region-
code.compute.internal  <none>           <none>
```

i. 確認在 IP 欄中已為預設服務指派 IPv6 地址。

```
kubectl get services -n kube-system -o wide
```

範例輸出如下。

```
NAME       TYPE        CLUSTER-IP          EXTERNAL-IP   PORT(S)         AGE
 SELECTOR
kube-dns   ClusterIP   fd30:3087:b6c2::a   <none>        53/UDP,53/TCP   57m   k8s-
app=kube-dns
```

j. （選用）部署範例應用程式或部署AWS Load Balancer控制器和範例應用程式，將 HTTP 應用程式與 the section called “應用程式負載平衡”或 網路流量與 負載平衡the section called “網路負載平衡”至 IPv6 Pod。

k. 完成您為本教學課程建立的叢集和節點之後，您應該清除使用下列命令建立的資源。刪除之前，請確定您未使用本教學課程以外的任何資源。

i. 如果您在與完成先前步驟不同的 Shell 中完成此步驟，請設定先前步驟中使用的所有變數值，將*###*取代為您完成先前步驟時指定的值。如果您在完成先前步驟的相同 shell 中完成此步驟，請跳至下一個步驟。

```
export region_code=region-code
export vpc_stack_name=my-eks-ipv6-vpc
export cluster_name=my-cluster
export nodegroup_name=my-nodegroup
export account_id=111122223333
export node_role_name=AmazonEKSNodeRole
export cluster_role_name=myAmazonEKSClusterRole
```

ii. 刪除節點群組。

```
aws eks delete-nodegroup --region $region_code --cluster-name $cluster_name --
nodegroup-name $nodegroup_name
```

刪除需要幾分鐘的時間。執行下列命令。如果傳回任何輸出，請勿繼續進行下一個步驟。

```
aws eks list-nodegroups --region $region_code --cluster-name $cluster_name --query
 nodegroups --output text
```

iii. 刪除叢集。

```
aws eks delete-cluster --region $region_code --name $cluster_name
```

叢集需要幾分鐘的時間刪除。在繼續之前，請務必使用下列命令來確定叢集已被刪除。

```
aws eks describe-cluster --region $region_code --name $cluster_name
```

在輸出與下列輸出類似之前，請勿繼續進行下一個步驟。

```
An error occurred (ResourceNotFoundException) when calling the DescribeCluster
 operation: No cluster found for name: my-cluster.
```

iv. 刪除您建立的 IAM 資源。將 *AmazonEKS_CNI_IPv6_Policy* 取代為您選擇的名稱 (如果您選擇的名稱與之前步驟中使用的名稱不同)。

```
aws iam detach-role-policy --role-name $cluster_role_name --policy-arn arn:aws:
 iam::aws:policy/AmazonEKSClusterPolicy
```

```
aws iam detach-role-policy --role-name $node_role_name --policy-arn arn:aws:
 iam::aws:policy/AmazonEKSWorkerNodePolicy
aws iam detach-role-policy --role-name $node_role_name --policy-arn arn:aws:
 iam::aws:policy/AmazonEC2ContainerRegistryReadOnly
aws iam detach-role-policy --role-name $node_role_name --policy-arn arn:aws: iam::
$account_id:policy/AmazonEKS_CNI_IPv6_Policy
aws iam delete-policy --policy-arn arn:aws: iam::$account_id:policy/
AmazonEKS_CNI_IPv6_Policy
aws iam delete-role --role-name $cluster_role_name
aws iam delete-role --role-name $node_role_name
```

v. 刪除建立 VPC 的 AWS CloudFormation 堆疊。

```
aws cloudformation delete-stack --region $region_code --stack-name $vpc_stack_name
```

啟用 Pod 的傳出網際網路存取

適用於：Linux IPv4 Fargate 節點、具有 Amazon EC2 執行個體的 Linux 節點

如果您使用 IPv6 系列部署叢集，則此主題中的資訊不適用於您的叢集，因為IPv6地址不會進行網路翻譯。如需搭配使用 IPv6 與叢集的詳細資訊，請參閱 the section called “IPv6”。

根據預設，叢集中的每個 Pod 都會從與部署 Pod 的 VPC 相關聯的無類別網域間路由 (CIDR) 區塊中指派一個私有IPv4地址。相同 VPC 中的 Pod 會使用這些私有 IP 地址做為端點彼此通訊。當 Pod 與與您 VPC 相關聯的 CIDR 區塊以外的任何IPv4地址通訊時，Amazon VPC CNI 外掛程式 （適用於 Linux 或 Windows) 預設會將 Pod IPv4地址轉譯為 Pod 正在執行之節點的主要彈性網路介面的主要私有IPv4地址 *。

> **Note**
>
> 對於 Windows 節點，還有其他要考慮的詳細資訊。根據預設，Windows 的 VPC CNI 外掛程式是以網路組態定義，其中 SNAT 會排除流向相同 VPC 內目的地的流量。這表示內部 VPC 通訊已停用 SNAT，且配置給 Pod 的 IP 地址可在 VPC 內路由。但是，傳送至 VPC 外部目的地的流量，其來源 Pod IP SNAT 會傳送至執行個體 ENI 的主要 IP 地址。此 Windows 預設組態可確保 Pod 能以與主機執行個體相同的方式存取 VPC 外部的網路。

由於此行為：

- 您的 Pod 只能與網際網路資源通訊，前提是其執行的節點具有指派給它的公有或彈性 IP 地址，且位於公有子網路中。公有子網路的關聯路由表具有網際網路閘道的路由。我們建議儘可能將節點部署到私有子網。
- 對於早於 的外掛程式版本1.8.0，使用 VPCs 對等互連、傳輸 VPC 或 AWS Direct Connect 連接到叢集 VPC 的網路或 VPC 中的資源無法在次要彈性網路介面後方啟動與 Pod 的通訊。不過，您的 Pod 可以啟動與這些資源的通訊，並從這些資源接收回應。

如果在您的環境中下列任一陳述式成立，請使用下列命令變更預設組態。

- 您的網路或 VPCs 中有使用 VPC 對等互連、傳輸 VPC 或 AWS Direct Connect 連接到叢集 VPC 的資源，這些資源需要使用 IPv4地址啟動與 Pod 的通訊，而且您的外掛程式版本早於 1.8.0。
- 您的 Pod 位於私有子網路中，需要向網際網路傳出通訊。子網路包含指向 NAT 閘道的路由。

```
kubectl set env daemonset -n kube-system aws-node AWS_VPC_K8S_CNI_EXTERNALSNAT=true
```

> **Note**
>
> AWS_VPC_K8S_CNI_EXTERNALSNAT 和 AWS_VPC_K8S_CNI_EXCLUDE_SNAT_CIDRS CNI 組態變數不適用於 Windows 節點。Windows 不支援停用 SNAT。對於從 SNAT 排除 IPv4 CIDRs 清單，您可以透過在 Windows 引導指令碼中指定 ExcludedSnatCIDRs 參數來定義此項目。如需使用此參數的詳細資訊，請參閱 the section called “引導指令碼組態參數”。

主機聯網

* 如果 Pod 的規格包含 hostNetwork=true（預設為 false），則其 IP 地址不會翻譯為不同的地址。根據預設， kube-proxy和 Amazon VPC CNI 外掛程式適用於叢集上執行的 Kubernetes Pod。對於這些 Pod，IP 地址與節點的主要 IP 地址相同，因此不會翻譯 Pod 的 IP 地址。如需 Pod hostNetwork設定的詳細資訊，請參閱 Kubernetes API 參考中的 PodSpec v1 核心。

使用 Kubernetes 網路政策限制 Pod 流量

根據預設，Kubernetes 不會限制叢集中任何 Pod 之間或任何其他網路中 Pod 和資源之間的 IP 地址、連接埠或連線。您可以使用 Kubernetes 網路政策來限制進出 Pod 的網路流量。如需詳細資訊，請參閱 Kubernetes 文件中的網路政策。

如果您的叢集上有適用於 Kubernetes 的 Amazon VPC CNI 外掛程式版本 1.13或更早版本，則需要實作第三方解決方案，將 Kubernetes 網路政策套用至您的叢集。外掛程式的版本 1.14或更新版本可以實作網路政策，因此您不需要使用第三方解決方案。在本主題中，您將了解如何將叢集設定為在叢集上使用 Kubernetes 網路政策，而不使用第三方附加元件。

下列組態支援適用於 Kubernetes 的 Amazon VPC CNI 外掛程式中的網路政策。

- 叢集上適用於 Kubernetes 的 Amazon VPC CNI 外掛程式 1.14 版或更新版本。
- 設定為 IPv4 或 IPv6 地址的叢集。
- 您可以將網路政策與 Pod 的安全群組搭配使用。有了網路政策，您就可以控制所有叢集內通訊。透過 Pod 的安全群組，您可以控制從 Pod 內的應用程式存取 AWS 服務。
- 您可以搭配自訂聯網和字首委派來使用網路政策。

考量事項

架構

- 使用適用於 Kubernetes 的 Amazon VPC CNI 外掛程式將適用於 Kubernetes 的 Kubernetes 網路政策的 Amazon VPC CNI 外掛程式套用至叢集時，您只能將政策套用至 Amazon EC2 Linux 節點。您無法將政策套用至 Fargate 或 Windows 節點。
- 網路政策只會套用 IPv4或 IPv6 地址，但不能同時套用兩者。在 IPv4叢集中，VPC CNI 會將IPv4地址指派給 Pod 並套用IPv4政策。在 IPv6叢集中，VPC CNI 會將IPv6地址指派給 Pod 並套用IPv6政策。套用至IPv6叢集的任何IPv4網路政策規則都會遭到忽略。套用至IPv4叢集的任何IPv6網路政策規則都會遭到忽略。

網路政策

- 網路政策只會套用至屬於部署一部分的 Pod。沒有metadata.ownerReferences集合的獨立 Pod 無法套用網路政策。
- 您可以將多個網路政策套用至相同的 Pod。設定選取相同 Pod 的兩個或多個政策時，所有政策都會套用至 Pod。
- 網路政策中每個通訊協定ingress:或egress:選擇器中每個通訊協定的唯一連接埠組合數目上限為 24。
- 對於任何 Kubernetes 服務，服務連接埠必須與容器連接埠相同。如果您使用的是具名連接埠，請在服務規格中使用相同的名稱。

遷移

- 如果您的叢集目前正在使用第三方解決方案來管理 Kubernetes 網路政策，您可以將這些相同的政策與適用於 Kubernetes 的 Amazon VPC CNI 外掛程式搭配使用。不過，您必須移除現有的解決方案，使其不會管理相同的政策。

安裝

- 網路政策功能會建立且需要稱為 `policyendpoints.networking.k8s.aws` 的 `PolicyEndpoint` 自訂資源定義 (CRD)。自訂資源的 `PolicyEndpoint` 物件是由 Amazon EKS 管理。您不應修改或刪除這些資源。
- 如果您執行使用執行個體角色 IAM 憑證的 Pod 或連線至 EC2 IMDS，請小心檢查是否有會封鎖 EC2 IMDS 存取的網路政策。您可能需要新增網路政策以允許 EC2 IMDS 的存取權。如需詳細資訊，請參閱《Amazon EC2 使用者指南》中的執行個體中繼資料和使用者資料。

  對服務帳戶或 EKS Pod Identity 使用 IAM 角色的 Pod 無法存取 EC2 IMDS。
- 適用於 Kubernetes 的 Amazon VPC CNI 外掛程式不會將網路政策套用至每個 Pod 的其他網路介面，只會套用每個 Pod 的主要介面 (`eth0`)。這會影響下列架構：
  - `ENABLE_V4_EGRESS` 變數設定為 `true` 的 `IPv6` Pod。此變數可讓 `IPv4` 輸出功能將 IPv6 Pod 連線到 `IPv4` 端點，例如叢集外部的端點。`IPv4` 輸出功能的運作方式是使用本機迴路 IPv4 地址建立額外的網路介面。
  - 使用鏈結的網絡外掛程式（例如 Multus）時。由於這些外掛程式會將網路介面新增至每個 Pod，因此網路政策不會套用至鏈結的網路外掛程式。

使用 Kubernetes 網路政策限制 Pod 網路流量

您可以使用 Kubernetes 網路政策來限制進出 Pod 的網路流量。如需詳細資訊，請參閱 Kubernetes 文件中的網路政策。

您必須設定下列項目才能使用此功能：

1. 在 Pod 啟動時設定政策強制執行。您可以在 VPC CNI 的`aws-node`容器中執行此操作`DaemonSet`。
2. 啟用附加元件的網路政策參數。
3. 設定您的叢集以使用 Kubernetes 網路政策

開始之前，請檢閱考量事項。如需詳細資訊，請參閱the section called “考量事項”。

先決條件

以下是此功能的先決條件：

最低叢集版本

現有 Amazon EKS 叢集。若要部署叢集，請參閱 開始使用。叢集必須執行下表所列的其中一種 Kubernetes 版本和平台版本，請注意，也支援任何高於列出的 Kubernetes 和平台版本。您可以檢查目前的 Kubernetes 版本，方法是將下列命令中的 *my-cluster* 取代為您的叢集名稱，然後執行修改後的命令：

```
aws eks describe-cluster --name my-cluster --query cluster.version --output text
```

Kubernetes 版本	平台版本
1.27.4	eks.5
1.26.7	eks.6

最低 VPC CNI 版本

叢集上適用於 Kubernetes 的 Amazon VPC CNI 外掛程式版本 1.14或更新版本。您可以使用下列命令來查看您目前擁有哪個版本。

```
kubectl describe daemonset aws-node --namespace kube-system | grep amazon-k8s-cni: |
 cut -d : -f 3
```

如果您的版本早於 1.14，請查看 the section called “更新 (EKS 附加元件)” 以升級到版本 1.14 或更高版本。

最低 Linux 核心版本

您的節點必須具有 Linux 核心版本 5.10 或更高版本。您可以使用 uname -r 來檢查您的核心版本。如果您使用的是最新版本的 Amazon EKS 最佳化 Amazon Linux、Amazon EKS 最佳化加速 Amazon Linux AMIs 和 Bottlerocket AMIs，則它們已經具有所需的核心版本。

Amazon EKS 最佳化加速 Amazon Linux AMI 版本 v20231116 或具有核心版本 5.10 的更新版本。

步驟 1：在 Pod 啟動時設定政策強制執行

適用於 Kubernetes 的 Amazon VPC CNI 外掛程式會設定與 Pod 佈建平行的 Pod 網路政策。在為新 Pod 設定所有政策之前，新 Pod 中的容器將以預設允許政策開始。這稱為標準模式。預設允許政策表示允許進出新 Pod 的所有輸入和輸出流量。例如，在使用作用中政策更新新 Pod 之前，Pod 不會強制執行任何防火牆規則（允許所有流量）。

將NETWORK_POLICY_ENFORCING_MODE變數設定為 時strict，使用 VPC CNI 的 Pod 會從預設拒絕政策開始，然後設定政策。這稱為嚴格模式。在嚴格模式下，您必須針對 Pod 在叢集中需要存取的每個端點設定網路政策。請注意，此要求適用於 CoreDNS Pod。使用主機聯網的 Pod 未設定預設拒絕政策。

您可以在 VPC CNI 的NETWORK_POLICY_ENFORCING_MODEstrictaws-node容器中將環境變數設定為 ，以變更預設網路政策DaemonSet。

```
env:
  - name: NETWORK_POLICY_ENFORCING_MODE
    value: "strict"
```

步驟 2：啟用附加元件的網路政策參數

依預設，網路政策功能會使用節點的連接埠 8162 做為指標。此外，該功能使用了連接埠 8163 進行健康狀態探查。如果您在需要使用這些連接埠的節點或 Pod 內部執行另一應用程式，則該應用程式將無法執行。在 VPC CNI 版本 v1.14.1 或更新版本中，您可以變更這些連接埠。

使用下列程序來啟用附加元件的網路政策參數。

AWS Management Console

1. 開啟 Amazon EKS 主控台。
2. 在左側導覽窗格中，選取叢集，然後選取您要為其設定 Amazon VPC CNI 附加元件的叢集名稱。
3. 選擇附加元件索引標籤。
4. 選取附加元件方塊右上方的方塊，然後選擇 Edit (編輯)。
5. 在設定**Amazon VPC CNI**頁面上：
   a. 在版本清單中選取 v1.14.0-eksbuild.3或更新版本。
   b. 展開選用組態設定。
   c. 在組態值中輸入 JSON 金鑰 "enableNetworkPolicy": 和值 "true"。產生的文字必須是有效的 JSON 物件。如果此金鑰和值是文字方塊中唯一的資料，請以大括號 { } 括住該金鑰和值。

下列範例已啟用網路政策功能，並將指標和運作狀態探查設定為預設連接埠號碼：

```
{
    "enableNetworkPolicy": "true",
    "nodeAgent": {
        "healthProbeBindAddr": "8163",
        "metricsBindAddr": "8162"
    }
}
```

Helm

如果您已透過 安裝適用於 Kubernetes 的 Amazon VPC CNI 外掛程式helm，您可以更新組態以變更連接埠。

1. 執行下列命令來變更連接埠。分別在金鑰 nodeAgent.metricsBindAddr 或金鑰 nodeAgent.healthProbeBindAddr 值設定連接埠號碼。

```
helm upgrade --set nodeAgent.metricsBindAddr=8162 --set
 nodeAgent.healthProbeBindAddr=8163 aws-vpc-cni --namespace kube-system eks/aws-vpc-
cni
```

kubectl

1. 在您的編輯器中開啟 aws-node DaemonSet。

```
kubectl edit daemonset -n kube-system aws-node
```

2. 在 VPC CNI aws-node daemonset 清單檔案的 aws-network-policy-agent 容器，取代 args: 中下列命令引數的連接埠號碼。

```
    - args:
            - --metrics-bind-addr=:8162
            - --health-probe-bind-addr=:8163
```

步驟 3：在您的節點上掛載 Berkeley 封包篩選條件 (BPF) 檔案系統

您必須在每個節點上掛載 Berkeley Packet Filter (BPF) 檔案系統。

> **Note**
>
> 如果您的叢集是版本 1.27 或更高版本，您可以跳過此步驟，因為所有 Amazon EKS 最佳化的 Amazon Linux 和 1.27 或更高版本的 Bottlerocket AMI 均已擁有此功能。
> 對於所有其他叢集版本，如果您將 Amazon EKS 最佳化的 Amazon Linux 升級到版本 v20230703 或者更高版本，或者您將 Bottlerocket AMI 升級到版本 v1.0.2 或更高版本，則可以跳過此步驟。

1. 在每個節點上掛載 Berkeley Packet Filter (BPF) 檔案系統。

```
sudo mount -t bpf bpffs /sys/fs/bpf
```

2. 接著，將相同的命令新增至您 Amazon EC2 Auto Scaling 群組的啟動範本中的使用者資料。

步驟 4：將叢集設定為使用 Kubernetes 網路政策

您可以為 Amazon EKS 附加元件或自我管理附加元件設定此值。

Amazon EKS 附加元件

使用 AWS CLI，您可以執行下列命令，將叢集設定為使用 Kubernetes 網路政策。將 my-cluster 取代為您的叢集名稱，並將 IAM 角色 ARN 取代為您正在使用的角色。

```
aws eks update-addon --cluster-name my-cluster --addon-name vpc-cni --addon-version
 v1.14.0-eksbuild.3 \
    --service-account-role-arn arn:aws: iam::123456789012:role/AmazonEKSVPCCNIRole \
    --resolve-conflicts PRESERVE --configuration-values '{"enableNetworkPolicy":
 "true"}'
```

若要使用 AWS 管理主控台設定此項目，請遵循下列步驟：

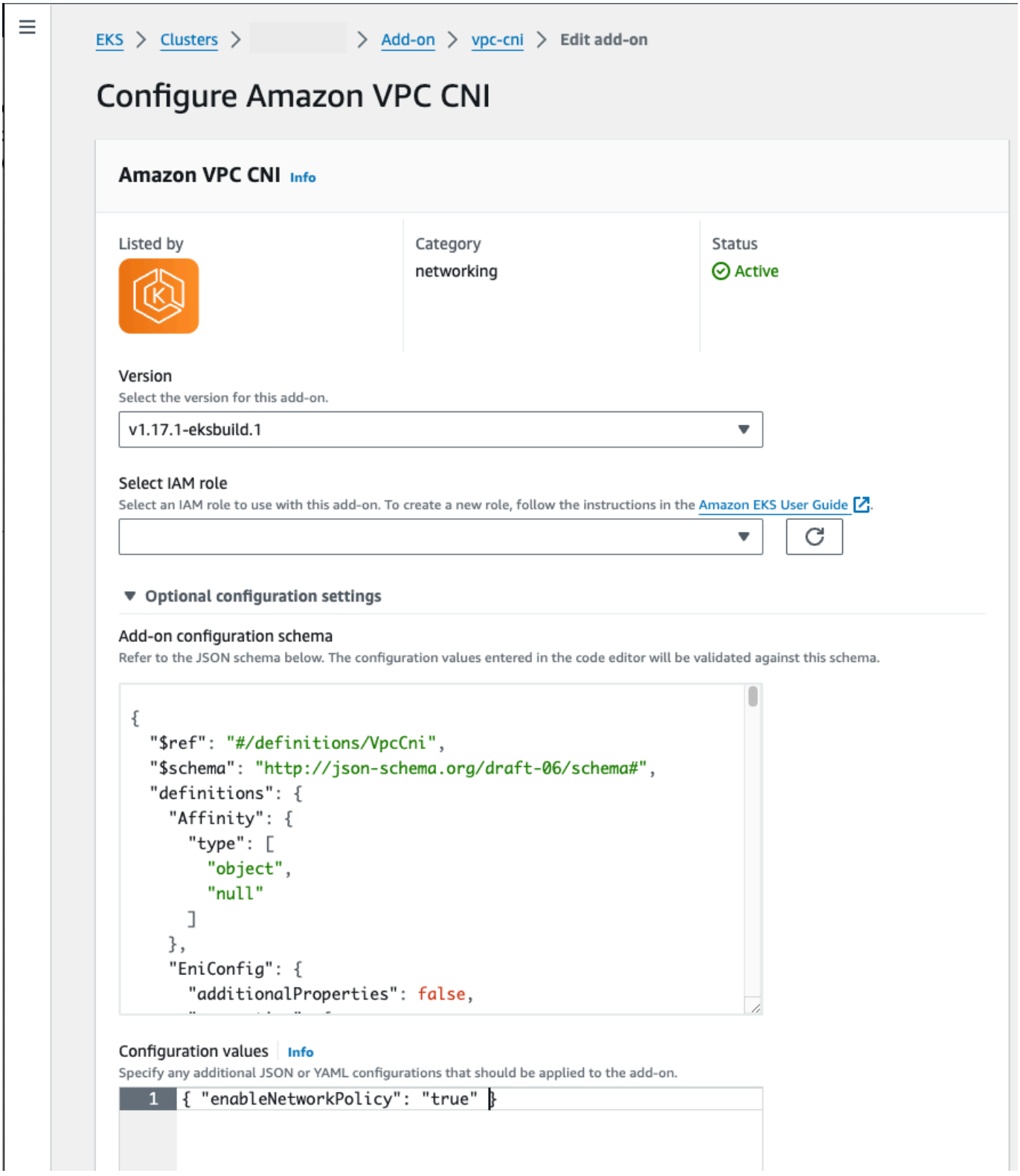
1. 開啟 Amazon EKS 主控台。
2. 在左側導覽窗格中，選取叢集，然後選取您要為其設定 Amazon VPC CNI 附加元件的叢集名稱。
3. 選擇附加元件索引標籤。
4. 選取附加元件方塊右上方的方塊，然後選擇 Edit (編輯)。
5. 在設定**Amazon VPC CNI**頁面上：
   a. 在版本清單中選取 v1.14.0-eksbuild.3或更新版本。

b. 展開選用組態設定。

c. 在組態值中輸入 JSON 金鑰 "enableNetworkPolicy": 和值 "true"。產生的文字必須是有效的 JSON 物件。如果此金鑰和值是文字方塊中唯一的資料，請以大括號 { } 括住該金鑰和值。下列範例顯示已啟用網路政策：

```
{ "enableNetworkPolicy": "true" }
```

下列螢幕擷取畫面展示了案例的範例。


EKS > Clusters > > Add-on > vpc-cni > Edit add-on
Configure Amazon VPC CNI
Amazon VPC CNI Info
Listed by
Category
networking
Status
Active
Version
Select the version for this add-on.
v1.17.1-eksbuild.1
Select IAM role
Select an IAM role to use with this add-on. To create a new role, follow the instructions in the Amazon EKS User Guide.
Optional configuration settings
Add-on configuration schema
Refer to the JSON schema below. The configuration values entered in the code editor will be validated against this schema.
{
"$ref": "#/definitions/VpcCni",
"$schema": "http://json-schema.org/draft-06/schema#",
"definitions": {
"Affinity": {
"type": [
"object",
"null"
]
},
"EniConfig": {
"additionalProperties": false,
Configuration values Info
Specify any additional JSON or YAML configurations that should be applied to the add-on.
1 { "enableNetworkPolicy": "true" }

自我管理附加元件

Helm

如果您已透過 安裝適用於 Kubernetes 的 Amazon VPC CNI 外掛程式helm，您可以更新組態以啟用網路政策。

1. 執行下列命令以啟用網路政策。

```
helm upgrade --set enableNetworkPolicy=true aws-vpc-cni --namespace kube-system eks/
aws-vpc-cni
```

kubectl

1. 在您的編輯器中開啟 amazon-vpc-cni ConfigMap。

```
kubectl edit configmap -n kube-system amazon-vpc-cni -o yaml
```

2. 在 ConfigMap 中，加入下列行至 data。

```
enable-network-policy-controller: "true"
```

新增該行之後，您的 ConfigMap應該會如下所示。

```
apiVersion: v1
 kind: ConfigMap
 metadata:
  name: amazon-vpc-cni
  namespace: kube-system
 data:
  enable-network-policy-controller: "true"
```

3. 在您的編輯器中開啟 aws-node DaemonSet。

```
kubectl edit daemonset -n kube-system aws-node
```

a. 在 VPC CNI aws-node daemonset 清單檔案的 aws-network-policy-agent 容器，以 true 取代 args: 中命令引數 --enable-network-policy=false 裡的 false。

```
    - args:
       - --enable-network-policy=true
```

步驟 5. 後續步驟

完成組態後，請確認 Pod aws-node 正在叢集上執行。

```
kubectl get pods -n kube-system | grep 'aws-node\|amazon'
```

範例輸出如下。

```
aws-node-gmqp7                                          2/2     Running   1 (24h ago)
 24h
aws-node-prnsh                                          2/2     Running   1 (24h ago)
 24h
```

版本 1.14和更新版本的 aws-node Pod 中有 2 個容器。在先前版本中，如果網路政策已停用，則 aws-node Pod 中只會有單一容器。

您現在可以將 Kubernetes 網路政策部署到您的叢集。

若要實作 Kubernetes 網路政策，您可以建立 Kubernetes NetworkPolicy 物件並將其部署到您的叢集。 NetworkPolicy 物件的範圍是命名空間。您可以實作政策，根據標籤選擇器、命名空間和 IP 地址範圍來允許或拒絕 Pod 之間的流量。如需建立NetworkPolicy物件的詳細資訊，請參閱 Kubernetes 文件中的網路政策。

Kubernetes NetworkPolicy 物件的強制執行是使用擴充 Berkeley 封包篩選條件 (eBPF) 實作。相對於 iptables 型實作，它提供了更低的延遲和效能特性，包含降低 CPU 使用率與避免循序查詢。此外，eBPF 探查可讓您存取內容豐富的資料，以協助偵錯複雜的核心層級問題並改善可觀測性。Amazon EKS 支援以 eBPF 為基礎的匯出工具，利用探查在每個節點上記錄政策結果，並將資料匯出至外部日誌收集器以協助偵錯。如需詳細資訊，請參閱 eBPF 文件。

停用 Amazon EKS Pod 網路流量的 Kubernetes 網路政策

停用 Kubernetes 網路政策以停止限制 Amazon EKS Pod 網路流量

1. 列出所有 Kubernetes 網路政策。

```
kubectl get netpol -A
```

2. 刪除每個 Kubernetes 網路政策。您必須先刪除所有網路政策，再停用網路政策。

```
kubectl delete netpol <policy-name>
```

3. 在編輯器中開啟 aws-node DaemonSet。

```
kubectl edit daemonset -n kube-system aws-node
```

4. 在 VPC CNI `aws-node` daemonset 清單檔案的 `aws-network-policy-agent` 容器，以 `false` 取代 `args:` 中命令引數 `--enable-network-policy=true` 裡的 `true`。

```
    - args:
       - --enable-network-policy=true
```

針對 Amazon EKS 的 Kubernetes 網路政策進行故障診斷

這是 Amazon VPC CNI 網路政策功能的疑難排解指南。

本指南涵蓋：

- 安裝資訊、CRD 和 RBAC 許可 the section called “新的 `policyendpoints` CRD 和許可”
- 診斷網路政策問題時要檢查的日誌 the section called “網路政策日誌”
- 執行 eBPF SDK 工具集合以進行疑難排解
- 已知問題和解決方案 the section called “已知問題和解決方案”

> **Note**
>
> 請注意，網路政策僅適用於 Kubernetes Deployments 製作的 Pod。如需 VPC CNI 中網路政策的更多限制，請參閱the section called “考量事項”。

您可以透過讀取網路政策日誌並從 eBPF SDK 執行工具，對使用網路政策的網路連線進行疑難排解和調查。 ???

新的 **`policyendpoints`** CRD 和許可

- CRD： `policyendpoints.networking.k8s.aws`
- Kubernetes API：`apiservice`稱為 `v1.networking.k8s.io`

- Kubernetes 資源： `Kind: NetworkPolicy`
- RBAC：`ClusterRole`稱為 `aws-node`(VPC CNI)，`ClusterRole`稱為 `eks:network-policy-controller`(EKS 叢集控制平面中的網路政策控制器)

對於網路政策，VPC CNI 會建立新的 `CustomResourceDefinition`(CRD)，稱為 `policyendpoints.networking.k8s.aws`。VPC CNI 必須具有許可，才能建立 CRD 並為此項目建立 CustomResources (CR)，以及 VPC CNI () 安裝的其他 CRD`eniconfigs.crd.k8s.amazonaws.com`。這兩個 CRDs都可在 GitHub 的 crds.yaml 檔案中取得。具體而言，VPC CNI 必須具有 的「get」、「list」和「watch」動詞許可`policyendpoints`。

Kubernetes 網路政策是`apiservice`名為 的一部分`v1.networking.k8s.io`，這`apiversion: networking.k8s.io/v1`位於您的政策 YAML 檔案中。VPC CNI `DaemonSet` 必須具有許可才能使用 Kubernetes API 的此部分。

VPC CNI 許可位於`ClusterRole`稱為 的 中`aws-node`。請注意，`ClusterRole`物件不會在命名空間中分組。以下顯示叢集`aws-node`的 ：

```
kubectl get clusterrole aws-node -o yaml
```

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  labels:
    app.kubernetes.io/instance: aws-vpc-cni
    app.kubernetes.io/managed-by: Helm
    app.kubernetes.io/name: aws-node
    app.kubernetes.io/version: v1.19.4
    helm.sh/chart: aws-vpc-cni-1.19.4
    k8s-app: aws-node
  name: aws-node
rules:
- apiGroups:
  - crd.k8s.amazonaws.com
  resources:
  - eniconfigs
  verbs:
  - list
  - watch
  - get
```

```
- apiGroups:
  - ""
  resources:
  - namespaces
  verbs:
  - list
  - watch
  - get
- apiGroups:
  - ""
  resources:
  - pods
  verbs:
  - list
  - watch
  - get
- apiGroups:
  - ""
  resources:
  - nodes
  verbs:
  - list
  - watch
  - get
- apiGroups:
  - ""
  - events.k8s.io
  resources:
  - events
  verbs:
  - create
  - patch
  - list
- apiGroups:
  - networking.k8s.aws
  resources:
  - policyendpoints
  verbs:
  - get
  - list
  - watch
- apiGroups:
  - networking.k8s.aws
  resources:
```

```
  - policyendpoints/status
  verbs:
  - get
- apiGroups:
  - vpcresources.k8s.aws
  resources:
  - cninodes
  verbs:
  - get
  - list
  - watch
  - patch
```

此外，新的控制器會在每個 EKS 叢集的控制平面中執行。控制器使用ClusterRole稱為 的 許可eks:network-policy-controller。以下顯示叢集eks:network-policy-controller的：

```
kubectl get clusterrole eks:network-policy-controller -o yaml
```

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  labels:
    app.kubernetes.io/name: amazon-network-policy-controller-k8s
  name: eks:network-policy-controller
rules:
- apiGroups:
  - ""
  resources:
  - namespaces
  verbs:
  - get
  - list
  - watch
- apiGroups:
  - ""
  resources:
  - pods
  verbs:
  - get
  - list
  - watch
```

```
- apiGroups:
  - ""
  resources:
  - services
  verbs:
  - get
  - list
  - watch
- apiGroups:
  - networking.k8s.aws
  resources:
  - policyendpoints
  verbs:
  - create
  - delete
  - get
  - list
  - patch
  - update
  - watch
- apiGroups:
  - networking.k8s.aws
  resources:
  - policyendpoints/finalizers
  verbs:
  - update
- apiGroups:
  - networking.k8s.aws
  resources:
  - policyendpoints/status
  verbs:
  - get
  - patch
  - update
- apiGroups:
  - networking.k8s.io
  resources:
  - networkpolicies
  verbs:
  - get
  - list
  - patch
  - update
```

```
  - watch
```

網路政策日誌

VPC CNI 的每個決策都會記錄在流程日誌中，網路政策是否允許或拒絕連線。每個節點上的網路政策日誌均包含具有網路政策的每個 Pod 之流程日誌。網路政策日誌會儲存於 `/var/log/aws-routed-eni/network-policy-agent.log`。以下範例來自於 `network-policy-agent.log` 檔案：

```
{"level":"info","timestamp":"2023-05-30T16:05:32.573Z","logger":"ebpf-
client","msg":"Flow Info: ","Src
IP":"192.168.87.155","Src Port":38971,"Dest IP":"64.6.160","Dest
Port":53,"Proto":"UDP","Verdict":"ACCEPT"}
```

網路政策日誌預設為停用。若要啟用網路政策日誌，請遵循下列步驟：

> **Note**
>
> 網路政策日誌需要 VPC CNI `aws-node`DaemonSet資訊清單中`aws-network-policy-agent`容器的額外 1 個 vCPU。

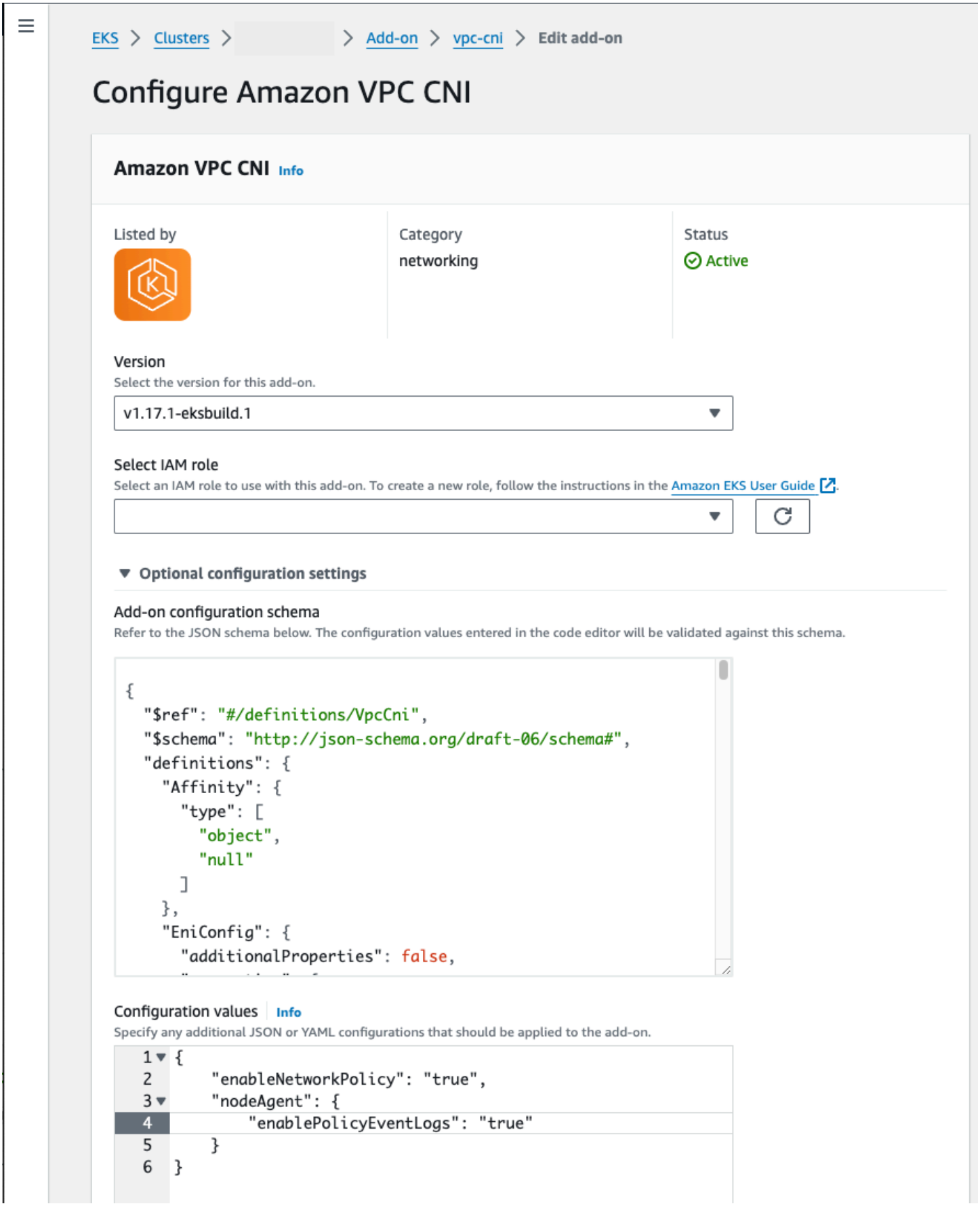
Amazon EKS 附加元件

AWS Management Console

a. 開啟 Amazon EKS 主控台。
b. 在左側導覽窗格中，選取叢集，然後選取您要為其設定 Amazon VPC CNI 附加元件的叢集名稱。
c. 選擇附加元件索引標籤。
d. 選取附加元件方塊右上方的方塊，然後選擇 Edit (編輯)。
e. 在設定 ***Amazon VPC CNI*** 頁面上：
   i. 在版本下拉式清單中，選取 `v1.14.0-eksbuild.3` 或更高版本。
   ii. 展開選用組態設定。
   iii. 輸入最上層 JSON 金鑰 `"nodeAgent":`，且值為具有組態值中的金鑰 `"enablePolicyEventLogs":` 與 `"true"` 的值之物件。產生的文字必須是有效的 JSON 物件。下列範例顯示網路政策和網路政策日誌已啟用，網路政策日誌會傳送至 CloudWatch Logs：

```
{
    "enableNetworkPolicy": "true",
    "nodeAgent": {
        "enablePolicyEventLogs": "true"
    }
}
```

下列螢幕擷取畫面展示了案例的範例。


EKS > Clusters > > Add-on > vpc-cni > Edit add-on
Configure Amazon VPC CNI
Amazon VPC CNI Info
Listed by
Category
networking
Status
Active
Version
Select the version for this add-on.
v1.17.1-eksbuild.1
Select IAM role
Select an IAM role to use with this add-on. To create a new role, follow the instructions in the Amazon EKS User Guide.
Optional configuration settings
Add-on configuration schema
Refer to the JSON schema below. The configuration values entered in the code editor will be validated against this schema.
{
"$ref": "#/definitions/VpcCni",
"$schema": "http://json-schema.org/draft-06/schema#",
"definitions": {
"Affinity": {
"type": [
"object",
"null"
]
},
"EniConfig": {
"additionalProperties": false,
Configuration values Info
Specify any additional JSON or YAML configurations that should be applied to the add-on.
1 {
2 "enableNetworkPolicy": "true",
3 "nodeAgent": {
4 "enablePolicyEventLogs": "true"
5 }
6 }

AWS CLI

a. 執行下列 AWS CLI 命令。將 my-cluster 取代為您的叢集名稱，並將 IAM 角色 ARN 取代為您正在使用的角色。

```
aws eks update-addon --cluster-name my-cluster --addon-name vpc-cni --addon-
version v1.14.0-eksbuild.3 \
    --service-account-role-arn arn:aws: iam::123456789012:role/AmazonEKSVPCCNIRole
 \
    --resolve-conflicts PRESERVE --configuration-values '{"nodeAgent":
 {"enablePolicyEventLogs": "true"}}'
```

自我管理附加元件

Helm

如果您已透過 安裝適用於 Kubernetes 的 Amazon VPC CNI 外掛程式helm，您可以更新組態以寫入網路政策日誌。

a. 執行下列命令以啟用網路政策。

```
helm upgrade --set nodeAgent.enablePolicyEventLogs=true aws-vpc-cni --namespace
 kube-system eks/aws-vpc-cni
```

kubectl

如果您已透過 安裝適用於 Kubernetes 的 Amazon VPC CNI 外掛程式kubectl，您可以更新組態以寫入網路政策日誌。

a. 在您的編輯器中開啟 aws-node DaemonSet。

```
kubectl edit daemonset -n kube-system aws-node
```

b. 在 VPC CNI aws-nodeDaemonSet資訊清單中 args: aws-network-policy-agent 容器中true的 --enable-policy-event-logs=false命令引數中false，將 取代為 。

```
    - args:
       - --enable-policy-event-logs=true
```

將網路政策日誌傳送到 Amazon CloudWatch Logs

您可以使用 Amazon CloudWatch Logs 這一類服務來監控網路政策日誌。您可以使用下列方法將網路政策日誌傳送到 CloudWatch Logs。

對於 EKS 叢集，政策日誌將位於 下，`/aws/eks/cluster-name/cluster/`對於自我管理的 K8S 叢集，日誌將放置在 下`/aws/k8s-cluster/cluster/`。

使用適用於 Kubernetes 的 Amazon VPC CNI 外掛程式傳送網路政策日誌

如果您啟用網路政策，系統會將第二個容器新增至節點代理程式的 `aws-node` Pod。此節點代理程式可以將網路政策日誌傳送到 CloudWatch Logs。

> **Note**
>
> 網路政策日誌僅會由節點代理程式傳送。不包含 VPC CNI 製作的其他日誌。

先決條件

- 將下列許可作為一節或個別政策新增至您用於 VPC CNI 的 IAM 角色。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogGroups",
                "logs:CreateLogGroup",
                "logs:CreateLogStream",
                "logs:PutLogEvents"
            ],
            "Resource": "*"
        }
    ]
}
```

Amazon EKS 附加元件

AWS Management Console

a. 開啟 Amazon EKS 主控台。

b. 在左側導覽窗格中，選取叢集，然後選取您要為其設定 Amazon VPC CNI 附加元件的叢集名稱。

c. 選擇附加元件索引標籤。

d. 選取附加元件方塊右上方的方塊，然後選擇 Edit (編輯)。

e. 在設定 ***Amazon VPC CNI*** 頁面上：

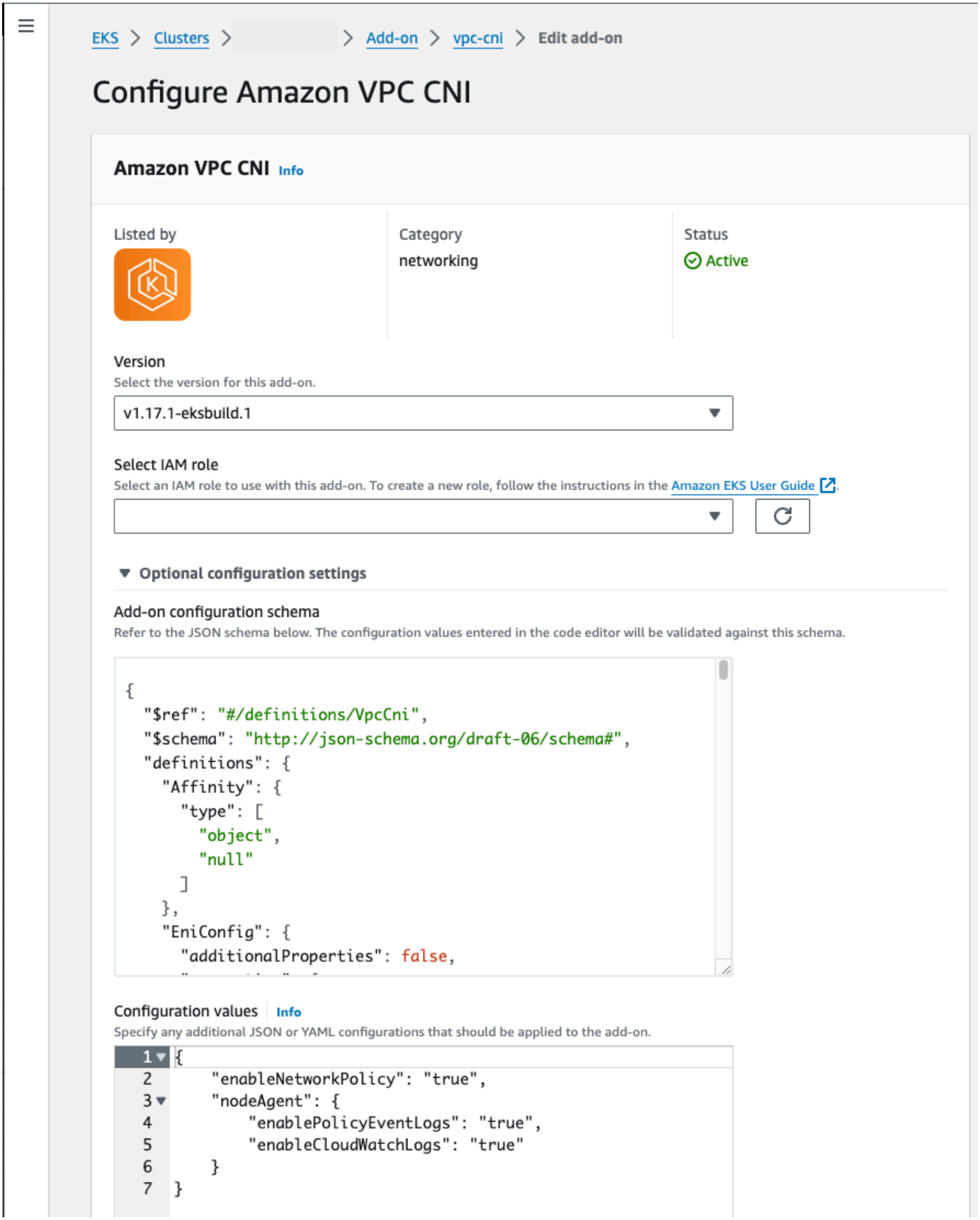
i. 在版本下拉式清單中，選取 `v1.14.0-eksbuild.3` 或更高版本。

ii. 展開選用組態設定。

iii. 輸入最上層 JSON 金鑰 `"nodeAgent":`，且值為具有組態值中的金鑰 `"enableCloudWatchLogs":` 與 `"true"` 的值之物件。產生的文字必須是有效的 JSON 物件。下列範例顯示網路政策和網路政策日誌已啟用，日誌會傳送至 CloudWatch Logs：

```
{
    "enableNetworkPolicy": "true",
    "nodeAgent": {
        "enablePolicyEventLogs": "true",
        "enableCloudWatchLogs": "true",
    }
}
```

下列螢幕擷取畫面展示了案例的範例。


EKS > Clusters > > Add-on > vpc-cni > Edit add-on
Configure Amazon VPC CNI
Amazon VPC CNI Info
Listed by
Category
networking
Status
Active
Version
Select the version for this add-on.
v1.17.1-eksbuild.1
Select IAM role
Select an IAM role to use with this add-on. To create a new role, follow the instructions in the Amazon EKS User Guide.
Optional configuration settings
Add-on configuration schema
Refer to the JSON schema below. The configuration values entered in the code editor will be validated against this schema.
{
"$ref": "#/definitions/VpcCni",
"$schema": "http://json-schema.org/draft-06/schema#",
"definitions": {
"Affinity": {
"type": [
"object",
"null"
]
},
"EniConfig": {
"additionalProperties": false,
Configuration values Info
Specify any additional JSON or YAML configurations that should be applied to the add-on.
1 {
2 "enableNetworkPolicy": "true",
3 "nodeAgent": {
4 "enablePolicyEventLogs": "true",
5 "enableCloudWatchLogs": "true"
6 }
7 }

AWS CLI

a. 執行下列 AWS CLI 命令。將 my-cluster 取代為您的叢集名稱，並將 IAM 角色 ARN 取代為您正在使用的角色。

```
aws eks update-addon --cluster-name my-cluster --addon-name vpc-cni --addon-
version v1.14.0-eksbuild.3 \
    --service-account-role-arn arn:aws: iam::123456789012:role/AmazonEKSVPCCNIRole
 \
    --resolve-conflicts PRESERVE --configuration-values '{"nodeAgent":
 {"enablePolicyEventLogs": "true", "enableCloudWatchLogs": "true"}}'
```

自我管理附加元件

Helm

如果您已透過 安裝適用於 Kubernetes 的 Amazon VPC CNI 外掛程式helm，您可以更新組態，將網路政策日誌傳送至 CloudWatch Logs。

a. 執行下列命令來啟用網路政策日誌，並將其傳送至 CloudWatch Logs。

```
helm upgrade --set nodeAgent.enablePolicyEventLogs=true --set
 nodeAgent.enableCloudWatchLogs=true aws-vpc-cni --namespace kube-system eks/aws-
vpc-cni
```

kubectl

a. 在您的編輯器中開啟 aws-node DaemonSet。

```
kubectl edit daemonset -n kube-system aws-node
```

b. false 將 取代true為兩個命令引數--enable-cloudwatch-logs=false中的 ，--enable-policy-event-logs=false並在 VPC CNI aws-nodeDaemonSet資訊清單中args:aws-network-policy-agent容器中的 中取代 。

```
    - args:
       - --enable-policy-event-logs=true
       - --enable-cloudwatch-logs=true
```

使用 Fluent Bit 傳送網路政策日誌 **DaemonSet**

如果您在 中使用 Fluent Bit 從節點DaemonSet傳送日誌，您可以新增組態，以包含來自網路政策的網路政策日誌。您可以使用下列範例組態：

```
    [INPUT]
        Name              tail
        Tag               eksnp.*
        Path              /var/log/aws-routed-eni/network-policy-agent*.log
        Parser            json
        DB                /var/log/aws-routed-eni/flb_npagent.db
        Mem_Buf_Limit     5MB
        Skip_Long_Lines   On
        Refresh_Interval  10
```

包含的 eBPF 開發套件

適用於 Kubernetes 的 Amazon VPC CNI 外掛程式會在節點上安裝 eBPF SDK 工具集合。您可以使用 eBPF SDK 工具來識別網路政策的問題。例如，下列命令會列出目前在節點上執行的程式。

```
sudo /opt/cni/bin/aws-eks-na-cli ebpf progs
```

若要執行此命令，您可以使用任何方法連接到節點。

已知問題和解決方案

下列各節說明 Amazon VPC CNI 網路政策功能及其解決方案的已知問題。

儘管 enable-policy-event-logs 設定為 false，但產生的網路政策日誌

問題：即使`enable-policy-event-logs`設定設為 ，EKS VPC CNI 也會產生網路政策日誌`false`。

解決方案： `enable-policy-event-logs`設定只會停用政策「決策」日誌，但不會停用所有網路政策代理程式日誌記錄。此行為記錄在 GitHub 上的 aws-network-policy-agent README 中。若要完全停用記錄，您可能需要調整其他記錄組態。

網路政策映射清除問題

問題：刪除 Pod 後，網路`policyendpoint`仍然存在且無法清除的問題。

解決方案：此問題是由 VPC CNI 附加元件 1.19.3-eksbuild.1 版的問題所造成。更新至較新版本的 VPC CNI 附加元件以解決此問題。

未套用網路政策

問題：在 Amazon VPC CNI 外掛程式中啟用網路政策功能，但網路政策未正確套用。

如果您建立網路政策，`kind: NetworkPolicy`但不會影響 Pod，請檢查政策端點物件是否在與 Pod 相同的命名空間中建立。如果命名空間中沒有`policyendpoint`物件，則網路政策控制器 (EKS 叢集的一部分） 無法為要套用的網路政策代理程式 (VPC CNI 的一部分） 建立網路政策規則。

解決方案：解決方案是修正 VPC CNI (`ClusterRole：aws-node`) 和網路政策控制器 (`ClusterRole：eks:network-policy-controller`) 的許可，並在 Kyverno 等任何政策強制執行工具中允許這些動作。確保 Kyverno 政策不會封鎖`policyendpoint`物件的建立。如需 中必要許可的許可，請參閱上一節the section called “新的 policyendpoints CRD 和許可”。

在嚴格模式下刪除政策後，Pod 不會返回預設拒絕狀態

問題：在嚴格模式下啟用網路政策時，Pod 會從預設拒絕政策開始。套用政策後，允許流量到指定的端點。不過，刪除政策時，Pod 不會返回預設拒絕狀態，而是進入預設允許狀態。

解決方案：此問題已在 VPC CNI 1.19.3 版中修正，其中包括網路政策代理程式 1.2.0 版。修正後，在啟用嚴格模式的情況下，一旦移除政策，Pod 會如預期回到預設拒絕狀態。

Pod 啟動延遲的安全群組

問題：在 EKS 中使用 Pod 安全群組功能時，Pod 啟動延遲會增加。

解決方案：延遲是由於資源控制器中`CreateNetworkInterface`來自 API 上的 API 限流的速率限制，VPC 資源控制器用於為 Pod 建立分支 ENIs。檢查您帳戶的此操作 API 限制，並視需要考慮請求提高限制。

由於 vpc.amazonaws.com/pod-eni 不足而FailedScheduling

問題：Pod 無法以錯誤排程： `FailedScheduling 2m53s (x28 over 137m) default-scheduler 0/5 nodes are available: 5 Insufficient vpc.amazonaws.com/pod-eni. preemption: 0/5 nodes are available: 5 No preemption victims found for incoming pod.`

解決方案：與上一個問題一樣，將安全群組指派給 Pod 會增加 Pod 排程延遲，並且可能會增加超過 CNI 閾值的時間來新增每個 ENI，導致啟動 Pod 的失敗。這是使用 Pod 安全群組時的預期行為。設計工作負載架構時，請考慮排程的影響。

IPAM 連線問題和分割錯誤

問題：發生多個錯誤，包括 IPAM 連線問題、限流請求和分割錯誤：

- `Checking for IPAM connectivity …`
- `Throttling request took 1.047064274s`
- `Retrying waiting for IPAM-D`
- `panic: runtime error: invalid memory address or nil pointer dereference`

解決方案：如果您`systemd-udev`在 AL2023 上安裝 ，就會發生此問題，因為檔案會以中斷政策重新寫入。當更新到具有更新套件或手動更新套件本身`releasever`的不同 時，可能會發生這種情況。避免在 AL2023 節點`systemd-udev`上安裝或更新 。

無法依名稱尋找裝置錯誤

問題：錯誤訊息：
`{"level":"error","ts":"2025-02-05T20:27:18.669Z","caller":"ebpf/bpf_client.go:578","msg":"failed to find device by name eni9ea69618bf0: %!w(netlink.LinkNotFoundError={0xc000115310})"}`

解決方案：此問題已在最新版本的 Amazon VPC CNI 網路政策代理程式 (v1.2.0) 中識別和修正。更新至 VPC CNI 的最新版本，以解決此問題。

Multus CNI 映像中的 CVE 漏洞

問題：增強型 EKS ImageScan CVE 報告可識別 Multus CNI 映像版本 v4.1.4-eksbuild.2_thick 中的漏洞。

解決方案：更新至 Multus CNI 映像的新版本，以及沒有漏洞的新網路政策控制器映像。您可以更新掃描器，以解決先前版本中找到的漏洞。

日誌中的流程資訊 DENY 判定

問題：網路政策日誌顯示 DENY 判定：
`{"level":"info","ts":"2024-11-25T13:34:24.808Z","logger":"ebpf-client","caller":"events/events.go:193","msg":"Flow Info: ","Src IP":"","Src Port":9096,"Dest IP":"","Dest Port":56830,"Proto":"TCP","Verdict":"DENY"}`

解決方案：此問題已在新版本的網路政策控制器中解決。更新至最新的 EKS 平台版本，以解決記錄問題。

從 Calico 遷移後的 Pod-to-pod 通訊問題

問題：將 EKS 叢集升級至 1.30 版，並針對網路政策從 Calico 切換到 Amazon VPC CNI 之後，套用網路政策時 pod-to-pod 通訊會失敗。刪除網路政策時會還原通訊。

解決方案：VPC CNI 中的網路政策代理程式無法擁有與 Calico 相同的指定連接埠數量。請改用網路政策中的連接埠範圍。網路政策中每個通訊協定`ingress:`或`egress:`選擇器中每個通訊協定的唯一連接埠組合數目上限為 24。使用連接埠範圍來減少唯一連接埠的數量，並避免此限制。

網路政策代理程式不支援獨立 Pod

問題：套用至獨立 Pod 的網路政策可能有不一致的行為。

解決方案：網路政策代理程式目前僅支援部署為部署/複本一部分的 Pod。如果網路政策套用至獨立 Pod，則行為中可能會有一些不一致。此頁面頂端、 the section called “考量事項”和 GitHub 的 aws-network-policy-agent GitHub 問題編號 327 GitHub。將 Pod 部署為部署或複本的一部分，以實現一致的網路政策行為。

Amazon EKS 網路政策的星星示範

此示範會在 Amazon EKS 叢集上建立前端、後端和用戶端服務。示範中亦將建立管理圖形使用者介面，其顯示各服務之間可用的輸入和輸出路徑。我們建議您在未執行生產工作負載的叢集上完成示範。

在您建立任何網路政策前，所有服務都可以雙向通訊。在您套用網路政策後，便可看到用戶端只能與前端服務通訊，同時後端只能接受來自前端的流量。

1. 套用前端、後端、用戶端和管理使用者介面服務：

```
kubectl apply -f https://raw.githubusercontent.com/aws-samples/eks-
workshop/2f9d29ed3f82ed6b083649e975a0e574fb8a4058/content/beginner/120_network-
policies/calico/stars_policy_demo/create_resources.files/namespace.yaml
kubectl apply -f https://raw.githubusercontent.com/aws-samples/eks-
workshop/2f9d29ed3f82ed6b083649e975a0e574fb8a4058/content/beginner/120_network-
policies/calico/stars_policy_demo/create_resources.files/management-ui.yaml
kubectl apply -f https://raw.githubusercontent.com/aws-samples/eks-
workshop/2f9d29ed3f82ed6b083649e975a0e574fb8a4058/content/beginner/120_network-
policies/calico/stars_policy_demo/create_resources.files/backend.yaml
kubectl apply -f https://raw.githubusercontent.com/aws-samples/eks-
workshop/2f9d29ed3f82ed6b083649e975a0e574fb8a4058/content/beginner/120_network-
policies/calico/stars_policy_demo/create_resources.files/frontend.yaml
```

```
kubectl apply -f https://raw.githubusercontent.com/aws-samples/eks-
workshop/2f9d29ed3f82ed6b083649e975a0e574fb8a4058/content/beginner/120_network-
policies/calico/stars_policy_demo/create_resources.files/client.yaml
```

2. 檢視叢集上的所有 Pod。

```
kubectl get pods -A
```

範例輸出如下。

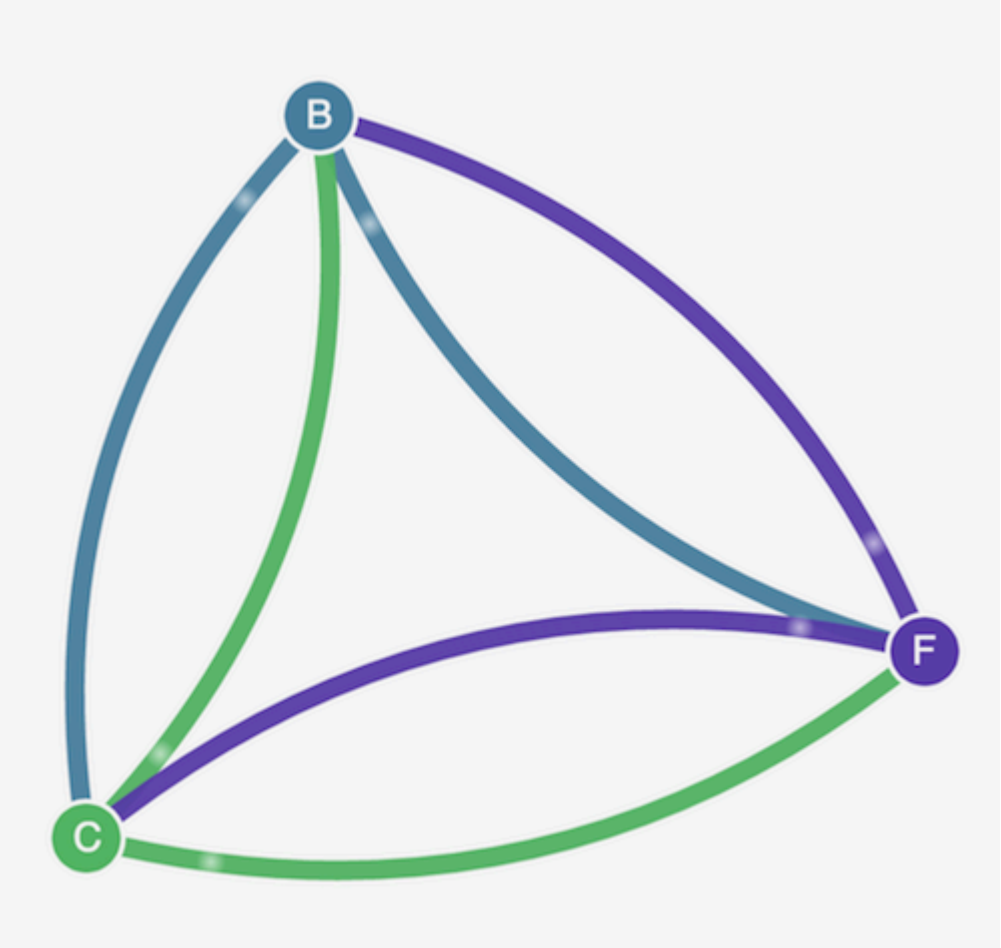
在輸出中，您應該在以下輸出中顯示的命名空間中看到 pod。*NAMES* 和 READY 直欄中的 Pod 數量與下列輸出中的數量不同。除非您看到具有類似名稱的 Pod，且它們都Running位於 STATUS欄中，否則請勿繼續。

```
NAMESPACE         NAME                                       READY   STATUS
 RESTARTS   AGE
[...]
client            client-xlffc                               1/1     Running   0
     5m19s
[...]
management-ui     management-ui-qrb2g                        1/1     Running   0
     5m24s
stars             backend-sz87q                              1/1     Running   0
     5m23s
stars             frontend-cscnf                             1/1     Running   0
     5m21s
[...]
```

3. 若要連接到管理使用者介面，請連接到在叢集上執行之服務的 EXTERNAL-IP：

```
kubectl get service/management-ui -n management-ui
```

4. 打開瀏覽器到上一個步驟的位置。您應該會看到管理使用者介面。C 節點是用戶端服務，而 F 節點是前端服務，且 B 節點是後端服務。每個節點都有對所有其他節點的完整通訊存取權 (如粗體、上顏色行的文字所指示)。



5. 在 stars 和 client 命名空間中套用以下網路政策來將服務彼此隔離：

```
kind: NetworkPolicy
apiVersion: networking.k8s.io/v1
metadata:
  name: default-deny
spec:
  podSelector:
    matchLabels: {}
```

您可以使用下列命令將政策同時套用至兩個命名空間：

```
kubectl apply -n stars -f https://raw.githubusercontent.com/aws-samples/eks-
workshop/2f9d29ed3f82ed6b083649e975a0e574fb8a4058/content/beginner/120_network-
policies/calico/stars_policy_demo/apply_network_policies.files/default-deny.yaml
kubectl apply -n client -f https://raw.githubusercontent.com/aws-samples/eks-
workshop/2f9d29ed3f82ed6b083649e975a0e574fb8a4058/content/beginner/120_network-
policies/calico/stars_policy_demo/apply_network_policies.files/default-deny.yaml
```

6. 重新整理您的瀏覽器。您看到管理使用者介面無法再連接任何節點，因此它們不會出現在使用者介面中。
7. 套用下列不同的網路政策，以允許管理使用者介面存取服務。套用此政策以允許 UI：

```
kind: NetworkPolicy
apiVersion: networking.k8s.io/v1
metadata:
  namespace: stars
  name: allow-ui
spec:
  podSelector:
    matchLabels: {}
  ingress:
    - from:
        - namespaceSelector:
            matchLabels:
              role: management-ui
```

套用此政策以允許用戶端：

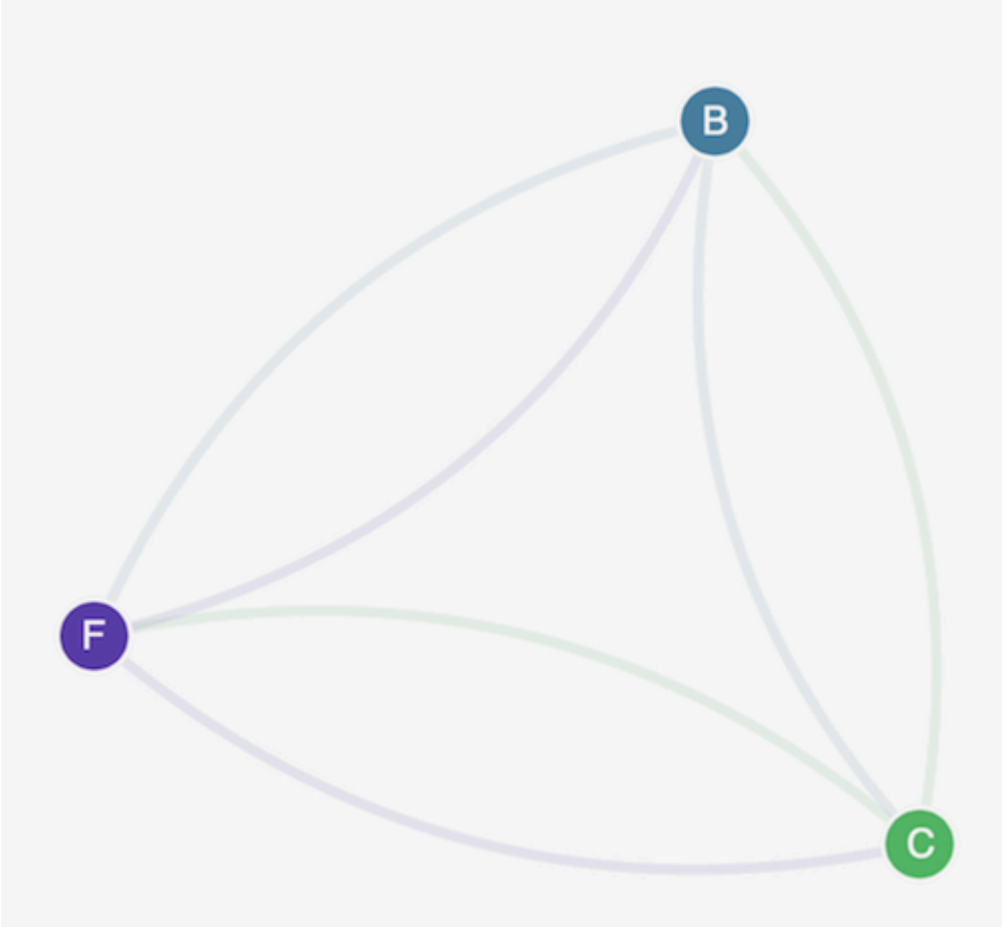
```
kind: NetworkPolicy
apiVersion: networking.k8s.io/v1
metadata:
  namespace: client
  name: allow-ui
spec:
  podSelector:
    matchLabels: {}
  ingress:
    - from:
        - namespaceSelector:
            matchLabels:
              role: management-ui
```

您可以使用下列命令來同時套用兩個政策：

```
kubectl apply -f https://raw.githubusercontent.com/aws-samples/eks-
workshop/2f9d29ed3f82ed6b083649e975a0e574fb8a4058/content/beginner/120_network-
policies/calico/stars_policy_demo/apply_network_policies.files/allow-ui.yaml
```

```
kubectl apply -f https://raw.githubusercontent.com/aws-samples/eks-
workshop/2f9d29ed3f82ed6b083649e975a0e574fb8a4058/content/beginner/120_network-
policies/calico/stars_policy_demo/apply_network_policies.files/allow-ui-client.yaml
```

8. 重新整理您的瀏覽器。您將看到管理使用者介面再次可到達各節點，但各節點無法互相通訊。

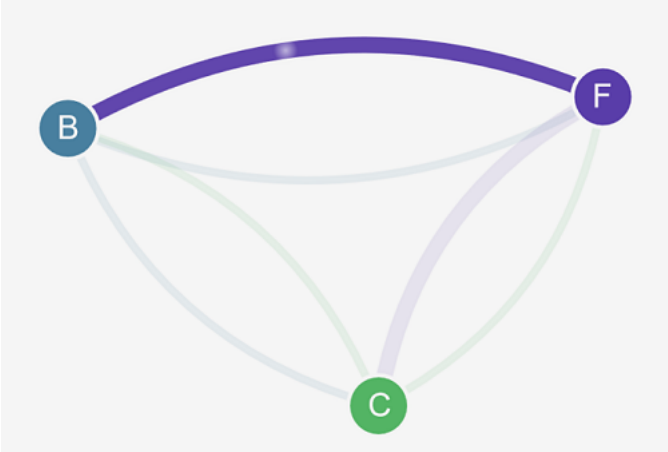


9. 套用以下網路政策以允許流量從前端服務流向後端服務：

```
kind: NetworkPolicy
apiVersion: networking.k8s.io/v1
metadata:
  namespace: stars
  name: backend-policy
spec:
  podSelector:
    matchLabels:
      role: backend
  ingress:
    - from:
        - podSelector:
```

```
            matchLabels:
              role: frontend
      ports:
        - protocol: TCP
          port: 6379
```

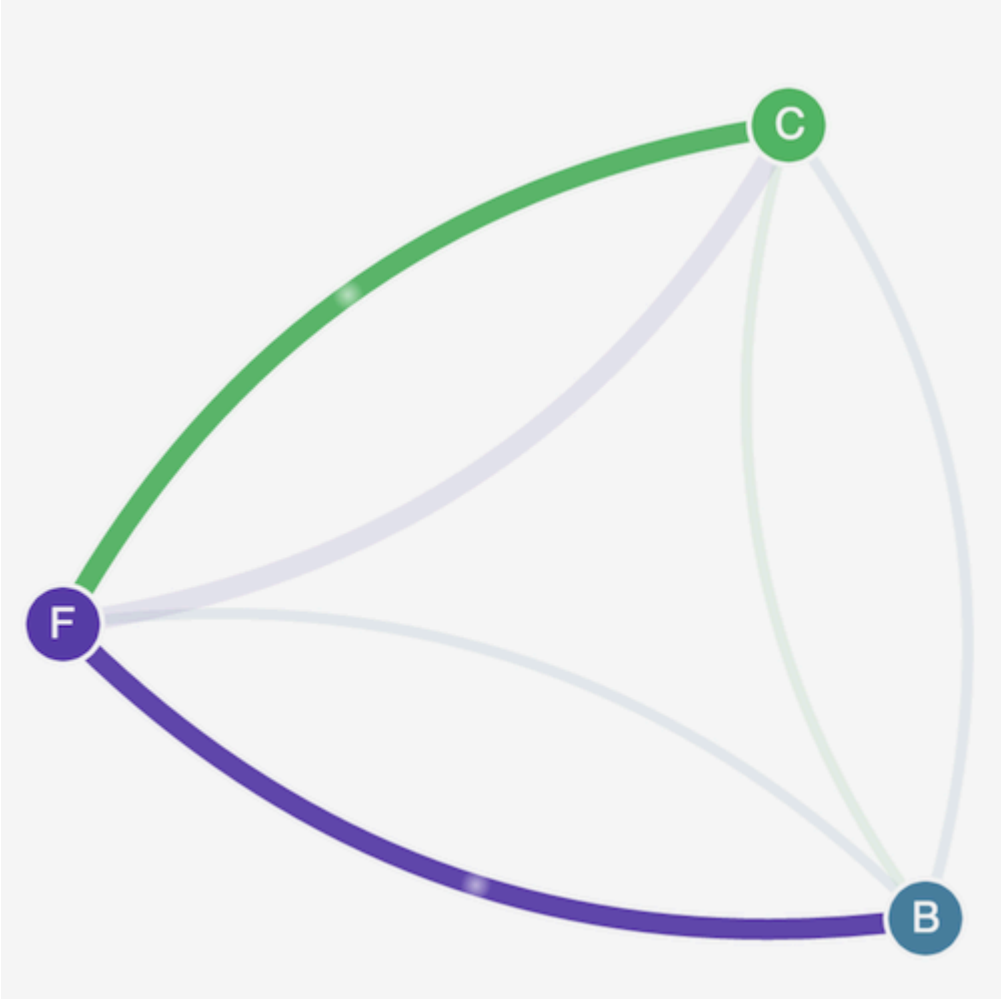
10重新整理您的瀏覽器。您可以看到前端服務可以與後端服務通訊。



11套用以下網路政策以允許流量從用戶端流向後端服務：

```
kind: NetworkPolicy
apiVersion: networking.k8s.io/v1
metadata:
  namespace: stars
  name: frontend-policy
spec:
  podSelector:
    matchLabels:
      role: frontend
  ingress:
    - from:
        - namespaceSelector:
            matchLabels:
              role: client
      ports:
        - protocol: TCP
          port: 80
```

12重新整理您的瀏覽器。您可以看到用戶端可以與前端服務通訊。前端服務仍然可以與後端服務通訊。



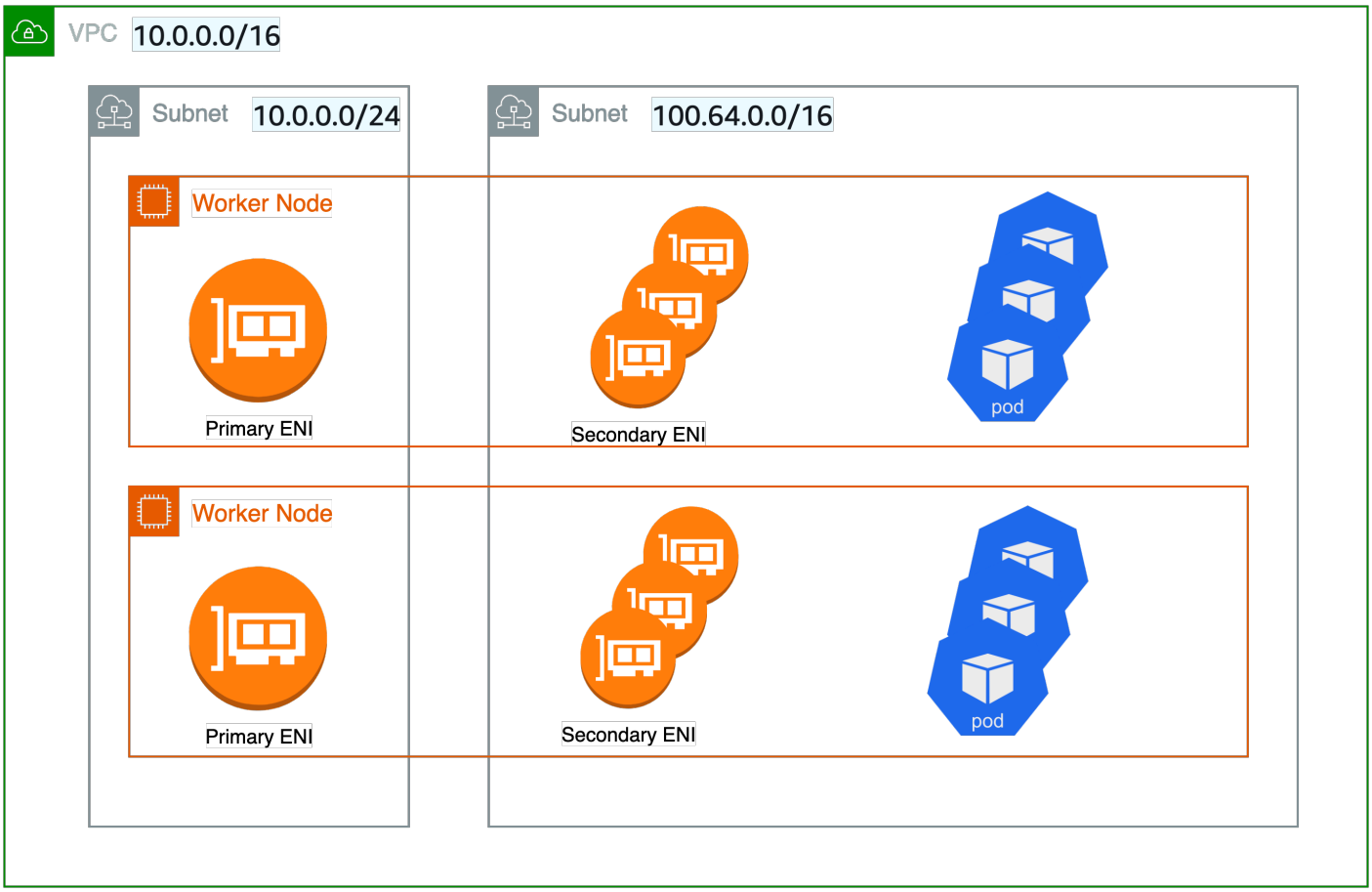
13(選用) 在完成示範後，您可以刪除其資源。

```
kubectl delete -f https://raw.githubusercontent.com/aws-samples/eks-
workshop/2f9d29ed3f82ed6b083649e975a0e574fb8a4058/content/beginner/120_network-
policies/calico/stars_policy_demo/create_resources.files/client.yaml
kubectl delete -f https://raw.githubusercontent.com/aws-samples/eks-
workshop/2f9d29ed3f82ed6b083649e975a0e574fb8a4058/content/beginner/120_network-
policies/calico/stars_policy_demo/create_resources.files/frontend.yaml
kubectl delete -f https://raw.githubusercontent.com/aws-samples/eks-
workshop/2f9d29ed3f82ed6b083649e975a0e574fb8a4058/content/beginner/120_network-
policies/calico/stars_policy_demo/create_resources.files/backend.yaml
kubectl delete -f https://raw.githubusercontent.com/aws-samples/eks-
workshop/2f9d29ed3f82ed6b083649e975a0e574fb8a4058/content/beginner/120_network-
policies/calico/stars_policy_demo/create_resources.files/management-ui.yaml
kubectl delete -f https://raw.githubusercontent.com/aws-samples/eks-
workshop/2f9d29ed3f82ed6b083649e975a0e574fb8a4058/content/beginner/120_network-
policies/calico/stars_policy_demo/create_resources.files/namespace.yaml
```

即使刪除資源之後，節點上仍可能有網路政策端點，這些端點可能會以非預期的方式干擾叢集中的聯網。移除這些規則的唯一確定方法，是重新啟動節點或終止所有節點並將其回收。若要終止所有節點，請將 Auto Scaling 群組所需的計數設定為 0，然後備份到所需的數字，或是只終止節點。

使用自訂聯網在替代子網路中部署 Pod

適用於：Linux IPv4 Fargate 節點、具有 Amazon EC2 執行個體的 Linux 節點



根據預設，當適用於 Kubernetes 的 Amazon VPC CNI 外掛程式為您的 Amazon EC2 節點建立次要[彈性網路介面](#)（網路介面）時，它會在與節點的主要網路介面相同的子網路中建立它們。它還會將與主網路介面關聯的相同安全群組關聯至次要網路介面。由於以下一或多個原因，您可能希望外掛程式在不同子網中建立次要網路介面，或者希望將不同的安全群組與次要網路介面關聯 (或者兩者兼具)：

- 主要網路界面所在的子網路中可用的IPv4地址數量有限。這可能會限制您可以在子網路中建立的 Pod 數量。透過為次要網路介面使用不同的子網路，您可以增加 Pod 可用的IPv4地址數量。
- 基於安全考量，您的 Pod 可能需要使用與節點的主要網路界面不同的子網路或安全群組。

- 節點是在公有子網路中設定，而您想要將 Pod 放置在私有子網路中。與公有子網相關聯的路由表包括網際網路閘道的路由。與私有子網路相關聯的路由表不包含網際網路閘道的路由。

> **Tip**
>
> 您也可以直接將新的或現有的子網路新增至 Amazon EKS 叢集，而無需使用自訂聯網。如需詳細資訊，請參閱the section called “從管理主控台將現有的 VPC 子網路新增至 Amazon EKS 叢集”。

考量事項

以下是使用 功能的考量。

- 啟用自訂聯網後，不會將指派給主要網路介面的 IP 地址指派給 Pod。只有來自次要網路介面的 IP 地址才會指派給 Pod。
- 如果您的叢集使用 IPv6 系列，則無法使用自訂聯網。
- 如果您計劃僅使用自訂聯網來協助緩解 IPv4 地址耗盡的情況，則可以改為使用 IPv6 系列來建立叢集。如需詳細資訊，請參閱the section called “IPv6”。
- 即使部署到為次要網路介面指定之子網路的 Pod 可以使用與節點主要網路介面不同的子網路和安全群組，子網路和安全群組必須與節點位於相同的 VPC 中。
- 對於 Fargate，子網路是透過 Fargate 設定檔控制。如需詳細資訊，請參閱the section called “定義設定檔”。

在 Amazon EKS 節點中自訂次要網路介面

開始教學課程之前，請先完成下列各項：

- 檢閱考量事項
- 熟悉適用於 Kubernetes 的 Amazon VPC CNI 外掛程式如何建立次要網路介面，並將 IP 地址指派給 Pod。如需詳細資訊，請參閱 GitHub 上的 ENI Allocation (ENI 分配)。
- 在您的裝置或 AWS CloudShell 上安裝和設定的 AWS 命令列界面 (AWS CLI) 版本 `1.27.160` `2.12.3`或更新版本。若要檢查您目前的版本，請使用 `aws --version | cut -d / -f2 | cut -d ' ' -f1`。適用於 macOS 的 `yum`、 `apt-get`或 Homebrew 等套件管理員通常是最新版本 CLI AWS 後面的幾個版本。若要安裝最新版本，請參閱《 AWS 命令列界面使用者指南》中的使用 aws 設定安裝 和 快速組態。 https://docs.aws.amazon.com/cli/latest/userguide/cli-configure-

quickstart.html#cli-configure-quickstart-config在 AWS CloudShell 中安裝的 AWS CLI 版本也可能是最新版本後面的幾個版本。若要更新它，請參閱《CloudShell AWS 使用者指南》中的將 CLI 安裝到您的主目錄。 AWS CloudShell

- kubectl 命令列工具安裝在您的裝置或 AWS CloudShell 上。若要安裝或升級 kubectl，請參閱 the section called “設定 kubectl 和 eksctl”。
- 建議您在 Bash shell 中完成本主題中的步驟。如果您不使用 Bash shell，一些指令碼命令，例如行接續字元和變數的設定和使用方式需要調整您的 shell。此外，您的 Shell 的引用及轉義規則可能會有所不同。如需詳細資訊，請參閱《 AWS 命令列界面使用者指南》中的使用引號搭配 AWS CLI 中的字串。

在本教學課程中，建議您使用範例值，但記下要取代這些值之處除外。完成生產叢集的步驟時，您可以取代任何範例值。我們建議您在同一終端中完成所有步驟。這是因為在整個步驟中設定和使用變數，並且不會存在於不同的終端機中。

本主題中的命令使用 AWS CLI 範例中列出的慣例進行格式化。如果您從命令列執行的命令，與您正在使用的 AWS CLI AWS 描述檔中定義的預設 AWS 區域不同，則需要將 --region us-west-2新增至命令，us-west-2以您的 AWS 區域取代 。

如果要將自訂聯網部署到生產叢集，請跳至 the section called “步驟 2：設定 VPC”。

步驟 1：建立測試 VPC 和叢集

下列程序可幫助您建立測試 VPC 和叢集，並為該叢集設定自訂聯網。我們不建議將測試叢集用於生產工作負載，因為本主題未涵蓋您可能在生產叢集上使用的幾個不相關功能。如需詳細資訊，請參閱the section called “建立叢集”。

1. 執行下列命令來定義 account_id變數。

```
account_id=$(aws sts get-caller-identity --query Account --output text)
```

2. 建立 VPC。

   a. 如果您要部署到測試系統，請使用 Amazon EKS AWS CloudFormation 範本建立 VPC。

```
aws cloudformation create-stack --stack-name my-eks-custom-networking-vpc \
  --template-url https://s3.us-west-2.amazonaws.com/amazon-eks/
cloudformation/2020-10-29/amazon-eks-vpc-private-subnets.yaml \
  --parameters ParameterKey=VpcBlock,ParameterValue=192.168.0.0/24 \
  ParameterKey=PrivateSubnet01Block,ParameterValue=192.168.0.64/27 \
  ParameterKey=PrivateSubnet02Block,ParameterValue=192.168.0.96/27 \
```

```
  ParameterKey=PublicSubnet01Block,ParameterValue=192.168.0.0/27 \
  ParameterKey=PublicSubnet02Block,ParameterValue=192.168.0.32/27
```

b. AWS CloudFormation 堆疊需要幾分鐘的時間才能建立。若要檢查堆疊的部署狀態，請執行下列命令。

```
aws cloudformation describe-stacks --stack-name my-eks-custom-networking-vpc --
query Stacks\[\].StackStatus  --output text
```

在命令的輸出為 之前，請勿繼續下一個步驟CREATE_COMPLETE。

c. 使用以範本建立的私有子網 ID 的值來定義變數。

```
subnet_id_1=$(aws cloudformation describe-stack-resources --stack-name my-eks-
custom-networking-vpc \
    --query "StackResources[?
LogicalResourceId=='PrivateSubnet01'].PhysicalResourceId" --output text)
subnet_id_2=$(aws cloudformation describe-stack-resources --stack-name my-eks-
custom-networking-vpc \
    --query "StackResources[?
LogicalResourceId=='PrivateSubnet02'].PhysicalResourceId" --output text)
```

d. 使用上一步中擷取的子網的可用區域來定義變數。

```
az_1=$(aws ec2 describe-subnets --subnet-ids $subnet_id_1 --query
 'Subnets[*].AvailabilityZone' --output text)
az_2=$(aws ec2 describe-subnets --subnet-ids $subnet_id_2 --query
 'Subnets[*].AvailabilityZone' --output text)
```

3. 建立叢集 IAM 角色。

a. 執行下列命令以建立 IAM 信任政策 JSON 檔案。

```
cat >eks-cluster-role-trust-policy.json <<EOF
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "eks.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
```

```
    }
  ]
}
EOF
```

b. 建立 Amazon EKS 叢集 IAM 角色。如有必要，請在 `eks-cluster-role-trust-policy.json` 前面加上您在上一步中寫入檔案的電腦的路徑。命令會將您在上一步驟中建立的信任策略與角色相關聯。若要建立 IAM 角色，必須為建立角色的 IAM 主體指派以下 `iam:CreateRole` 動作 (許可)。

```
aws iam create-role --role-name myCustomNetworkingAmazonEKSClusterRole --assume-role-policy-document file://"eks-cluster-role-trust-policy.json"
```

c. 將名為 AmazonEKSClusterPolicy 的 Amazon EKS 受管政策連接至角色。若要將 IAM 政策連接至 IAM 主體，必須為連接政策的 IAM 實體指派以下 IAM 動作之一 (許可)：`iam:AttachUserPolicy` 或 `iam:AttachRolePolicy`。

```
aws iam attach-role-policy --policy-arn arn:aws: iam::aws:policy/AmazonEKSClusterPolicy --role-name myCustomNetworkingAmazonEKSClusterRole
```

4. 建立 Amazon EKS 叢集，並設定您的裝置以便與其進行通訊。

a. 建立叢集。

```
aws eks create-cluster --name my-custom-networking-cluster \
   --role-arn arn:aws: iam::$account_id:role/myCustomNetworkingAmazonEKSClusterRole \
   --resources-vpc-config subnetIds="$subnet_id_1","$subnet_id_2"
```

> **Note**
>
> 您可能會收到錯誤，告知您請求中的其中一個可用區域沒有足夠的容量來建立 Amazon EKS 叢集。如果發生這種情況，錯誤輸出包含的可用區域可支援新的叢集。使用至少兩個位於帳戶的支援可用區域子網路來建立您的叢集。如需詳細資訊，請參閱the section called “容量不足”。

b. 建立叢集需要幾分鐘才能完成。若要檢查叢集的部署狀態，請執行下列命令。

```
aws eks describe-cluster --name my-custom-networking-cluster --query
 cluster.status
```

在命令的輸出為 之前，請勿繼續下一個步驟"ACTIVE"。

c. 設定 kubectl 以便與叢集通訊。

```
aws eks update-kubeconfig --name my-custom-networking-cluster
```

步驟 2：設定 VPC

本教學課程需要在 the section called “步驟 1：建立測試 VPC 和叢集” 中建立的 VPC。對於生產叢集，透過將所有範例值取代為您自己的值，為您的 VPC 相應地調整步驟。

1. 確認您目前安裝的 Kubernetes 專用 Amazon VPC CNI 外掛程式是最新版本。若要確定 Amazon EKS 附加元件類型的最新版本並更新您的版本，請參閱 the section called “更新 附加元件”。若要確定自我管理附加元件類型的最新版本並更新您的版本，請參閱 the section called “Amazon VPC CNI”。
2. 擷取叢集 VPC 的 ID，並將其存放在變數中以供稍後步驟使用。

```
vpc_id=$(aws eks describe-cluster --name my-custom-networking-cluster --query
 "cluster.resourcesVpcConfig.vpcId" --output text)
```

3. 將額外的無類別網域間路由 (CIDR) 區塊與叢集的 VPC 建立關聯。CIDR 區塊不能與任何現有的關聯 CIDR 區塊重疊。

a. 檢視目前與您的 VPC 關聯的 CIDR 區塊。

```
aws ec2 describe-vpcs --vpc-ids $vpc_id \
    --query 'Vpcs[*].CidrBlockAssociationSet[*].{CIDRBlock: CidrBlock, State:
 CidrBlockState.State}' --out table
```

範例輸出如下。

```
----------------------------------
|          DescribeVpcs          |
+-----------------+--------------+
|    CIDRBlock    |    State     |
+-----------------+--------------+
|  192.168.0.0/24 |  associated  |
+-----------------+--------------+
```

b. 將其他 CIDR 區塊與 VPC 建立關聯。在下列命令中取代 CIDR 區塊值。如需詳細資訊，請參閱《Amazon VPC 使用者指南》中的將其他 IPv4 CIDR 區塊與 VPC 建立關聯。

```
aws ec2 associate-vpc-cidr-block --vpc-id $vpc_id --cidr-block 192.168.1.0/24
```

c. 確認新區塊已關聯。

```
aws ec2 describe-vpcs --vpc-ids $vpc_id --query
 'Vpcs[*].CidrBlockAssociationSet[*].{CIDRBlock: CidrBlock, State:
 CidrBlockState.State}' --out table
```

範例輸出如下。

```
----------------------------------
|          DescribeVpcs          |
+-----------------+--------------+
|    CIDRBlock    |    State     |
+-----------------+--------------+
|  192.168.0.0/24 |  associated  |
|  192.168.1.0/24 |  associated  |
+-----------------+--------------+
```

在新 CIDR 區塊的 State為 之前，請勿繼續進行下一個步驟associated。

4. 在現有子網所在的每個可用區域中建立任意數量的子網。指定 CIDR 區塊，該區塊位於您在上一個步驟中與 VPC 相關聯的 CIDR 區塊內。

a. 建立新子網。在下列命令中取代 CIDR 區塊值。子網必須在不同於現有子網所在的 VPC CIDR 區塊中建立，但與現有子網位於相同的可用區域中。在此範例中，會在目前私有子網所在的每個可用區域的新 CIDR 區塊中，建立一個子網。建立的子網 ID 會儲存在變數中，以便在稍後步驟中使用。Name 值與指派給在上一步中使用 Amazon EKS VPC 範本建立的子網的值相符。不需要名稱。您可以使用不同的名稱。

```
new_subnet_id_1=$(aws ec2 create-subnet --vpc-id $vpc_id --availability-zone $az_1
 --cidr-block 192.168.1.0/27 \
    --tag-specifications 'ResourceType=subnet,Tags=[{Key=Name,Value=my-eks-custom-
networking-vpc-PrivateSubnet01},{Key=kubernetes.io/role/internal-elb,Value=1}]' \
    --query Subnet.SubnetId --output text)
new_subnet_id_2=$(aws ec2 create-subnet --vpc-id $vpc_id --availability-zone $az_2
 --cidr-block 192.168.1.32/27 \
```

```
    --tag-specifications 'ResourceType=subnet,Tags=[{Key=Name,Value=my-eks-custom-
networking-vpc-PrivateSubnet02},{Key=kubernetes.io/role/internal-elb,Value=1}]' \
    --query Subnet.SubnetId --output text)
```

> **Important**
>
> 根據預設，您的新子網路會隱含地與 VPC 的主要路由表相關聯。此路由表允許在 VPC 中部署的所有資源之間進行通訊。不過，它不允許與 IP 地址位於與您 VPC 相關聯 CIDR 區塊之外的資源進行通訊。您可以將自己的路由表與子網相關聯以變更此行為。如需詳細資訊，請參閱《Amazon VPC 使用者指南》中的子網由表。

b. 檢視 VPC 中的目前子網。

```
aws ec2 describe-subnets --filters "Name=vpc-id,Values=$vpc_id" \
    --query 'Subnets[*].{SubnetId: SubnetId,AvailabilityZone:
 AvailabilityZone,CidrBlock: CidrBlock}' \
    --output table
```

範例輸出如下。

```
-------------------------------------------------------------------
|                          DescribeSubnets                        |
+------------------+--------------------+---------------------------+
| AvailabilityZone |     CidrBlock      |         SubnetId          |
+------------------+--------------------+---------------------------+
|  us-west-2d      |  192.168.0.0/27    |     subnet-example1       |
|  us-west-2a      |  192.168.0.32/27   |     subnet-example2       |
|  us-west-2a      |  192.168.0.64/27   |     subnet-example3       |
|  us-west-2d      |  192.168.0.96/27   |     subnet-example4       |
|  us-west-2a      |  192.168.1.0/27    |     subnet-example5       |
|  us-west-2d      |  192.168.1.32/27   |     subnet-example6       |
+------------------+--------------------+---------------------------+
```

您可以看到您建立的 192.168.1.0 CIDR 區塊中的子網，與 192.168.0.0 CIDR 區塊中的子網位於相同的可用區域中。

步驟 3：設定 Kubernetes 資源

1. 在 aws-node DaemonSet `true`中將`AWS_VPC_K8S_CNI_CUSTOM_NETWORK_CFG`環境變數設定為 。

```
kubectl set env daemonset aws-node -n kube-system
 AWS_VPC_K8S_CNI_CUSTOM_NETWORK_CFG=true
```

2. 擷取叢集安全群組的 ID，並將其存放在變數中以供下一步驟使用。在您建立叢集時，Amazon EKS 會自動建立此安全群組。

```
cluster_security_group_id=$(aws eks describe-cluster --name my-custom-networking-
cluster --query cluster.resourcesVpcConfig.clusterSecurityGroupId --output text)
```

3. 為您要部署 Pod 的每個子網路建立`ENIConfig`自訂資源。
   a. 為每個網路介面組態建立唯一的檔案。

   以下命令為在上一步中建立的兩個子網建立單獨的 ENIConfig 檔案。name 的值必須是唯一的值。此名稱與子網所在的可用區域相同。叢集安全群組指派給 ENIConfig。

```
cat >$az_1.yaml <<EOF
apiVersion: crd.k8s.amazonaws.com/v1alpha1
kind: ENIConfig
metadata:
  name: $az_1
spec:
  securityGroups:
    - $cluster_security_group_id
  subnet: $new_subnet_id_1
EOF
```

```
cat >$az_2.yaml <<EOF
apiVersion: crd.k8s.amazonaws.com/v1alpha1
kind: ENIConfig
metadata:
  name: $az_2
spec:
  securityGroups:
    - $cluster_security_group_id
  subnet: $new_subnet_id_2
EOF
```

對於生產叢集，您可以對前述命令進行以下變更：

- 以您要用於每個 的現有安全群組 ID 取代 $cluster_security_group_idENIConfig。
- 我們建議您盡可能將 ENIConfigs 命名為與您將用於 ENIConfig的可用區域相同的 。由於各種原因，您可能需要為您的 ENIConfigs 使用與可用區域名稱不同的名稱。例如，如果您在同一可用區域中有兩個以上的子網，並且希望將其與自訂聯網一起使用，則需要多個 ENIConfigs 用於相同的可用區域。由於每個 ENIConfig都需要唯一的名稱，因此您無法使用可用區域名稱ENIConfigs來命名多個 。

  如果您的ENIConfig名稱與可用區域名稱不同，請在先前的命令中將 $az_1 和 $az_2 取代為您自己的名稱，並在本教學稍後的 ENIConfig 中註釋您的節點。

  > **Note**
  >
  > 如果您未指定用於生產叢集的有效安全群組，且正在使用：

- 適用於 Kubernetes 的 Amazon VPC CNI 外掛程式版本 1.8.0或更新版本，則會使用與節點主要彈性網路界面相關聯的安全群組。
- 適用於 Kubernetes 的 Amazon VPC CNI 外掛程式版本早於 1.8.0，則 VPC 的預設安全群組會指派給次要網路介面。

> **Important**
>
> - AWS_VPC_K8S_CNI_EXTERNALSNAT=false 是適用於 Kubernetes 的 Amazon VPC CNI 外掛程式組態中的預設設定。如果您使用的是預設設定，則目的地不在與 VPC 相關聯之其中一個 CIDR 區塊內的 IP 地址流量會使用節點主要網路界面的安全群組和子網路。您 中定義用來建立次要網路介面ENIConfigs的子網路和安全群組不會用於此流量。如需有關此設定的詳細資訊，請參閱 the section called “傳出流量”。
> - 如果您也使用 Pod 的安全群組，SecurityGroupPolicy則會使用 中指定的安全群組，而不是 中指定的安全群組ENIConfigs。如需詳細資訊，請參閱the section called “Pod 的安全群組”。

b. 使用下列命令，將您建立的每個自訂資源檔案套用到叢集。

```
kubectl apply -f $az_1.yaml
kubectl apply -f $az_2.yaml
```

4. 確認您的 ENIConfigs 已建立。

```
kubectl get ENIConfigs
```

範例輸出如下。

```
NAME         AGE
us-west-2a   117s
us-west-2d   105s
```

5. 如果您在生產叢集上啟用自訂聯網，並將您的 命名為您使用它們的可用區域以外的ENIConfigs名稱，請跳到下一個步驟來部署 Amazon EC2 節點。

   啟用 Kubernetes 以將可用區域的 ENIConfig 自動套用至叢集中建立的任何新 Amazon EC2 節點。

   a. 對於本教學課程中的測試叢集，請跳至下一步。

      對於生產叢集，請檢查 aws-node DaemonSet 的容器規格中是否存在具有 ENI_CONFIG_ANNOTATION_DEF 環境變數 k8s.amazonaws.com/eniConfig 金鑰的註釋。

      ```
      kubectl describe daemonset aws-node -n kube-system | grep
       ENI_CONFIG_ANNOTATION_DEF
      ```

      如果傳回輸出，則表示註釋已存在。如果未傳回輸出，表示變數尚未設定。對於生產叢集，您可以使用此設定或以下步驟中的設定。如果使用此設定，將會覆寫以下步驟中的設定。在本教學課程中，將使用下一步中的設定。

   b. 更新您的 aws-node DaemonSet，以自動ENIConfig將可用區域的 套用至叢集中建立的任何新 Amazon EC2 節點。

      ```
      kubectl set env daemonset aws-node -n kube-system
       ENI_CONFIG_LABEL_DEF=topology.kubernetes.io/zone
      ```

步驟 4：部署 Amazon EC2 節點

1. 建立節點 IAM 角色。

   a. 執行下列命令以建立 IAM 信任政策 JSON 檔案。

      ```
      cat >node-role-trust-relationship.json <<EOF
      {
      ```

```
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "ec2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF
```

b. 建立 IAM 角色，並將其傳回的 Amazon Resource Name (ARN) 存放在變數中，以供後續步驟使用。

```
node_role_arn=$(aws iam create-role --role-name myCustomNetworkingNodeRole --
assume-role-policy-document file://"node-role-trust-relationship.json" \
    --query Role.Arn --output text)
```

c. 將三個所需的 IAM 受管政策連接到 IAM 角色。

```
aws iam attach-role-policy \
  --policy-arn arn:aws: iam::aws:policy/AmazonEKSWorkerNodePolicy \
  --role-name myCustomNetworkingNodeRole
aws iam attach-role-policy \
  --policy-arn arn:aws: iam::aws:policy/AmazonEC2ContainerRegistryReadOnly \
  --role-name myCustomNetworkingNodeRole
aws iam attach-role-policy \
    --policy-arn arn:aws: iam::aws:policy/AmazonEKS_CNI_Policy \
    --role-name myCustomNetworkingNodeRole
```

> **Important**
>
> 為了簡化本教學課程，AmazonEKS_CNI_Policy 政策會連接至節點 IAM 角色。不過，在生產叢集中，我們建議將政策連接至單獨的 IAM 角色，該角色僅適用於 Kubernetes 的 Amazon VPC CNI 外掛程式。如需詳細資訊，請參閱the section called “為 IRSA 設定”。

2. 建立下列其中一種節點群組類型。若要確定您想要部署的執行個體類型，請參閱 the section called “Amazon EC2 執行個體類型”。在本教學課程中，請完成受管、沒有啟動範本或未指定 AMI ID 的啟

動範本選項。如果您要將節點群組用於生產工作負載，建議您先熟悉所有受管節點群組和自我管理節點群組選項，再部署節點群組。

- 受管：使用下列其中一個選項，部署節點群組：
  - 沒有啟動範本，或有啟動範本，但沒有指定 AMI ID：執行下列命令。對於本教學課程，請使用範例值。對於生產節點群組，請將所有範例值取代為您自己的值。節點群組名稱不能超過 63 個字元。它必須以字母或數字開頭，但剩餘字元也可以包含連字符和底線。

```
aws eks create-nodegroup --cluster-name my-custom-networking-cluster --nodegroup-
name my-nodegroup \
    --subnets $subnet_id_1 $subnet_id_2 --instance-types t3.medium --node-role
 $node_role_arn
```

  - 使用具有指定 AMI ID 的啟動範本

    A. 判斷節點的 Amazon EKS 建議 Pod 數量上限。針對每個 Amazon EC2 執行個體類型，請遵循 Amazon EKS 建議的最大 Pod 中的指示，將 `--cni-custom-networking-enabled`新增至該主題中的步驟 3。請記下輸出，以便在下一個步驟中使用。

    B. 在啟動範本中，指定 Amazon EKS 最佳化 AMI ID 或基於 Amazon EKS 最佳化 AMI 的自訂 AMI，然後使用啟動範本部署節點群組，並在啟動範本中提供下列使用者資料。此使用者資料會將引數傳遞至 `bootstrap.sh` 檔案。如需引導檔案的詳細資訊，請參閱 GitHub 上的 bootstrap.sh。您可將 `20` 取代為上一個步驟的值 (建議) 或您自己的值。

```
/etc/eks/bootstrap.sh my-custom-networking-cluster --use-max-pods false --
kubelet-extra-args '--max-pods=20'
```

    如果您建立的自訂 AMI 並非以 Amazon EKS 最佳化 AMI 為基礎，則需要自行自訂建立組態。

- 自我管理

  i. 判斷節點的 Amazon EKS 建議 Pod 數量上限。請遵循 the section called “Amazon EKS 為每種 Amazon EC2 執行個體類型建議 Pod 數量上限” 中的指示，將 `--cni-custom-networking-enabled` 新增至該主題的步驟 3。請記下輸出，以便在下一個步驟中使用。

  ii. 使用 the section called “Amazon Linux” 中的指示部署節點群組。為 BootstrapArguments 參數指定以下文字。您可將 `20` 取代為上一個步驟的值 (建議) 或您自己的值。

```
--use-max-pods false --kubelet-extra-args '--max-pods=20'
```

> **Note**
>
> 如果您希望生產叢集中的節點支援大量 Pod，請the section called “Amazon EKS 為每種 Amazon EC2 執行個體類型建議 Pod 數量上限”再次在 中執行指令碼。此外，將 `--cni-prefix-delegation-enabled` 選項新增制命令中。例如：`m5.large` 執行個體類型會傳回 `110`。如需有關如何啟用此功能的說明，請參閱 the section called “增加 IP 地址”。您可將此功能與自訂聯網搭配使用。

3. 建立節點群組需要幾分鐘的時間。您可以使用以下命令檢查受管節點群組的建立狀態。

```
aws eks describe-nodegroup --cluster-name my-custom-networking-cluster --nodegroup-
name my-nodegroup --query nodegroup.status --output text
```

   在傳回的輸出為 之前，請勿繼續下一個步驟`ACTIVE`。

4. 針對本教學課程，您可以略過此步驟。

   對於生產叢集，如果您未將 命名為與正在使用的可用區域`ENIConfigs`相同，則必須使用應與節點搭配使用`ENIConfig`的名稱來註釋節點。如果您在每個可用區域中只有一個子網路，而且以與可用區域相同的名稱命名您的 `ENIConfigs`，則不需要此步驟。這是因為當您在上一個步驟中啟用正確時，適用於 Kubernetes 的 Amazon VPC CNI 外掛程式會自動將其`ENIConfig`與節點建立關聯。

   a. 取得您叢集中的節點清單。

```
kubectl get nodes
```

      範例輸出如下。

```
NAME                                          STATUS   ROLES    AGE     VERSION
ip-192-168-0-126.us-west-2.compute.internal   Ready    <none>   8m49s   v1.22.9-
eks-810597c
ip-192-168-0-92.us-west-2.compute.internal    Ready    <none>   8m34s   v1.22.9-
eks-810597c
```

   b. 確定每個節點所在的可用區域。針對上一個步驟中傳回的每個節點執行下列命令，根據上一個輸出取代 IP 地址。

```
aws ec2 describe-instances --filters Name=network-interface.private-dns-
name,Values=ip-192-168-0-126.us-west-2.compute.internal \
```

```
--query 'Reservations[].Instances[].{AvailabilityZone: Placement.AvailabilityZone,
 SubnetId: SubnetId}'
```

範例輸出如下。

```
[
    {
        "AvailabilityZone": "us-west-2d",
        "SubnetId": "subnet-Example5"
    }
]
```

c. 使用您為子網 ID 和可用區域建立的 ENIConfig 註釋每個節點。您只能使用一個 ENIConfig 註釋一個節點，但可以使用相同的 ENIConfig 註釋多個節點。使用自己的取代範例值。

```
kubectl annotate node ip-192-168-0-126.us-west-2.compute.internal
 k8s.amazonaws.com/eniConfig=EniConfigName1
kubectl annotate node ip-192-168-0-92.us-west-2.compute.internal
 k8s.amazonaws.com/eniConfig=EniConfigName2
```

5. 如果您在使用自訂聯網功能切換到 之前，生產叢集中有執行 Pod 的節點，請完成下列任務：

a. 確保您具有使用自訂聯網功能的可用節點。

b. 繫結並耗盡節點以正常關閉 Pod。如需詳細資訊，請參閱 Kubernetes 文件中的安全地耗盡節點。

c. 終止節點。如果節點位於現有的受管節點群組中，則可以刪除該節點群組。執行下列命令。

```
aws eks delete-nodegroup --cluster-name my-custom-networking-cluster --nodegroup-
name my-nodegroup
```

只有使用 k8s.amazonaws.com/eniConfig 標籤註冊的新節點能使用自訂聯網功能。

6. 確認已從與您在上一個步驟中建立的其中一個子網路相關聯的 CIDR 區塊指派 IP 地址給 Pod。

```
kubectl get pods -A -o wide
```

範例輸出如下。

```
NAMESPACE     NAME                          READY   STATUS    RESTARTS   AGE     IP
           NODE                                          NOMINATED NODE   READINESS
 GATES
```

```
kube-system   aws-node-2rkn4             1/1     Running   0          7m19s
 192.168.0.92    ip-192-168-0-92.us-west-2.compute.internal    <none>
 <none>
kube-system   aws-node-k96wp             1/1     Running   0          7m15s
 192.168.0.126   ip-192-168-0-126.us-west-2.compute.internal   <none>
 <none>
kube-system   coredns-657694c6f4-smcgr   1/1     Running   0          56m
 192.168.1.23    ip-192-168-0-92.us-west-2.compute.internal    <none>
 <none>
kube-system   coredns-657694c6f4-stwv9   1/1     Running   0          56m
 192.168.1.28    ip-192-168-0-92.us-west-2.compute.internal    <none>
 <none>
kube-system   kube-proxy-jgshq           1/1     Running   0          7m19s
 192.168.0.92    ip-192-168-0-92.us-west-2.compute.internal    <none>
 <none>
kube-system   kube-proxy-wx9vk           1/1     Running   0          7m15s
 192.168.0.126   ip-192-168-0-126.us-west-2.compute.internal   <none>
 <none>
```

您可以看到核心 Pod 是從您新增至 VPC 的 192.168.1.0 CIDR 區塊指派的 IP 地址。如果沒有自訂網路，他們將被指派來自 192.168.0.0 CIDR 區塊的地址，因為其是最初與 VPC 關聯的唯一 CIDR 區塊。

如果 Pod spec包含 hostNetwork=true，則會指派節點的主要 IP 地址。它不會從您新增的子網路指派地址。依預設，此值是設為 false。此值true針對叢集上執行的 Kubernetes (aws-node) Pod 的 kube-proxy和 Amazon VPC CNI 外掛程式設為 。這就是為什麼 kube-proxy和外掛程式的 aws-node Pod 在先前的輸出中未指派 192.168.1.x 地址的原因。如需 Pod hostNetwork設定的詳細資訊，請參閱 Kubernetes API 參考中的 PodSpec v1 核心。

步驟 5：刪除教學課程資源

完成本教學課程後，建議您刪除您建立的資源。然後，您可以調整步驟以啟用生產叢集的自訂聯網。

1. 如果您建立的節點群組僅用於測試，請將其刪除。

```
aws eks delete-nodegroup --cluster-name my-custom-networking-cluster --nodegroup-name
 my-nodegroup
```

2. 即使 CLI AWS 輸出顯示叢集已刪除後，刪除程序仍可能實際上未完成。刪除程序需要幾分鐘。執行下列命令以確認已完成。

```
aws eks describe-nodegroup --cluster-name my-custom-networking-cluster --nodegroup-
name my-nodegroup --query nodegroup.status --output text
```

在傳回的輸出類似於下列輸出之前，請勿繼續。

```
An error occurred (ResourceNotFoundException) when calling the DescribeNodegroup
 operation: No node group found for name: my-nodegroup.
```

3. 如果您建立的節點群組僅用於測試，則請刪除節點 IAM 角色。

   a. 將策略與角色分離。

   ```
   aws iam detach-role-policy --role-name myCustomNetworkingNodeRole --policy-arn
    arn:aws: iam::aws:policy/AmazonEKSWorkerNodePolicy
   aws iam detach-role-policy --role-name myCustomNetworkingNodeRole --policy-arn
    arn:aws: iam::aws:policy/AmazonEC2ContainerRegistryReadOnly
   aws iam detach-role-policy --role-name myCustomNetworkingNodeRole --policy-arn
    arn:aws: iam::aws:policy/AmazonEKS_CNI_Policy
   ```

   b. 刪除角色。

   ```
   aws iam delete-role --role-name myCustomNetworkingNodeRole
   ```

4. 刪除叢集。

```
aws eks delete-cluster --name my-custom-networking-cluster
```

請使用下列命令來確認刪除叢集。

```
aws eks describe-cluster --name my-custom-networking-cluster --query cluster.status
 --output text
```

如果傳回類似以下的輸出，則已成功刪除叢集。

```
An error occurred (ResourceNotFoundException) when calling the DescribeCluster
 operation: No cluster found for name: my-custom-networking-cluster.
```

5. 刪除叢集 IAM 角色。

   a. 將策略與角色分離。

```
aws iam detach-role-policy --role-name myCustomNetworkingAmazonEKSClusterRole --
policy-arn arn:aws: iam::aws:policy/AmazonEKSClusterPolicy
```

b. 刪除角色。

```
aws iam delete-role --role-name myCustomNetworkingAmazonEKSClusterRole
```

6. 刪除您在上一個步驟中建立的子網。

```
aws ec2 delete-subnet --subnet-id $new_subnet_id_1
aws ec2 delete-subnet --subnet-id $new_subnet_id_2
```

7. 刪除您建立的 VPC。

```
aws cloudformation delete-stack --stack-name my-eks-custom-networking-vpc
```

使用字首將更多 IP 地址指派給 Amazon EKS 節點

適用於：具有 Amazon EC2 執行個體的 Linux 和 Windows 節點

適用於：公有和私有子網路

每個 Amazon EC2 執行個體都支援最大數量的彈性網路介面，以及可指派給每個網路介面的最大數量的 IP 地址。每個節點的每個網路介面都需要一個 IP 地址。所有其他可用的 IP 地址都可以指派給 Pods。每個 Pod 都需要專屬的 IP 地址。因此，您可能有具有可用運算和記憶體資源的節點，但無法容納其他 ，Pods因為節點已用完要指派給 的 IP 地址Pods。

您可以透過指派 IP 字首Pods來增加節點可指派的 IP 地址數量，而不是將個別次要 IP 地址指派給節點。每個字首都包含多個 IP 地址。如果您未設定叢集進行 IP 字首指派，您的叢集必須進行更多 Amazon EC2 應用程式設計介面 (API) 呼叫，以設定 Pod 連線所需的網路介面和 IP 地址。隨著叢集大小的增加，這些 API 呼叫的頻率可能會導致更長的 Pod 和執行個體啟動時間。這樣一來，會導致擴展延遲以滿足大型和高峰工作負載的需求，並增加成本和管理負荷，因為您需要佈建額外的叢集和 VPC 以符合擴展需求。如需詳細資訊，請參閱 GitHub 上的 Kubernetes 可擴展性閾值。

與適用於 Kubernetes 功能的 Amazon VPC CNI 外掛程式的相容性

您可以將 IP 字首與下列功能搭配使用：

- IPv4 來源網路地址轉譯 - 如需詳細資訊，請參閱 the section called “傳出流量”。
- 叢集、Pod 和服務的 IPv6 地址 - 如需詳細資訊，請參閱 the section called “IPv6”。

- 使用 Kubernetes 網路政策限制流量 - 如需詳細資訊，請參閱 the section called “Kubernetes 政策”。

下列清單提供適用的 Amazon VPC CNI 外掛程式設定相關資訊。如需每個設定的詳細資訊，請參閱 GitHub 上的 amazon-vpc-cni-k8s。

- `WARM_IP_TARGET`
- `MINIMUM_IP_TARGET`
- `WARM_PREFIX_TARGET`

考量事項

使用此功能時，請考慮下列事項：

- 每個 Amazon EC2 執行個體類型都支援最大數量的 Pod。如果您的受管節點群組包含多個執行個體類型，則叢集中執行個體的最小 Pod 上限會套用至叢集中的所有節點。
- 依預設，您最多可在節點上執行 110 個 Pods，但您可以變更該數量。如果您變更該數量且擁有現有受管節點群組，則節點群組的下一個 AMI 或啟動範本更新會導致新的工作節點出現變更後的值。
- 從指派 IP 地址轉換為指派 IP 字首時，建議您建立新的節點群組來增加可用 IP 地址的數量，而不是對現有節點執行輪流取代。在同時指派 IP 地址和字首的節點上執行 Pod 可能會導致公告 IP 地址容量不一致，進而影響節點上的未來工作負載。如需執行轉換的建議方式，請參閱《Amazon EKS 最佳實務指南》中的在次要 IP 模式與字首委派模式相互遷移期間取代所有節點。
- 安全群組範圍位於節點層級 - 如需詳細資訊，請參閱安全群組。
- 指派給網路介面的 IP 字首支援每個節點的高 Pod 密度，並具有最佳的啟動時間。
- IP 字首和 IP 地址與標準 Amazon EC2 彈性網路介面相關聯。需要特定安全群組的 Pod 會被指派分支網路介面的主要 IP 地址。您可以混合取得 IP 地址的 Pod，或來自 IP 字首的 IP 地址與取得相同節點上分支網路介面的 Pod。
- 僅適用於具有 Linux 節點的叢集。
  - 設定附加元件以將字首指派給網路介面後，您無法將 Kubernetes 的 Amazon VPC CNI 外掛程式降級為低於 `1.9.0` (或 `1.10.1`) 的版本，而無需移除叢集中所有節點群組中的所有節點。
  - 如果您也使用 Pod 的安全群組，其中 `POD_SECURITY_GROUP_ENFORCING_MODE=standard` 和 `AWS_VPC_K8S_CNI_EXTERNALSNAT=false`，當您的 Pod 與 VPC 外部的端點通訊時，會使用節點的安全群組，而不是您指派給 Pod 的任何安全群組。

    如果您也使用 Pod 的安全群組，且 `POD_SECURITY_GROUP_ENFORCING_MODE=` 為 `strict`，當您與 VPC 外部的端點Pods通訊時，會使用Pod’s安全群組。

增加 Amazon EKS 節點的可用 IP 地址

您可以透過指派 IP 字首來增加節點可指派給 Pod 的 IP 地址數量，而不是將個別次要 IP 地址指派給節點。

先決條件

- 您需要現有的叢集。若要部署叢集，請參閱 the section called “建立叢集”。
- Amazon EKS 節點所在的子網路必須具有足夠的連續 /28 (針對 IPv4 叢集) 或 /80 (針對 IPv6 叢集) 無類別域間路由 (CIDR) 區塊。IPv6 叢集中只能包含 Linux 節點。如果 IP 地址分散在子網路 CIDR 中，則使用 IP 字首可能會失敗。我們建議下列作法：
  - 使用子網路 CIDR 保留，即使保留範圍內的任何 IP 地址仍在使用中，也不會在其發行時重新指派 IP 地址。此舉可確保字首無需分割，即可用於配置。
  - 使用專門用於執行指派 IP 字首之工作負載的新子網路。指派 IP 字首時，Windows 和 Linux 工作負載都可以在相同的子網路中執行。
- 若要將 IP 字首指派給節點，您的節點必須是 AWS Nitro 型。非 Nitro 型執行個體會繼續配置個別次要 IP 地址，但指派給 Pod 的 IP 地址數量遠低於 Nitro 型執行個體。
- 僅適用於具有 Linux 節點的叢集 – 如果您的叢集已針對 IPv4 系列設定，您必須安裝適用於 Kubernetes 附加元件的 Amazon VPC CNI 外掛程式版本 1.9.0或更新版本。您可以使用下列命令來檢查目前版本：

```
kubectl describe daemonset aws-node --namespace kube-system | grep Image | cut -d "/"
 -f 2
```

  如果您的叢集針對 IPv6 系列設定，則必須安裝附加元件的版本 1.10.1。如果您的外掛程式版本早於所需版本，則必須對其進行更新。如需詳細資訊，請參閱使用 Amazon VPC CNI 將 IPs 指派給 Pod 的更新章節。
- 僅適用於具有 Windows 節點的叢集
  - 您的叢集及其平台版本必須與下表中的版本相同或為更新版本。若要升級叢集版本，請參閱 the section called “更新 Kubernetes 版本”。如果您的叢集不是最低平台版本，則在 Amazon EKS 更新您的平台版本之前，您無法將 IP 字首指派給節點。

Kubernetes 版本	平台版本
1.27	eks.3

Kubernetes 版本	平台版本
1.26	eks.4

您可以檢查目前的 Kubernetes 和平台版本，方法是將下列命令中的 *my-cluster* 取代為您的叢集名稱，然後執行修改後的命令：`aws eks describe-cluster --name my-cluster --query 'cluster.{"Kubernetes Version": version, "Platform Version": platformVersion}'`。

- 您必須為叢集啟用 Windows 支援。如需詳細資訊，請參閱the section called “啟用 Windows 支援”。

將 IP 地址字首指派給節點

設定叢集以將 IP 地址字首指派給節點。完成符合您節點作業系統的程序。

Linux

1. 啟用 參數，將字首指派給 Amazon VPC CNI DaemonSet 的網路介面。當您部署叢集時，適用於 Kubernetes 附加元件的 Amazon VPC CNI 外掛程式版本 `1.10.1` 或更新版本會與其一起部署。如果您使用 IPv6 系列建立叢集，則此設定會預設設為 `true`。如果您使用 IPv4 系列建立叢集，則此設定會預設設為 `false`。

```
kubectl set env daemonset aws-node -n kube-system ENABLE_PREFIX_DELEGATION=true
```

> **Important**
>
> 即使您的子網路有可用的 IP 地址，如果子網路沒有任何連續的可用/28區塊，您會在 Kubernetes 日誌的 Amazon VPC CNI 外掛程式中看到下列錯誤。
>
> ```
> InsufficientCidrBlocks: The specified subnet does not have enough free cidr
>  blocks to satisfy the request
> ```
>
> 這可能是由分散在子網路上的現有次要 IP 地址的分段造成。若要解決此錯誤，請建立新的子網路並在該處啟動 Pod，或使用 Amazon EC2 子網路 CIDR 保留來保留子網路內的空間，以便與字首指派搭配使用。如需詳細資訊，請參閱《Amazon VPC 使用者指南》中的子網路 CIDR 保留。

2. 如果您計劃在沒有啟動範本的情況下，或使用尚未指定 AMI ID 的啟動範本來部署受管節點群組，而且您正在使用 Kubernetes 專用 Amazon VPC CNI 外掛程式的版本，或比先決條件中列出的版本更高，請跳到下一個步驟。受管節點群組會自動為您計算 Pod 的數量上限。

   如果您要使用已在其中指定 AMI ID 的啟動範本來部署自我管理節點群組或受管節點群組，則必須判斷節點的 Amazon EKS 建議最大 Pod 數量。針對每個 Amazon EC2 執行個體類型，請遵循 Amazon EKS 建議的最大 Pod 中的指示，將 `--cni-prefix-delegation-enabled`新增至步驟 3。請記下輸出，以便在稍後步驟中使用。

   > **Important**
   >
   > 受管節點群組會強制對 `maxPods` 的值執行數量上限。對於少於 30 個 vCPU 的執行個體，數量上限為 110；對於所有其他執行個體，數量上限為 250。無論是否啟用字首委派，都會套用此數量上限。

3. 如果您使用的是為 設定的叢集IPv6，請跳至下一個步驟。

   在下列其中一個選項中指定參數。若要判斷哪個選項適合您，以及提供哪個值，請參閱 GitHub 上的 WARM_PREFIX_TARGET、WARM_IP_TARGET 和 MINIMUM_IP_TARGET。

   您可以用大於零的值取代 *example values*。

   - `WARM_PREFIX_TARGET`

     ```
     kubectl set env ds aws-node -n kube-system WARM_PREFIX_TARGET=1
     ```

   - `WARM_IP_TARGET` 或 `MINIMUM_IP_TARGET`：如果此值已設定，則其會覆寫為 `WARM_PREFIX_TARGET` 設定的任何值。

     ```
     kubectl set env ds aws-node -n kube-system WARM_IP_TARGET=5
     ```

     ```
     kubectl set env ds aws-node -n kube-system MINIMUM_IP_TARGET=2
     ```

4. 使用至少一個 Amazon EC2 Nitro Amazon Linux 2 執行個體類型，建立下列類型的節點群組之一。如需 Nitro 執行個體類型的清單，請參閱《Amazon EC2 使用者指南》中的在 Nitro 系統上建置的執行個體。Windows 不支援這項功能。對於包含 *110* 的選項，使用步驟 3 的值 (建議) 或您自己的值將其取代。

   - 自我管理 – 使用建立自我管理 Amazon Linux 節點中的指示來部署節點群組。為 BootstrapArguments 參數指定以下文字。

```
--use-max-pods false --kubelet-extra-args '--max-pods=110'
```

如果您使用 eksctl來建立節點群組，您可以使用下列命令。

```
eksctl create nodegroup --cluster my-cluster --managed=false --max-pods-per-node
 110
```

- 受管：使用下列其中一個選項，部署節點群組：
  - 沒有啟動範本或沒有指定 AMI ID 的啟動範本 – 完成為叢集建立受管節點群組中的程序。受管節點群組會自動為您計算 Amazon EKS 建議的 max-pods 值。
  - 使用具有指定 AMI ID 的啟動範本：在啟動範本中，指定 Amazon EKS 最佳化 AMI ID 或基於 Amazon EKS 最佳化 AMI 的自訂 AMI，然後使用啟動範本部署節點群組，並在啟動範本中提供下列使用者資料。此使用者資料會將引數傳遞至 bootstrap.sh 檔案。如需引導檔案的詳細資訊，請參閱 GitHub 上的 bootstrap.sh。

    ```
    /etc/eks/bootstrap.sh my-cluster \
      --use-max-pods false \
      --kubelet-extra-args '--max-pods=110'
    ```

    如果您使用 eksctl來建立節點群組，您可以使用下列命令。

    ```
    eksctl create nodegroup --cluster my-cluster --max-pods-per-node 110
    ```

    如果您建立的自訂 AMI 不是以 Amazon EKS 最佳化 AMI 為基礎，則需要自行自訂建立組態。

> **Note**
>
> 如果您也想要從與執行個體不同的子網路將 IP 地址指派給 Pod，則需要在此步驟中啟用 功能。如需詳細資訊，請參閱the section called “自訂聯網”。

Windows

1. 啟用 IP 字首指派。
   a. 開啟 amazon-vpc-cni ConfigMap 進行編輯。

```
kubectl edit configmap -n kube-system amazon-vpc-cni -o yaml
```

b. 將下行新增至 data 區段：

```
  enable-windows-prefix-delegation: "true"
```

c. 儲存檔案並關閉編輯器。

d. 確認行已新增至 ConfigMap。

```
kubectl get configmap -n kube-system amazon-vpc-cni -o "jsonpath={.data.enable-
windows-prefix-delegation}"
```

如果傳回的輸出不是 true，則可能發生錯誤。請嘗試再次完成該步驟。

> **Important**
>
> 即使您的子網路有可用的 IP 地址，如果子網路沒有任何連續的可用/28區塊，您會在 Kubernetes 日誌的 Amazon VPC CNI 外掛程式中看到下列錯誤。
>
> ```
> InsufficientCidrBlocks: The specified subnet does not have enough free cidr
>  blocks to satisfy the request
> ```
>
> 這可能是由分散在子網路上的現有次要 IP 地址的分段造成。若要解決此錯誤，請建立新的子網路並在該處啟動 Pod，或使用 Amazon EC2 子網路 CIDR 保留來保留子網路內的空間，以便與字首指派搭配使用。如需詳細資訊，請參閱《Amazon VPC 使用者指南》中的子網路 CIDR 保留。

2. (選用) 指定其他組態來控制叢集的預先擴展和動態擴展行為。如需詳細資訊，請參閱 GitHub 上在 Windows 上使用字首委派模式的組態選項。

a. 開啟 amazon-vpc-cni ConfigMap 進行編輯。

```
kubectl edit configmap -n kube-system amazon-vpc-cni -o yaml
```

b. 將###取代為大於零的值，並將您需要的項目新增至 的 data區段ConfigMap。如果您為 warm-ip-target 或 minimum-ip-target 設定了值，則該值會覆寫任何為 warm-prefix-target 設定的值。

```
  warm-prefix-target: "1"
```

```
  warm-ip-target: "5"
  minimum-ip-target: "2"
```

c. 儲存檔案並關閉編輯器。

3. 建立至少具有一種 Amazon EC2 Nitro 執行個體類型的 Windows 節點群組。如需 Nitro 執行個體類型的清單，請參閱《Amazon EC2 使用者指南》中的在 Nitro 系統上建置的執行個體。根據預設，您可以部署到節點的 Pod 數量上限為 110。如果您要增加或減少該數量，請在引導組態的使用者資料中指定以下內容：將 *max-pods-quantity* 取代為您的 max Pod 值。

```
-KubeletExtraArgs '--max-pods=max-pods-quantity'
```

如果您要部署受管節點群組，則需要在啟動範本中新增此組態。如需詳細資訊，請參閱the section called “啟動範本”。如需 Windows 引導指令碼組態參數的詳細資訊，請參閱 the section called “引導指令碼組態參數”。

判斷最大 Pod 和可用的 IP 地址

1. 部署您的節點後，請檢視叢集中的節點。

```
kubectl get nodes
```

範例輸出如下。

```
NAME                                              STATUS     ROLES    AGE   VERSION
ip-192-168-22-103.region-code.compute.internal    Ready      <none>   19m   v1.XX.X-
eks-6b7464
ip-192-168-97-94.region-code.compute.internal     Ready      <none>   19m   v1.XX.X-
eks-6b7464
```

2. 描述其中一個節點，判斷該節點的 max-pods 值以及可用 IP 地址的數量。將 *192.168.30.193* 取代為上一個輸出中傳回的其中一個節點名稱中的IPv4地址。

```
kubectl describe node ip-192-168-30-193.region-code.compute.internal | grep 'pods\|
PrivateIPv4Address'
```

範例輸出如下。

```
pods:                                       110
```

```
vpc.amazonaws.com/PrivateIPv4Address:  144
```

在先前的輸出中， 110是 Kubernetes 將部署到節點的 Pod 數量上限，即使有 *144* 個 IP 地址可用。

將安全群組指派給個別 Pod

適用於：具有 Amazon EC2 執行個體的 Linux 節點

適用於：私有子網路

Pod 的安全群組將 Amazon EC2 安全群組與 Kubernetes Pod 整合。您可以使用 Amazon EC2 安全群組來定義規則，允許傳入和傳出網路流量進出您部署到在許多 Amazon EC2 執行個體類型和 Fargate 上執行的節點的 Pod。如需此功能的詳細說明，請參閱 Pod 的安全群組簡介部落格文章。

與適用於 Kubernetes 功能的 Amazon VPC CNI 外掛程式的相容性

您可以針對具有下列功能的 Pod 使用安全群組：

- IPv4 來源網路地址轉譯 - 如需詳細資訊，請參閱 the section called “傳出流量”。
- 叢集、Pod 和服務的 IPv6 地址 - 如需詳細資訊，請參閱 the section called “IPv6”。
- 使用 Kubernetes 網路政策限制流量 - 如需詳細資訊，請參閱 the section called “Kubernetes 政策”。

考量事項

部署 Pod 的安全群組之前，請考慮下列限制和條件：

- Pod 的安全群組無法與 Windows 節點搭配使用。
- Pod 的安全群組可與針對包含 Amazon EC2 節點的 `IPv6` 系列設定的叢集搭配使用，方法是使用 Amazon VPC CNI 外掛程式的 1.16.0 版或更新版本。您可以使用 1.7.7 版或更新版本的 Amazon VPC CNI 外掛程式，針對叢集設定僅包含 Fargate 節點`IPv6`系列之 Pod 使用安全群組。如需詳細資訊，請參閱the section called “IPv6”
- 大多數 Nitro 型 Amazon EC2 執行個體系列都支援 Pod 的安全群組，但並非所有世代的系列都支援。例如，支援 `m5`、`c5`、`r5`、`c6g`、 `m6g`和 `r6g`執行個體系列和世代。不支援 `t` 系列中的執行個體類型。如需支援執行個體類型的完整清單，請參閱 GitHub 上的 limit.go 檔案。您的節點必須為該檔案列出的具有 `IsTrunkingCompatible: true` 的執行個體類型之一。
- 如果您同時使用 Pod 的自訂聯網和安全群組，則會使用 Pod 安全群組指定的安全群組，而不是 中指定的安全群組`ENIConfig`。

- 如果您使用的是 版本 1.10.2或更早版本的 Amazon VPC CNI 外掛程式，且在 Pod 規格中包含 terminationGracePeriodSeconds設定，則設定的值不能為零。
- 如果您使用的是 版本 1.10或更早版本的 Amazon VPC CNI 外掛程式，或1.11具有 POD_SECURITY_GROUP_ENFORCING_MODE= 的版本strict，這是預設設定，則您指派安全群組的 Pod Local不支援類型 NodePort 和LoadBalancer使用externalTrafficPolicy執行個體目標設定為 的 Kubernetes 服務。如需將負載平衡器與執行個體目標搭配使用的詳細資訊，請參閱 the section called “網路負載平衡”。
- 如果您使用版本 1.10或更早版本的 Amazon VPC CNI 外掛程式，或1.11具有 POD_SECURITY_GROUP_ENFORCING_MODE= 的版本strict，這是預設設定，則來源 NAT 會針對來自具有指派安全群組之 Pod 的傳出流量停用，以便套用傳出安全群組規則。若要存取網際網路，必須在以 NAT 閘道或執行個體設定之私有子網路中部署的節點上啟動具有指派安全群組的 Pod。已指派安全群組部署至公有子網路的 Pod 無法存取網際網路。

  如果您使用版本 1.11或更新版本的外掛程式搭配 POD_SECURITY_GROUP_ENFORCING_MODE=standard，則目的地為 VPC 外部的 Pod 流量會轉譯為執行個體主要網路界面的 IP 地址。對於此流量，會使用主要網路界面安全群組中的規則，而不是 Pod 安全群組中的規則。
- 若要將 Calico 網路政策與具有關聯安全群組的 Pod 搭配使用，您必須使用 1.11.0或更新版本的 Amazon VPC CNI 外掛程式，並設定 POD_SECURITY_GROUP_ENFORCING_MODE=standard。否則，進出具有相關聯安全群組之 Pod 的流量不受 Calico 網路政策強制執行的約束，且僅限於 Amazon EC2 安全群組強制執行。若要更新 Amazon VPC CNI 版本，請參閱 the section called “Amazon VPC CNI”
- 在使用 NodeLocal DNSCache 的叢集中使用安全群組的 Amazon EC2 節點上執行的 Pod 僅支援版本 1.11.0或更新版本的 Amazon VPC CNI 外掛程式和 POD_SECURITY_GROUP_ENFORCING_MODE=standard。若要更新您的 Amazon VPC CNI 外掛程式版本，請參閱 the section called “Amazon VPC CNI”
- 對於流失率較高的 Pod，Pod 的安全群組可能會導致較高的 Pod 啟動延遲。原因是資源控制器的速率限制。
- EC2 安全群組範圍位於 Pod 層級 - 如需詳細資訊，請參閱安全群組。

  如果您設定 POD_SECURITY_GROUP_ENFORCING_MODE=standard和 AWS_VPC_K8S_CNI_EXTERNALSNAT=false，則目的地為 VPC 外部端點的流量會使用節點的安全群組，而不是 Pod 的安全群組。

為 Amazon EKS Pod 的安全群組設定 Kubernetes 的 Amazon VPC CNI 外掛程式

如果您搭配 Amazon EC2 執行個體使用 Pod，則需要為安全群組設定 Kubernetes 的 Amazon VPC CNI 外掛程式

如果您僅使用 Fargate Pod，而且叢集中沒有任何 Amazon EC2 節點，請參閱 the section called “SecurityGroupPolicy”。

1. 使用以下命令檢查適用於 Kubernetes 版本的目前 Amazon VPC CNI 外掛程式：

   ```
   kubectl describe daemonset aws-node --namespace kube-system | grep amazon-k8s-cni: |
    cut -d : -f 3
   ```

   範例輸出如下。

   ```
   v1.7.6
   ```

   如果您的適用於 Kubernetes 的 Amazon VPC CNI 外掛程式早於 1.7.7，請將外掛程式更新為版本 1.7.7或更新版本。如需詳細資訊，請參閱the section called “Amazon VPC CNI”

2. 將 link：iam/home#/policies/arn:aws: iam： ：aws：policy/AmazonEKSVPCResourceController【AmazonEKSVPCResourceController，type="console"】 受管 IAM 政策新增至與您的 Amazon EKS 叢集相關聯的叢集角色。此政策允許角色管理網路介面、其私有 IP 地址以及其與網路執行個體之間的連接和分離。

   a. 擷取叢集 IAM 角色的名稱，並存放在變數中。使用您叢集的名稱取代 *my-cluster*。

      ```
      cluster_role=$(aws eks describe-cluster --name my-cluster --query cluster.roleArn
       --output text | cut -d / -f 2)
      ```

   b. 將政策連接到角色。

      ```
      aws iam attach-role-policy --policy-arn arn:aws: iam::aws:policy/
      AmazonEKSVPCResourceController --role-name $cluster_role
      ```

3. 在 aws-node DaemonSet true中將 ENABLE_POD_ENI變數設定為 ，以啟用 Amazon VPC CNI 附加元件來管理 Pod 的網路介面。一旦此設定設為 true，則針對叢集中的每個節點，附加元件會建立cninode自訂資源。VPC 資源控制器會建立並連接一個稱為幹線網路介面與描述 aws-k8s-trunk-eni 的特殊網路介面。

   ```
   kubectl set env daemonset aws-node -n kube-system ENABLE_POD_ENI=true
   ```

> **Note**
>
> 幹線網路介面包含在執行個體類型所支援的最大網路介面數量中。如需每個執行個體類型支援的最大網路介面數量清單，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。如果您的節點已經連接到標準網路介面的最大數量，那麼 VPC 資源控制器將保留一個空間。您必須向下擴展執行中的 Pod，控制器才能分離和刪除標準網路介面、建立幹線網路介面，並將其連接至執行個體。

4. 您可以使用以下命令查看您的哪些節點擁有 CNINode 自訂資源。如果傳回 No resources found，則等待幾秒鐘後再重試一次。上一個步驟需要重新啟動 Kubernetes Pod 的 Amazon VPC CNI 外掛程式，這需要幾秒鐘的時間。

```
kubectl get cninode -A
     NAME FEATURES
     ip-192-168-64-141.us-west-2.compute.internal [{"name":"SecurityGroupsForPods"}]
     ip-192-168-7-203.us-west-2.compute.internal [{"name":"SecurityGroupsForPods"}]
```

如果您使用的 VPC CNI 版本低於 1.15，則會使用節點標籤而非 CNINode 自訂資源。您可以使用true下列命令查看哪些節點將節點標籤aws-k8s-trunk-eni設為 。如果傳回 No resources found，則等待幾秒鐘後再重試一次。上一個步驟需要重新啟動 Kubernetes Pod 的 Amazon VPC CNI 外掛程式，這需要幾秒鐘的時間。

```
kubectl get nodes -o wide -l vpc.amazonaws.com/has-trunk-attached=true
```

建立幹線網路介面後，會從幹線或標準網路介面指派次要 IP 地址給 Pod。如果刪除節點，則會自動刪除幹線介面。

當您在後續步驟中部署 Pod 的安全群組時，VPC 資源控制器會建立稱為分支網路介面的特殊網路介面，其中包含 的說明，aws-k8s-branch-eni並將安全群組與其建立關聯。除了連接至節點的標準和幹線網路介面之外，還會建立分支網路介面。

如果您使用即時或整備探查，則還需要停用 TCP 早期 demux，以便 kubelet可以使用 TCP 連接到分支網路介面上的 Pod。若要停用 TCP early demux，請執行下列命令：

```
kubectl patch daemonset aws-node -n kube-system \
  -p '{"spec": {"template": {"spec": {"initContainers": [{"env":
[{"name":"DISABLE_TCP_EARLY_DEMUX","value":"true"}],"name":"aws-vpc-cni-init"}]}}}}'
```

> **Note**
>
> 如果您使用的是適用於 Kubernetes 附加元件的 Amazon VPC CNI 外掛程式`1.11.0`或更新版本，並設定 `POD_SECURITY_GROUP_ENFORCING_MODE=standard`，如下一個步驟所述，則不需要執行先前的命令。

5. 如果您的叢集使用 `NodeLocal DNSCache`，或者您想要將 Calico 網路政策與具有自己的安全群組的 Pod 搭配使用，或者您有 類型 的 Kubernetes 服務`LoadBalancer`，`NodePort`並使用執行個體目標搭配 `externalTrafficPolicy` 設定為您要指派安全群組`Local`的 Pod，則您必須使用適用於 Kubernetes 附加元件的 Amazon VPC CNI 外掛程式版本 `1.11.0`或更新版本，而且您必須啟用下列設定：

   ```
   kubectl set env daemonset aws-node -n kube-system
    POD_SECURITY_GROUP_ENFORCING_MODE=standard
   ```

   重要：Pod 安全群組規則不會套用至 Pod 之間的流量，或 Pod 與位於相同節點的服務之間的流量**`nodeLocalDNS`**，例如 **`kubelet`**或 。在同一節點上使用不同安全群組的 Pod 無法通訊，因為它們是在不同的子網路中設定，而且在這些子網路之間停用路由。 從 Pod 到 VPC 外部地址的傳出流量，會轉譯為執行個體主要網路界面的 IP 地址 （除非您也已設定 `AWS_VPC_K8S_CNI_EXTERNALSNAT=true`)。對於此流量，會使用主要網路界面安全群組中的規則，而不是 Pod 安全群組中的規則。** 若要將此設定套用至現有的 Pod，您必須重新啟動 Pod 或 Pod 正在執行的節點。

6. 若要了解如何使用 Pod 的安全群組政策，請參閱 the section called “SecurityGroupPolicy”。

使用 Amazon EKS Pod 的安全群組政策

若要使用 Pod 的安全群組，您必須擁有現有的安全群組。下列步驟說明如何使用 Pod 的安全群組政策。除非另有說明，否則請從相同的終端機完成所有步驟，因為變數會用於以下步驟，這些步驟不會跨終端機保留。

如果您有 Pod 搭配 Amazon EC2 執行個體，您必須先設定外掛程式，才能使用此程序。如需詳細資訊，請參閱the section called “設定”。

1. 建立 Kubernetes 命名空間以部署 資源。您可以將 *`my-namespace`* 取代為要使用的命名空間的名稱。

```
kubectl create namespace my-namespace
```

2. 將 Amazon EKS SecurityGroupPolicy 部署至您的叢集。

   a. 將以下內容複製到您的裝置。serviceAccountSelector 如果您想要根據服務帳戶標籤選取 Pod，您可以將 *podSelector* 取代為 。您必須指定其中一個選取器或其他工具。空白 podSelector (範例：podSelector: {}) 會選取命名空間中的所有 Pod。您可以將 *my-role* 變更為您的角色名稱。空白 serviceAccountSelector 會選取命名空間中的所有服務帳戶。您可以將 *my-security-group-policy* 取代為您的 SecurityGroupPolicy 的名稱，將 *my-namespace* 取代為要在其中建立 SecurityGroupPolicy 的命名空間。

   您必須將 *my_pod_security_group_id* 取代為現有安全群組的 ID。如果您沒有現有的安全群組，則必須建立一個。如需詳細資訊，請參閱《Amazon EC2 使用者指南》https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/中的適用於 Linux 執行個體的 Amazon EC2 安全群組一節。您可以指定 1-5 個安全群組 ID。如果您指定多個 ID，則所有安全群組中所有規則的組合對選取的 Pod 有效。

```
cat >my-security-group-policy.yaml <<EOF
apiVersion: vpcresources.k8s.aws/v1beta1
kind: SecurityGroupPolicy
metadata:
  name: my-security-group-policy
  namespace: my-namespace
spec:
  podSelector:
    matchLabels:
      role: my-role
  securityGroups:
    groupIds:
      - my_pod_security_group_id
EOF
```

> **Important**
>
> 您為 Pod 指定的安全群組必須符合下列條件：
>
> - 它們必須存在。如果它們不存在，則當您部署符合選取器的 Pod 時，您的 Pod 仍會卡在建立程序中。如果您描述 Pod，您會看到類似下列的錯誤訊息：An error occurred (InvalidSecurityGroupID.NotFound) when calling

the CreateNetworkInterface operation: The securityGroup ID '*sg-05b1d815d1EXAMPLE*' does not exist。

- 它們必須允許透過您已設定探查的任何連接埠，從套用至節點的安全群組 （適用於 kubelet) 傳入通訊。
- 它們必須允許透過 TCP和UDP連接埠 53 與指派給執行 CoreDNS 之 Pod （或 Pod 執行所在的節點） 的安全群組進行傳出通訊。CoreDNS Pod 的安全群組必須允許來自您指定之安全群組的傳入TCP和UDP連接埠 53 流量。
- 他們必須具有必要的傳入和傳出規則，才能與需要通訊的其他 Pod 通訊。
- 如果您將安全群組與 Fargate 搭配使用，它們必須具有允許 Pod 與 Kubernetes 控制平面通訊的規則。執行此動作最簡單的方法是指定叢集安全群組為其中一個安全群組。

安全群組政策僅適用於新排程的 Pod。它們不會影響執行中的 Pod。

b. 部署政策。

```
kubectl apply -f my-security-group-policy.yaml
```

3. 部署具有符合您在上一個步驟中指定之 *podSelector* 的 *my-role* 值之標籤的範例應用程式。

a. 將以下內容複製到您的裝置。使用您自己的值取代*###*，然後執行修改後的命令。如果您取代 *my-role*，請確定它與您在上一個步驟中為選取器指定的值相同。

```
cat >sample-application.yaml <<EOF
apiVersion: apps/v1
kind: Deployment
metadata:
  name: my-deployment
  namespace: my-namespace
  labels:
    app: my-app
spec:
  replicas: 4
  selector:
    matchLabels:
      app: my-app
  template:
    metadata:
      labels:
        app: my-app
        role: my-role
    spec:
```

```
      terminationGracePeriodSeconds: 120
      containers:
      - name: nginx
        image: public.ecr.aws/nginx/nginx:1.23
        ports:
        - containerPort: 80
---
apiVersion: v1
kind: Service
metadata:
  name: my-app
  namespace: my-namespace
  labels:
    app: my-app
spec:
  selector:
    app: my-app
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80
EOF
```

b. 使用下列命令部署應用程式。當您部署應用程式時，適用於 Kubernetes 的 Amazon VPC CNI 外掛程式會比對`role`標籤，而且您在上一個步驟中指定的安全群組會套用至 Pod。

```
kubectl apply -f sample-application.yaml
```

4. 檢視使用範例應用程式部署的 Pod。對於本主題的餘數，該終端機稱為 `TerminalA`。

```
kubectl get pods -n my-namespace -o wide
```

範例輸出如下。

```
NAME                             READY   STATUS    RESTARTS   AGE     IP
  NODE                                             NOMINATED NODE   READINESS GATES
my-deployment-5df6f7687b-4fbjm   1/1     Running   0          7m51s   192.168.53.48
  ip-192-168-33-28.region-code.compute.internal   <none>           <none>
my-deployment-5df6f7687b-j9fl4   1/1     Running   0          7m51s   192.168.70.145
  ip-192-168-92-33.region-code.compute.internal   <none>           <none>
my-deployment-5df6f7687b-rjxcz   1/1     Running   0          7m51s   192.168.73.207
  ip-192-168-92-33.region-code.compute.internal   <none>           <none>
```

```
my-deployment-5df6f7687b-zmb42    1/1     Running   0          7m51s   192.168.63.27
  ip-192-168-33-28.region-code.compute.internal   <none>           <none>
```

> **Note**
>
> 如果有任何 Pod 卡住，請嘗試這些提示。
>
> - 如果有任何 Pod 卡在 `Waiting` 狀態，請執行 `kubectl describe pod my-deployment-xxxxxxxxxx-xxxxx -n my-namespace` 。若您看到 `Insufficient permissions: Unable to create Elastic Network Interface.`，請確認您已在上一步驟中將 IAM 政策新增至 IAM 叢集角色。
> - 如果有任何 Pod 卡在 `Pending` 狀態，請確認您的節點執行個體類型已列在 limit.go 中，且執行個體類型支援的分支網路介面數目上限乘以節點群組中節點數目的乘積尚未滿足。例如，`m5.large` 執行個體支援九個分支網路介面。如果您的節點群組有五個節點，則最多可以為節點群組建立 45 個分支網路介面。您嘗試部署的第 46 個 Pod 將處於 `Pending` 狀態，直到刪除另一個具有相關聯安全群組的 Pod 為止。

如果您執行 `kubectl describe pod my-deployment-xxxxxxxxxx-xxxxx -n my-namespace` ，並看到類似下列訊息的訊息，則可以放心地將其忽略。當 Kubernetes 的 Amazon VPC CNI 外掛程式嘗試設定主機聯網，並在建立網路介面時失敗時，可能會出現此訊息。外掛程式會記錄此事件，直到網路介面建立為止。

```
Failed to create Pod sandbox: rpc error: code = Unknown desc = failed to set up
 sandbox container "e24268322e55c8185721f52df6493684f6c2c3bf4fd59c9c121fd4cdc894579f"
 network for Pod "my-deployment-5df6f7687b-4fbjm": networkPlugin
cni failed to set up Pod "my-deployment-5df6f7687b-4fbjm-c89wx_my-namespace" network:
 add cmd: failed to assign an IP address to container
```

您無法超過執行個體類型上執行的 Pod 數量上限。如需每個執行個體類型可執行的 Pod 數量上限清單，請參閱 GitHub 上的 eni-max-pods.txt。當您刪除具有關聯安全群組的 Pod，或刪除執行 Pod 的節點時，VPC 資源控制器會刪除分支網路界面。如果您使用安全群組的 Pod 刪除具有 Pod 的叢集，則控制器不會刪除分支網路介面，因此您需要自行刪除它們。如需有關如何刪除網路介面的資訊，請參閱《Amazon EC2 使用者指南》中的刪除網路介面。

5. 在單獨的終端機中，將 shell 插入其中一個 Pod。對於本主題的餘數，該終端機稱為 `TerminalB`。將 *`5df6f7687b-4fbjm`* 取代為上一個步驟輸出中傳回的其中一個 Pod 的 ID。

```
kubectl exec -it -n my-namespace my-deployment-5df6f7687b-4fbjm -- /bin/bash
```

6. 從 TerminalB 中的 shell，確認範例應用程式是否正常運作。

```
curl my-app
```

範例輸出如下。

```
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
[...]
```

您收到輸出，因為執行應用程式的所有 Pod 都與您建立的安全群組相關聯。該群組包含規則，允許安全群組相關聯的所有 Pod 之間的所有流量。允許從該安全群組傳出 DNS 流量到與節點關聯的叢集安全群組。節點正在執行 CoreDNS Pod，您的 Pod 會對其進行名稱查詢。

7. 從 TerminalA 中，移除允許從安全群組與叢集進行 DNS 通訊的安全群組規則。如果您在上一個步驟中沒有將 DNS 規則新增至叢集安全群組，請將 *$my_cluster_security_group_id* 取代為您建立規則的安全群組 ID。

```
aws ec2 revoke-security-group-ingress --group-id $my_cluster_security_group_id --
security-group-rule-ids $my_tcp_rule_id
aws ec2 revoke-security-group-ingress --group-id $my_cluster_security_group_id --
security-group-rule-ids $my_udp_rule_id
```

8. 從 TerminalB，再次嘗試存取該應用程式。

```
curl my-app
```

範例輸出如下。

```
curl: (6) Could not resolve host: my-app
```

嘗試失敗，因為 Pod 無法再存取具有與其相關聯的叢集安全群組的 CoreDNS Pod。叢集安全群組不再具有允許來自與 Pod 相關聯之安全群組的 DNS 通訊的安全群組規則。

如果您嘗試使用上一個步驟中針對其中一個 Pod 傳回的 IP 地址存取應用程式，您仍然會收到回應，因為具有與其相關聯之安全群組的 Pod 之間允許所有連接埠，而且不需要名稱查詢。

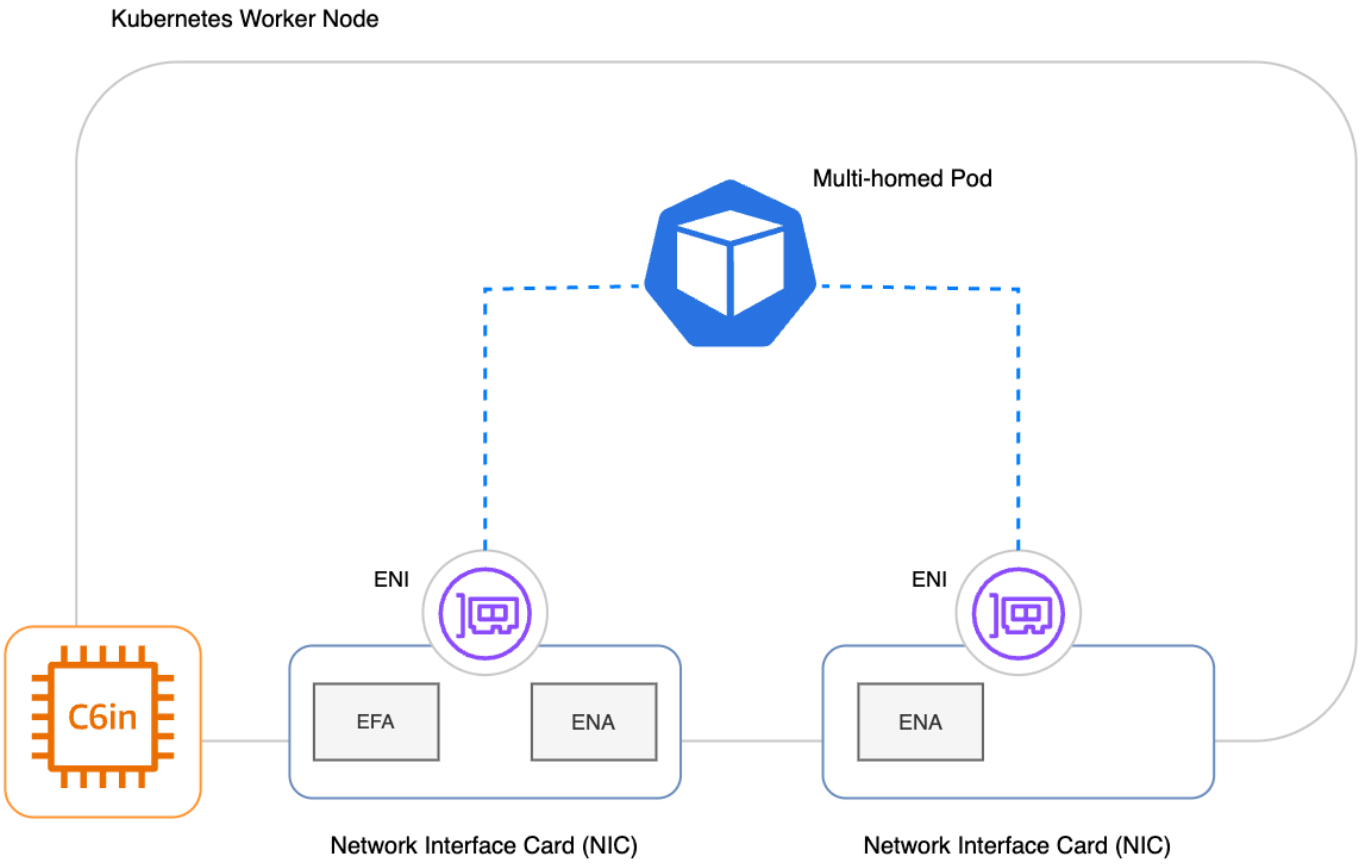
9. 完成實驗後，您可以移除您建立的範例安全群組政策、應用程式和安全群組。從 TerminalA 執行下列命令。

```
kubectl delete namespace my-namespace
aws ec2 revoke-security-group-ingress --group-id $my_pod_security_group_id --
security-group-rule-ids $my_inbound_self_rule_id
wait
sleep 45s
aws ec2 delete-security-group --group-id $my_pod_security_group_id
```

將多個網路介面連接至 Pod

根據預設，Amazon VPC CNI 外掛程式會為每個 Pod 指派一個 IP 地址。此 IP 地址會連接到彈性網路界面，以處理 Pod 的所有傳入和傳出流量。若要提高頻寬和每秒封包速率效能，您可以使用 VPC CNI 的多 NIC 功能來設定多主目錄 Pod。多主目錄 Pod 是使用多個網路介面 （和多個 IP 地址） 的單一 Kubernetes Pod。透過執行多主目錄 Pod，您可以使用並行連線將其應用程式流量分散到多個網路介面。這對於人工智慧 (AI)、Machine Learning(ML) 和高效能運算 (HPC) 使用案例特別有用。

下圖顯示在使用多個網路介面卡 (NICs) 的工作者節點上執行的多主目錄 Pod。



背景介紹

在 Amazon EC2 上，彈性網路界面是 VPC 中代表虛擬網路卡的邏輯聯網元件。對於許多 EC2 執行個體類型，網路介面在硬體中共用單一網路介面卡 (NIC)。此單一 NIC 具有最大頻寬和每秒封包速率。

如果已啟用多 NIC 功能，VPC CNI 不會大量指派 IP 地址，預設會這麼做。相反地，VPC CNI 會在新的 Pod 啟動時，隨需為每個網路卡上的網路介面指派一個 IP 地址。此行為會降低 IP 地址耗盡的速率，這會因為使用多主目錄 Pod 而增加。由於 VPC CNI 正在隨需指派 IP 地址，因此在啟用多 NIC 功能的執行個體上，Pod 可能需要更長的時間才能啟動。

考量事項

- 確保您的 Kubernetes 叢集正在執行 VPC CNI 版本 `1.20.0` 和更新版本。多 NIC 功能僅適用於 VPC CNI `1.20.0`或更新版本。
- 在 VPC CNI 外掛程式中啟用`ENABLE_MULTI_NIC`環境變數。您可以執行下列命令來設定 變數，並啟動 DaemonSet 的部署。
  - `kubectl set env daemonset aws-node -n kube-system ENABLE_MULTI_NIC=true`

- 請確定您建立具有多個網路介面卡 (NICs工作者節點。如需具有多個網路介面卡的 EC2 執行個體清單，請參閱《Amazon EC2 使用者指南》中的網路卡。
- 如果已啟用多 NIC 功能，VPC CNI 不會大量指派 IP 地址，預設會這麼做。由於 VPC CNI 正在隨需指派 IP 地址，因此在啟用多 NIC 功能的執行個體上，Pod 可能需要更長的時間才能啟動。如需詳細資訊，請參閱前一 the section called “背景介紹” 節。
- 啟用多 NIC 功能時，Pod 預設不會有多個網路介面。您必須將每個工作負載設定為使用多重 NIC。將k8s.amazonaws.com/nicConfig: multi-nic-attachment註釋新增至應具有多個網路介面的工作負載。

IPv6 考量

- 自訂 IAM 政策 - 對於IPv6叢集，請為 VPC CNI 建立並使用下列自訂 IAM 政策。此政策專屬於多 NIC。如需搭配IPv6叢集使用 VPC CNI 的一般資訊，請參閱 the section called “IPv6”。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AmazonEKSCNIPolicyIPv6MultiNIC",
            "Effect": "Allow",
            "Action": [
                "ec2:CreateNetworkInterface",
                "ec2:DescribeInstances",
                "ec2:AssignIpv6Addresses",
                "ec2:DetachNetworkInterface",
                "ec2:DescribeNetworkInterfaces",
                "ec2:DescribeTags",
                "ec2:ModifyNetworkInterfaceAttribute",
                "ec2:DeleteNetworkInterface",
                "ec2:DescribeInstanceTypes",
                "ec2:UnassignIpv6Addresses",
                "ec2:AttachNetworkInterface",
                "ec2:DescribeSubnets"
            ],
            "Resource": "*"
        },
        {
            "Sid": "AmazonEKSCNIPolicyENITagIPv6MultiNIC",
            "Effect": "Allow",
            "Action": "ec2:CreateTags",
            "Resource": "arn:aws:ec2:*:*:network-interface/*"
```

```
        }
    ]
}
```

- IPv6 無法使用轉換機制 - 如果您使用多 NIC 功能，VPC CNI 不會將IPv4地址指派給IPv6叢集上的 Pod。否則，VPC CNI 會為每個 Pod 指派主機本機IPv4地址，以便 Pod 可以與其他 Amazon VPC 或網際網路中的外部IPv4資源通訊。

用量

在 VPC CNI 中啟用多 NIC aws-node 功能且 Pod 重新啟動後，您可以將每個工作負載設定為多主目錄。下列具有必要註釋的 YAML 組態範例：

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: orders-deployment
  namespace: ecommerce
  labels:
    app: orders
spec:
  replicas: 3
  selector:
    matchLabels:
      app: orders
  template:
    metadata:
      annotations:
         k8s.amazonaws.com/nicConfig: multi-nic-attachment
      labels:
        app: orders
    spec:
...
```

常見問答集

1。什麼是網路介面卡 (NIC)？

網路介面卡 (NIC) 也稱為網路卡，是一種實體裝置，可為基礎雲端運算硬體啟用網路連線。在現代 EC2 伺服器中，這是指 Nitro 網路卡。Elastic Network Interface (ENI) 是此基礎網路卡的虛擬表示法。

有些 EC2 執行個體類型具有多個 NICs可提供更高的頻寬和封包速率效能。對於這類執行個體，您可以將次要 ENIs 指派給其他網路卡。例如，ENI #1 可以做為連接到網路卡索引 0 之 NIC 的界面，而 ENI #2 可以做為連接到個別網路卡索引之 NIC 的界面。

2. 什麼是多主目錄 Pod？

多主目錄 Pod 是具有多個網路介面的單一 Kubernetes Pod （並透過隱含多個 IP 地址）。每個 Pod 網路界面都與彈性網路界面 (ENI) 相關聯，這些 ENIs 是基礎工作者節點上個別 NICs 的邏輯表示法。使用多個網路介面時，多主目錄 Pod 具有額外的資料傳輸容量，這也會提高其資料傳輸率。

> **Important**
>
> VPC CNI 只能在具有多個 NICs 的執行個體類型上設定多主目錄 Pod。

3. 為什麼我應該使用此功能？

如果您需要在以 Kubernetes 為基礎的工作負載中擴展網路效能，您可以使用 多 NIC 功能來執行多主目錄 Pod，該 Pod 會與其連接 ENA 裝置的所有基礎 NICs 連接。利用其他網路卡可將應用程式流量分散到多個並行連線，以提高應用程式的頻寬容量和封包速率效能。這對於人工智慧 (AI)、Machine Learning(ML) 和高效能運算 (HPC) 使用案例特別有用。

4. 如何使用此功能？

1. 首先，您必須確保您的 Kubernetes 叢集使用 VPC CNI 1.20 版或更新版本。如需將 VPC CNI 更新為 EKS 附加元件的步驟，請參閱 the section called “更新 (EKS 附加元件)”。
2. 然後，您必須使用環境ENABLE_MULTI_NIC變數在 VPC CNI 中啟用多 NIC 支援。
3. 然後，您必須確保建立和聯結具有多個網路卡的節點。如需具有多個網路卡的 EC2 執行個體類型清單，請參閱《Amazon EC2 使用者指南》中的網路卡。
4. 最後，您將每個工作負載設定為使用多個網路介面 （多主目錄 Pod) 或使用單一網路介面。

5. 如何將工作負載設定為在支援的工作者節點上使用多個 NICs？

若要使用多主目錄 Pod，您需要新增下列註釋：`k8s.amazonaws.com/nicConfig: multi-nic-attachment`。這會將基礎執行個體中每個 NIC 的 ENI 連接到 Pod (Pod 和 NICs 之間的一個到多個映射）。

如果缺少此註釋，VPC CNI 會假設您的 Pod 只需要 1 個網路介面，並在任何可用的 NIC 上指派來自 ENI 的 IP。

6. 此功能支援哪些網路介面轉接器？

如果您有至少一個 ENA 連接到基礎網路卡的 IP 流量，您可以使用任何網路介面轉接器。如需 ENA 的詳細資訊，請參閱《Amazon EC2 使用者指南》中的彈性網路轉接器 (ENA)。

支援的網路裝置組態：

- ENA 介面提供支援 VPC IP 聯網所需的所有傳統 IP 聯網和路由功能。如需詳細資訊，請參閱在 EC2 執行個體上啟用 ENA 增強型聯網。
- EFA (EFA 搭配 ENA) 介面提供用於 IP 聯網的 ENA 裝置和用於低延遲、高輸送量通訊的 EFA 裝置。

> **Important**
>
> 如果網路卡只連接 EFA 限定轉接器，VPC CNI 將在佈建多主目錄 Pod 的網路連線時略過它。不過，如果您將僅限 EFA 轉接器與網路卡上的 ENA 轉接器結合，則 VPC CNI 也會管理此裝置上的 ENIs。若要搭配 EKS 叢集使用僅限 EFA 的介面，請參閱 the section called “使用 EFA 設定訓練叢集”。

7. 我是否可以查看叢集中的節點是否支援 ENA？

可以，您可以使用 AWS CLI 或 EC2 API 來擷取叢集中 EC2 執行個體的網路資訊。這可提供執行個體是否支援 ENA 的詳細資訊。在下列範例中，將 <your-instance-id>取代為節點的 EC2 執行個體 ID。

AWS CLI 範例：

```
aws ec2 describe-instances --instance-ids <your-instance-id> --query
 "Reservations[].Instances[].EnaSupport"
```

輸出範例：

```
[ true ]
```

8. 我可以看到與 Pod 相關聯的不同 IP 地址嗎？

不，不容易。不過，您可以從節點使用 nsenter 來執行常見的網路工具，例如 ，ip route show並查看其他 IP 地址和介面。

9. 我可以控制 Pod 的網路介面數量嗎？

否。將工作負載設定為在支援的執行個體上使用多個 NICs 時，單一 Pod 會自動擁有執行個體上每個網路卡的 IP 地址。或者，單一主目錄 Pod 將有一個網路介面連接到執行個體上的一個 NIC。

> **Important**
>
> VPC CNI 會略過僅連接 EFA 限定裝置的網路卡。

10. 我可以將 Pod 設定為使用特定 NIC 嗎？

否，不支援。如果 Pod 具有相關註釋，則 VPC CNI 會自動將其設定為在工作者節點上使用每個 NIC 搭配 ENA 轉接器。

11. 此功能是否適用於其他 VPC CNI 網路功能？

是，VPC CNI 中的多 NIC 功能適用於自訂聯網和增強型子網路探索。不過，多主目錄 Pod 不會使用自訂子網路或安全群組。反之，VPC CNI 會將 IP 地址和網路介面指派給與節點具有相同子網路和安全群組組態的多主目錄 Pod。如需自訂聯網的詳細資訊，請參閱 the section called “自訂聯網”。

VPC CNI 中的多 NIC 功能不適用於 ，也無法與 Pod 的安全群組結合使用。

12. 我可以將網路政策與此功能搭配使用嗎？

可以，您可以將 Kubernetes 網路政策與多 NIC 搭配使用。Kubernetes 網路政策會限制往返 Pod 的網路流量。如需使用 VPC CNI 套用網路政策的詳細資訊，請參閱 the section called “Kubernetes 政策”。

13. 是否在 EKS Auto Mode 中啟用多 NIC 支援？

EKS Auto Mode 叢集不支援 Multi-NIC。

Amazon EKS 叢集的替代 CNI 外掛程式

適用於 Kubernetes 的 Amazon VPC CNI 外掛程式是具有 Amazon EC2 節點的 Amazon EKS 唯一支援的 CNI 外掛程式。Amazon EKS 支援適用於 Amazon EKS 混合節點的 Cilium 和 Calico 核心功能。Amazon EKS 執行上游 Kubernetes，因此您可以將替代相容的 CNI 外掛程式安裝到叢集中的 Amazon EC2 節點。如果您的叢集中有 Fargate 節點，則適用於 Kubernetes 的 Amazon VPC CNI 外掛程式已存在於 Fargate 節點上。這是您可以搭配 Fargate 節點使用的唯一 CNI 外掛程式。嘗試在 Fargate 節點上安裝替代 CNI 外掛程式失敗。

如果您打算在 Amazon EC2 節點中使用替代 CNI 外掛程式，那麼建議您取得外掛程式的商業支援，或擁有內部專業知識來排除問題，並為 CNI 外掛程式專案提供修正方法。

Amazon EKS 會持續與提供替代相容 CNI 外掛程式支援的合作夥伴網路建立關係。請參閱下列合作夥伴文件，獲取有關版本、資格和所執行測試的詳細資訊。

合作夥伴	產品	文件
Tigera	Calico	安裝說明
Isovalent	Cilium	安裝說明
Juniper	雲端原生 Contrail Networking (CN2)	安裝說明
VMware	Antrea	安裝說明

Amazon EKS 旨在為您提供各種選擇來涵蓋所有使用案例。

替代相容網路政策外掛程式

Calico 是廣泛採用的容器聯網和安全性解決方案。在 EKS 上使用 Calico 為您的 EKS 叢集提供完全合規的網路政策強制執行。此外，您可以選擇使用 Calico 的聯網，這會保留基礎 VPC 的 IP 地址。Calico Cloud 增強了 Calico Open Source 的功能，提供進階的安全性和可觀測性功能。

具有關聯安全群組的 Pod 的流量不受 Calico 網路政策強制執行的約束，且僅限於 Amazon VPC 安全群組強制執行。

如果您使用 Calico 網路政策強制執行，建議您將環境變數設定為 `ANNOTATE_POD_IPtrue`，以避免 Kubernetes 的已知問題。若要使用此功能，您必須將 Pod 的`patch`許可新增至 `aws-node` ClusterRole。請注意，將修補程式許可新增至 `aws-node` DaemonSet 會增加外掛程式的安全範圍。如需詳細資訊，請參閱 GitHub 上 VPC CNI 儲存庫中的 ANNOTATE_POD_IP。

Amazon EKS Auto 模式的考量事項

Amazon EKS Auto Mode 不支援替代 CNI 外掛程式或網路政策外掛程式。如需詳細資訊，請參閱EKS 自動模式。

使用 Multus 將多個網路介面連接至 Pod

Multus CNI 是 Amazon EKS 的容器網路介面 (CNI) 外掛程式，可讓 將多個網路介面連接至 Pod。如需詳細資訊，請參閱 GitHub 上的 Multus-CNI文檔。

在 Amazon EKS 中，每個 Pod 都有一個由 Amazon VPC CNI 外掛程式指派的網路介面。使用 Multus，您可以建立具有多個介面的多主目錄 Pod。這由做為「中繼外掛程式」的 Multus 完成；可呼叫多個其他 CNI 外掛程式的 CNI 外掛程式。Multus 的 AWS 支援使用 Amazon VPC CNI 外掛程式設定為預設委派外掛程式。

- Amazon EKS 不會建置和發佈單一根 I/O 虛擬化 (SR-IOV) 和資料平面開發套件 (DPDK) CNI 外掛程式。不過，您可以透過 Multus 受管主機裝置和 `ipvlan` 外掛程式直接連接至 Amazon EC2 彈性網路介面 (ENA)，進而達成封包加速。
- Amazon EKS 支援 Multus，它提供了一個通用的程序，可以簡單地鏈接其他 CNI 外掛程式。支援 Multus 和鏈結程序，但 AWS 不支援所有可鏈結的相容 CNI 外掛程式，或這些與鏈結組態無關的 CNI 外掛程式中可能發生的問題。
- Amazon EKS 為 Multus 外掛程式提供支援和生命週期管理，但不負責與額外網路介面相關聯的任何 IP 地址或其他管理。使用 Amazon VPC CNI 外掛程式之預設網路介面的 IP 地址和管理維持不變。
- mazon VPC CNI 外掛程式僅獲預設委派外掛程式的正式支援。如果您選擇不將 Amazon VPC CNI 外掛程式用於主要聯網，則需要修改已發佈的 Multus 安裝清單檔案，以將預設委派外掛程式重新設定為替代 CNI。
- 只有在使用 Amazon VPC CNI 作為主要 CNI 時，才支援 Multus。用於高階、次要或其他介面時，我們不支援 Amazon VPC CNI。
- 若要防止 Amazon VPC CNI 外掛程式嘗試管理指派給 Pod 的其他網路介面，請將下列標籤新增至網路介面：

  金鑰

  ```
  : node.k8s.amazonaws.com/no_manage
  ```

  值

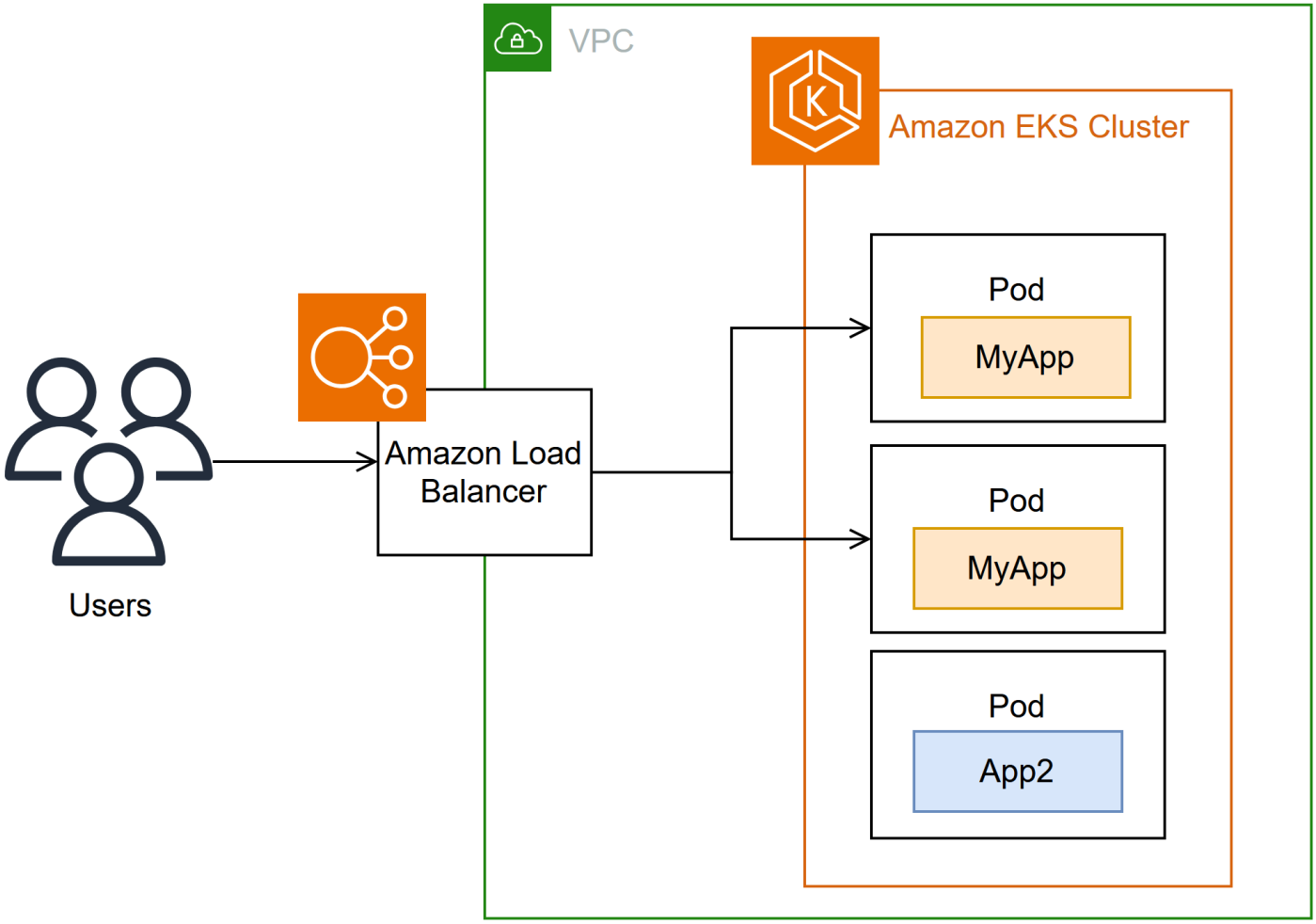
  ```
  : true
  ```

- Multus 與網路政策相容，但政策必須擴充，以包含連接埠和 IP 地址，這些連接埠和 IP 地址可能是連接至 Pod 的其他網路介面的一部分。

如需實作逐步示範，請參閱 GitHub 上的 Multus 設定指南。

使用 AWS Load Balancer控制器路由網際網路流量

The AWS Load Balancer 控制器會管理 Kubernetes 叢集的 AWS Elastic Load Balancer。您可以使用控制器將叢集應用程式公開至網際網路。控制器會佈建指向叢集服務或輸入資源的 AWS 負載平衡器。換句話說，控制器會建立指向叢集中多個 Pod 的單一 IP 地址或 DNS 名稱。



控制器會監看 Kubernetes Ingress 或服務資源。為了回應，它會建立適當的 AWS Elastic Load Balancing資源。您可以將註釋套用至 Kubernetes 資源，以設定負載平衡器的特定行為。例如，您可以使用註釋將 AWS 安全群組連接至負載平衡器。

控制器會佈建以下資源：

Kubernetes **`Ingress`**

當您建立 Kubernetes 時，LBC 會建立 AWS Application Load Balancer (ALB)`Ingress`。檢閱您可以套用至傳入資源的註釋。

`LoadBalancer`類型的 Kubernetes 服務

當您建立類型為 的 Kubernetes 服務時，LBC 會建立 AWS Network Load Balancer (NLB)`LoadBalancer`。檢閱您可以套用至服務資源的註釋。

過去，Kubernetes 網路負載平衡器用於執行個體目標，但 LBC 用於 IP 目標。使用 AWS Load Balancer控制器版本 `2.3.0` 或更新版本，您可以使用任一目標類型建立 NLBs。如需 NLB 目標類型的詳細資訊，請參閱《Network Load Balancer 使用者指南》中的目標類型。

控制器是 GitHub 上管理的開放原始碼專案。

部署控制器之前，建議您使用 Application Load Balancer 和 檢閱 Route 應用程式和 HTTP 流量中的先決條件和考量事項the section called “網路負載平衡”。在這些主題中，您將部署包含 AWS 負載平衡器的範例應用程式。

安裝控制器

您可以使用下列其中一個程序來安裝 AWS Load Balancer控制器：

- 如果您是初次使用 Amazon EKS，我們建議您使用 Helm 進行安裝，因為它可簡化 AWS Load Balancer控制器安裝。如需詳細資訊，請參閱the section called “使用 Helm 安裝”。
- 對於進階組態，例如對公有容器登錄檔的網路存取受限的叢集，請使用 Kubernetes 資訊清單。如需詳細資訊，請參閱the section called “使用資訊清單安裝”。

從已淘汰的控制器版本遷移

- 如果您已安裝已棄用版本的 AWS Load Balancer控制器，請參閱 the section called “從已棄用的 遷移”。
- 已棄用的版本無法升級。必須移除它們，並安裝目前版本的 AWS Load Balancer控制器。
- 已棄用版本包括：
  - AWS 適用於 Kubernetes 的 ALB 輸入控制器 (「輸入控制器」)，是 AWS Load Balancer控制器的前身。
  - 任何`0.1.`*`x`* 版本的 AWS Load Balancer控制器

舊版雲端供應商

Kubernetes 包含 的舊版雲端供應商 AWS。舊版雲端提供者能夠佈建 AWS 負載平衡器，類似於 AWS Load Balancer控制器。舊版雲端提供者會建立 Classic Load Balancer。如果您未安裝 AWS Load Balancer控制器，Kubernetes 會預設為使用舊版雲端提供者。您應該安裝 AWS Load Balancer控制器，並避免使用舊版雲端提供者。

> **Important**
>
> 在 2.5 版和更新版本中， AWS Load Balancer控制器會使用 成為 Kubernetes 服務資源的預設控制器，`type: LoadBalancer`並為每個服務建立 AWS Network Load Balancer (NLB)。它透過為服務製作變異的 webhook 來做到這一點，此 webhook 會為 `type: LoadBalancer` 的新服務將 `spec.loadBalancerClass` 欄位設定至 `service.k8s.aws/nlb`。您可以關閉此功能並恢復為使用舊式雲端供應商作為預設控制器，方法是將 Helm Chart 值 `enableServiceMutatorWebhook` 設定為 `false`。除非您關閉此功能，否則叢集不會為您的服務佈建新的 Classic Load Balancer。現有的 Classic Load Balancer 會繼續運作。

搭配 Helm 的 Install AWS Load Balancer 控制器

> **Tip**
>
> 使用 Amazon EKS Auto Mode，您不需要安裝或升級聯網附加元件。自動模式包含 Pod 聯網和負載平衡功能。
> 如需詳細資訊，請參閱EKS 自動模式。

本主題說明如何使用 Helm、Kubernetes 套件管理員 和 安裝 AWS Load Balancer控制器`eksctl`。控制器已安裝預設選項。如需控制器的詳細資訊，包括使用註釋設定控制器的詳細資訊，請參閱 GitHub 上的AWS Load Balancer控制器文件。

在下列步驟中，將`###`取代為您自己的值。

先決條件

開始本教學課程之前，您必須完成下列步驟：

- 建立 Amazon EKS 叢集。若要建立服務角色，請參閱開始使用。
- 在本機電腦上安裝 Helm。

- 請確定 Kubernetes、 `kube-proxy`和 CoreDNS 附加元件的 Amazon VPC CNI 外掛程式處於服務帳戶字符中列出的最低版本。
- 了解 AWS Elastic Load Balancing概念。如需詳細資訊，請參閱《Elastic Load Balancing 使用者指南》。
- 了解 Kubernetes 服務和輸入資源。

考量事項

在繼續此頁面上的組態步驟之前，請考慮下列事項：

- IAM 政策和角色 (`AmazonEKSLoadBalancerControllerRole`) 可以在相同 AWS 帳戶中的多個 EKS 叢集之間重複使用。
- 如果您要在最初建立角色 (`AmazonEKSLoadBalancerControllerRole`) 的相同叢集上安裝控制器，請在驗證角色存在之後，前往步驟 2：安裝Load Balancer控制器。
- 如果您使用服務帳戶 (IRSA) 的 IAM 角色，則必須為每個叢集設定 IRSA，而且角色信任政策中的 OpenID Connect (OIDC) 供應商 ARN 專屬於每個 EKS 叢集。此外，如果您要在具有現有 的新叢集上安裝控制器`AmazonEKSLoadBalancerControllerRole`，請更新角色的信任政策，以包含新叢集的 OIDC 供應商，並使用適當的角色註釋建立新的服務帳戶。若要判斷您是否已經有 OIDC 提供者，或要建立提供者，請參閱 the section called “IAM OIDC 供應商”。

步驟 1：使用 建立 IAM 角色 `eksctl`

下列步驟參考 AWS Load Balancer控制器 v2.13.3 發行版本。如需所有版本的詳細資訊，請參閱 GitHub 上的AWS Load Balancer控制器版本頁面。

1. 下載 AWS Load Balancer控制器的 IAM 政策，以允許其代表您呼叫 AWS APIs。

```
curl -O https://raw.githubusercontent.com/kubernetes-sigs/aws-load-balancer-
controller/v2.13.3/docs/install/iam_policy.json
```

   - 如果您是非標準 AWS 分割區，例如政府或中國區域，請檢閱 GitHub 上的政策，並下載適用於您區域的適當政策。

2. 使用上一個步驟中下載的政策，建立 IAM 政策。

```
aws iam create-policy \
    --policy-name AWSLoadBalancerControllerIAMPolicy \
    --policy-document file://iam_policy.json
```

> **Note**
>
> 如果您在 中檢視政策 AWS Management Console，主控台會顯示 ELB 服務的警告，但不會顯示 ELB v2 服務的警告。發生這種情況是由於政策中存在適用於 ELB v2 的某些動作，但不適用於 ELB。您可以忽略這些對 ELB 發出的警告。

3. 取代叢集名稱、區域代碼和帳戶 ID 的值。

```
eksctl create iamserviceaccount \
    --cluster=<cluster-name> \
    --namespace=kube-system \
    --name=aws-load-balancer-controller \
    --attach-policy-arn=arn:aws: iam::<AWS_ACCOUNT_ID>:policy/
AWSLoadBalancerControllerIAMPolicy \
    --override-existing-serviceaccounts \
    --region <aws-region-code> \
    --approve
```

步驟 2：Install AWS Load Balancer 控制器

1. 新增 `eks-charts` Helm Chart 儲存庫。 會在 GitHub 上 AWS 維護此儲存庫。

```
helm repo add eks https://aws.github.io/eks-charts
```

2. 更新您的本機儲存庫，以確定您擁有最新的圖表。

```
helm repo update eks
```

3. 安裝 AWS Load Balancer控制器。

   如果您要將控制器部署到限制存取 Amazon EC2 執行個體中繼資料服務 (IMDS) 的 Amazon EC2 節點，或如果您要部署到 Fargate 或 Amazon EKS 混合節點，請將下列旗標新增至下列`helm`命令：Amazon EC2

   - `--set region=`*`region-code`*
   - `--set vpcId=`*`vpc-xxxxxxxx`*

   使用您叢集的名稱取代 *`my-cluster`*。在下列命令中，`aws-load-balancer-controller` 是您在上一個步驟中建立的 Kubernetes 服務帳戶。

如需設定 helm Chart 的詳細資訊，請參閱 GitHub 上的 values.yaml。

```
helm install aws-load-balancer-controller eks/aws-load-balancer-controller \
  -n kube-system \
  --set clusterName=my-cluster \
  --set serviceAccount.create=false \
  --set serviceAccount.name=aws-load-balancer-controller \
  --version 1.13.0
```

Important

部署的圖表不會自動接收安全性更新。當圖表可用時，您需要手動升級到較新的圖表。升級時，請在上一個命令upgrade中將##變更為 。

`helm install` 命令會自動安裝控制器的自訂資源定義 (CRDs)。`helm upgrade` 命令不會。如果您使用 `helm upgrade`, ，則必須手動安裝 CRDs。執行下列命令來安裝 CRDs：

```
wget https://raw.githubusercontent.com/aws/eks-charts/master/stable/aws-load-balancer-controller/crds/crds.yaml
kubectl apply -f crds.yaml
```

步驟 3：確認控制器已安裝

1. 確認控制器已安裝。

   ```
   kubectl get deployment -n kube-system aws-load-balancer-controller
   ```

   範例輸出如下。

   ```
   NAME                           READY   UP-TO-DATE   AVAILABLE   AGE
   aws-load-balancer-controller   2/2     2            2           84s
   ```

   如果使用 Helm 進行部署，則會收到先前的輸出。如果使用 Kubernetes 清單檔案進行部署，則您只有一個複本。

2. 使用控制器佈建 AWS 資源之前，您的叢集必須符合特定需求。如需詳細資訊，請參閱the section called “應用程式負載平衡”及the section called “網路負載平衡”。

具有資訊清單的 Install AWS Load Balancer 控制器

> **Tip**
>
> 使用 Amazon EKS Auto Mode，您不需要安裝或升級聯網附加元件。自動模式包含 Pod 聯網和負載平衡功能。
> 如需詳細資訊，請參閱EKS 自動模式。

本主題說明如何透過下載和套用 Kubernetes 資訊清單來安裝控制器。您可以檢視 GitHub 上的完整控制器文件。

在下列步驟中，將###取代為您自己的值。

先決條件

開始本教學課程之前，您必須完成下列步驟：

- 建立 Amazon EKS 叢集。若要建立服務角色，請參閱開始使用。
- 在本機電腦上安裝 Helm。
- 請確定 Kubernetes、 `kube-proxy`和 CoreDNS 附加元件的 Amazon VPC CNI 外掛程式處於服務帳戶字符中列出的最低版本。
- 了解 AWS Elastic Load Balancing概念。如需詳細資訊，請參閱《Elastic Load Balancing 使用者指南》。
- 了解 Kubernetes 服務和輸入資源。

考量事項

在繼續此頁面上的組態步驟之前，請考慮下列事項：

- IAM 政策和角色 (`AmazonEKSLoadBalancerControllerRole`) 可以在相同 AWS 帳戶中的多個 EKS 叢集之間重複使用。
- 如果您要在最初建立角色 (`AmazonEKSLoadBalancerControllerRole`) 的相同叢集上安裝控制器，請在驗證角色存在之後，前往步驟 2：安裝 cert-manager。
- 如果您使用服務帳戶 (IRSA) 的 IAM 角色，則必須為每個叢集設定 IRSA，而且角色信任政策中的 OpenID Connect (OIDC) 供應商 ARN 專屬於每個 EKS 叢集。此外，如果您要在具有現有 的新叢集上安裝控制器`AmazonEKSLoadBalancerControllerRole`，請更新角色的信任政策，以包含新叢

集的 OIDC 供應商，並使用適當的角色註釋建立新的服務帳戶。若要判斷您是否已經有 OIDC 提供者，或要建立提供者，請參閱 the section called “IAM OIDC 供應商”。

步驟 1：設定 IAM

下列步驟參考 AWS Load Balancer控制器 v2.13.3 發行版本。如需所有版本的詳細資訊，請參閱 GitHub 上的AWS Load Balancer控制器版本頁面。

1. 下載 AWS Load Balancer控制器的 IAM 政策，以允許其代表您呼叫 AWS APIs。

   Example

   AWS

   ```
   curl -O https://raw.githubusercontent.com/kubernetes-sigs/aws-load-balancer-controller/v2.13.3/docs/install/iam_policy.json
   ```

   AWS GovCloud (US)

   ```
   curl -O https://raw.githubusercontent.com/kubernetes-sigs/aws-load-balancer-controller/v2.13.3/docs/install/iam_policy_us-gov.json
   ```

   ```
   mv iam_policy_us-gov.json iam_policy.json
   ```

2. 使用上一個步驟中下載的政策，建立 IAM 政策。

   ```
   aws iam create-policy \
       --policy-name AWSLoadBalancerControllerIAMPolicy \
       --policy-document file://iam_policy.json
   ```

   > **Note**
   >
   > 如果您在 中檢視政策 AWS Management Console，主控台會顯示 ELB 服務的警告，但不會顯示 ELB v2 服務的警告。發生這種情況是由於政策中存在適用於 ELB v2 的某些動作，但不適用於 ELB。您可以忽略這些對 ELB 發出的警告。

Example

eksctl

a. 使用叢集名稱取代 *my-cluster*，並使用帳戶 ID 取代 *111122223333*，然後執行命令。

```
eksctl create iamserviceaccount \
  --cluster=my-cluster \
  --namespace=kube-system \
  --name=aws-load-balancer-controller \
  --role-name AmazonEKSLoadBalancerControllerRole \
  --attach-policy-arn=arn:aws: iam::111122223333:policy/
AWSLoadBalancerControllerIAMPolicy \
  --approve
```

AWS CLI and kubectl

a. 擷取叢集的 OIDC 提供者 ID，並將其存放在變數中。

```
oidc_id=$(aws eks describe-cluster --name my-cluster --query
 "cluster.identity.oidc.issuer" --output text | cut -d '/' -f 5)
```

b. 判斷具有叢集 ID 的 IAM OIDC 供應商是否已在您的帳戶中。您需要為叢集和 IAM 設定 OIDC。

```
aws iam list-open-id-connect-providers | grep $oidc_id | cut -d "/" -f4
```

如果傳回輸出，表示您已有叢集的 IAM OIDC 提供者。如果未傳回任何輸出，則您必須為叢集建立 IAM OIDC 提供商。如需詳細資訊，請參閱the section called “IAM OIDC 供應商”。

c. 將以下內容複製到您的裝置。使用您的帳戶 ID 取代 *111122223333*。將 *region-code* 取代為您的叢集所在的 AWS 區域。使用前一個步驟傳回的輸出值取代 *EXAMPLED539D4633E53DE1B71EXAMPLE*。

```
cat >load-balancer-role-trust-policy.json <<EOF
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
```

```
                "Federated": "arn:aws: iam::111122223333:oidc-provider/
oidc.eks.region-code.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE"
            },
            "Action": "sts:AssumeRoleWithWebIdentity",
            "Condition": {
                "StringEquals": {
                    "oidc.eks.region-code.amazonaws.com/id/
EXAMPLED539D4633E53DE1B71EXAMPLE:aud": "sts.amazonaws.com",
                    "oidc.eks.region-code.amazonaws.com/id/
EXAMPLED539D4633E53DE1B71EXAMPLE:sub": "system:serviceaccount:kube-system:aws-
load-balancer-controller"
                }
            }
        }
    ]
}
EOF
```

d. 建立 IAM 角色。

```
aws iam create-role \
  --role-name AmazonEKSLoadBalancerControllerRole \
  --assume-role-policy-document file://"load-balancer-role-trust-policy.json"
```

e. 將必要的 Amazon EKS 受管 IAM 政策連接到 IAM 角色。使用您的帳戶 ID 取代 *111122223333*。

```
aws iam attach-role-policy \
  --policy-arn arn:aws: iam::111122223333:policy/
AWSLoadBalancerControllerIAMPolicy \
  --role-name AmazonEKSLoadBalancerControllerRole
```

f. 將以下內容複製到您的裝置。使用您的帳戶 ID 取代 *111122223333*。取代文字後，請執行已修改的命令以建立 `aws-load-balancer-controller-service-account.yaml` 檔案。

```
cat >aws-load-balancer-controller-service-account.yaml <<EOF
apiVersion: v1
kind: ServiceAccount
metadata:
  labels:
    app.kubernetes.io/component: controller
    app.kubernetes.io/name: aws-load-balancer-controller
  name: aws-load-balancer-controller
```

```
  namespace: kube-system
  annotations:
    eks.amazonaws.com/role-arn: arn:aws: iam::111122223333:role/
AmazonEKSLoadBalancerControllerRole
EOF
```

g. 在您的叢集上建立 Kubernetes 服務帳戶。名為 aws-load-balancer-controller 的 Kubernetes 服務帳戶使用您建立名為 *AmazonEKSLoadBalancerControllerRole* 的 IAM 角色進行註釋。

```
kubectl apply -f aws-load-balancer-controller-service-account.yaml
```

步驟 2：安裝 cert-manager

使用以下其中一個方法安裝 cert-manager，將憑證組態注入 Webhook。如需詳細資訊，請參閱 cert-manager 文件中的入門。

建議使用quay.io容器登錄檔來安裝 cert-manager。如果您的節點無法存取quay.io容器登錄檔，cert-manager請使用 Amazon ECR 安裝（請參閱下文）。

Example

Quay.io

a. 如果您的節點可以存取 quay.io 容器登錄檔，請安裝 cert-manager 以將憑證組態注入 Webhook。

```
kubectl apply \
    --validate=false \
    -f https://github.com/jetstack/cert-manager/releases/download/v1.13.5/cert-
manager.yaml
```

Amazon ECR

a. 使用以下其中一個方法安裝 cert-manager，將憑證組態注入 Webhook。如需詳細資訊，請參閱 cert-manager 文件中的入門。

b. 下載清單檔案。

```
curl -Lo cert-manager.yaml https://github.com/jetstack/cert-manager/releases/
download/v1.13.5/cert-manager.yaml
```

c. 提取以下映像並將其推送到節點可存取的儲存庫。有關如何提取、標記映像並將其推送到您儲存庫的詳細資訊，請參閱 the section called “將映像複製到儲存庫”。

```
quay.io/jetstack/cert-manager-cainjector:v1.13.5
quay.io/jetstack/cert-manager-controller:v1.13.5
quay.io/jetstack/cert-manager-webhook:v1.13.5
```

d. 使用您的登錄檔名稱取代清單檔案中三個映像的 quay.io。下列命令假設您的私有儲存庫名稱與來源儲存庫相同。將 *111122223333.dkr.ecr.region-code.amazonaws.com* 取代為您的私有登錄檔。

```
sed -i.bak -e 's|quay.io|111122223333.dkr.ecr.region-code.amazonaws.com|' ./cert-
manager.yaml
```

e. 套用清單檔案。

```
kubectl apply \
    --validate=false \
    -f ./cert-manager.yaml
```

步驟 3：Install AWS Load Balancer 控制器

1. 下載控制器規格。如需控制器的詳細資訊，請參閱 GitHub 上的文件。

```
curl -Lo v2_13_3_full.yaml https://github.com/kubernetes-sigs/aws-load-balancer-
controller/releases/download/v2.13.3/v2_13_3_full.yaml
```

2. 對檔案進行以下編輯。

a. 如果您已下載 v2_13_3_full.yaml 檔案，請執行下列命令以移除清單檔案中的 ServiceAccount 區段。如果您不移除本節，則會覆寫您在上一個步驟中對服務帳戶所做的必要註釋。如果您刪除控制器，移除此區段還可保留您在上一個步驟中建立的服務帳戶。

```
sed -i.bak -e '690,698d' ./v2_13_3_full.yaml
```

如果您下載不同的檔案版本，請在編輯器中開啟檔案並移除下列各行。

```
apiVersion: v1
kind: ServiceAccount
metadata:
  labels:
    app.kubernetes.io/component: controller
    app.kubernetes.io/name: aws-load-balancer-controller
  name: aws-load-balancer-controller
  namespace: kube-system
---
```

b. 使用叢集名稱取代 *my-cluster*，從而使用叢集名稱取代檔案的 Deployment spec 區段中的 your-cluster-name。

```
sed -i.bak -e 's|your-cluster-name|my-cluster|' ./v2_13_3_full.yaml
```

c. 如果您的節點無法存取 Amazon EKS Amazon ECR 映像儲存庫，則需要提取下列映像並將其推送到節點可存取的儲存庫。有關如何提取、標記映像並將其推送到您儲存庫的詳細資訊，請參閱 the section called “將映像複製到儲存庫”。

```
public.ecr.aws/eks/aws-load-balancer-controller:v2.13.3
```

將登錄檔的名稱新增至資訊清單。下列命令假設私有儲存庫的名稱與來源儲存庫相同，並將私有登錄檔的名稱新增至 檔案。將 *111122223333.dkr.ecr.region-code.amazonaws.com* 取代為您的登錄檔。此行假定您將私有儲存庫命名為與來源儲存庫相同的名稱。如果沒有，請將私有登錄檔名稱後的 eks/aws-load-balancer-controller 文字變更為儲存庫名稱。

```
sed -i.bak -e 's|public.ecr.aws/eks/aws-load-balancer-controller|
111122223333.dkr.ecr.region-code.amazonaws.com/eks/aws-load-balancer-
controller|' ./v2_13_3_full.yaml
```

d. （僅適用於 Fargate 或受限制 IMDS)

如果您要將控制器部署到限制存取 Amazon EC2 執行個體中繼資料服務 (IMDS) 的 Amazon EC2 節點，或者如果您要部署到 Fargate 或 Amazon EKS 混合節點，請在 following parameters下新增 - args:。 Amazon EC2

```
[...]
spec:
      containers:
        - args:
```

```
            - --cluster-name=your-cluster-name
            - --ingress-class=alb
            - --aws-vpc-id=vpc-xxxxxxxx
            - --aws-region=region-code


[...]
```

3. 套用檔案。

```
kubectl apply -f v2_13_3_full.yaml
```

4. 將 IngressClass 和 IngressClassParams 清單檔案下載至叢集。

```
curl -Lo v2_13_3_ingclass.yaml https://github.com/kubernetes-sigs/aws-load-balancer-
controller/releases/download/v2.13.3/v2_13_3_ingclass.yaml
```

5. 將清單檔案套用至叢集。

```
kubectl apply -f v2_13_3_ingclass.yaml
```

步驟 4：確認控制器已安裝

1. 確認控制器已安裝。

```
kubectl get deployment -n kube-system aws-load-balancer-controller
```

範例輸出如下。

```
NAME                           READY   UP-TO-DATE   AVAILABLE   AGE
aws-load-balancer-controller   2/2     2            2           84s
```

如果使用 Helm 進行部署，則會收到先前的輸出。如果使用 Kubernetes 清單檔案進行部署，則您只有一個複本。

2. 使用控制器佈建 AWS 資源之前，您的叢集必須符合特定需求。如需詳細資訊，請參閱the section called “應用程式負載平衡”及the section called “網路負載平衡”。

從已棄用的 ALB 傳入控制器遷移應用程式

本主題說明如何從已取代的控制器版本遷移。更具體地說，它描述了如何移除已棄用版本的 AWS Load Balancer控制器。

- 已棄用的版本無法升級。您必須先移除它們，然後安裝目前的版本。
- 已棄用版本包括：
  - AWS 適用於 Kubernetes 的 ALB 輸入控制器（「輸入控制器」），是 AWS Load Balancer控制器的前身。
  - 任何`0.1.`*`x`* 版本的 AWS Load Balancer控制器

移除已棄用的控制器版本

> **Note**
>
> 您可能已經使用 Helm 或使用 Kubernetes 資訊清單手動安裝已棄用版本。請使用最初使用安裝的工具來完成此程序。

1. 如果您安裝了 `incubator/aws-alb-ingress-controller` Helm Chart，請將其解除安裝。

   ```
   helm delete aws-alb-ingress-controller -n kube-system
   ```

2. 如果您安裝了第 `0.1.`*`x`* 版的 `eks-charts/aws-load-balancer-controller` 圖表，請將其解除安裝。由於 Webhook API 版本不相容，從 `0.1.`*`x`* 升級至 版本`1.0.0`無法運作。

   ```
   helm delete aws-load-balancer-controller -n kube-system
   ```

3. 檢查目前是否已安裝控制器。

   ```
   kubectl get deployment -n kube-system alb-ingress-controller
   ```

   如果未安裝控制器，這是輸出。

   ```
   Error from server (NotFound): deployments.apps "alb-ingress-controller" not found
   ```

   如果已安裝控制器，則此為輸出。

```
NAME                       READY UP-TO-DATE AVAILABLE AGE
alb-ingress-controller 1/1   1          1         122d
```

4. 輸入下列命令以移除控制器。

```
kubectl delete -f https://raw.githubusercontent.com/kubernetes-sigs/aws-alb-ingress-
controller/v1.1.8/docs/examples/alb-ingress-controller.yaml
kubectl delete -f https://raw.githubusercontent.com/kubernetes-sigs/aws-alb-ingress-
controller/v1.1.8/docs/examples/rbac-role.yaml
```

遷移至 AWS Load Balancer控制器

若要從 Kubernetes 的 ALB 輸入控制器遷移至 AWS Load Balancer控制器，您需要：

1. 移除 ALB 傳入控制器 （請參閱上述）。
2. 安裝 AWS Load Balancer控制器。
3. 將其他政策新增至 AWS Load Balancer控制器所使用的 IAM 角色。此政策允許 LBC 管理 ALB Ingress Controller for Kubernetes 建立的資源。
4. 下載 IAM 政策。此政策允許 AWS Load Balancer控制器管理 ALB 輸入控制器為 Kubernetes 建立的資源。您也可以檢視政策。

```
curl -O https://raw.githubusercontent.com/kubernetes-sigs/aws-load-balancer-
controller/v2.13.3/docs/install/iam_policy_v1_to_v2_additional.json
```

5. 如果您的叢集位於 AWS GovCloud （美國東部） 或 AWS GovCloud （美國西部） AWS 區域，請將 取代 `arn:aws:` 為 `arn:aws-us-gov:`。

```
sed -i.bak -e 's|arn:aws:|arn:aws-us-gov:|' iam_policy_v1_to_v2_additional.json
```

6. 建立 IAM 政策並記下傳回的 ARN。

```
aws iam create-policy \
  --policy-name AWSLoadBalancerControllerAdditionalIAMPolicy \
  --policy-document file://iam_policy_v1_to_v2_additional.json
```

7. 將 IAM 政策連接至 AWS Load Balancer控制器所使用的 IAM 角色。將 *your-role-name* 取代為角色的名稱，例如 `AmazonEKSLoadBalancerControllerRole`。

如果您使用 建立角色`eksctl`，則若要尋找建立的角色名稱，請開啟 AWS CloudFormation 主控台，然後選取 eksctl-***my-cluster***-addon-iamserviceaccount-kube-system-aws-load-balancer-controller 堆疊。選取 Resources (資源) 標籤。角色名稱位於 Physical ID (實體 ID) 欄。

```
aws iam attach-role-policy \
  --role-name your-role-name \
  --policy-arn arn:aws: iam::111122223333:policy/
AWSLoadBalancerControllerAdditionalIAMPolicy
```

在 Amazon EKS 叢集中管理 DNS 的 CoreDNS

> **Tip**
>
> 使用 Amazon EKS Auto Mode，您不需要安裝或升級聯網附加元件。自動模式包含 Pod 聯網和負載平衡功能。
> 如需詳細資訊，請參閱EKS 自動模式。

CoreDNS 是一個靈活、可擴展的 DNS 伺服器，可作為 Kubernetes 集群 DNS。當您啟動具有至少一個節點的 Amazon EKS 叢集時，依預設會部署兩個 CoreDNS 映像複本，而不論叢集中部署的節點數量為何。CoreDNS Pod 為叢集中的所有 Pod 提供名稱解析。如果您的叢集包含具有符合 CoreDNS 命名空間的 Fargate 設定檔，則 CoreDNS Pod 可以部署到 Fargate 節點`deployment`。如需 CoreDNS 的詳細資訊，請參閱 Kubernetes 文件中的使用 CoreDNS 進行服務探索。

CoreDNS 版本

下表列出每個 Kubernetes 版本的 Amazon EKS 附加元件類型最新版本。

Kubernetes 版本	CoreDNS 版本
1.33	v1.12.2-eksbuild.4
1.32	v1.11.4-eksbuild.14
1.31	v1.11.4-eksbuild.14
1.30	v1.11.4-eksbuild.14

Kubernetes 版本	CoreDNS 版本
1.29	v1.11.4-eksbuild.14
1.28	v1.10.1-eksbuild.35
1.27	v1.10.1-eksbuild.35
1.26	v1.9.3-eksbuild.39

Important

如果您要自我管理此附加元件，資料表中的版本可能與可用的自我管理版本不同。如需更新此附加元件之自我管理類型的詳細資訊，請參閱 the section called “更新 （ 自我管理 ）”。

重要的 CoreDNS 升級考量事項

- CoreDNS 更新使用 PodDisruptionBudget，以協助在更新程序期間維持 DNS 服務可用性。
- 為了改善 CoreDNS 部署的穩定性和可用性， `v1.9.3-eksbuild.6`和更新版本 `v1.10.1-eksbuild.3` 和 會使用 部署`PodDisruptionBudget`。如果您已部署現有的 `PodDisruptionBudget`，則升級到這些版本可能會失敗。如果升級失敗，完成下列其中一項任務即可解決問題：
  - 升級 Amazon EKS 附加元件時，選擇將覆寫現有設定作為衝突解決方案選項。如果您已對 部署進行其他自訂設定，請務必在升級之前備份設定，以便在升級之後重新套用其他自訂設定。
  - 移除現有的 `PodDisruptionBudget`，然後再次嘗試升級。
- 在 EKS 附加元件版本 `v1.9.3-eksbuild.3` 和更新版本 和更新版本 `v1.10.1-eksbuild.6` 和更新版本中，CoreDNS 部署會將 `readinessProbe` 設定為使用`/ready`端點。此端點會在 CoreDNS 的`Corefile`組態檔案中啟用。

  如果您使用自訂 `Corefile`，則必須將`ready`外掛程式新增至 組態，讓`/ready`端點在 CoreDNS 中處於作用中狀態，以供探查使用。
- 在 EKS 附加元件版本以`v1.9.3-eksbuild.7`及更新版本`v1.10.1-eksbuild.4`和更新版本中，您可以變更 `PodDisruptionBudget`。您可以使用下列範例的欄位，在選擇性組態設定中編輯附加元件並變更這些設定。此範例顯示預設值`PodDisruptionBudget`。

```
{
    "podDisruptionBudget": {
        "enabled": true,
        "maxUnavailable": 1
        }
}
```

您可以設定 maxUnavailable或 minAvailable，但無法在單一 中同時設定兩者PodDisruptionBudget。如需 的詳細資訊PodDisruptionBudgets，請參閱 Kubernetes 文件中的指定 PodDisruptionBudget。

請注意，如果您將 enabled設定為 false，PodDisruptionBudget則不會移除 。將此欄位設定為false之後，您必須刪除PodDisruptionBudget物件。同樣地，如果您在升級至具有 的版本後，編輯附加元件以使用較舊版本的附加元件 （降級附加元件）PodDisruptionBudget，PodDisruptionBudget則不會移除 。執行下列命令以刪除 PodDisruptionBudget：

```
kubectl delete poddisruptionbudget coredns -n kube-system
```

- 在 EKS 附加元件版本 v1.10.1-eksbuild.5和更新版本中，將預設公差從 變更為 node-role.kubernetes.io/master:NoSchedulenode-role.kubernetes.io/control-plane:NoSchedule，以符合 KEP 2067。如需 KEP 2067 的詳細資訊，請參閱 GitHub 上 Kubernetes Enhancement Proposals (KEP) 中的 KEP-2067：重新命名 kubeadm "master" 標籤和污點。 KEPs

  在 EKS 附加元件版本 v1.8.7-eksbuild.8 和更新版本 和更新版本 v1.9.3-eksbuild.9和更新版本中，兩個容錯都設定為與每個 Kubernetes 版本相容。

- 在 EKS 附加元件版本 v1.9.3-eksbuild.11 和 v1.10.1-eksbuild.7及更新版本中，CoreDNS 部署會設定 的預設值topologySpreadConstraints。如果多個可用區域中有節點可用，預設值可確保 CoreDNS Pod 會分散到可用區域。您可以設定將使用的自訂值，而非預設值。預設值如下：

```
topologySpreadConstraints:
  - maxSkew: 1
    topologyKey: topology.kubernetes.io/zone
    whenUnsatisfiable: ScheduleAnyway
    labelSelector:
      matchLabels:
```

```
        k8s-app: kube-dns
```

CoreDNS **v1.11** 升級考量事項

- 在 EKS 附加元件版本 `v1.11.1-eksbuild.4` 和更新版本中，容器映像是以 Amazon EKS Distro 維護的最小基礎映像為基礎，其中包含最少的套件且沒有 shell。如需詳細資訊，請參閱 Amazon EKS Distro。CoreDNS 映像的使用和疑難排解保持不變。

建立 CoreDNS Amazon EKS 附加元件

建立 CoreDNS Amazon EKS 附加元件。您必須先擁有叢集，才能建立附加元件。如需詳細資訊，請參閱the section called “建立叢集”。

1. 查看叢集上目前安裝了哪些附加元件版本。

   ```
   kubectl describe deployment coredns --namespace kube-system | grep coredns: | cut -
   d : -f 3
   ```

   範例輸出如下。

   ```
   v1.10.1-eksbuild.13
   ```

2. 查看叢集上安裝的附加元件類型。視您用來建立叢集的工具而定，您的叢集上目前可能沒有安裝 Amazon EKS 附加元件類型。使用您叢集的名稱取代 *my-cluster*。

   ```
   aws eks describe-addon --cluster-name my-cluster --addon-name coredns --query
    addon.addonVersion --output text
   ```

   如果傳回版本編號，表示您的叢集上安裝了 Amazon EKS 類型的附加元件，而且不需要完成此程序中的其餘步驟。如果傳回錯誤，表示叢集上未安裝 Amazon EKS 類型的附加元件。完成此程序的剩餘步驟以安裝該類型。

3. 儲存您目前安裝的附加元件。

   ```
   kubectl get deployment coredns -n kube-system -o yaml > aws-k8s-coredns-old.yaml
   ```

4. 使用 CLI AWS 建立附加元件。如果您想要使用 AWS Management Console 或 `eksctl` 建立附加元件，請參閱 `coredns` the section called “建立附加元件”並指定附加元件名稱。將隨後的命令複製到您的裝置。視需要對命令進行下列修改，然後執行修改後的命令。

- 使用您叢集的名稱取代 *my-cluster*。
- 將 *v1.11.3-eksbuild.1* 取代為叢集版本最新版本資料表中列出的最新版本。

```
aws eks create-addon --cluster-name my-cluster --addon-name coredns --addon-version
 v1.11.3-eksbuild.1
```

如果您已將自訂設定套用至與 Amazon EKS 附加元件預設設定衝突的目前附加元件，則建立可能會失敗。若建立失敗，您會收到錯誤，其中的訊息有助於您解決問題。或者，您可以將 `--resolve-conflicts OVERWRITE` 新增至上一條命令。這可讓附加元件覆寫任何現有的自訂設定。建立附加元件之後，您可以使用自訂設定進行更新。

5. 確認叢集的 Kubernetes 版本的最新版本附加元件已新增至叢集。使用您叢集的名稱取代 *my-cluster*。

```
aws eks describe-addon --cluster-name my-cluster --addon-name coredns --query
 addon.addonVersion --output text
```

建立附加元件的動作可能需要幾秒鐘的時間才能完成。

範例輸出如下。

```
v1.11.3-eksbuild.1
```

6. 如果您對原始附加元件制定自訂設定，請在建立 Amazon EKS 附加元件之前，使用在上一步中儲存的組態，以您的自訂設定更新 Amazon EKS 附加元件。如需更新附加元件的說明，請參閱 the section called “更新 (EKS 附加元件) ”。

更新 CoreDNS Amazon EKS 附加元件

更新 Amazon EKS 類型的附加元件。如果您尚未將 Amazon EKS 附加元件新增至叢集，請新增它或參閱 the section called “更新 （ 自我管理 ） ”。

開始之前，請檢閱升級考量事項。如需詳細資訊，請參閱the section called “重要的 CoreDNS 升級考量事項”。

1. 查看叢集上目前安裝了哪些附加元件版本。使用您的叢集名稱取代 *my-cluster*。

```
aws eks describe-addon --cluster-name my-cluster --addon-name coredns --query
 "addon.addonVersion" --output text
```

範例輸出如下。

```
v1.10.1-eksbuild.13
```

如果傳回的版本與最新版本資料表中叢集 Kubernetes 版本的版本相同，表示您已在叢集上安裝最新版本，不需要完成此程序的其餘部分。如果您收到錯誤，而不是輸出中的版本號碼，則表示叢集上沒有安裝 Amazon EKS 類型的附加元件。您必須先建立附加元件，才能使用此程序進行更新。

2. 儲存您目前安裝的附加元件。

```
kubectl get deployment coredns -n kube-system -o yaml > aws-k8s-coredns-old.yaml
```

3. 使用 CLI AWS 更新您的附加元件。如果您想要使用 AWS Management Console 或 `eksctl` 來更新附加元件，請參閱 the section called “更新 附加元件”。將隨後的命令複製到您的裝置。視需要對命令進行下列修改，然後執行修改後的命令。
   - 使用您叢集的名稱取代 *my-cluster*。
   - 將 *v1.11.3-eksbuild.1* 取代為叢集版本最新版本資料表中列出的最新版本。
   - `--resolve-conflicts`*PRESERVE* 選項會保留附加元件的現有組態值。如果您已設定附加元件設定的自訂值，而且不使用此選項，Amazon EKS 會以其預設值覆寫您的值。如果您使用此選項，建議您在更新生產叢集上的附加元件之前，測試非生產叢集上的任何欄位和值變更。如果您將此值變更為 OVERWRITE，則所有設定都會變更為 Amazon EKS 預設值。如果您已為任何設定設定自訂值，可能會以 Amazon EKS 預設值覆寫這些值。如果您將此值變更為 none，Amazon EKS 不會變更任何設定的值，但更新可能會失敗。若更新失敗，您會收到錯誤訊息，以協助您解決衝突。
   - 如果您未更新組態設定，`--configuration-values '{`*"replicaCount":3*`}'`請從命令中移除 。如果您要更新組態設定，請將 *"replicaCount"#3* 取代為您要設定的設定。在此範例中，CoreDNS 的複本數量設定為 3。您指定的值必須對組態結構描述有效。如果您不知道組態結構描述，請執行 `aws eks describe-addon-configuration --addon-name coredns --addon-version` *v1.11.3-eksbuild.1* ，將 *v1.11.3-eksbuild.1* 取代為您要查看組態的附加元件版本編號。結構描述會在輸出中傳回。如果您有任何現有的自訂組態，並且想要全部移除，同時將所有設定的值設定回 Amazon EKS 預設值，請從命令中移除 *"replicaCount":3* ，這樣就會得到空白的 {}。如需 CoreDNS 設定的詳細資訊，請參閱 Kubernetes 文件中的自訂 DNS 服務。

```
aws eks update-addon --cluster-name my-cluster --addon-name coredns --addon-version
 v1.11.3-eksbuild.1 \
```

```
    --resolve-conflicts PRESERVE --configuration-values '{"replicaCount":3}'
```

更新動作可能需要幾秒鐘的時間才能完成。

4. 確認附加元件版本已更新。使用您叢集的名稱取代 *my-cluster*。

```
aws eks describe-addon --cluster-name my-cluster --addon-name coredns
```

更新動作可能需要幾秒鐘的時間才能完成。

範例輸出如下。

```
{
    "addon": {
        "addonName": "coredns",
        "clusterName": "my-cluster",
        "status": "ACTIVE",
        "addonVersion": "v1.11.3-eksbuild.1",
        "health": {
            "issues": []
        },
        "addonArn": "arn:aws: eks:region:111122223333:addon/my-cluster/coredns/
d2c34f06-1111-2222-1eb0-24f64ce37fa4",
        "createdAt": "2023-03-01T16:41:32.442000+00:00",
        "modifiedAt": "2023-03-01T18:16:54.332000+00:00",
        "tags": {},
        "configurationValues": "{\"replicaCount\":3}"
    }
}
```

更新 CoreDNS Amazon EKS 自我管理附加元件

> **Important**
>
> 建議將附加元件的 Amazon EKS 類型新增到叢集，而不是使用附加元件的自我管理類型。如果您不熟悉類型之間的差異，請參閱 the section called “Amazon EKS 附加元件”。如需將 Amazon EKS 附加元件新增至叢集的詳細資訊，請參閱 the section called “建立附加元件”。如果您無法使用 Amazon EKS 附加元件，建議您提交有關為何無法傳送至容器藍圖 GitHub 儲存庫的問題。

開始之前，請檢閱升級考量事項。如需詳細資訊，請參閱the section called “重要的 CoreDNS 升級考量事項”。

1. 確認您的叢集上已安裝附加元件的自我管理類型。使用您叢集的名稱取代 *my-cluster*。

```
aws eks describe-addon --cluster-name my-cluster --addon-name coredns --query
 addon.addonVersion --output text
```

   如果傳回錯誤訊息，則表明您的叢集上安裝了附加元件的自我管理類型。完成此程序的剩餘步驟。如果傳回版本編號，則表明已在叢集上安裝附加元件的 Amazon EKS 類型。若要更新附加元件的 Amazon EKS 類型，請使用更新 CoreDNS Amazon EKS 附加元件中的程序，而不是使用此程序。如果您不熟悉附加元件類型之間的差異，請參閱 the section called “Amazon EKS 附加元件”。

2. 查看叢集上目前安裝了哪些容器映像版本。

```
kubectl describe deployment coredns -n kube-system | grep Image | cut -d ":" -f 3
```

   範例輸出如下。

```
v1.8.7-eksbuild.2
```

3. 如果您目前的 CoreDNS 版本是 `v1.5.0`或更新版本，但早於 CoreDNS 版本表中列出的版本，請略過此步驟。如果您目前的版本早於 `1.5.0`，則需要修改 `ConfigMap` for CoreDNS 以使用轉送附加元件，而不是代理附加元件。

   a. `ConfigMap` 使用以下命令開啟 。

```
kubectl edit configmap coredns -n kube-system
```

   b. 使用 `forward` 取代為下列行中的 `proxy`：儲存檔案，然後退出編輯器。

```
proxy . /etc/resolv.conf
```

4. 如果您原本在 Kubernetes `1.17`或更早版本上部署叢集，則可能需要從 CoreDNS 資訊清單中移除已中斷的行。

> **Important**
>
> 您必須先完成此步驟，才能更新至 CoreDNS 版本 `1.7.0`，但建議您完成此步驟，即使您更新至較早的版本。

a. 檢查您的 CoreDNS 資訊清單是否有這一行。

```
kubectl get configmap coredns -n kube-system -o jsonpath='{$.data.Corefile}' |
 grep upstream
```

如果未傳回任何輸出，則資訊清單不會有 行，您可以跳到下一個步驟來更新 CoreDNS。如果傳回輸出，那麼您需要移除該行。

b. 使用下列命令編輯 ConfigMap，移除檔案中有文字 upstream 的行。不要變更檔案中的任何其他內容。移除行之後，儲存變更。

```
kubectl edit configmap coredns -n kube-system -o yaml
```

5. 擷取您目前的 CoreDNS 映像版本：

```
kubectl describe deployment coredns -n kube-system | grep Image
```

範例輸出如下。

```
602401143452.dkr.ecr.region-code.amazonaws.com/eks/coredns:v1.8.7-eksbuild.2
```

6. 如果您要更新至 CoreDNS 1.8.3或更新版本，則需要將 endpointslices許可新增至 system:coredns Kubernetes clusterrole。

```
kubectl edit clusterrole system:coredns -n kube-system
```

在檔案的 rules 區段的現有許可行下，新增以下行。

```
[...]
- apiGroups:
  - discovery.k8s.io
  resources:
  - endpointslices
  verbs:
  - list
  - watch
[...]
```

7. 使用上一個步驟中傳回的輸出值取代 *602401143452* 和 *region-code*，以更新 CoreDNS 附加元件。將 *v1.11.3-eksbuild.1* 取代為 Kubernetes 版本最新版本資料表中列出的 CoreDNS 版本。

```
kubectl set image deployment.apps/coredns -n kube-system
 coredns=602401143452.dkr.ecr.region-code.amazonaws.com/eks/coredns:v1.11.3-
eksbuild.1
```

範例輸出如下。

```
deployment.apps/coredns image updated
```

8. 再次檢查容器映像版本，以確認其已更新至您在上一步中指定的版本。

```
kubectl describe deployment coredns -n kube-system | grep Image | cut -d ":" -f 3
```

範例輸出如下。

```
v1.11.3-eksbuild.1
```

擴展高 DNS 流量的 CoreDNS Pod

當您啟動具有至少一個節點的 Amazon EKS 叢集時，無論叢集中部署的節點數目為何，預設都會部署兩個 CoreDNS 映像複本的部署。CoreDNS Pod 為叢集中的所有 Pod 提供名稱解析。應用程式使用名稱解析來連線至叢集中的 Pod 和服務，以及連線至叢集外部的服務。隨著 Pod 的名稱解析（查詢）請求數量增加，CoreDNS Pod 可能會變得不堪重負和速度變慢，並拒絕 Pod 無法處理的請求。

若要處理 CoreDNS Pod 上增加的負載，請考慮 CoreDNS 的自動擴展系統。Amazon EKS 可以在 CoreDNS 的 EKS 附加元件版本中管理 CoreDNS 部署的自動擴展。此 CoreDNS 自動擴展器會持續監控叢集狀態，包括節點和 CPU 核心的數量。根據該資訊，控制器將動態調整 EKS 叢集中 CoreDNS 部署的複本數量。此功能適用於 CoreDNS v1.9和更新版本。如需哪些版本與 CoreDNS Autoscaling 相容的詳細資訊，請參閱下一節。

系統會根據叢集中的節點和 CPU 核心數量，使用動態公式自動管理 CoreDNS 複本，計算方式為 (numberOfNodes 除以 16) 和 (numberOfCPUCores以 256) 的最大值。它會評估 10 分鐘尖峰時段的需求，在需要時立即擴展以處理增加的 DNS 查詢負載，同時每 3 分鐘減少 33% 的複本，以維持系統穩定性並避免中斷。

我們建議您將此功能與其他 EKS Cluster Autoscaling 最佳實務搭配使用，以改善整體應用程式可用性和叢集可擴展性。

先決條件

若要讓 Amazon EKS 擴展您的 CoreDNS 部署，有三個先決條件：

- 您必須使用 CoreDNS 的 EKS 附加元件版本。
- 您的叢集必須至少執行最低叢集版本和平台版本。
- 您的叢集必須至少執行 CoreDNS 的 EKS 附加元件最低版本。

最低叢集版本

CoreDNS 的自動擴展由 Amazon EKS 管理的叢集控制平面中的新元件完成。因此，您必須將叢集升級至支援具有新元件之最低平台版本的 EKS 版本。

新的 Amazon EKS 叢集。若要部署叢集，請參閱 開始使用。叢集必須執行下表或更新版本中列出的其中一個 Kubernetes 版本和平台版本。請注意，也支援任何高於列出的 Kubernetes 和平台版本。您可以檢查目前的 Kubernetes 版本，方法是將下列命令中的 *my-cluster* 取代為您的叢集名稱，然後執行修改後的命令：

```
aws eks describe-cluster --name my-cluster --query cluster.version --output text
```

Kubernetes 版本	平台版本
1.29.3	eks.7
1.28.8	eks.13
1.27.12	eks.17
1.26.15	eks.18

> **Note**
>
> 也支援更新 Kubernetes 版本的每個平台版本，例如 1.30 Kubernetes 版本的 eks.1和更新版本。

最低 EKS 附加元件版本

Kubernetes 版本	1.29	1.28	1.27	1.26
	v1.11.1-eksbuild.9	v1.10.1-eksbuild.11	v1.10.1-eksbuild.11	v1.9.3-eksbuild.15

在 中設定 CoreDNS 自動調整規模 AWS Management Console

1. 確保您的叢集處於或高於最低叢集版本。

   Amazon EKS 會自動升級相同 Kubernetes 版本的平台版本之間的叢集，您無法自行啟動此程序。反之，您可以將叢集升級至下一個 Kubernetes 版本，而叢集將升級至該 K8s 版本和最新的平台版本。

   新的 Kubernetes 版本有時會加入重大變更。因此，我們建議您在更新生產叢集之前，先使用新 Kubernetes 版本的個別叢集來測試應用程式的行為。

   若要將叢集升級至新的 Kubernetes 版本，請遵循將現有叢集更新為新的 Kubernetes 版本中的程序。
2. 請確定您擁有 CoreDNS 的 EKS 附加元件，而非自我管理的 CoreDNS 部署。

   視您用來建立叢集的工具而定，您的叢集上目前可能沒有安裝 Amazon EKS 附加元件類型。若要查看叢集上安裝的附加元件類型，您可以執行下列命令。使用您叢集的名稱取代 `my-cluster`。

   ```
   aws eks describe-addon --cluster-name my-cluster --addon-name coredns --query
    addon.addonVersion --output text
   ```

   如果傳回版本號碼，您的叢集上安裝了 Amazon EKS 類型的附加元件，您可以繼續下一個步驟。如果傳回錯誤，表示叢集上未安裝 Amazon EKS 類型的附加元件。完成程序的其餘步驟 建立 CoreDNS Amazon EKS 附加元件，以使用 Amazon EKS 附加元件取代自我管理版本。
3. 確保您的 CoreDNS 的 EKS 附加元件版本等於或高於最低 EKS 附加元件版本。

   查看叢集上目前安裝了哪些附加元件版本。您可以在 中查看 AWS Management Console 或執行下列命令：

   ```
   kubectl describe deployment coredns --namespace kube-system | grep coredns: | cut -
   d : -f 3
   ```

範例輸出如下。

```
v1.10.1-eksbuild.13
```

將此版本與上一節中的最低 EKS 附加元件版本進行比較。如有需要，請依照更新 CoreDNS Amazon EKS 附加元件的程序，將 EKS 附加元件升級至更高版本。

4. 將自動調整規模組態新增至 EKS 附加元件的選用組態設定。
   a. 開啟 Amazon EKS 主控台。
   b. 在左側導覽窗格中，選取 Clusters (叢集)，然後選取您要為其設定附加元件之叢集的名稱。
   c. 選擇附加元件索引標籤。
   d. 選取 CoreDNS 附加元件方塊右上角的方塊，然後選擇編輯。
   e. 在設定 CoreDNS 頁面上：
      i. 選取您要使用的版本。我們建議您保留與上一個步驟相同的版本，並以不同的動作更新版本和組態。
      ii. 展開選用組態設定。
      iii. 在組態值`true`中輸入具有金鑰`"autoscaling":`和值的巢狀 JSON 物件的 JSON 金鑰`"enabled":`和值。 產生的文字必須是有效的 JSON 物件。如果此金鑰和值是文字方塊中唯一的資料，請以大括號 { } 括住該金鑰和值。下列範例顯示已啟用自動調整規模：

```
{
  "autoScaling": {
    "enabled": true
  }
}
```

      iv. （選用） 您可以提供自動擴展可以擴展 CoreDNS Pod 數量的最小值和最大值。

下列範例顯示已啟用自動調整規模，且所有選用索引鍵都有值。我們建議 CoreDNS Pod 的最小數量一律大於 2，以提供叢集中 DNS 服務的彈性。

```
{
  "autoScaling": {
    "enabled": true,
    "minReplicas": 2,
    "maxReplicas": 10
  }
```

```
}
```

f. 若要透過取代 CoreDNS Pod 來套用新組態，請選擇儲存變更。

Amazon EKS 會使用 Kubernetes Deployment for CoreDNS 的推出，將變更套用至 EKS 附加元件。 CoreDNS 您可以在 和 的附加元件更新歷史記錄中追蹤推展的狀態 AWS Management Console `kubectl rollout status deployment/coredns --namespace kube-system`。

`kubectl rollout` 具有下列命令：

```
kubectl rollout

history  -- View rollout history
pause    -- Mark the provided resource as paused
restart  -- Restart a resource
resume   -- Resume a paused resource
status   -- Show the status of the rollout
undo     -- Undo a previous rollout
```

如果推展時間過長，Amazon EKS 會復原推展，且具有附加元件更新類型且狀態為失敗的訊息會新增至附加元件的更新歷史記錄。若要調查任何問題，請從推展的歷史記錄開始，並在 CoreDNS Pod `kubectl logs`上執行以查看 CoreDNS 的日誌。

5. 如果更新歷史記錄中的新項目狀態為成功，則推展已完成，且附加元件在所有 CoreDNS Pod 中使用新組態。當您變更叢集中節點的節點和 CPU 核心數量時，Amazon EKS 會擴展 CoreDNS 部署的複本數量。

在 AWS 命令列界面中設定 CoreDNS 自動調整規模

1. 確保您的叢集處於或高於最低叢集版本。

Amazon EKS 會自動升級相同 Kubernetes 版本的平台版本之間的叢集，您無法自行啟動此程序。反之，您可以將叢集升級至下一個 Kubernetes 版本，而叢集將升級至該 K8s 版本和最新的平台版本。

新的 Kubernetes 版本有時會加入重大變更。因此，我們建議您在更新生產叢集之前，先使用新 Kubernetes 版本的個別叢集來測試應用程式的行為。

若要將叢集升級至新的 Kubernetes 版本，請遵循將現有叢集更新為新的 Kubernetes 版本中的程序。

2. 請確定您擁有 CoreDNS 的 EKS 附加元件，而非自我管理的 CoreDNS 部署。

   視您用來建立叢集的工具而定，您的叢集上目前可能沒有安裝 Amazon EKS 附加元件類型。若要查看叢集上安裝的附加元件類型，您可以執行下列命令。使用您叢集的名稱取代 `my-cluster`。

   ```
   aws eks describe-addon --cluster-name my-cluster --addon-name coredns --query
    addon.addonVersion --output text
   ```

   如果傳回版本編號，則表明已在叢集上安裝附加元件的 Amazon EKS 類型。如果傳回錯誤，表示叢集上未安裝 Amazon EKS 類型的附加元件。完成程序的其餘步驟 建立 CoreDNS Amazon EKS 附加元件，以使用 Amazon EKS 附加元件取代自我管理版本。

3. 確保您的 CoreDNS 的 EKS 附加元件版本等於或高於最低 EKS 附加元件版本。

   查看叢集上目前安裝了哪些附加元件版本。您可以在 中查看 AWS Management Console 或執行下列命令：

   ```
   kubectl describe deployment coredns --namespace kube-system | grep coredns: | cut -
   d : -f 3
   ```

   範例輸出如下。

   ```
   v1.10.1-eksbuild.13
   ```

   將此版本與上一節中的最低 EKS 附加元件版本進行比較。如有需要，請依照更新 CoreDNS Amazon EKS 附加元件的程序，將 EKS 附加元件升級至更高版本。

4. 將自動調整規模組態新增至 EKS 附加元件的選用組態設定。

   執行下列 AWS CLI 命令。將 `my-cluster` 取代為您的叢集名稱，並將 IAM 角色 ARN 取代為您正在使用的角色。

   ```
   aws eks update-addon --cluster-name my-cluster --addon-name coredns \
       --resolve-conflicts PRESERVE --configuration-values '{"autoScaling":
   {"enabled":true}}'
   ```

   Amazon EKS 會使用 Kubernetes Deployment for CoreDNS 的推出，將變更套用至 EKS 附加元件。 CoreDNS 您可以在 和 中的附加元件更新歷史記錄中追蹤推展的狀態 AWS Management Console `kubectl rollout status deployment/coredns --namespace kube-system`。

kubectl rollout 具有下列命令：

```
kubectl rollout

history  -- View rollout history
pause    -- Mark the provided resource as paused
restart  -- Restart a resource
resume   -- Resume a paused resource
status   -- Show the status of the rollout
undo     -- Undo a previous rollout
```

如果推展時間過長，Amazon EKS 會復原推展，且具有附加元件更新類型且狀態為失敗的訊息會新增至附加元件的更新歷史記錄。若要調查任何問題，請從推展的歷史記錄開始，並在 CoreDNS Pod kubectl logs上執行以查看 CoreDNS 的日誌。

5. （選用）您可以提供自動擴展可以擴展 CoreDNS Pod 數量的最小值和最大值。

   下列範例顯示已啟用自動調整規模，且所有選用索引鍵都有值。我們建議 CoreDNS Pod 的最小數量一律大於 2，以提供叢集中 DNS 服務的彈性。

```
aws eks update-addon --cluster-name my-cluster --addon-name coredns \
    --resolve-conflicts PRESERVE --configuration-values '{"autoScaling":
{"enabled":true,"minReplicas":2,"maxReplicas":10}}'
```

6. 執行下列命令，檢查附加元件的更新狀態：

```
aws eks describe-addon --cluster-name my-cluster --addon-name coredns
```

   如果您看到此行："status": "ACTIVE"，則推展已完成，且附加元件在所有 CoreDNS Pod 中使用新組態。當您變更叢集中節點的節點和 CPU 核心數量時，Amazon EKS 會擴展 CoreDNS 部署的複本數量。

使用 CoreDNS 指標監控 Kubernetes DNS 解析

CoreDNS 做為 EKS 附加元件，9153以kube-dns服務的 Prometheus 格式公開來自連接埠上 CoreDNS 的指標。您可以使用 Prometheus、Amazon CloudWatch 代理程式或任何其他相容的系統來湊集 (收集) 這些指標。

如需與 Prometheus 和 CloudWatch 代理程式相容的範例湊集組態，請參閱《Amazon CloudWatch 使用者指南》中的適用於 Prometheus 的 CloudWatch 代理程式組態。

在 Amazon EKS 叢集kube-proxy中管理

> **Tip**
>
> 使用 Amazon EKS Auto Mode，您不需要安裝或升級聯網附加元件。自動模式包含 Pod 聯網和負載平衡功能。
> 如需詳細資訊，請參閱EKS 自動模式。

建議將附加元件的 Amazon EKS 類型新增到叢集，而不是使用附加元件的自我管理類型。如果您不熟悉類型之間的差異，請參閱 the section called “Amazon EKS 附加元件”。如需將 Amazon EKS 附加元件新增至叢集的詳細資訊，請參閱 the section called “建立附加元件”。如果您無法使用 Amazon EKS 附加元件，建議您提交問題，說明為何無法前往容器藍圖 GitHub 儲存庫。

kube-proxy 附加元件會部署在 Amazon EKS 叢集中的每個 Amazon EC2 節點上。它在您的節點上維護網路規則，並啟用與 Pod 的網路通訊。附加元件不會部署到叢集中的 Fargate 節點。如需詳細資訊，請參閱 Kubernetes 文件中的 kube-proxy。

安裝為 Amazon EKS 附加元件

kube-proxy 版本

下表列出每個 Kubernetes 版本的 Amazon EKS 附加元件類型最新版本。

Kubernetes 版本	kube-proxy 版本
1.33	v1.33.0-eksbuild.2
1.32	v1.32.6-eksbuild.2
1.31	v1.31.10-eksbuild.2
1.30	v1.30.14-eksbuild.2
1.29	v1.29.15-eksbuild.10
1.28	v1.28.15-eksbuild.25

Kubernetes 版本	**kube-proxy** 版本
1.27	v1.27.16-eksbuild.35
1.26	v1.26.15-eksbuild.35

> **Note**
>
> 文件的較早版本不正確。 `v1.26.12` `v1.28.5`、 `v1.27.9`和 `kube-proxy`版本無法使用。如果您要自我管理此附加元件，資料表中的版本可能與可用的自我管理版本不同。

kube-proxy 容器映像

`kube-proxy` 容器映像是以 Amazon EKS Distro 維護的最小基礎映像為基礎，其中包含最少的套件且沒有 shell。如需詳細資訊，請參閱 Amazon EKS Distro。

下表列出每個 Amazon EKS 叢集版本的最新可用自我管理`kube-proxy`容器映像版本。

版本	kube-proxy
1.33	v1.33.0-minimal-eksbuild.2
1.32	v1.32.6-minimal-eksbuild.2
1.31	v1.31.10-minimal-eksbuild.2
1.30	v1.30.14-minimal-eksbuild.2
1.29	v1.29.15-minimal-eksbuild.10
1.28	v1.28.15-minimal-eksbuild.25
1.27	v1.27.16-minimal-eksbuild.35
1.26	v1.26.15-minimal-eksbuild.35

更新 Amazon EKS 附加元件類型時，您必須指定有效的 Amazon EKS 附加元件版本，該版本可能不是此資料表中列出的版本。這是因為 Amazon EKS 附加元件版本不一定符合更新此附加元件自我管理類

型時指定的容器映像版本。當您更新此附加元件的自我管理類型時，請指定此資料表中列出的有效容器映像版本。

更新 Kubernetes **kube-proxy**自我管理附加元件

> **Important**
>
> 建議將附加元件的 Amazon EKS 類型新增到叢集，而不是使用附加元件的自我管理類型。如果您不熟悉類型之間的差異，請參閱 the section called “Amazon EKS 附加元件”。如需將 Amazon EKS 附加元件新增至叢集的詳細資訊，請參閱 the section called “建立附加元件”。如果您無法使用 Amazon EKS 附加元件，建議您提交有關為何無法傳送至容器藍圖 GitHub 儲存庫的問題。

先決條件

- 現有 Amazon EKS 叢集。若要部署叢集，請參閱 開始使用。

考量事項

- Kube-proxy Amazon EKS 叢集上的相容性和扭曲政策與 Kubernetes 相同。了解如何驗證 Amazon EKS 附加元件版本與叢集的相容性。

1. 確認您的叢集上已安裝附加元件的自我管理類型。使用您叢集的名稱取代 *my-cluster*。

```
aws eks describe-addon --cluster-name my-cluster --addon-name kube-proxy --query
 addon.addonVersion --output text
```

   如果傳回錯誤訊息，則表明您的叢集上安裝了附加元件的自我管理類型。本主題中的其餘步驟用於更新附加元件的自我管理類型。如果傳回版本編號，則表明已在叢集上安裝附加元件的 Amazon EKS 類型。若要更新它，請使用更新 Amazon EKS 附加元件中的程序，而不是使用本主題中的程序。如果您不熟悉附加元件類型之間的差異，請參閱 the section called “Amazon EKS 附加元件”。

2. 查看叢集上目前安裝了哪些容器映像版本。

```
kubectl describe daemonset kube-proxy -n kube-system | grep Image
```

   範例輸出如下。

```
Image:    602401143452.dkr.ecr.region-code.amazonaws.com/eks/kube-proxy:v1.29.1-
eksbuild.2
```

在範例輸出中，*v1.29.1-eksbuild.2* 是安裝在叢集上的版本。

3. 將 *602401143452* 和 *region-code* 取代為上一個步驟中輸出的值，以更新kube-proxy附加元件。將 *v1.30.6-eksbuild.3* 取代為每個 Amazon EKS 叢集kube-proxy版本資料表的最新可用自我管理 kube-proxy 容器映像版本中列出的版本。 ???

> **Important**
>
> 每個影像類型的資訊清單不同，而且在預設或最小影像類型之間不相容。您必須使用與上一個影像相同的影像類型，以便進入點和引數相符。

```
kubectl set image daemonset.apps/kube-proxy -n kube-system kube-
proxy=602401143452.dkr.ecr.region-code.amazonaws.com/eks/kube-proxy:v1.30.6-
eksbuild.3
```

範例輸出如下。

```
daemonset.apps/kube-proxy image updated
```

4. 確認您的叢集上現在已安裝新版本。

```
kubectl describe daemonset kube-proxy -n kube-system | grep Image | cut -d ":" -f 3
```

範例輸出如下。

```
v1.30.0-eksbuild.3
```

5. 如果您在相同叢集中使用 x86和 Arm節點，且叢集是在 2020 年 8 月 17 日之前部署。然後，編輯您的 kube-proxy 清單檔案，以包含多個硬體架構的節點選取器，並使用以下命令。這是一次性操作。將選擇器新增至資訊清單後，您不需要在每次更新附加元件時新增它。如果您的叢集是在 2020 年 8 月 17 日或之後部署的，則 kube-proxy 已經具備多架構能力。

```
kubectl edit -n kube-system daemonset/kube-proxy
```

將下列節點選取器新增至編輯器中的檔案，然後儲存檔案。如需在編輯器中包含此文字的範例，請參閱 GitHub 上的 CNI 清單檔案檔案。這可讓 Kubernetes 根據節點的硬體架構提取正確的硬體映像。

```
- key: "kubernetes.io/arch"
  operator: In
  values:
  - amd64
  - arm64
```

6. 如果您的叢集最初是使用 Kubernetes 版本 1.14 或更新版本建立，則可以略過此步驟，因為 kube-proxy已包含此 Affinity Rule。如果您最初使用 Kubernetes 版本 1.13或更早版本建立 Amazon EKS 叢集，並打算在叢集中使用 Fargate 節點，請編輯資訊kube-proxy清單以包含NodeAffinity規則，以防止 kube-proxy Pod 在 Fargate 節點上排程。這是一次性編輯。將 Affinity Rule新增至資訊清單後，就不需要在每次更新附加元件時新增它。編輯您的 kube-proxy DaemonSet。

```
kubectl edit -n kube-system daemonset/kube-proxy
```

在編輯器中將以下內容Affinity Rule新增至檔案的 DaemonSet spec區段，然後儲存檔案。如需在編輯器中包含此文字的範例，請參閱 GitHub 上的 CNI 清單檔案檔案。

```
- key: eks.amazonaws.com/compute-type
  operator: NotIn
  values:
  - fargate
```

了解如何將工作負載和附加元件部署至 Amazon EKS

您的工作負載會部署在容器，這些容器會部署在 Kubernetes 的 Pod 中。Pod 包含一或多個容器。一般而言，一或多個提供相同服務的 Pod 會部署在 Kubernetes 服務中。部署多個提供相同服務的 Pod 後，您可以：

- 使用 AWS Management Console檢視在每個叢集上執行之工作負載的相關資訊。
- 使用 Kubernetes Vertical Pod Autoscaler 縱向擴展或縮減 Pod。
- 使用 Kubernetes Horizontal Pod Autoscaler 水平擴展或縮減滿足需求所需的 Pod 數量。
- 建立外部 （適用於可存取網際網路的 Pod) 或內部 （適用於私有 Pod) 網路負載平衡器，以平衡跨 Pod 的網路流量。負載平衡器會在 OSI 模型的第 4 層路由流量。
- 建立 Application Load Balancer 以平衡跨 Pod 的應用程式流量。Application Load Balancer 會在 OSI 模型的第 7 層路由流量。
- 如果您是初次使用 Kubernetes，本主題可協助您部署範例應用程式。
- 您可以使用 `externalIPs` 限制可指派給服務的 IP 地址。

在 Linux 上部署範例應用程式

在本主題中，您會將範例應用程式部署至 linux 節點上的叢集。

先決條件

- 具有至少一個節點的現有 Kubernetes 叢集。如果您沒有現有的 Amazon EKS 叢集，您可以使用 中的其中一個指南來部署叢集開始使用。
- 安裝在電腦上的 `Kubectl`。如需詳細資訊，請參閱the section called “設定 kubectl 和 eksctl”。
- 設定好的 `Kubectl`，以便與叢集通訊。如需詳細資訊，請參閱the section called “使用 kubectl 存取叢集”。
- 如果打算將範例工作負載部署至 Fargate，則必須擁有現有的 Fargate 設定檔，其中包含在本教學課程中建立的相同命名空間，也就是 `eks-sample-app`，除非您變更名稱。如果您使用 中的其中一個 gudes 建立叢集開始使用，則必須建立新的設定檔，或將命名空間新增至現有的設定檔，因為入門指南中建立的設定檔不會指定本教學課程中所使用的命名空間。您的 VPC 也必須至少有一個私有子網路。

雖然許多變數可在下列步驟中變更，但建議只在指定的位置變更變值。一旦您更了解 Kubernetes Pod、部署和服務，您就可以嘗試變更其他值。

建立命名空間。

命名空間可讓您將 Kubernetes 中的資源分組。如需詳細資訊，請參閱 Kubernetes 文件中的命名空間。如果您打算部署範例應用程式以使用 AWS Fargate 簡化運算管理，請確定在啟動時定義哪些 Pod 使用 AWS Fargate namespace的 值為 eks-sample-app。

```
kubectl create namespace eks-sample-app
```

建立 Kubernetes 部署

建立 Kubernetes 部署。此範例部署會從公有儲存庫提取容器映像，並將其中的三個複本 （ 個別 Pod) 部署到您的叢集。若要進一步了解，請參閱 Kubernetes 文件中的部署。

1. 將下列內容儲存到名為 eks-sample-deployment.yaml 的檔案中。範例應用程式中的容器不會使用網路儲存，但您可能需要應用程式。如需詳細資訊，請參閱應用程式資料儲存。
   - 在 kubernetes.io/arch 金鑰下方的 amd64 或 arm64 values 代表可以部署應用程式至其中一個硬體架構 (如果叢集中有兩個)。此操作可行，因為此映像是一個多架構映像，但並非所有映像都是。您可以在要從中提取映像的儲存庫中檢視映像詳細資訊，以判斷映像支援的硬體架構。部署不支援硬體架構類型的映像時，或您不希望將映像部署到其中時，請從資訊清單中移除該類型。如需詳細資訊，請參閱 Kubernetes 文件中的已知標籤、註釋和污點。
   - kubernetes.io/os: linux nodeSelector 表示，如果叢集中有 Linux 和 Windows 節點 (舉例來說)，則該映像只會部署至 Linux 節點。如需詳細資訊，請參閱 Kubernetes 文件中的已知標籤、註釋和污點。

   ```
   apiVersion: apps/v1
   kind: Deployment
   metadata:
     name: eks-sample-linux-deployment
     namespace: eks-sample-app
     labels:
       app: eks-sample-linux-app
   spec:
     replicas: 3
     selector:
       matchLabels:
         app: eks-sample-linux-app
     template:
   ```

```
    metadata:
      labels:
        app: eks-sample-linux-app
    spec:
      affinity:
        nodeAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
            nodeSelectorTerms:
            - matchExpressions:
              - key: kubernetes.io/arch
                operator: In
                values:
                - amd64
                - arm64
      containers:
      - name: nginx
        image: public.ecr.aws/nginx/nginx:1.23
        ports:
        - name: http
          containerPort: 80
        imagePullPolicy: IfNotPresent
      nodeSelector:
        kubernetes.io/os: linux
```

2. 將部署清單檔案套用至叢集。

```
kubectl apply -f eks-sample-deployment.yaml
```

建立服務

服務可讓您透過單一 IP 地址或名稱存取所有複本。如需詳細資訊，請參閱 Kubernetes 文件中的服務。雖然未在範例應用程式中實作，但如果您有應用程式需要與其他 AWS 服務互動，我們建議您為 Pod 建立 Kubernetes 服務帳戶，並將其與 AWS IAM 帳戶建立關聯。指定服務帳戶後，您的 Pod 僅能擁有您為它們指定與其他服務互動所需的最低許可。如需詳細資訊，請參閱the section called “具有 IRSA 的登入資料”。

1. 將下列內容儲存到名為 `eks-sample-service.yaml` 的檔案中。Kubernetes 會為服務指派自己的 IP 地址，這些地址只能從叢集內存取。若要從叢集外部存取服務，請部署 AWS Load Balancer 控制器，以將應用程式或網路流量負載平衡至服務。

```
apiVersion: v1
```

```
kind: Service
metadata:
  name: eks-sample-linux-service
  namespace: eks-sample-app
  labels:
    app: eks-sample-linux-app
spec:
  selector:
    app: eks-sample-linux-app
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80
```

2. 將服務資訊清單套用至叢集。

```
kubectl apply -f eks-sample-service.yaml
```

檢閱已建立的資源

1. 檢視 eks-sample-app 命名空間中存在的所有資源。

```
kubectl get all -n eks-sample-app
```

範例輸出如下。

```
NAME                                                READY   STATUS    RESTARTS   AGE
pod/eks-sample-linux-deployment-65b7669776-m6qxz    1/1     Running   0          27m
pod/eks-sample-linux-deployment-65b7669776-mmxvd    1/1     Running   0          27m
pod/eks-sample-linux-deployment-65b7669776-qzn22    1/1     Running   0          27m

NAME                                TYPE         CLUSTER-IP      EXTERNAL-IP   PORT(S)
   AGE
service/eks-sample-linux-service    ClusterIP    10.100.74.8     <none>        80/TCP
   32m

NAME                                          READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/eks-sample-linux-deployment   3/3     3            3           27m

NAME                                                      DESIRED   CURRENT   READY
 AGE
```

```
replicaset.apps/eks-sample-linux-deployment-776d8f8fd8    3         3         3
 27m
```

在輸出中，您會看到在先前步驟中所部署範例清單檔案中指定的服務和部署。您也會看到三個 Pod。這是因為範例清單檔案中指定了 3 個 `replicas`。如需 Pod 的詳細資訊，請參閱 Kubernetes 文件中的 Pod。Kubernetes 會自動建立`replicaset`資源，即使未在範例資訊清單中指定。如需 `ReplicaSets` 的詳細資訊，請參閱 Kubernetes 文件中的 ReplicaSet。

> **Note**
>
> Kubernetes 會維護清單檔案中指定的複本數量。如果這是生產部署，而且您希望 Kubernetes 水平擴展複本數量或垂直擴展 Pod 的運算資源，請使用 Scale Pod 部署搭配 Horizontal Pod Autoscaler，並使用調整 Pod 資源搭配 Vertical Pod Autoscaler 來執行此操作。

2. 檢視已部署服務的詳細資訊。

```
kubectl -n eks-sample-app describe service eks-sample-linux-service
```

範例輸出如下。

```
Name:              eks-sample-linux-service
Namespace:         eks-sample-app
Labels:            app=eks-sample-linux-app
Annotations:       <none>
Selector:          app=eks-sample-linux-app
Type:              ClusterIP
IP Families:       <none>
IP:                10.100.74.8
IPs:               10.100.74.8
Port:              <unset>  80/TCP
TargetPort:        80/TCP
Endpoints:         192.168.24.212:80,192.168.50.185:80,192.168.63.93:80
Session Affinity:  None
Events:            <none>
```

在先前的輸出中， 的值`IP:`是可從叢集內的任何節點或 Pod 到達的唯一 IP 地址，但無法從叢集外部到達。的值`Endpoints`是從 VPC 內指派給屬於服務一部分之 Pod 的 IP 地址。

3. 在先前步驟中檢視命名空間時，請檢視輸出中列出的其中一個 Pod 詳細資訊。將 *776d8f8fd8-78w66* 取代為其中一個 Pod 傳回的值。

```
kubectl -n eks-sample-app describe pod eks-sample-linux-deployment-65b7669776-m6qxz
```

縮寫範例輸出

```
Name:         eks-sample-linux-deployment-65b7669776-m6qxz
Namespace:    eks-sample-app
Priority:     0
Node:         ip-192-168-45-132.us-west-2.compute.internal/192.168.45.132
[...]
IP:           192.168.63.93
IPs:
  IP:           192.168.63.93
Controlled By:  ReplicaSet/eks-sample-linux-deployment-65b7669776
[...]
Conditions:
  Type              Status
  Initialized       True
  Ready             True
  ContainersReady   True
  PodScheduled      True
[...]
Events:
  Type    Reason     Age    From
 Message
  ----    ------     ----   ----
 -------
  Normal  Scheduled  3m20s  default-scheduler
 Successfully assigned eks-sample-app/eks-sample-linux-deployment-65b7669776-m6qxz to
 ip-192-168-45-132.us-west-2.compute.internal
[...]
```

在先前的輸出中， 的值IP:是唯一的 IP，從指派給節點所在子網路的 CIDR 區塊指派給 Pod。如果希望從不同的 CIDR 區塊指派 Pod IP 地址，則可以變更預設行為。如需詳細資訊，請參閱the section called “自訂聯網”。您也可以看到 Kubernetes 排程器在 上排程 PodNode，其 IP 地址為 *192.168.45.132*。

> **Tip**
>
> 您可以檢視 中的許多 Pod、服務、部署和其他 Kubernetes 資源詳細資訊，而不是使用命令列 AWS Management Console。如需詳細資訊，請參閱the section called “存取叢集資源”。

在 Pod 上執行 shell

1. 在先前步驟中描述的 Pod 上執行 Shell，將 *65b7669776-m6qxz* 取代為其中一個 Pod 的 ID。

```
kubectl exec -it eks-sample-linux-deployment-65b7669776-m6qxz -n eks-sample-app -- /
bin/bash
```

2. 從 Pod shell 中，檢視在上一個步驟中與部署一起安裝的 Web 伺服器輸出。您只需要指定服務名稱。根據預設，CoreDNS 會解析為服務的 IP 地址，該地址會與 Amazon EKS 叢集一起部署。

```
curl eks-sample-linux-service
```

   範例輸出如下。

```
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
[...]
```

3. 從 Pod Shell 中檢視 Pod 的 DNS 伺服器。

```
cat /etc/resolv.conf
```

   範例輸出如下。

```
nameserver 10.100.0.10
search eks-sample-app.svc.cluster.local svc.cluster.local cluster.local us-
west-2.compute.internal
options ndots:5
```

   在先前的輸出中，`10.100.0.10` 會自動指派為讓所有 Pod 部署至叢集的 `nameserver`。

4. 透過輸入 `exit` 中斷從 Pod 的連線。

5. 完成範例應用程式後，您可以使用下列命令移除範例命名空間、服務和部署。

```
kubectl delete namespace eks-sample-app
```

後續步驟

部署範例應用程式之後，建議您嘗試下列一些練習：

- 使用 Application Load Balancer 路由應用程式和 HTTP 流量
- 使用 Network Load Balancer 路由 TCP 和 UDP 流量

在 Windows 上部署範例應用程式

在本主題中，您會將範例應用程式部署到 Windows 節點上的叢集。

先決條件

- 具有至少一個節點的現有 Kubernetes 叢集。如果您沒有現有的 Amazon EKS 叢集，您可以使用 中的其中一個指南來部署叢集開始使用。您必須為叢集和至少一個 Amazon EC2 Windows 節點啟用 Windows 支援。
- 安裝在電腦上的 `Kubectl`。如需詳細資訊，請參閱the section called “設定 kubectl 和 eksctl”。
- 設定好的 `Kubectl`，以便與叢集通訊。如需詳細資訊，請參閱the section called “使用 kubectl 存取叢集”。
- 如果打算將範例工作負載部署至 Fargate，則必須擁有現有的 Fargate 設定檔，其中包含在本教學課程中建立的相同命名空間，也就是 `eks-sample-app`，除非您變更名稱。如果您在 中使用其中一個 gudes 建立叢集開始使用，則您必須建立新的設定檔，或將命名空間新增至現有的設定檔，因為入門指南中建立的設定檔不會指定本教學課程中所使用的命名空間。您的 VPC 也必須至少有一個私有子網路。

雖然許多變數可在下列步驟中變更，但建議只在指定的位置變更變值。一旦您更了解 Kubernetes Pod、部署和服務，您就可以嘗試變更其他值。

建立命名空間。

命名空間可讓您將 Kubernetes 中的資源分組。如需詳細資訊，請參閱 Kubernetes 文件中的命名空間。如果您打算部署範例應用程式以使用 AWS Fargate 簡化運算管理，請確定 定義啟動時使用 AWS Fargate 之 Pod namespace的 值為 eks-sample-app。

```
kubectl create namespace eks-sample-app
```

建立 Kubernetes 部署

此範例部署會從公有儲存庫提取容器映像，並將其三個複本 （ 個別 Pod) 部署到您的叢集。若要進一步了解，請參閱 Kubernetes 文件中的部署。

1. 將下列內容儲存到名為 eks-sample-deployment.yaml 的檔案中。範例應用程式中的容器不會使用網路儲存，但您可能有需要的應用程式。如需詳細資訊，請參閱應用程式資料儲存。
   - kubernetes.io/os: windows nodeSelector 表示，如果叢集中有 Windows 和 Linux 節點 (舉例來說)，則該映像只會部署至 Windows 節點。如需詳細資訊，請參閱 Kubernetes 文件中的已知標籤、註釋和污點。

     ```
     apiVersion: apps/v1
     kind: Deployment
     metadata:
       name: eks-sample-windows-deployment
       namespace: eks-sample-app
       labels:
         app: eks-sample-windows-app
     spec:
       replicas: 3
       selector:
         matchLabels:
           app: eks-sample-windows-app
       template:
         metadata:
           labels:
             app: eks-sample-windows-app
         spec:
           affinity:
             nodeAffinity:
               requiredDuringSchedulingIgnoredDuringExecution:
                 nodeSelectorTerms:
                 - matchExpressions:
     ```

```
              - key: kubernetes.io/arch
                operator: In
                values:
                - amd64
      containers:
      - name: windows-server-iis
        image: mcr.microsoft.com/windows/servercore:ltsc2019
        ports:
        - name: http
          containerPort: 80
        imagePullPolicy: IfNotPresent
        command:
        - powershell.exe
        - -command
        - "Add-WindowsFeature Web-Server; Invoke-WebRequest -UseBasicParsing
 -Uri 'https://dotnetbinaries.blob.core.windows.net/servicemonitor/2.0.1.6/
ServiceMonitor.exe' -OutFile 'C:\\ServiceMonitor.exe'; echo '<html><body><br/
><br/><marquee><H1>Hello EKS!!!<H1><marquee></body><html>' > C:\\inetpub\\wwwroot\
\default.html; C:\\ServiceMonitor.exe 'w3svc'; "
      nodeSelector:
        kubernetes.io/os: windows
```

2. 將部署清單檔案套用至叢集。

```
kubectl apply -f eks-sample-deployment.yaml
```

建立服務

服務可讓您透過單一 IP 地址或名稱存取所有複本。如需詳細資訊，請參閱 Kubernetes 文件中的服務。雖然未在範例應用程式中實作，但如果您的應用程式需要與其他 AWS 服務互動，我們建議您為 Pod 建立 Kubernetes 服務帳戶，並將其與 AWS IAM 帳戶建立關聯。指定服務帳戶後，您的 Pod 僅能擁有您為它們指定與其他服務互動所需的最低許可。如需詳細資訊，請參閱the section called “具有 IRSA 的登入資料”。

1. 將下列內容儲存到名為 `eks-sample-service.yaml` 的檔案中。Kubernetes 會為服務指派自己的 IP 地址，這些地址只能從叢集內存取。若要從叢集外部存取服務，請部署 AWS Load Balancer 控制器，以將應用程式或網路流量負載平衡至服務。

```
apiVersion: v1
kind: Service
metadata:
```

```
  name: eks-sample-windows-service
  namespace: eks-sample-app
  labels:
    app: eks-sample-windows-app
spec:
  selector:
    app: eks-sample-windows-app
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80
```

2. 將服務資訊清單套用至叢集。

```
kubectl apply -f eks-sample-service.yaml
```

檢閱已建立的資源

1. 檢視 eks-sample-app 命名空間中存在的所有資源。

```
kubectl get all -n eks-sample-app
```

範例輸出如下。

```
NAME                                                 READY   STATUS    RESTARTS   AGE
pod/eks-sample-windows-deployment-65b7669776-m6qxz   1/1     Running   0          27m
pod/eks-sample-windows-deployment-65b7669776-mmxvd   1/1     Running   0          27m
pod/eks-sample-windows-deployment-65b7669776-qzn22   1/1     Running   0          27m

NAME                                 TYPE         CLUSTER-IP      EXTERNAL-IP   PORT(S)
   AGE
service/eks-sample-windows-service   ClusterIP    10.100.74.8     <none>        80/
TCP    32m

NAME                                            READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/eks-sample-windows-deployment 3/3     3            3           27m

NAME                                                       DESIRED   CURRENT   READY
 AGE
```

```
replicaset.apps/eks-sample-windows-deployment-776d8f8fd8    3         3         3
   27m
```

在輸出中，您會看到在先前步驟中所部署範例清單檔案中指定的服務和部署。您也會看到三個 Pod。這是因為範例清單檔案中指定了 3 個 `replicas`。如需 Pod 的詳細資訊，請參閱 Kubernetes 文件中的 Pod。Kubernetes 會自動建立`replicaset`資源，即使未在範例資訊清單中指定。如需 `ReplicaSets` 的詳細資訊，請參閱 Kubernetes 文件中的 ReplicaSet。

> **Note**
>
> Kubernetes 會維護清單檔案中指定的複本數量。如果這是生產部署，而且您希望 Kubernetes 水平擴展複本數量或垂直擴展 Pod 的運算資源，請使用 Scale Pod 部署搭配 Horizontal Pod Autoscaler，並使用 調整 Pod 資源搭配 Vertical Pod Autoscaler 來執行此操作。

2. 檢視已部署服務的詳細資訊。

```
kubectl -n eks-sample-app describe service eks-sample-windows-service
```

範例輸出如下。

```
Name:              eks-sample-windows-service
Namespace:         eks-sample-app
Labels:            app=eks-sample-windows-app
Annotations:       <none>
Selector:          app=eks-sample-windows-app
Type:              ClusterIP
IP Families:       <none>
IP:                10.100.74.8
IPs:               10.100.74.8
Port:              <unset>  80/TCP
TargetPort:        80/TCP
Endpoints:         192.168.24.212:80,192.168.50.185:80,192.168.63.93:80
Session Affinity:  None
Events:            <none>
```

在先前的輸出中， 的值`IP:`是可從叢集中的任何節點或 Pod 到達的唯一 IP 地址，但無法從叢集外部到達。的值`Endpoints`是從 VPC 內指派給屬於服務一部分之 Pod 的 IP 地址。

3. 在先前步驟中檢視命名空間時，請檢視輸出中列出的其中一個 Pod 詳細資訊。將 *776d8f8fd8-78w66* 取代為其中一個 Pod 傳回的值。

```
kubectl -n eks-sample-app describe pod eks-sample-windows-deployment-65b7669776-m6qxz
```

縮寫範例輸出

```
Name:         eks-sample-windows-deployment-65b7669776-m6qxz
Namespace:    eks-sample-app
Priority:     0
Node:         ip-192-168-45-132.us-west-2.compute.internal/192.168.45.132
[...]
IP:           192.168.63.93
IPs:
  IP:           192.168.63.93
Controlled By:  ReplicaSet/eks-sample-windows-deployment-65b7669776
[...]
Conditions:
  Type              Status
  Initialized       True
  Ready             True
  ContainersReady   True
  PodScheduled      True
[...]
Events:
  Type    Reason     Age    From
 Message
  ----    ------     ----   ----
 -------
  Normal  Scheduled  3m20s  default-scheduler
 Successfully assigned eks-sample-app/eks-sample-windows-deployment-65b7669776-m6qxz
 to ip-192-168-45-132.us-west-2.compute.internal
[...]
```

在先前的輸出中，的值`IP:`是從指派給節點所在子網路的 CIDR 區塊指派給 Pod 的唯一 IP。如果希望從不同的 CIDR 區塊指派 Pod IP 地址，則可以變更預設行為。如需詳細資訊，請參閱the section called “自訂聯網”。您也可以看到 Kubernetes 排程器使用 Node IP 地址 *192.168.45.132* 在 上排程 Pod。

> **Tip**
>
> 與其使用命令列，您可以在 中檢視 Pod、服務、部署和其他 Kubernetes 資源的許多詳細資訊 AWS Management Console。如需詳細資訊，請參閱the section called “存取叢集資源”。

在 Pod 上執行 shell

1. 在先前步驟中描述的 Pod 上執行 Shell，將 *65b7669776-m6qxz* 取代為其中一個 Pod 的 ID。

```
kubectl exec -it eks-sample-windows-deployment-65b7669776-m6qxz -n eks-sample-app --
 powershell.exe
```

2. 從 Pod shell 中，檢視在上一個步驟中與您的部署一起安裝之 Web 伺服器的輸出。您只需要指定服務名稱。根據預設，CoreDNS 會解析為服務的 IP 地址，該地址會與 Amazon EKS 叢集一起部署。

```
Invoke-WebRequest -uri eks-sample-windows-service/default.html -UseBasicParsing
```

   範例輸出如下。

```
StatusCode        : 200
StatusDescription : OK
Content           : < h t m l > < b o d y > < b r / > < b r / > < m a r q u e e > < H
 1 > H e l l o
                      E K S ! ! ! < H 1 > < m a r q u e e > < / b o d y > < h t m l >
```

3. 從 Pod Shell 中檢視 Pod 的 DNS 伺服器。

```
Get-NetIPConfiguration
```

   縮寫輸出

```
InterfaceAlias       : vEthernet
[...]
IPv4Address          : 192.168.63.14
[...]
DNSServer            : 10.100.0.10
```

在先前的輸出中，10.100.0.10 會自動指派為讓所有 Pod 部署至叢集的 DNS 伺服器。

4. 透過輸入 exit 中斷從 Pod 的連線。
5. 完成範例應用程式後，您可以使用下列命令移除範例命名空間、服務和部署。

```
kubectl delete namespace eks-sample-app
```

後續步驟

部署範例應用程式之後，建議您嘗試下列一些練習：

- 使用 Application Load Balancer 路由應用程式和 HTTP 流量
- 使用 Network Load Balancer 路由 TCP 和 UDP 流量

使用 Vertical Pod Autoscaler 調整 Pod 資源

Kubernetes Vertical Pod Autoscaler 會自動調整 Pod 的 CPU 和記憶體保留，以協助「正確大小」您的應用程式。此調整可以改善叢集資源使用率，並釋放其他 Pod 的 CPU 和記憶體。本主題協助您將 Vertical Pod Autoscaler 部署到您的叢集，並驗證是否正常運作。

- 您擁有現有的 Amazon EKS 叢集。如果沒有，請參閱 開始使用。
- 您已安裝 Kubernetes 指標伺服器。如需詳細資訊，請參閱 the section called “指標伺服器”。
- 您所使用的 kubectl 用戶端已設定將與您的 Amazon EKS 叢集通訊。
- 裝置上已安裝 OpenSSL 1.1.1 或更新版本。

部署 Vertical Pod Autoscaler

在本節中，您會將 Vertical Pod Autoscaler 部署到叢集。

1. 開啟終端機視窗，導覽至您要下載 Vertical Pod Autoscaler 原始程式碼的目錄。
2. 複製 kubernetes/autoscalerGitHub GitHub儲存庫。

```
git clone https://github.com/kubernetes/autoscaler.git
```

3. 切換至 vertical-pod-autoscaler 目錄。

```
cd autoscaler/vertical-pod-autoscaler/
```

4. (選用) 如果您已部署另一個版本的 Vertical Pod Autoscaler，請使用下列命令將其移除。

```
./hack/vpa-down.sh
```

5. 如果您的節點無法存取registry.k8s.io容器登錄檔的網際網路，則需要提取下列映像，並將其推送至您自己的私有儲存庫。如需有關如何提取映像以及將映像推送到您的私有儲存庫的詳細資訊，請參閱 the section called “將映像複製到儲存庫”。

```
registry.k8s.io/autoscaling/vpa-admission-controller:0.10.0
registry.k8s.io/autoscaling/vpa-recommender:0.10.0
registry.k8s.io/autoscaling/vpa-updater:0.10.0
```

   如果您要將映像推送至私有 Amazon ECR 儲存庫，請將registry.k8s.io資訊清單中的 取代為登錄檔。使用您的帳戶 ID 取代 *111122223333*。將 *region-code* 取代為您叢集所在的 AWS 區域。以下命令假定您將自己的儲存庫命名為與清單檔案中儲存庫相同的名稱。如果您將儲存庫命名為不同的名稱，則您也需要變更它。

```
sed -i.bak -e 's/registry.k8s.io/111122223333.dkr.ecr.region-code.amazonaws.com/' ./
deploy/admission-controller-deployment.yaml
sed -i.bak -e 's/registry.k8s.io/111122223333.dkr.ecr.region-code.amazonaws.com/' ./
deploy/recommender-deployment.yaml
sed -i.bak -e 's/registry.k8s.io/111122223333.dkr.ecr.region-code.amazonaws.com/' ./
deploy/updater-deployment.yaml
```

6. 使用下列命令，將 Vertical Pod Autoscaler 部署到您的叢集。

```
./hack/vpa-up.sh
```

7. 確認垂直 Pod Autoscaler Pod 已成功建立。

```
kubectl get pods -n kube-system
```

   範例輸出如下。

```
NAME                                        READY   STATUS    RESTARTS   AGE
[...]
metrics-server-8459fc497-kfj8w              1/1     Running   0          83m
```

```
vpa-admission-controller-68c748777d-ppspd   1/1     Running   0          7s
vpa-recommender-6fc8c67d85-gljpl            1/1     Running   0          8s
vpa-updater-786b96955c-bgp9d                1/1     Running   0          8s
```

測試您的 Vertical Pod Autoscaler 安裝

在本節中，您將部署範例應用程式，以驗證 Vertical Pod Autoscaler 是否正常運作。

1. 使用以下命令部署 `hamster.yaml` Vertical Pod Autoscaler 範例。

```
kubectl apply -f examples/hamster.yaml
```

2. 從`hamster`範例應用程式取得 Pod。

```
kubectl get pods -l app=hamster
```

範例輸出如下。

```
hamster-c7d89d6db-rglf5   1/1     Running   0          48s
hamster-c7d89d6db-znvz5   1/1     Running   0          48s
```

3. 描述其中一個 Pod 以檢視其 `cpu`和 `memory`保留。使用在上一個步驟傳回的輸出中的其中一個 ID 取代 *`c7d89d6db-rglf5`*。

```
kubectl describe pod hamster-c7d89d6db-rglf5
```

範例輸出如下。

```
[...]
Containers:
  hamster:
    Container ID:  docker://
e76c2413fc720ac395c33b64588c82094fc8e5d590e373d5f818f3978f577e24
    Image:         registry.k8s.io/ubuntu-slim:0.1
    Image ID:      docker-pullable://registry.k8s.io/ubuntu-
slim@sha256:b6f8c3885f5880a4f1a7cf717c07242eb4858fdd5a84b5ffe35b1cf680ea17b1
    Port:          <none>
    Host Port:     <none>
    Command:
      /bin/sh
```

```
    Args:
      -c
      while true; do timeout 0.5s yes >/dev/null; sleep 0.5s; done
    State:          Running
      Started:      Fri, 27 Sep 2019 10:35:16 -0700
    Ready:          True
    Restart Count:  0
    Requests:
      cpu:        100m
      memory:     50Mi
[...]
```

您可以看到原始 Pod 保留 100 毫秒的 CPU 和 50 MB 的記憶體。對於此範例應用程式，100 millicpu 小於 Pod 需要執行的 Pod，因此會受到 CPU 限制。保留的記憶體也遠少於所需。Vertical Pod Autoscaler `vpa-recommender` 部署會分析倉鼠 Pod，以查看 CPU 和記憶體需求是否適當。如果需要調整， 會以更新的值`vpa-updater`重新啟動 Pod。

4. 等待 `vpa-updater`啟動新的倉鼠 Pod。這需要一兩分鐘。您可以使用下列命令來監控 Pod。

> **Note**
>
> 如果您不確定新 Pod 是否已啟動，請將 Pod 名稱與先前的清單進行比較。當新的 Pod 啟動時，您會看到新的 Pod 名稱。

```
kubectl get --watch Pods -l app=hamster
```

5. 當新的倉鼠 Pod 啟動時，請加以描述並檢視更新的 CPU 和記憶體保留。

```
kubectl describe pod hamster-c7d89d6db-jxgfv
```

範例輸出如下。

```
[...]
Containers:
  hamster:
    Container ID:
 docker://2c3e7b6fb7ce0d8c86444334df654af6fb3fc88aad4c5d710eac3b1e7c58f7db
    Image:         registry.k8s.io/ubuntu-slim:0.1
    Image ID:      docker-pullable://registry.k8s.io/ubuntu-
slim@sha256:b6f8c3885f5880a4f1a7cf717c07242eb4858fdd5a84b5ffe35b1cf680ea17b1
```

```
    Port:          <none>
    Host Port:     <none>
    Command:
      /bin/sh
    Args:
      -c
      while true; do timeout 0.5s yes >/dev/null; sleep 0.5s; done
    State:          Running
      Started:      Fri, 27 Sep 2019 10:37:08 -0700
    Ready:          True
    Restart Count:  0
    Requests:
      cpu:        587m
      memory:     262144k
[...]
```

在先前的輸出中，您可以看到 cpu 保留已增加到 587 millicpu，這已是原始值的 5 倍。memory 已增加到 262,144 KB，大約是 250 MiB 或原始值的 5 倍。此 Pod 的資源不足，而 Vertical Pod Autoscaler 以更適當的值更正了預估值。

6. 描述 hamster-vpa 資源以檢視新的建議。

```
kubectl describe vpa/hamster-vpa
```

範例輸出如下。

```
Name:         hamster-vpa
Namespace:    default
Labels:       <none>
Annotations:  kubectl.kubernetes.io/last-applied-configuration:
                {"apiVersion":"autoscaling.k8s.io/
v1beta2","kind":"VerticalPodAutoscaler","metadata":{"annotations":{},"name":"hamster-
vpa","namespace":"d...
API Version:  autoscaling.k8s.io/v1beta2
Kind:         VerticalPodAutoscaler
Metadata:
  Creation Timestamp:  2019-09-27T18:22:51Z
  Generation:          23
  Resource Version:    14411
  Self Link:           /apis/autoscaling.k8s.io/v1beta2/namespaces/default/
verticalpodautoscalers/hamster-vpa
  UID:                 d0d85fb9-e153-11e9-ae53-0205785d75b0
```

```
Spec:
  Target Ref:
    API Version:  apps/v1
    Kind:         Deployment
    Name:         hamster
Status:
  Conditions:
    Last Transition Time:  2019-09-27T18:23:28Z
    Status:                True
    Type:                  RecommendationProvided
  Recommendation:
    Container Recommendations:
      Container Name:  hamster
      Lower Bound:
        Cpu:     550m
        Memory:  262144k
      Target:
        Cpu:     587m
        Memory:  262144k
      Uncapped Target:
        Cpu:     587m
        Memory:  262144k
      Upper Bound:
        Cpu:     21147m
        Memory:  387863636
Events:          <none>
```

7. 當您完成範例應用程式的實驗後，您可以使用下列命令將其刪除。

```
kubectl delete -f examples/hamster.yaml
```

使用 Horizontal Pod Autoscaler 擴展 Pod 部署

Kubernetes Horizontal Pod Autoscaler 會根據資源的 CPU 使用率，自動擴展部署、複寫控制器或複本集中的 Pod 數量。這可協助您的應用程式水平擴展以滿足增加的需求，或在不需要資源時縮減，從而釋出節點供其他應用程式使用。當您設定目標 CPU 使用率百分比時，Horizontal Pod Autoscaler 會擴展或縮減您的應用程式，以嘗試滿足該目標。

Horizontal Pod Autoscaler 是 Kubernetes 中的標準 API 資源，只需要在 Amazon EKS 叢集上安裝指標來源 (例如 Kubernetes 指標伺服器) 即可運作。您不需要在叢集上部署或安裝 Horizontal Pod

Autoscaler，即可開始調整您的應用程式。如需詳細資訊，請參閱 Kubernetes 文件中的 Horizontal Pod Autoscaler。

使用此主題為您的 Amazon EKS 叢集準備 Horizontal Pod Autoscaler，並以範例應用程式驗證是否正常運作。

> **Note**
>
> 本主題是以 Kubernetes 文件中的 Horizontal Pod 自動擴展器演練為基礎。

- 您擁有現有的 Amazon EKS 叢集。如果沒有，請參閱 開始使用。
- 您已安裝 Kubernetes 指標伺服器。如需詳細資訊，請參閱 the section called “指標伺服器”。
- 您所使用的 `kubectl` 用戶端已設定將與您的 Amazon EKS 叢集通訊。

執行 Horizontal Pod Autoscaler 測試應用程式

在本節中，您將部署範例應用程式，以驗證 Horizontal Pod Autoscaler 是否正常運作。

> **Note**
>
> 此範例以 Kubernetes 文件中的 Horizontal Pod 自動擴展器演練為基礎。

1. 使用下列命令部署簡單的 Apache Web 伺服器應用程式。

   ```
   kubectl apply -f https://k8s.io/examples/application/php-apache.yaml
   ```

   此 Apache Web 伺服器 Pod 有 500 millicpu CPU 限制，且可在連接埠 80 上運作。
2. 建立 `php-apache` 部署的 Horizontal Pod Autoscaler 資源。

   ```
   kubectl autoscale deployment php-apache --cpu-percent=50 --min=1 --max=10
   ```

   此命令會建立自動擴展器，以部署 50% 的 CPU 使用率為目標，最少一個 Pod，最多十個 Pod。當平均 CPU 負載低於 50% 時，自動擴展器會嘗試將部署中的 Pod 數量減少到最少 1 個。當負載大於 50% 時，自動擴展器會嘗試增加部署中的 Pod 數量，最多增加 10 個。如需詳細資訊，請參閱 Kubernetes 文件中的 HorizontalPodAutoscaler 如何運作？。

3. 使用以下命令描述自動擴展器，以檢視其詳細資訊。

```
kubectl get hpa
```

範例輸出如下。

```
NAME         REFERENCE                TARGETS   MINPODS   MAXPODS   REPLICAS   AGE
php-apache   Deployment/php-apache    0%/50%    1         10        1          51s
```

如您所見，目前的 CPU 負載為 0%，因為伺服器上尚無負載。Pod 計數已經位於最低界限（一個），因此無法縮減。

4. 執行容器，為 Web 伺服器建立負載。

```
kubectl run -i \
    --tty load-generator \
    --rm --image=busybox \
    --restart=Never \
    -- /bin/sh -c "while sleep 0.01; do wget -q -O- http://php-apache; done"
```

5. 若要觀察向外擴展部署，請定期在不同的終端機中，從您執行上一個步驟的終端機執行下列命令。

```
kubectl get hpa php-apache
```

範例輸出如下。

```
NAME         REFERENCE                TARGETS    MINPODS   MAXPODS   REPLICAS   AGE
php-apache   Deployment/php-apache    250%/50%   1         10        5          4m44s
```

複本計數可能需要一分多鐘的時間才會增加。只要實際 CPU 百分比高於目標百分比，複本計數就會增加，最多可增加至 10。在這種情況下，它是 250%，因此 的數量會REPLICAS繼續增加。

> **Note**
>
> 您可能需要幾分鐘才會看到複本計數達到其最大值。例如，如果只需要 6 個複本，CPU 負載才會維持在 50% 或以下，則負載不會擴展超過 6 個複本。

6. 停止該負載。在您要產生載入的終端機視窗中，按住 Ctrl+C鍵來停止載入。您可以在正在監看擴展的終端機中再次執行下列命令，以監看複本縮減至 1。

```
kubectl get hpa
```

範例輸出如下。

```
NAME         REFERENCE                  TARGETS   MINPODS   MAXPODS   REPLICAS   AGE
php-apache   Deployment/php-apache      0%/50%    1         10        1          25m
```

> **Note**
>
> 縮減的預設時間範圍為 5 分鐘，因此需要一些時間才會看到複本計數再次達到 1，即使目前的 CPU 百分比為 0。時間範圍是可修改的。如需詳細資訊，請參閱 Kubernetes 文件中的 Horizontal Pod Autoscaler。

7. 當您完成範例應用程式的實驗後，請刪除 `php-apache` 資源。

```
kubectl delete deployment.apps/php-apache service/php-apache
 horizontalpodautoscaler.autoscaling/php-apache
```

使用 Network Load Balancer 路由 TCP 和 UDP 流量

> **Note**
>
> 新功能：Amazon EKS Auto Mode 會自動執行負載平衡的例行任務。如需詳細資訊，請參閱：
>
> - the section called “部署負載平衡器”
> - the section called “建立服務”

網路流量在 OSI 模型的 L4 處於負載平衡狀態。若要在 上負載平衡應用程式流量L7，您可以部署 Kubernetes `ingress`，其會佈建 AWS Application Load Balancer。如需詳細資訊，請參閱the section called “應用程式負載平衡”。若要進一步了解兩種負載平衡類型之間的差異，請參閱 AWS 網站上的 Elastic Load Balancing 功能。

當您建立類型為 `Service`的 Kubernetes 時`LoadBalancer`， AWS 雲端提供者負載平衡器控制器預設會建立 AWS Classic Load Balancer，但也可以建立 AWS Network Load Balancer。此控制器將來

僅接收關鍵錯誤修正。如需使用 AWS 雲端提供者負載平衡器 的詳細資訊，請參閱 Kubernetes 文件中的AWS 雲端提供者負載平衡器控制器。本主題中不涉及其使用方式。

建議您使用版本 2.7.2或更新版本的 AWS Load Balancer Controller，而非 AWS 雲端提供者負載平衡器控制器。The AWS Load Balancer 控制器會建立 AWS Network Load Balancer，但不會建立 AWS Classic Load Balancer。本主題的其餘部分是關於使用 AWS Load Balancer控制器。

AWS Network Load Balancer 可以將網路流量負載平衡到部署到 Amazon EC2 IP 和執行個體目標的 Pod、 AWS Fargate IP 目標，或部署到 Amazon EKS 混合節點做為 IP 目標。如需詳細資訊，請參閱 GitHub 上的 AWS Load Balancer 控制器。

先決條件

您必須先符合下列要求，才能使用 AWS Load Balancer網路流量。

- 擁有現有的叢集。如果您沒有現有的叢集，請參閱 開始使用。如果您需要更新現有叢集的版本，請參閱 the section called “更新 Kubernetes 版本”。
- 在您的叢集上部署 AWS Load Balancer控制器。如需詳細資訊，請參閱the section called “ AWS Load Balancer控制器”。我們建議使用 2.7.2 版或更新版本。
- 至少有一個子網路。若在可用區域中找到多個標記的子網路，控制器會根據其子網路 ID 依詞典編纂順序來選擇第一個子網路。子網路必須至少有 8 個可用的 IP 地址。
- 如果您使用的是 AWS Load Balancer控制器版本 2.1.1或更早版本，子網路必須標記如下。如果使用版本 2.1.2 或更新版本，則此標籤是選用的。如果您的多個叢集在相同 VPC 中執行，或多個 AWS 服務在 VPC 中共用子網路，並希望對每個叢集佈建負載平衡器的位置有更多控制權，則您可能想要標記子網路。如果您明確指定子網路 IDs做為服務物件上的註釋，則 Kubernetes 和 AWS Load Balancer控制器會直接使用這些子網路來建立負載平衡器。如果您選擇使用此方法來佈建負載平衡器，而且您可以略過下列私有和公有子網路標記需求，則不需要子網路標記。使用您的叢集名稱取代 *my-cluster*。
  - 索引鍵：kubernetes.io/cluster/<my-cluster>
  - 值：shared 或 owned
- 您的公有和私有子網路必須符合下列要求，除非您明確指定子網路 ID 作為服務或傳入物件上的註釋。如果您透過明確指定子網路 IDs做為服務或傳入物件的註釋來佈建負載平衡器，則 Kubernetes 和 AWS Load Balancer控制器會直接使用這些子網路來建立負載平衡器，而且不需要下列標籤。
  - 私有子網路：必須使用下列格式進行標記。這樣 Kubernetes 和 AWS Load Balancer控制器就會知道子網路可用於內部負載平衡器。如果您在 2020 年 3 月 26 日之後使用 eksctl或 Amazon EKS AWS AWS CloudFormation 範本來建立 VPC，則子網路會在建立時適當地加上標籤。如需

Amazon EKS AWS AWS CloudFormation VPC 範本的詳細資訊，請參閱 the section called “建立 VPC”。

- 索引鍵：kubernetes.io/role/internal-elb
- 值：1

- 公有子網路：必須使用下列格式進行標記。這是為了讓 Kubernetes 僅以這些子網路用於外部負載平衡器，而非在每個可用區域 (依據子網路 ID 的字典順序) 中選擇公有子網路。如果您在 2020 年 3 月 26 日之後使用 eksctl或 Amazon EKS AWS CloudFormation 範本建立 VPC，則子網路會在建立時適當地加上標籤。如需 Amazon EKS AWS CloudFormation VPC 範本的詳細資訊，請參閱 the section called “建立 VPC”。
  - 索引鍵：kubernetes.io/role/elb
  - 值：1

如果未明確新增子網路角色標籤，Kubernetes 服務控制器會檢查叢集 VPC 子網路的路由表，以判斷子網路是私有還是公有。我們建議您不要依賴此行為，而是明確新增私有或公有角色標籤。The AWS Load Balancer 控制器不會檢查路由表，並且需要私有和公有標籤才能成功自動探索。

考量事項

- 負載平衡器的組態是由加入至該服務清單檔案的註釋所控制。使用 AWS Load Balancer控制器的服務註釋與使用 AWS 雲端提供者負載平衡器控制器的服務註釋不同。部署服務之前，請務必檢閱 AWS Load Balancer控制器的註釋。
- 使用適用於 Kubernetes 的 Amazon VPC CNI 外掛程式時， AWS Load Balancer控制器可以將負載平衡至 Amazon EC2 IP 或執行個體目標和 Fargate IP 目標。使用替代相容 CNI 外掛程式時，控制器只能對執行個體目標進行負載平衡，除非您要對 Amazon EKS 混合節點進行負載平衡。對於混合節點，控制器可以負載平衡 IP 目標。如需 Network Load Balancer 目標類型的詳細資訊，請參閱 Network Load Balancer 使用者指南中的目標類型。
- 如果您想要在建立負載平衡器時或之後將標籤新增至負載平衡器，請在您的服務規格中新增下列註釋。如需詳細資訊，請參閱 AWS Load Balancer控制器文件中AWS 的資源標籤。

```
service.beta.kubernetes.io/aws-load-balancer-additional-resource-tags
```

- 您可以透過新增以下注釋將彈性 IP 地址指派至 Network Load Balancer。將###取代為彈性 IP 地址Allocation IDs的 。Allocation IDs 的數量必須與負載平衡器所使用的子網路數量一致。如需詳細資訊，請參閱 AWS Load Balancer 控制器說明文件。

```
service.beta.kubernetes.io/aws-load-balancer-eip-allocations: eipalloc-
xxxxxxxxxxxxxxxx,eipalloc-yyyyyyyyyyyyyyyyy
```

- Amazon EKS 會將一個傳入規則新增至節點的安全群組，以用於用戶端流量，並針對 VPC 中的每個負載平衡器子網路新增一個規則，以針對您建立的每個 Network Load Balancer 進行運作狀態檢查。如果 Amazon EKS 嘗試建立超過安全群組允許之規則數量上限配額的規則，則 `LoadBalancer` 類型的服務部署可能會失敗。如需詳細資訊，請參閱《Amazon VPC 使用者指南》中 Amazon VPC 配額的安全群組。請考慮下列選項，將超過安全群組規則數量上限的機會降至最低：
  - 請求提高每個安全群組規則的配額。如需詳細資訊，請參閱《Service Quotas 使用者指南》中的請求提高配額。
  - 使用 IP 目標，而不是執行個體目標。使用 IP 目標時，您可以共用相同目標連接埠的規則。您可以使用註釋手動指定負載平衡器子網路。如需詳細資訊，請參閱 GitHub 上的注釋。
  - 使用傳入，而不是類型服務 `LoadBalancer` 將流量傳送至您的服務。The AWS Application Load Balancer 所需的規則比 Network Load Balancer 少。您可在多個輸入之間共用 ALB。如需詳細資訊，請參閱the section called “應用程式負載平衡”。您無法跨多個服務共用 Network Load Balancer。
  - 將叢集部署到多個帳戶。
- 如果您的 Pod 在 Amazon EKS 叢集的 Windows 上執行，具有負載平衡器的單一服務最多可支援 1024 個後端 Pod。每個 Pod 都有自己的唯一 IP 地址。
- 我們建議僅使用負載平衡器控制器建立新的 Network AWS Load Balancer。嘗試取代使用 AWS 雲端提供者負載平衡器控制器建立的現有 Network Load Balancer 可能會導致多個 Network Load Balancer，這可能會導致應用程式停機。

建立 Network Load Balancer

您可以建立具有 IP 或執行個體目標的 Network Load Balancer。

建立網路負載平衡器 — IP 目標

- 您可以將 IP 目標與部署到 Amazon EC2 節點、Fargate 或 Amazon EKS 混合節點的 Pod 搭配使用。您的 Kubernetes 服務必須以 `LoadBalancer` 類型建立。如需詳細資訊，請參閱 Kubernetes 文件中的 LoadBalancer 類型。

  若要建立使用 IP 目標的負載平衡器，請將下列註解新增至服務清單檔案，然後部署您的服務。`external` 的值`aws-load-balancer-type`是導致 AWS Load Balancer控制器而非 AWS 雲

端提供者負載平衡器控制器建立 Network Load Balancer 的原因。您可以檢視具有注釋的範例服務清單檔案。

```
service.beta.kubernetes.io/aws-load-balancer-type: "external"
service.beta.kubernetes.io/aws-load-balancer-nlb-target-type: "ip"
```

> **Note**
>
> 如果您要將負載平衡載入 IPv6 Pod，請新增下列註釋。您只能在 IPv6 上負載平衡至 IP 目標，而不能負載平衡執行個體目標。如果沒有此註釋，負載平衡將在 IPv4 上執行。

```
service.beta.kubernetes.io/aws-load-balancer-ip-address-type: dualstack
```

依預設，使用 `internal` `aws-load-balancer-scheme` 建立 Network Load Balancer。您可以在叢集 VPC 中的任何子網路中啟動 Network Load Balancer，包括建立叢集時未指定的子網路。

Kubernetes 會檢查您子網路的路由表，以判別其為公有或私有。公有子網路具備使用網際網路閘道直接通向網際網路的路由，而私有子網路則不然。

若要在公有子網路中建立 Network Load Balancer 對 Amazon EC2 節點進行負載平衡 (Fargate 只能為私有)，請指定具有以下註釋的 `internet-facing`：

```
service.beta.kubernetes.io/aws-load-balancer-scheme: "internet-facing"
```

> **Note**
>
> 仍支援 `service.beta.kubernetes.io/aws-load-balancer-type: "nlb-ip"` 註釋以提供回溯相容性。不過，建議您針對新的負載平衡器使用先前的註釋，而不是 `service.beta.kubernetes.io/aws-load-balancer-type: "nlb-ip"`。

> **Important**
>
> 建立服務後，請勿編輯註釋。如果需要進行修改，請刪除服務物件，並使用此註釋所需的值重新建立服務物件。

建立網路負載平衡器 — 執行個體目標

- AWS 雲端提供者負載平衡器控制器只會使用執行個體目標建立 Network Load Balancer。版本 2.2.0 AWS Load Balancer控制器也會建立具有執行個體目標的 Network Load Balancer。建議您使用它，而不是 AWS 雲端提供者負載平衡器控制器，來建立新的 Network Load Balancer。您可以將 Network Load Balancer 執行個體目標與部署到 Amazon EC2 節點的 Pod 搭配使用，但不能使用 Fargate。若要在部署到 Fargate 的 Pod 之間負載平衡網路流量，則必須使用 IP 目標。

  若要將 Network Load Balancer 部署至私有子網路，則您的服務規格必須具有下列註釋。您可以檢視具有注釋的範例服務清單檔案。`external` 的值`aws-load-balancer-type`是導致 AWS Load Balancer控制器而非 AWS 雲端提供者負載平衡器控制器建立 Network Load Balancer 的原因。

  ```
  service.beta.kubernetes.io/aws-load-balancer-type: "external"
  service.beta.kubernetes.io/aws-load-balancer-nlb-target-type: "instance"
  ```

  依預設，使用 `internal` `aws-load-balancer-scheme` 建立 Network Load Balancer。若為內部 Network Load Balancer，您的 Amazon EKS 叢集必須設定至少使用 VPC 中的一個私有子網路。Kubernetes 會檢查您子網路的路由表，以判別其為公有或私有。公有子網路具備使用網際網路閘道直接通向網際網路的路由，而私有子網路則不然。

  若要在公有子網路中建立 Network Load Balancer 對 Amazon EC2 節點進行負載平衡，請指定具有以下註釋的 `internet-facing`：

  ```
  service.beta.kubernetes.io/aws-load-balancer-scheme: "internet-facing"
  ```

  > **Important**
  >
  > 建立服務後，請勿編輯註釋。如果需要進行修改，請刪除服務物件，並使用此註釋所需的值重新建立服務物件。

(選用) 部署範例應用程式

- 叢集 VPC 中必須至少有一個公有或私有子網路。
- 在您的叢集上部署 AWS Load Balancer控制器。如需詳細資訊，請參閱the section called “ AWS Load Balancer控制器”。我們建議使用 2.7.2 版或更新版本。

1. 如果您要部署到 Fargate，請確定 VPC 中有可用的私有子網路，並建立 Fargate 設定檔。如果您未部署到 Fargate，請略過此步驟。您可以透過執行下列命令來建立描述檔，也可以使用命令中 name 和 namespace 相同的值藉由 AWS Management Console 進行。使用自己的取代###。

```
eksctl create fargateprofile \
    --cluster my-cluster \
    --region region-code \
    --name nlb-sample-app \
    --namespace nlb-sample-app
```

2. 部署範例應用程式。

   a. 建立應用程式的命名空間。

```
kubectl create namespace nlb-sample-app
```

   b. 將下列內容儲存到電腦上名為 sample-deployment.yaml 的檔案中。

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nlb-sample-app
  namespace: nlb-sample-app
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
        - name: nginx
          image: public.ecr.aws/nginx/nginx:1.23
          ports:
            - name: tcp
              containerPort: 80
```

   c. 將清單檔案套用至叢集。

```
kubectl apply -f sample-deployment.yaml
```

3. 使用負載平衡至 IP 目標的網際網路取向的 Network Load Balancer 建立服務。

   a. 將下列內容儲存至電腦上名為 sample-service.yaml 檔案的 檔案。如果您要部署到 Fargate 節點，請移除該service.beta.kubernetes.io/aws-load-balancer-scheme: internet-facing行。

```
apiVersion: v1
kind: Service
metadata:
  name: nlb-sample-service
  namespace: nlb-sample-app
  annotations:
    service.beta.kubernetes.io/aws-load-balancer-type: external
    service.beta.kubernetes.io/aws-load-balancer-nlb-target-type: ip
    service.beta.kubernetes.io/aws-load-balancer-scheme: internet-facing
spec:
  ports:
    - port: 80
      targetPort: 80
      protocol: TCP
  type: LoadBalancer
  selector:
    app: nginx
```

   b. 將清單檔案套用至叢集。

```
kubectl apply -f sample-service.yaml
```

4. 確認服務已部署。

```
kubectl get svc nlb-sample-service -n nlb-sample-app
```

   範例輸出如下。

```
NAME             TYPE           CLUSTER-IP         EXTERNAL-IP
                                                 PORT(S)        AGE
sample-service   LoadBalancer   10.100.240.137   k8s-nlbsampl-nlbsampl-xxxxxxxxxx-
xxxxxxxxxxxxxxxx.elb.region-code.amazonaws.com  80:32400/TCP   16h
```

> **Note**
>
> *10.100.240.137* 和 *xxxxxxxxxx-xxxxxxxxxxxxxxxx* 的值將與範例輸出不同（它們對於您的負載平衡器是唯一的），而且 *us-west-2* 可能有所不同，取決於叢集所在的 AWS 區域。

5. 開啟 Amazon EC2 AWS Management Console。選取左側導覽窗格中的 Target Groups (目標群組) (在 Load Balancing (負載平衡) 下)。在名稱欄中，選取目標群組的名稱，其中負載平衡器欄中的值與上一個步驟中輸出EXTERNAL-IP欄中的一部分名稱相符。例如，k8s-default-samplese-*xxxxxxxxxx* 如果您的輸出與上一個輸出相同，您可以選擇名為 的目標群組。Target type (目標類型) 為 IP，因為其已在範例服務清單檔案中指定。
6. 選取 Target group (目標群組)，然後選擇 Targets (目標) 索引標籤。在 Registered targets (已註冊目標) 下，您應該會看到在上一個步驟中部署之三個複本的三個 IP 地址。待所有目標狀態變為 healthy (狀態良好) 再繼續。可能需要幾分鐘的時間，所有目標才能變為 healthy。目標在變更為 healthy 狀態前，可能處於 unhealthy 狀態。
7. 將流量傳送至 服務，將 *xxxxxxxxxx-xxxxxxxxxxxxxxxx* 和 *us-west-2* 取代為上一個步驟的輸出中傳回的值EXTERNAL-IP。如果您部署到私有子網路，則需要從 VPC 內的裝置檢視頁面，例如堡壘主機。如需詳細資訊，請參閱 AWS上的 Linux 堡壘主機。

   ```
   curl k8s-default-samplese-xxxxxxxxxx-xxxxxxxxxxxxxxxx.elb.region-code.amazonaws.com
   ```

   範例輸出如下。

   ```
   <!DOCTYPE html>
   <html>
   <head>
   <title>Welcome to nginx!</title>
   [...]
   ```

8. 完成範例部署、服務和命名空間後，請將其移除。

   ```
   kubectl delete namespace nlb-sample-app
   ```

使用 Application Load Balancer 路由應用程式和 HTTP 流量

> **Note**
>
> 新功能：Amazon EKS Auto Mode 會自動執行負載平衡的例行任務。如需詳細資訊，請參閱：
>
> - the section called “部署負載平衡器”
> - the section called “建立輸入類別”

當您建立 Kubernetes 時`ingress`，會佈建 an AWS Application Load Balancer (ALB)，以平衡應用程式流量。如需進一步了解，請參閱《Application Load Balancer 使用者指南》中的什麼是 Application Load Balancer？和 Kubernetes 文件中的傳入。ALBs可與部署到節點或 AWS Fargate 的 Pod 搭配使用。您可以將 ALB 部署到公有或私有子網路。

應用程式流量會在 OSI 模型的 L7 平衡。若要負載平衡 L4 的網路流量，您可以部署 `LoadBalancer` 類型的 Kubernetes `service`。此類型會佈建 AWS Network Load Balancer。如需詳細資訊，請參閱the section called “網路負載平衡”。若要進一步了解兩種負載平衡類型之間的差異，請參閱 AWS 網站上的 Elastic Load Balancing 功能。

先決條件

在將應用程式流量負載平衡到應用程式之前，您必須符合以下要求。

- 擁有現有的叢集。如果您沒有現有的叢集，請參閱 開始使用。如果您需要更新現有叢集的版本，請參閱 the section called “更新 Kubernetes 版本”。
- 在您的叢集上部署 AWS Load Balancer控制器。如需詳細資訊，請參閱the section called “ AWS Load Balancer控制器”。我們建議使用 `2.7.2` 版或更新版本。
- 至少兩個子網路位於不同的可用區域。 AWS Load Balancer控制器會從每個可用區域選擇一個子網路。當在可用區域中找到多個標記的子網路時，控制器會根據其子網路 ID 依詞典編纂順序來選擇子網路。每個子網路必須至少有 8 個可用的 IP 地址。

  如果您使用的是連接到工作者節點的多個安全群組，則必須僅標記一個安全群組，如下所示。使用您的叢集名稱取代 *`my-cluster`*。

  - 索引鍵：`kubernetes.io/cluster/<my-cluster>`
  - 值：`shared` 或 `owned`

- 如果您使用的是 AWS Load Balancer控制器版本 `2.1.1`或更早版本，子網路必須以下列格式標記。如果您使用的是版本 `2.1.2`或更新版本，標記是選用的。不過，如果出現下列任何情況，我們建議您標記子網路。您有多個叢集正在相同的 VPC 中執行，或是有多個 AWS 服務在 VPC 中共用子網路。或者，您希望更好地控制為每個叢集佈建負載平衡器的位置。使用您的叢集名稱取代 *`my-cluster`*。
  - 索引鍵：`kubernetes.io/cluster/<my-cluster>`
  - 值：`shared` 或 `owned`
- 您的公有和私有子網路必須符合下列要求，除非明確指定子網路 ID 作為服務或傳入物件上的註釋。假設您透過明確指定子網路 ID 作為服務或傳入物件上的註釋來佈建負載平衡器。在此情況下，Kubernetes 和 AWS 負載平衡器控制器會直接使用這些子網路來建立負載平衡器，而且不需要下列標籤。
  - 私有子網路：必須使用下列格式進行標記。這是為了讓 Kubernetes 和 AWS 負載平衡器控制器知道子網路可用於內部負載平衡器。如果您在 2020 年 3 月 26 日之後使用 `eksctl`或 Amazon EKS AWS CloudFormation 範本建立 VPC，子網路會在建立時適當標記。如需 Amazon EKS AWS CloudFormation VPC 範本的詳細資訊，請參閱 the section called “建立 VPC”。
    - 索引鍵：`kubernetes.io/role/internal-elb`
    - 值：`1`
  - 公有子網路：必須使用下列格式進行標記。這是為了讓 Kubernetes 知道只使用為外部負載平衡器指定的子網路。如此一來，Kubernetes 就不會在每個可用區域中選擇公有子網路（以詞典方式根據其子網路 ID)。如果您在 2020 年 3 月 26 日之後使用 `eksctl`或 Amazon EKS AWS CloudFormation 範本建立 VPC，子網路會在建立時適當標記。如需 Amazon EKS AWS CloudFormation VPC 範本的詳細資訊，請參閱 the section called “建立 VPC”。
    - 索引鍵：`kubernetes.io/role/elb`
    - 值：`1`

  如果未明確新增子網路角色標籤，Kubernetes 服務控制器會檢查叢集 VPC 子網路的路由表。這是為了判斷子網路是私有還是公有。我們建議您不要依賴此行為。而是明確地新增私有或公有角色標籤。AWS Load Balancer控制器不會檢查路由表。其還需要存在私有和公有標籤才能成功自動探索。
- 每當在具有 `kubernetes.io/ingress.class: alb`註釋的叢集上建立 Kubernetes 輸入 AWS 資源時，AWS Load Balancer控制器都會建立 ALBs 和必要的支援資源。輸入資源會設定 ALB 將 HTTP 或 HTTPS 流量路由到叢集中的不同 Pod。為了確保您的輸入物件使用 AWS Load Balancer 控制器，請將下列註釋新增至 Kubernetes 輸入規格。如需詳細資訊，請參閱 GitHub 上的傳入規格。

```
annotations:
```

```
    kubernetes.io/ingress.class: alb
```

> **Note**
>
> 如果您要將負載平衡載入 IPv6 Pod，請將下列註釋新增至您的輸入規格。您只能在 IPv6 上負載平衡至 IP 目標，而不能負載平衡執行個體目標。如果沒有此註釋，負載平衡將在 IPv4 上執行。

```
alb.ingress.kubernetes.io/ip-address-type: dualstack
```

- AWS Load Balancer控制器支援下列流量模式：
  - 執行個體 – 將叢集中的節點註冊作為 ALB 的目標。到達 ALB 的流量會針對NodePort您的服務路由至 ，然後代理至您的 Pod。這是預設的流量模式。您也可以使用 `alb.ingress.kubernetes.io/target-type: instance` 註釋明確指定它。

    > **Note**
    >
    > 您的 Kubernetes 服務必須指定 NodePort或 LoadBalancer類型，才能使用此流量模式。

  - IP：註冊 Pod 作為 ALB 的目標。到達 ALB 的流量會直接路由至您服務的 Pod。您必須指定 `alb.ingress.kubernetes.io/target-type: ip` 註釋才能使用此流量模式。當目標 Pod 在 Fargate 或 Amazon EKS 混合節點上執行時，需要 IP 目標類型。
- 要標記由控制器建立的 ALB，請新增以下註釋至控制器：`alb.ingress.kubernetes.io/tags`。如需 AWS Load Balancer控制器支援的所有可用註釋清單，請參閱 GitHub 上的傳入註釋。
- 升級或降級 ALB 控制器版本可能會導致依賴控制器的功能發生重大變化。如需有關每個版本中引入的重大變更的詳細資訊，請參閱 GitHub 上的 ALB 控制器版本備註。

將 ALBs與輸入群組重複使用

您可以使用 跨多個服務資源共用應用程式負載平衡器IngressGroups。

若要將傳入加入群組，請將下列註解新增至 Kubernetes 傳入資源規格。

```
alb.ingress.kubernetes.io/group.name: my-group
```

群組名稱必須：

- 是 63 個字元或以下的長度。
- 由小寫字母、數字、- 和 . 組成
- 開頭和結尾為字母或數字。

控制器會自動合併相同傳入群組中所有傳入的傳入規則。控制器透過單個 ALB 提供支援。在傳入中定義的大多數註解僅適用於該傳入定義的路徑。根據預設，輸入資源不屬於任何輸入群組。

> **Warning**
>
> 潛在的安全風險
> 只有當具有建立或修改輸入資源之 RBAC 許可的所有 Kubernetes 使用者都位於相同的信任界限內時，才能指定傳入的傳入群組。如果您使用群組名稱新增註釋，其他 Kubernetes 使用者可以建立或修改其傳入，使其屬於同一傳入群組。這樣做可能會產生不良行為，例如以較高優先順序規則覆寫現有規則。

您可以新增傳入資源的訂單編號。

```
alb.ingress.kubernetes.io/group.order: '10'
```

該編號可以是 1 到 1000 之間的數字。系統會先評估相同傳入群組中所有傳入數量的最低數量。系統會使用 0 值評估沒有此注釋的所有傳入。具有較高數量的重複規則可以覆寫具有較低數量的規則。根據預設，相同傳入群組中傳入之間的規則順序是由命名空間和名稱依詞典編纂順序決定。

> **Important**
>
> 確定相同傳入群組中的每個傳入都有唯一的優先順序編號。您無法跨輸入擁有重複的訂單號碼。

(選用) 部署範例應用程式

- 叢集 VPC 中必須至少有一個公有或私有子網路。
- 在您的叢集上部署 AWS Load Balancer控制器。如需詳細資訊，請參閱the section called “ AWS Load Balancer控制器”。我們建議使用 2.7.2 版或更新版本。

您可以在具有 Amazon EC2 節點、Fargate Pod 或兩者兼有的叢集上執行範例應用程式。

1. 如果您未部署到 Fargate，請略過此步驟。如果您要部署到 Fargate，請建立 Fargate 設定檔。您可以透過執行下列命令來建立描述檔，也可以使用命令中 `name` 和 `namespace` 相同的值藉由 AWS Management Console 進行。使用自己的取代`###`。

```
eksctl create fargateprofile \
    --cluster my-cluster \
    --region region-code \
    --name alb-sample-app \
    --namespace game-2048
```

2. 將遊戲 2048 部署為範例應用程式，以確認 AWS Load Balancer控制器建立 ALB AWS 作為輸入物件的結果。完成您要部署之子網路類型的步驟。

   a. 如果您要部署到使用 `IPv6` 系列建立的叢集中的 Pod，請跳到下一個步驟。

   - 公有：

```
kubectl apply -f https://raw.githubusercontent.com/kubernetes-sigs/aws-load-balancer-controller/v2.13.3/docs/examples/2048/2048_full.yaml
```

   - 私有：

     A. 下載清單檔案。

```
curl -O https://raw.githubusercontent.com/kubernetes-sigs/aws-load-balancer-controller/v2.13.3/docs/examples/2048/2048_full.yaml
```

     B. 編輯檔案並尋找顯示 `alb.ingress.kubernetes.io/scheme: internet-facing` 的行。

     C. 將`######### internal`並儲存檔案。

     D. 將清單檔案套用至叢集。

```
kubectl apply -f 2048_full.yaml
```

   b. 如果您要部署到使用 IPv6 系列建立的叢集中的 Pod，請完成下列步驟。

      i. 下載清單檔案。

```
curl -O https://raw.githubusercontent.com/kubernetes-sigs/aws-load-balancer-controller/v2.13.3/docs/examples/2048/2048_full.yaml
```

ii. 在編輯器中開啟檔案，然後在傳入規格中新增以下一行註釋。

```
alb.ingress.kubernetes.io/ip-address-type: dualstack
```

iii. 如果您要將負載平衡至內部 Pod，而不是面向網際網路的 Pod，請將 的行變更為 `alb.ingress.kubernetes.io/scheme: internet-facing` `alb.ingress.kubernetes.io/scheme: internal`

iv. 儲存檔案。

v. 將清單檔案套用至叢集。

```
kubectl apply -f 2048_full.yaml
```

3. 幾分鐘後，使用下列指令來驗證傳入資源已建立。

```
kubectl get ingress/ingress-2048 -n game-2048
```

範例輸出如下。

```
NAME           CLASS    HOSTS   ADDRESS
                     PORTS   AGE
ingress-2048   <none>   *       k8s-game2048-ingress2-xxxxxxxxxx-yyyyyyyyyy.region-
code.elb.amazonaws.com   80      2m32s
```

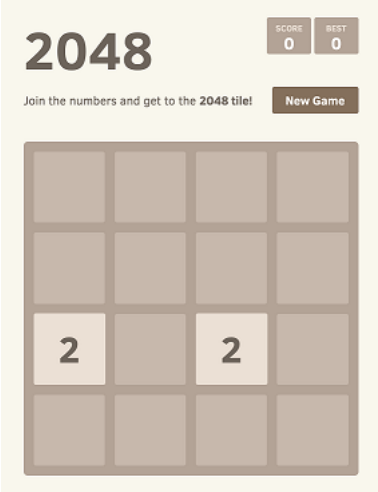
> **Note**
>
> 若在私有子網路中建立負載平衡器，則先前輸出中位於 ADDRESS 下的值前面會加上 `internal-`。

如果您的輸入在幾分鐘後未成功建立，請執行下列命令來檢視 AWS Load Balancer控制器日誌。這些日誌可能包含錯誤訊息，您可用於診斷部署的相關問題。

```
kubectl logs -f -n kube-system -l app.kubernetes.io/instance=aws-load-balancer-
controller
```

1. 如果您部署公有子網路，請開啟瀏覽器，再導覽至先前指令輸出中的 ADDRESS URL 查看範例應用程式。如果您沒有看到任何內容，請重新整理瀏覽器，然後再試一次。如果您部署到私有子網路，

則需要從 VPC 內的裝置檢視頁面，例如堡壘主機。如需詳細資訊，請參閱 AWS上的 Linux 堡壘主機。



2. 完成對範例應用程式的實驗後，請透過執行下列命令之一將其刪除。
   - 如果您應用了清單檔案，而不是應用下載的複本，請使用以下命令。

```
kubectl delete -f https://raw.githubusercontent.com/kubernetes-sigs/aws-load-balancer-controller/v2.13.3/docs/examples/2048/2048_full.yaml
```

   - 如果您下載並編輯清單檔案，請使用下列命令。

```
kubectl delete -f 2048_full.yaml
```

限制可指派給 服務的外部 IP 地址

Kubernetes 服務可透過以下方式從叢集內部連線：

- Kubernetes 自動指派的叢集 IP 地址
- 您為服務規則中的 `externalIPs` 屬性指定的 IP 地址。外部 IP 地址不是由 Kubernetes 管理，是叢集管理員的責任。使用 `externalIPs` 指定的外部 IP 地址不同於雲端提供商指派給類型服務 `LoadBalancer` 的外部 IP 地址。

如需進一步了解 Kubernetes 服務，請參閱 Kubernetes 文件中的服務。您可以限制可以在服務規格中為 `externalIPs` 指定的 IP 地址。

1. 部署 `cert-manager` 以管理網路掛鉤憑證。如需詳細資訊，請參閱 cert-manager 文件。

```
kubectl apply -f https://github.com/jetstack/cert-manager/releases/download/v1.5.4/
cert-manager.yaml
```

2. 確認 `cert-manager` Pod 正在執行。

```
kubectl get pods -n cert-manager
```

範例輸出如下。

```
NAME                                       READY   STATUS    RESTARTS   AGE
cert-manager-58c8844bb8-nlx7q              1/1     Running   0          15s
cert-manager-cainjector-745768f6ff-696h5   1/1     Running   0          15s
cert-manager-webhook-67cc76975b-4v4nk      1/1     Running   0          14s
```

3. 檢閱現有的 服務，確保其中沒有任何外部 IP 地址指派給他們，而這些地址不包含在您要限制地址的 CIDR 區塊中。

```
kubectl get services -A
```

範例輸出如下。

```
NAMESPACE                     NAME                                      TYPE
 CLUSTER-IP       EXTERNAL-IP     PORT(S)         AGE
cert-manager                  cert-manager                              ClusterIP
 10.100.102.137   <none>          9402/TCP        20m
cert-manager                  cert-manager-webhook                      ClusterIP
 10.100.6.136     <none>          443/TCP         20m
default                       kubernetes                                ClusterIP
 10.100.0.1       <none>          443/TCP         2d1h
externalip-validation-system  externalip-validation-webhook-service     ClusterIP
 10.100.234.179   <none>          443/TCP         16s
kube-system                   kube-dns                                  ClusterIP
 10.100.0.10      <none>          53/UDP,53/TCP   2d1h
my-namespace                  my-service                                ClusterIP
 10.100.128.10    192.168.1.1     80/TCP          149m
```

如果任何值是不在您想要限制存取之區塊內的 IP 地址，您將需要將地址變更為位於區塊內，然後重新部署服務。例如，上一個輸出中的 `my-service`服務已指派一個外部 IP 地址，該地址不在步驟 5 的 CIDR 區塊範例中。

4. 下載外部 IP Webhook 清單檔案。您也可以檢視 GitHub 上的 Webhook 來源程式碼

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/docs/externalip-webhook.yaml
```

5. 指定 CIDR 區塊。在編輯器中開啟下載的檔案，並在下列這幾行的開頭移除 \#。

```
#args:
#- --allowed-external-ip-cidrs=10.0.0.0/8
```

使用您自己的 CIDR 區塊取代 10.0.0.0/8。您可以指定需要的區塊，不限數量。如果指定多個區塊，則請在區塊之間加上逗號。

6. 如果您的叢集不在 us-west-2 AWS 區域中，請使用下列命令取代 amazonaws.com檔案中的 us-west-2602401143452、 和 。執行命令之前，請將 *region-code* 和 *111122223333* 取代為 Amazon EKS 附加元件檢視 Amazon 容器映像登錄檔清單中 AWS 的區域值。

```
sed -i.bak -e 's|602401143452|111122223333|' externalip-webhook.yaml
sed -i.bak -e 's|us-west-2|region-code|' externalip-webhook.yaml
sed -i.bak -e 's|amazonaws.com||' externalip-webhook.yaml
```

7. 將清單檔案套用至叢集。

```
kubectl apply -f externalip-webhook.yaml
```

嘗試使用為 指定的 IP 地址將服務部署到您的叢集externalIPs，而該 IP 地址未包含在您在指定 CIDR 區塊步驟中指定的區塊中，將會失敗。

將容器映像從一個儲存庫複製到另一個儲存庫

本主題說明如何從節點無法存取的儲存庫中提取容器映像，並將映像推送至節點可存取的儲存庫。您可以將映像推送到 Amazon ECR 或節點可存取的替代儲存庫。

- 電腦上已安裝和設定 Docker 引擎。如需相關說明，請參閱 Docker 文件中的安裝 Docker 引擎。
- 在您的裝置或 AWS CloudShell 上安裝和設定的 AWS 命令列界面 (AWS CLI) 版本 1.27.160 2.12.3或更新版本。若要檢查您目前的版本，請使用 aws --version | cut -d / -f2 | cut -d ' ' -f1。適用於 macOS 的 yum、 apt-get或 Homebrew 等套件管理員通常是最新版本 CLI AWS 後面的數個版本。若要安裝最新版本，請參閱《 AWS 命令列界面使用者指南》中的使用 aws 設定安裝 和快速組態。 https://docs.aws.amazon.com/cli/latest/userguide/cli-configure-quickstart.html#cli-configure-quickstart-config在 AWS CloudShell 中安裝的 AWS CLI 版本也可能是

最新版本後面的數個版本。若要更新它，請參閱《CloudShell AWS 使用者指南》中的將 CLI 安裝到您的主目錄。 AWS CloudShell

- 如果您希望節點透過 Amazon 網路從私有 Amazon ECR 儲存庫提取容器映像或將容器映像推送至私有 Amazon ECR 儲存庫，則為 Amazon ECR 的介面 VPC 端點。如需詳細資訊，請參閱《Amazon 彈性容器登錄檔使用者指南》中的建立適用於 Amazon ECR 的 VPC 端點。

請完成以下步驟，以從某個儲存庫中提取容器映像，並將其推送到您的儲存庫。在本主題提供的下列範例中，會提取 Kubernetes 指標協助程式的 Amazon VPC CNI 外掛程式映像。當您依照以下步驟操作時，請務必使用您的值取代 *example values*。

1. 如果您還沒有 Amazon ECR 儲存庫或其他儲存庫，請建立節點可存取的儲存庫。以下命令會建立 Amazon ECR 私有儲存庫。Amazon ECR 私有儲存庫名稱必須以字母開頭。名稱僅能包含小寫字母、數字、連字號 (-)、底線 (_) 和正斜線 (/)。如需詳細資訊，請參閱《Amazon 彈性容器登錄檔使用者指南》中的建立私有儲存庫。

   您可以使用所選的任何項目取代 *cni-metrics-helper*。最佳實務是為每個映像建立個別儲存庫。建議您採取此做法，因為映像標籤在儲存庫中必須是唯一的。將 *region-code* 取代為 AWS Amazon ECR 支援的 區域。

   ```
   aws ecr create-repository --region region-code --repository-name cni-metrics-helper
   ```

2. 判定節點需要提取的映像的登錄檔、儲存庫和標籤 (選用)。此資訊採用 `registry/repository[:tag]` 格式。

   許多有關於安裝映像的 Amazon EKS 主題都要求您套用清單檔案檔案或使用 Helm Chart 來安裝映像。不過，在您套用資訊清單檔案或安裝 Helm Chart 之前，請先檢視資訊清單或圖表`values.yaml`檔案的內容。如此一來，您可以判定要提取的登錄檔、儲存庫和標籤。

   例如，您可以在適用於 Kubernetes 指標協助程式的 Amazon VPC CNI 外掛程式資訊清單檔案中找到下列行。登錄檔為 `602401143452.dkr.ecr.us-west-2.amazonaws.com`，這是一個 Amazon ECR 私有登錄檔。儲存庫為 `cni-metrics-helper`。

   ```
   image: "602401143452.dkr.ecr.us-west-2.amazonaws.com/cni-metrics-helper:v1.12.6"
   ```

   您可能會看到如以下映像位置的變化：

   - 僅有 `repository-name:tag`。在此案例中，`docker.io` 通常是未經指定的登錄檔，因為在沒有指定登錄檔的情況下，依照預設 Kubernetes 會將其置於儲存庫名稱的前面。

- `repository-name/repository-namespace/repository:tag`。儲存庫命名空間為選用，但有時會由儲存庫擁有者指定，以將映像分類。例如，Amazon ECR Public Gallery 中的所有 Amazon EC2 映像都使用 `aws-ec2` 命名空間。

  在使用 Helm 安裝映像之前，請檢視 Helm `values.yaml` 檔案，以判定映像位置。例如，適用於 Kubernetes 指標協助程式的 Amazon VPC CNI 外掛程式的 values.yaml 檔案包含下列幾行。

```
image:
  region: us-west-2
  tag: v1.12.6
  account: "602401143452"
  domain: "amazonaws.com"
```

3. 提取清單檔案檔案中指定的容器映像。

   a. 如果您要從公有登錄檔中提取，例如 Amazon ECR Public Gallery，您可以跳到下一個子步驟，因為不需要身分驗證。在此範例中，您將向包含 CNI 指標協助程式映像的儲存庫的 Amazon ECR 私有登錄檔進行身分驗證。Amazon EKS 會在檢視 Amazon EKS 附加元件的 Amazon 容器映像登錄檔中列出的每個登錄檔中維護映像。您可以將 *602401143452* 和 *region-code* 取代為不同登錄檔的資訊，以對任何登錄檔進行身分驗證。AWS 每個支援 Amazon EKS 的區域都有單獨的登錄檔。

```
aws ecr get-login-password --region region-code | docker login --username AWS --
password-stdin 602401143452.dkr.ecr.region-code.amazonaws.com
```

   b. 提取映像。在此範例中，您將從上一個子步驟中所驗證的登錄表中提取。以您在上一個子步驟中提供的資訊取代 *602401143452* 和 *region-code*。

```
docker pull 602401143452.dkr.ecr.region-code.amazonaws.com/cni-metrics-
helper:v1.12.6
```

4. 標記與登錄檔、儲存庫和標籤一起提取的映像。以下範例假設您從清單檔案檔案中提取映像，並將其推送到您在第一個步驟中建立的 Amazon ECR 私有儲存庫。使用您的帳戶 ID 取代 *111122223333*。將 *region-code* 取代為您建立 Amazon ECR 私有儲存庫 AWS 的區域。

```
docker tag cni-metrics-helper:v1.12.6 111122223333.dkr.ecr.region-code.amazonaws.com/
cni-metrics-helper:v1.12.6
```

5. 對登錄檔進行身分驗證。在此範例中，您需對您在第一個步驟中建立的 Amazon ECR 私有登錄檔進行身分驗證。如需詳細資訊，請參閱《Amazon 彈性容器登錄檔使用者指南》中的登錄檔身分驗證。

```
aws ecr get-login-password --region region-code | docker login --username AWS --
password-stdin 111122223333.dkr.ecr.region-code.amazonaws.com
```

6. 將映像推送到您的儲存庫。在此範例中，您需將映像推送到您在第一個步驟中建立的 Amazon ECR 私有儲存庫。如需詳細資訊，請參閱《Amazon 彈性容器登錄檔使用者指南》中的推送 Docker 映像。

```
docker push 111122223333.dkr.ecr.region-code.amazonaws.com/cni-metrics-helper:v1.12.6
```

7. 針對您所推送的映像，使用 `registry/repository:tag` 更新您在先前步驟中用於確定映像的清單檔案檔案。如果您要使用 Helm Chart 安裝 ，通常可以選擇指定 `registry/repository:tag`。安裝圖表時，請為推送到儲存庫的映像指定 `registry/repository:tag`。

檢視 Amazon EKS 附加元件的 Amazon 容器映像登錄檔

將 AWS Amazon EKS 附加元件部署到叢集時，節點會從附加元件安裝機制中指定的登錄檔中提取所需容器映像，例如，安裝清單檔案或 Helm `values.yaml` 檔案。從 Amazon EKS Amazon ECR 私有儲存庫中提取映像。Amazon EKS 會將映像複寫到每個 Amazon EKS 支援 AWS 區域中的儲存庫。您的節點可以透過網際網路，從以下任何登錄檔中提取容器映像。或者，如果您在 VPC 中為 Amazon ECR (AWS PrivateLink) 建立界面 VPC 端點，則節點可以透過 Amazon 網路提取映像。登錄檔需要 IAM AWS 帳戶的身分驗證。您的節點使用 Amazon EKS 節點 IAM 角色進行驗證，該角色具有與其相關聯 AmazonEC2ContainerRegistryReadOnly 受管 IAM 政策的權限。

AWS 區域	登錄檔
af-south-1	877085696533.dkr.ecr.af-south-1.amazonaws.com
ap-east-1	800184023465.dkr.ecr.ap-east-1.amazonaws.com
ap-east-2	533267051163.dkr.ecr.ap-east-1.amazonaws.com

AWS 區域	登錄檔
ap-southeast-3	296578399912.dkr.ecr.ap-southeast-3.amazonaws.com
ap-south-2	900889452093.dkr.ecr.ap-south-2.amazonaws.com
ap-southeast-4	491585149902.dkr.ecr.ap-southeast-4.amazonaws.com
ap-south-1	602401143452.dkr.ecr.ap-south-1.amazonaws.com
ap-northeast-3	602401143452.dkr.ecr.ap-northeast-3.amazonaws.com
ap-northeast-2	602401143452.dkr.ecr.ap-northeast-2.amazonaws.com
ap-southeast-1	602401143452.dkr.ecr.ap-southeast-1.amazonaws.com
ap-southeast-2	602401143452.dkr.ecr.ap-southeast-2.amazonaws.com
ap-southeast-7	121268973566.dkr.ecr.ap-southeast-7.amazonaws.com
ap-northeast-1	602401143452.dkr.ecr.ap-northeast-1.amazonaws.com
cn-north-1	918309763551.dkr.ecr.cn-north-1.amazonaws.com.cn
cn-northwest-1	961992271922.dkr.ecr.cn-northwest-1.amazonaws.com.cn
eu-central-1	602401143452.dkr.ecr.eu-central-1.amazonaws.com

AWS 區域	登錄檔
eu-west-1	602401143452.dkr.ecr.eu-west-1.amazonaws.com
eu-west-2	602401143452.dkr.ecr.eu-west-2.amazonaws.com
eu-south-1	590381155156.dkr.ecr.eu-south-1.amazonaws.com
eu-west-3	602401143452.dkr.ecr.eu-west-3.amazonaws.com
eu-south-2	455263428931.dkr.ecr.eu-south-2.amazonaws.com
eu-north-1	602401143452.dkr.ecr.eu-north-1.amazonaws.com
eu-central-2	900612956339.dkr.ecr.eu-central-2.amazonaws.com
il-central-1	066635153087.dkr.ecr.il-central-1.amazonaws.com
mx-central-1	730335286997.dkr.ecr.mx-central-1.amazonaws.com
me-south-1	558608220178.dkr.ecr.me-south-1.amazonaws.com
me-central-1	759879836304.dkr.ecr.me-central-1.amazonaws.com
us-east-1	602401143452.dkr.ecr.us-east-1.amazonaws.com
us-east-2	602401143452.dkr.ecr.us-east-2.amazonaws.com

AWS 區域	登錄檔
us-west-1	602401143452.dkr.ecr.us-west-1.amazonaws.com
us-west-2	602401143452.dkr.ecr.us-west-2.amazonaws.com
ca-central-1	602401143452.dkr.ecr.ca-central-1.amazonaws.com
ca-west-1	761377655185.dkr.ecr.ca-west-1.amazonaws.com
sa-east-1	602401143452.dkr.ecr.sa-east-1.amazonaws.com
us-gov-east-1	151742754352.dkr.ecr.us-gov-east-1.amazonaws.com
us-gov-west-1	013241004608.dkr.ecr.us-gov-west-1.amazonaws.com

Amazon EKS 附加元件

附加元件是提供 Kubernetes 應用程式支援操作功能的軟體，但並不專用於應用程式。這包括可觀測性代理程式或 Kubernetes 驅動程式等軟體，允許叢集與基礎 AWS 資源互動，以進行聯網、運算和儲存。附加元件軟體通常由 Kubernetes 社群、雲端供應商或 AWS第三方供應商建置和維護。Amazon EKS 會自動為每個叢集安裝自我管理的附加元件，例如適用於 Kubernetes、 `kube-proxy`和 CoreDNS 的 Amazon VPC CNI 外掛程式。請注意，VPC CNI 附加元件與 Amazon EKS 混合節點不相容，而且不會部署到混合節點。您可以變更附加元件的預設設定，並在需要時進行更新。

Amazon EKS 附加元件提供 Amazon EKS 叢集精選附加元件集的安裝和管理。所有 Amazon EKS 附加元件都包含最新的安全修補程式、錯誤修正，並由 驗證 AWS 以使用 Amazon EKS。Amazon EKS 附加元件可讓您持續確保 Amazon EKS 叢集安全穩定，並降低安裝、設定和更新附加元件所需的工作量。如果是自我管理的附加元件 (例如 `kube-proxy`) 已在叢集上執行，且可作為 Amazon EKS 附加元件使用，則可以安裝 `kube-proxy` Amazon EKS 附加元件開始受益於 Amazon EKS 附加元件的功能。

您可以透過 Amazon EKS API 更新 Amazon EKS 附加元件的特定 Amazon EKS 受管組態欄位。您也可以在附加元件啟動後，直接在 Kubernetes 叢集中修改非由 Amazon EKS 管理的組態欄位。這包括定義附加元件適用的特定組態欄位。這些變更一旦進行，就不會被 Amazon EKS 覆寫。可以使用 Kubernetes 伺服器端套用功能來實現。如需詳細資訊，請參閱the section called “您可以自訂的欄位”。

您可以使用 Amazon EKS 附加元件搭配任何 Amazon EKS 節點類型。如需詳細資訊，請參閱管理運算。

您可以使用 Amazon EKS API、、 AWS CLI 和 來新增 AWS Management Console、更新或刪除 Amazon EKS 附加元件`eksctl`。您也可以使用 AWS CloudFormation 建立 Amazon EKS 附加元件。

考量事項

當您使用 Amazon EKS 附加元件時，請考慮下列事項：

- 若要設定叢集的附加元件，則您的 IAM 主體必須具有 IAM 許可才能使用附加元件。如需詳細資訊，請參閱 Amazon Elastic Kubernetes Service 定義的動作中在其名稱中具有 Addon 的動作。
- Amazon EKS 附加元件會在您為叢集佈建或設定的節點上執行。節點類型包括 Amazon EC2 執行個體、Fargate 和混合節點。
- 您可以修改非由 Amazon EKS 管理的欄位，以自訂 Amazon EKS 附加元件的安裝。如需詳細資訊，請參閱the section called “您可以自訂的欄位”。
- 如果您使用 建立叢集 AWS Management Console，Amazon EKS `kube-proxy`、適用於 Kubernetes 的 Amazon VPC CNI 外掛程式和 CoreDNS Amazon EKS 附加元件會自動新增至您的叢集。若您使用 `eksctl` 來建立具有 `config` 檔案的叢集，則 `eksctl` 也可以使用 Amazon EKS 附加元件建立叢集。如果您使用 建立叢集，`eksctl`而不使用 `config` 檔案或任何其他工具，則會安裝自我管理的 `kube-proxy`、適用於 Kubernetes 的 Amazon VPC CNI 外掛程式，以及 CoreDNS 附加元件，而不是 Amazon EKS 附加元件。您可以自行管理這些附加元件，也可以在建立叢集之後手動新增 Amazon EKS 附加元件。無論您使用何種方法來建立叢集，VPC CNI 附加元件都不會安裝在混合節點上。
- 會將 `eks:addon-cluster-admin`ClusterRoleBinding繫結`cluster-admin`ClusterRole至 `eks:addon-manager` Kubernetes 身分。角色具有`eks:addon-manager`身分的必要許可，以建立 Kubernetes 命名空間並將附加元件安裝到命名空間。若已移除 `eks:addon-cluster-admin` ClusterRoleBinding，Amazon EKS 叢集會繼續運作，但 Amazon EKS 將無法再管理任何附加元件。以下平台版本開始的所有叢集皆會使用新的 `ClusterRoleBinding`。

- 的 EKS 附加元件子集 AWS 已驗證與 Amazon EKS 混合節點的相容性。如需詳細資訊，請參閱 上的相容性資料表the section called “ AWS 附加元件”。

Amazon EKS Auto 模式的考量事項

Amazon EKS Auto 模式包含提供基本叢集功能的功能，包括：

- Pod 聯網
- 服務聯網
- 叢集 DNS
- 自動擴展
- 區塊儲存
- 負載平衡器控制器
- Pod Identity 代理程式
- 節點監控代理程式

使用自動模式運算時，許多常用的 EKS 附加元件會變成備援，例如：

- Amazon VPC CNI
- kube-proxy
- CoreDNS
- Amazon EBS CSI 驅動程式
- EKS Pod 身分識別代理程式

不過，如果您的叢集將自動模式與其他運算選項結合，例如自我管理的 EC2 執行個體、受管節點群組或 AWS Fargate，這些附加元件仍然是必要的。 AWS 具有增強型 EKS 附加元件與反親和性規則，可自動確保附加元件 Pod 僅排程在支援的運算類型上。此外，使用者現在可以利用 EKS 附加元件 `DescribeAddonVersions` API 來驗證每個附加元件及其特定版本的支援的 computeTypes。此外，使用 EKS Auto 模式時，上述控制器會在 AWS 擁有的基礎設施上執行。因此，除非您使用 EKS 自動模式搭配其他類型的運算，否則您甚至無法在帳戶中看到它們。在這種情況下，您會看到您在叢集上安裝的控制器。

如果您打算在現有叢集上啟用 EKS Auto Mode，您可能需要升級特定附加元件的版本。如需詳細資訊，請參閱 the section called “必要的附加元件版本” for EKS Auto Mode。

支援

AWS 會發佈具有不同支援層級的多種附加元件類型。

- AWS 附加元件：這些附加元件由 建置並完全支援 AWS。
  - 使用 AWS 附加元件來處理其他 AWS 服務，例如 Amazon EFS。
  - 如需詳細資訊，請參閱the section called “ AWS 附加元件”。
- AWS Marketplace 附加元件：這些附加元件由 掃描 AWS 並由獨立 AWS 合作夥伴支援。
  - 使用 Marketplace 附加元件為您的叢集新增重要且複雜的功能，例如使用 Splunk 進行監控。
  - 如需詳細資訊，請參閱the section called “Marketplace 附加元件”。
- 社群附加元件：這些附加元件由 掃描， AWS 但由開放原始碼社群支援。
  - 使用社群附加元件來降低安裝常見開放原始碼軟體的複雜性，例如 Kubernetes Metrics Server。
  - 社群附加元件並非由 建置 AWS。 AWS 只會驗證社群附加元件的版本相容性。
  - 如需詳細資訊，請參閱the section called “社群附加元件”。

下表詳細說明每個附加元件類型的支援範圍：

類別	功能	AWS 附加元件	AWS Marketplace 附加元件	社群附加元件
開發	由 建置 AWS	是	否	是
開發	驗證者 AWS	是	否	是*
開發	經 AWS 合作夥伴驗證	否	是	否
維護	掃描者 AWS	是	是	是
維護	修補者 AWS	是	否	是
維護	由 AWS 合作夥伴修補	否	是	否
發佈	發佈者 AWS	是	否	是

類別	功能	AWS 附加元件	AWS Marketplace 附加元件	社群附加元件
發佈	由 AWS 合作夥伴發佈	否	是	否
支援	的基本安裝支援 AWS	是	是	是
支援	完整 AWS 支援	是	否	否
支援	完整 AWS 合作夥伴支援	否	是	否

*：社群附加元件的驗證僅包含 Kubernetes 版本相容性。例如，如果您在叢集上安裝社群附加元件，會 AWS 檢查它是否與叢集的 Kubernetes 版本相容。

AWS Marketplace 附加元件可以從外部來源下載其他軟體相依性 AWS。這些外部相依性不會由 掃描或驗證 AWS。部署擷取外部相依性的 AWS Marketplace 附加元件時，請考慮您的安全需求。

AWS 附加元件

您可在叢集上建立下列 Amazon EKS 附加元件。您可以使用 `eksctl`、 AWS Management Console 或 AWS CLI 檢視最新的可用附加元件清單。若要查看所有可用的附加元件或安裝附加元件，請參閱 the section called “建立附加元件”。如果附加元件需要 IAM 許可，則您必須擁有叢集的 IAM OpenID Connect (OIDC) 提供者。若要判斷您是否已經擁有一個，或是需要建立一個，請參閱 the section called “IAM OIDC 供應商”。您可以在安裝附加元件之後建立或刪除附加元件。如需詳細資訊，請參閱 the section called “更新 附加元件” 或 the section called “移除附加元件” 。如需使用 Amazon EKS 混合節點執行 EKS 附加元件之特定考量的詳細資訊，請參閱 the section called “設定附加元件”。

您可以使用下列任何 Amazon EKS 附加元件。

描述	進一步了解	相容運算類型
為您的叢集提供原生 VPC 聯網	the section called “Kubernetes 專用 Amazon VPC CNI 外掛程式”	EC2

描述	進一步了解	相容運算類型
彈性、可擴展的 DNS 伺服器，可作為 Kubernetes 叢集 DNS	the section called “CoreDNS”	EC2、Fargate、EKS Auto Mode、EKS 混合節點
維護每個 Amazon EC2 節點的網路規則	the section called “Kube-proxy ”	EC2、EKS 混合節點
為您的叢集提供 Amazon EBS 儲存體	the section called “Amazon EBS CSI 驅動程式”	EC2
為您的叢集提供 Amazon EFS 儲存	the section called “Amazon EFS CSI 驅動程式”	EC2、EKS 自動模式
為您的叢集提供 Amazon FSx for Lustre 儲存	the section called “Amazon FSx CSI 驅動程式”	EC2、EKS 自動模式
為您的叢集提供 Amazon S3 儲存體	the section called “Amazon S3 CSI 驅動程式掛載點”	EC2、EKS 自動模式
偵測其他節點運作狀態問題	the section called “節點監控代理程式”	EC2、EKS 混合節點
在相容的 CSI 驅動程式中啟用快照功能，例如 Amazon EBS CSI 驅動程式	the section called “CSI 快照控制器”	EC2、Fargate、EKS Auto Mode、EKS 混合節點
SageMaker HyperPod 任務控管可最佳化 Amazon EKS 叢集中團隊之間的運算資源分配和用量，解決任務優先順序和資源共用方面的效率低下問題。	the section called “Amazon SageMaker HyperPod 任務控管”	EC2、EKS Auto 模式、
Amazon SageMaker HyperPod 可觀測性 AddOn 為 HyperPod 叢集提供全面的監控和可觀測性功能。	the section called “Amazon SageMaker HyperPod 可觀測性附加元件”	EC2、EKS Auto 模式、

描述	進一步了解	相容運算類型
Kubernetes 代理程式，可收集網路流程資料並將其回報給 Amazon CloudWatch，以便全面監控叢集節點之間的 TCP 連線。	the section called “AWS 網路流量監控代理程式”	EC2、EKS 自動模式
OpenTelemetry 專案的安全、生產就緒、 AWS 支援的分發	the section called “AWS Distro for OpenTelemetry”	EC2、Fargate、EKS Auto Mode、EKS 混合節點
分析和處理基礎資料來源的安全監控服務，包括 AWS CloudTrail 管理事件和 Amazon VPC 流程日誌。Amazon GuardDuty 也會處理功能，例如 Kubernetes 稽核日誌和執行期監控	the section called “Amazon GuardDuty 代理程式”	EC2、EKS 自動模式
提供的監控和可觀測性服務 AWS。此附加元件會安裝 CloudWatch Agent，並為 Amazon EKS 啟用 CloudWatch Application Signals 和 CloudWatch Container Insights 並具有增強的可觀測性	the section called “Amazon CloudWatch 可觀測性代理程式”	EC2、EKS Auto Mode、EKS 混合節點
能夠管理應用程式的登入資料，類似於 EC2 執行個體描述檔提供登入資料給 EC2 執行個體的方式	the section called “EKS Pod 身分識別代理程式”	EC2、EKS 混合節點
讓 cert-manager 從 AWS Private CA 發行 X.509 憑證。需要 cert-manager。	the section called “AWS Kubernetes 專用私有 CA 連接器”	EC2、Fargate、EKS Auto Mode、EKS 混合節點

描述	進一步了解	相容運算類型
產生有關 SR-IOV 網路裝置效能的 Prometheus 指標	the section called “SR-IOV 網路指標匯出工具”	EC2

Kubernetes 專用 Amazon VPC CNI 外掛程式

適用於 Kubernetes Amazon EKS 附加元件的 Amazon VPC CNI 外掛程式是一種 Kubernetes 容器網路介面 (CNI) 外掛程式，可為叢集提供原生 VPC 聯網。此附加元件的自我管理類型或受管類型預設會安裝在每個 Amazon EC2 節點上。如需詳細資訊，請參閱 Kubernetes 容器網路介面 (CNI) 外掛程式。

> **Note**
>
> 您不需要在 Amazon EKS Auto Mode 叢集上安裝此附加元件。如需詳細資訊，請參閱the section called “Amazon EKS Auto 模式的考量事項”。

Amazon EKS 附加元件名稱為 `vpc-cni`。

所需的 IAM 許可

此附加元件會將 IAM 角色用於 Amazon EKS 的服務帳戶功能。如需詳細資訊，請參閱the section called “具有 IRSA 的登入資料”。

如果您的叢集使用 `IPv4` 系列，則需要 AmazonEKS_CNI_Policy 中的許可。如果您的叢集使用 `IPv6` 系列，則必須在 IPv6 模式下建立具有許可的 IAM 政策。您可以建立 IAM 角色、將其中一個政策連接至該角色，並使用下列命令註釋附加元件使用的 Kubernetes 服務帳戶。

將 *`my-cluster`* 取代為您的叢集名稱，並將 *`AmazonEKSVPCCNIRole`* 取代為您的角色名稱。如果您的叢集使用 `IPv6` 系列，則請將 *`AmazonEKS_CNI_Policy`* 取代為您建立的政策名稱。此命令需要您在裝置上安裝 eksctl。如果您需要使用不同的工具來建立角色、將政策連接至角色，並註釋 Kubernetes 服務帳戶，請參閱 the section called “指派 IAM 角色”。

```
eksctl create iamserviceaccount --name aws-node --namespace kube-system --cluster my-
cluster --role-name AmazonEKSVPCCNIRole \
    --role-only --attach-policy-arn arn:aws: iam::aws:policy/AmazonEKS_CNI_Policy --
approve
```

更新資訊

您一次只能更新一個次要版本。例如，如果目前的版本是 1.28.*x*-eksbuild.*y* ，並且您想要更新至 1.30.*x*-eksbuild.*y* ，則您必須將目前的版本更新至 1.29.*x*-eksbuild.*y* ，然後再更新至 1.30.*x*-eksbuild.*y* 。如需更新附加元件的詳細資訊，請參閱 the section called “更新 (EKS 附加元件) ”。

CoreDNS

CoreDNS Amazon EKS 附加元件是彈性、可擴展的 DNS 伺服器，可作為 Kubernetes 叢集 DNS。根據預設，在您建立叢集時，會安裝此附加元件的自我管理類型或受管理類型。當您啟動具有至少一個節點的 Amazon EKS 叢集時，依預設會部署兩個 CoreDNS 映像複本，而不論叢集中部署的節點數量為何。CoreDNS Pod 為叢集中的所有 Pod 提供名稱解析。如果您的叢集包含具有符合 CoreDNS 部署命名空間的 Fargate 設定檔，您可以將 CoreDNS Pod 部署到 Fargate 節點。如需詳細資訊，請參閱the section called “定義設定檔”

> **Note**
>
> 您不需要在 Amazon EKS Auto Mode 叢集上安裝此附加元件。如需詳細資訊，請參閱the section called “Amazon EKS Auto 模式的考量事項”。

Amazon EKS 附加元件名稱為 `coredns`。

所需的 IAM 許可

此附加元件不需要任何許可。

其他資訊

若要進一步了解 CoreDNS，請參閱 Kubernetes 文件中的使用 CoreDNS 進行服務探索和自訂 DNS 服務。

Kube-proxy

Kube-proxy Amazon EKS 附加元件會維護每個 Amazon EC2 節點的網路規則。它可啟用與 Pod 的網路通訊。此附加元件的自我管理類型或受管類型預設會安裝在叢集的每個 Amazon EC2 節點上。

> **Note**
>
> 您不需要在 Amazon EKS Auto Mode 叢集上安裝此附加元件。如需詳細資訊，請參閱the section called “Amazon EKS Auto 模式的考量事項”。

Amazon EKS 附加元件名稱為 `kube-proxy`。

所需的 IAM 許可

此附加元件不需要任何許可。

更新資訊

更新目前版本之前，請考慮下列需求：

- `Kube-proxy` Amazon EKS 叢集上的相容性和扭曲政策與 Kubernetes 相同。

其他資訊

若要進一步了解 `kube-proxy`，請參閱 Kubernetes 文件中的 kube-proxy。

Amazon EBS CSI 驅動程式

Amazon EBS CSI 驅動程式 Amazon EKS 附加元件是 Kubernetes Container Storage Interface (CSI) 外掛程式，可為叢集提供 Amazon EBS 儲存。

> **Note**
>
> 您不需要在 Amazon EKS Auto Mode 叢集上安裝此附加元件。自動模式包含區塊儲存功能。如需詳細資訊，請參閱the section called “部署具狀態工作負載”。

Amazon EKS 附加元件名稱為 `aws-ebs-csi-driver`。

所需的 IAM 許可

此附加元件會將 IAM 角色用於 Amazon EKS 的服務帳戶功能。如需詳細資訊，請參閱the section called “具有 IRSA 的登入資料”。AmazonEBSCSIDriverPolicy AWS 受管政策中的許可是必要的。建立 IAM 角色，並按照下列命令連接受管政策。將 *`my-cluster`* 取代為您的叢集名稱，並將 *`AmazonEKS_EBS_CSI_DriverRole`* 取代為您的角色名稱。此命令需要您在裝置上安裝 eksctl。如果

您需要使用其他工具，或需要使用自訂 KMS 金鑰進行加密，請參閱 the section called “步驟 1：建立 IAM 角色”。

```
eksctl create iamserviceaccount \
    --name ebs-csi-controller-sa \
    --namespace kube-system \
    --cluster my-cluster \
    --role-name AmazonEKS_EBS_CSI_DriverRole \
    --role-only \
    --attach-policy-arn arn:aws: iam::aws:policy/service-role/AmazonEBSCSIDriverPolicy
 \
    --approve
```

其他資訊

若要進一步了解 附加元件，請參閱 the section called “Amazon EBS”。

Amazon EFS CSI 驅動程式

Amazon EFS CSI 驅動程式 Amazon EKS 附加元件是 Kubernetes Container Storage Interface (CSI) 外掛程式，可為叢集提供 Amazon EFS 儲存。

Amazon EKS 附加元件名稱為 `aws-efs-csi-driver`。

所需的 IAM 許可

必要的 IAM 許可 – 此附加元件會將 IAM 角色用於 Amazon EKS 的服務帳戶功能。如需詳細資訊，請參閱the section called “具有 IRSA 的登入資料”。AmazonEFSCSIDriverPolicy AWS 受管政策中的許可是必要的。您可以建立 IAM 角色，並使用下列命令連接受管政策。將 *`my-cluster`* 取代為您的叢集名稱，並將 *`AmazonEKS_EFS_CSI_DriverRole`* 取代為您的角色名稱。這些命令需要您在裝置上安裝 eksctl。如果您需要使用其他工具，請參閱 the section called “步驟 1：建立 IAM 角色”。

```
export cluster_name=my-cluster
export role_name=AmazonEKS_EFS_CSI_DriverRole
eksctl create iamserviceaccount \
    --name efs-csi-controller-sa \
    --namespace kube-system \
    --cluster $cluster_name \
    --role-name $role_name \
    --role-only \
    --attach-policy-arn arn:aws: iam::aws:policy/service-role/AmazonEFSCSIDriverPolicy
 \
```

```
    --approve
TRUST_POLICY=$(aws iam get-role --output json --role-name $role_name --query
 'Role.AssumeRolePolicyDocument' | \
    sed -e 's/efs-csi-controller-sa/efs-csi-*/' -e 's/StringEquals/StringLike/')
aws iam update-assume-role-policy --role-name $role_name --policy-document
 "$TRUST_POLICY"
```

其他資訊

若要進一步了解 附加元件，請參閱 the section called “Amazon EFS”。

Amazon FSx CSI 驅動程式

Amazon FSx CSI 驅動程式 Amazon EKS 附加元件是 Kubernetes Container Storage Interface (CSI) 外掛程式，可為叢集提供 Amazon FSx for Lustre 儲存。

Amazon EKS 附加元件名稱為 `aws-fsx-csi-driver`。

> **Note**
>
> - 叢集中預先存在的 Amazon FSx CSI 驅動程式安裝可能會導致附加元件安裝失敗。當您嘗試在存在非 EKS FSx CSI 驅動程式的情況下安裝 Amazon EKS 附加元件版本時，安裝會因為資源衝突而失敗。在安裝期間使用 `OVERWRITE`旗標來解決此問題：
>
>   ```
>   aws eks create-addon --addon-name aws-fsx-csi-driver --cluster-name my-cluster
>    --resolve-conflicts OVERWRITE
>   ```
>
> - Amazon FSx CSI 驅動程式 EKS 附加元件需要 EKS Pod Identity 代理程式進行身分驗證。如果沒有此元件，附加元件將會失敗並出現錯誤 `Amazon EKS Pod Identity agent is not installed in the cluster`，導致磁碟區無法運作。在部署 FSx CSI 驅動程式附加元件之前或之後安裝 Pod Identity 代理程式。如需詳細資訊，請參閱the section called “設定 代理程式”。

所需的 IAM 許可

此附加元件會將 IAM 角色用於 Amazon EKS 的服務帳戶功能。如需詳細資訊，請參閱the section called “具有 IRSA 的登入資料”。AmazonFSxFullAccess AWS 受管政策中的許可是必要的。建立 IAM 角色，並按照下列命令連接受管政策。將 *`my-cluster`* 取代為您的叢集名稱，並將 *`AmazonEKS_FSx_CSI_DriverRole`* 取代為您的角色名稱。此命令需要您在裝置上安裝 eksctl。

```
eksctl create iamserviceaccount \
    --name fsx-csi-controller-sa \
    --namespace kube-system \
    --cluster my-cluster \
    --role-name AmazonEKS_FSx_CSI_DriverRole \
    --role-only \
    --attach-policy-arn arn:aws: iam::aws:policy/AmazonFSxFullAccess \
    --approve
```

其他資訊

若要進一步了解 附加元件，請參閱 the section called “Amazon FSx for Lustre”。

Amazon S3 CSI 驅動程式掛載點

Amazon S3 CSI 驅動程式 Amazon EKS 附加元件掛載點是 Kubernetes Container Storage Interface (CSI) 外掛程式，可為叢集提供 Amazon S3 儲存。

Amazon EKS 附加元件名稱為 `aws-mountpoint-s3-csi-driver`。

所需的 IAM 許可

此附加元件會將 IAM 角色用於 Amazon EKS 的服務帳戶功能。如需詳細資訊，請參閱the section called “具有 IRSA 的登入資料”。

建立的 IAM 角色將需要提供 S3 存取權的政策。建立政策時，請遵循掛載點 IAM 許可建議。或者，您可以使用 AWS 受管政策連結：iam/home？#/policies/arn:aws: iam：：aws：policy/AmazonS3FullAccess$jsonEditor【AmazonS3FullAccess，type="console"】，但此受管政策授予比掛載點所需的許可更多。

您可以建立 IAM 角色，並使用下列命令將政策連接至該角色。將 *my-cluster* 取代為您的叢集名稱、將 *region-code* 取代為正確的 AWS 區域碼、將 *AmazonEKS_S3_CSI_DriverRole* 取代為您的角色名稱，並將 *AmazonEKS_S3_CSI_DriverRole_ARN* 取代為角色 ARN。這些命令需要您在裝置上安裝 eksctl。如需使用 IAM 主控台或 CLI AWS 的說明，請參閱 the section called “步驟 2：建立 IAM 角色”。

```
CLUSTER_NAME=my-cluster
REGION=region-code
ROLE_NAME=AmazonEKS_S3_CSI_DriverRole
POLICY_ARN=AmazonEKS_S3_CSI_DriverRole_ARN
eksctl create iamserviceaccount \
    --name s3-csi-driver-sa \
```

```
    --namespace kube-system \
    --cluster $CLUSTER_NAME \
    --attach-policy-arn $POLICY_ARN \
    --approve \
    --role-name $ROLE_NAME \
    --region $REGION \
    --role-only
```

其他資訊

若要進一步了解 附加元件，請參閱 the section called “適用於 Amazon S3 的掛載點”。

CSI 快照控制器

容器儲存介面 (CSI) 快照控制器可在相容的 CSI 驅動程式中使用快照功能，例如 Amazon EBS CSI 驅動程式。

Amazon EKS 附加元件名稱為 `snapshot-controller`。

所需的 IAM 許可

此附加元件不需要任何許可。

其他資訊

若要進一步了解 附加元件，請參閱 the section called “CSI 快照控制器”。

Amazon SageMaker HyperPod 任務控管

SageMaker HyperPod 任務控管是一種強大的管理系統，旨在簡化資源配置，並確保 Amazon EKS 叢集跨團隊和專案有效利用運算資源。這可讓管理員能夠設定：

- 各種任務的優先順序層級
- 每個團隊的運算配置
- 每個團隊如何借出閒置運算
- 如果團隊先佔自己的任務

HyperPod 任務控管也提供 Amazon EKS 叢集可觀測性，提供叢集容量的即時可見性。這包括運算可用性和用量、團隊配置和使用率，以及任務執行和等待時間資訊，讓您為明智的決策和主動資源管理做好準備。

Amazon EKS 附加元件名稱為 `amazon-sagemaker-hyperpod-taskgovernance`。

所需的 IAM 許可

此附加元件不需要任何許可。

其他資訊

若要進一步了解附加元件，請參閱 SageMaker HyperPod 任務控管

Amazon SageMaker HyperPod 可觀測性附加元件

Amazon SageMaker HyperPod 可觀測性附加元件可為 HyperPod 叢集提供全面的監控和可觀測性功能。此附加元件會自動部署和管理基本監控元件，包括節點匯出器、DCGM 匯出器、kube-state-metrics和 EFA 匯出器。它會收集指標並將其轉送至客戶指定的 Amazon Managed Prometheus (AMP) 執行個體，並公開自訂指標的 OTLP 端點，以及客戶訓練任務的事件擷取。

附加元件透過從包括 HyperPod 任務控管附加元件、HyperPod Training Operator、Kubeflow 和 KEDA 等各種元件抓取指標，與更廣泛的 HyperPod 生態系統整合。所有收集的指標都集中在 Amazon Managed Prometheus 中，讓客戶能夠透過 Amazon Managed Grafana 儀表板實現統一的可觀測性檢視。這可提供整個 HyperPod 環境中叢集運作狀態、資源使用率和訓練任務效能的end-to-end可見性。

Amazon EKS 附加元件名稱為 `amazon-sagemaker-hyperpod-observability`。

所需的 IAM 許可

此附加元件會將 IAM 角色用於 Amazon EKS 的服務帳戶功能。如需詳細資訊，請參閱the section called “具有 IRSA 的登入資料”。需要下列 受管政策：

- `AmazonPrometheusRemoteWriteAccess` - 用於將指標從叢集遠端寫入 AMP
- `CloudWatchAgentServerPolicy` - 用於將日誌從叢集遠端寫入 CloudWatch

其他資訊

若要進一步了解附加元件及其功能，請參閱 SageMaker HyperPod 可觀測性。

AWS 網路流量監控代理程式

Amazon CloudWatch Network Flow Monitor Agent 是一種 Kubernetes 應用程式，可從叢集中的所有節點收集 TCP 連線統計資料，並將網路流量報告發佈至 Amazon CloudWatch Network Flow Monitor Ingestion APIs。

Amazon EKS 附加元件名稱為 `aws-network-flow-monitoring-agent`。

所需的 IAM 許可

此附加元件需要 IAM 許可。

您需要將 `CloudWatchNetworkFlowMonitorAgentPublishPolicy`受管政策連接至 附加元件。

如需所需 IAM 設定的詳細資訊，請參閱 Amazon CloudWatch Network Flow Monitor Agent GitHub 儲存庫上的 IAM 政策。

如需 受管政策的詳細資訊，請參閱《Amazon CloudWatch CloudWatch 使用者指南》中的 CloudWatchNetworkFlowMonitorAgentPublishPolicy。

其他資訊

若要進一步了解 附加元件，請參閱 Amazon CloudWatch Network Flow Monitor Agent GitHub 儲存庫。

節點監控代理程式

節點監控代理程式 Amazon EKS 附加元件可以偵測其他節點運作狀態問題。選用的節點自動修復功能也可以利用這些額外的運作狀態訊號，視需要自動取代節點。

> **Note**
>
> 您不需要在 Amazon EKS Auto Mode 叢集上安裝此附加元件。如需詳細資訊，請參閱the section called “Amazon EKS Auto 模式的考量事項”。

Amazon EKS 附加元件名稱為 `eks-node-monitoring-agent`。

所需的 IAM 許可

此附加元件不需要額外的許可。

其他資訊

如需詳細資訊，請參閱the section called “節點運作狀態”。

AWS Distro for OpenTelemetry

AWS Distro for OpenTelemetry Amazon EKS 附加元件是 OpenTelemetry 專案的安全、生產就緒、AWS 支援的分佈。如需詳細資訊，請參閱 GitHub 上的 AWS Distro for OpenTelemetry。

Amazon EKS 附加元件名稱為 `adot`。

所需的 IAM 許可

如果您使用的是其中一個可以透過進階組態選擇加入的預先設定自訂資源，則此附加元件只需要 IAM 許可。

其他資訊

如需詳細資訊，請參閱 AWS Distro for OpenTelemetry 文件中的使用 EKS 附加元件的 Distro for OpenTelemetry 入門。 AWS OpenTelemetry

ADOT 需要`cert-manager`將附加元件部署在叢集上做為先決條件，否則如果直接使用 https://registry.terraform.io/modules/terraform-aws-modules/eks/aws/latest `cluster_addons` 屬性部署，此附加元件將無法運作。如需更多需求，請參閱 AWS Distro for OpenTelemetry 文件中的使用 EKS 附加元件的 Distro for OpenTelemetry 入門需求。 AWS OpenTelemetry

Amazon GuardDuty 代理程式

Amazon GuardDuty 代理程式 Amazon EKS 附加元件會從 EKS 叢集節點收集執行期事件 （檔案存取、程序執行、網路連線），以供 GuardDuty 執行期監控分析。GuardDuty 本身 （而非代理程式） 是安全性監控服務，可分析和處理基礎資料來源，包括 AWS CloudTrail 管理事件和 Amazon VPC 流程日誌，以及 Kubernetes 稽核日誌和執行期監控等功能。

Amazon EKS 附加元件名稱為 `aws-guardduty-agent`。

所需的 IAM 許可

此附加元件不需要任何許可。

其他資訊

如需詳細資訊，請參閱 Amazon GuardDuty 中 Amazon EKS 叢集的執行期監控。

- 若要偵測 Amazon EKS 叢集中潛在的安全威脅，請啟用 Amazon GuardDuty 執行時間監控，並將 GuardDuty 安全代理程式部署到您的 Amazon EKS 叢集。

Amazon CloudWatch 可觀測性代理程式

Amazon CloudWatch 可觀測性代理程式 Amazon EKS 附加元件提供的監控和可觀測性服務 AWS。此附加元件會安裝 CloudWatch 代理程式，並搭配 Amazon EKS 的增強可觀測性，啟用 CloudWatch Application Signals 和 CloudWatch Container Insights。如需詳細資訊，請參閱 Amazon CloudWatch Agent。

Amazon EKS 附加元件名稱為 `amazon-cloudwatch-observability`。

所需的 IAM 許可

此附加元件會將 IAM 角色用於 Amazon EKS 的服務帳戶功能。如需詳細資訊，請參閱the section called “具有 IRSA 的登入資料”。需要 link：iam/home#/policies/arn:aws: iam： ：aws：policy/AWSXrayWriteOnlyAccess【AWSXrayWriteOnlyAccess，type="console"】 和 link：iam/home#/policies/arn:aws: iam：aws：policy/CloudWatchAgentServerPolicy【CloudWatchAgentServerPolicy，type="console"】 AWS 受管政策中的許可。您可以建立 IAM 角色、將受管政策連接至該角色，並使用下列命令註釋附加元件使用的 Kubernetes 服務帳戶。將 *`my-cluster`* 取代為您的叢集名稱，並將 *`AmazonEKS_Observability_role`* 取代為您的角色名稱。此命令需要您在裝置上安裝 eksctl。如果您需要使用不同的工具來建立角色、將政策連接至角色，並註釋 Kubernetes 服務帳戶，請參閱 the section called “指派 IAM 角色”。

```
eksctl create iamserviceaccount \
    --name cloudwatch-agent \
    --namespace amazon-cloudwatch \
    --cluster my-cluster \
    --role-name AmazonEKS_Observability_Role \
    --role-only \
    --attach-policy-arn arn:aws: iam::aws:policy/AWSXrayWriteOnlyAccess \
    --attach-policy-arn arn:aws: iam::aws:policy/CloudWatchAgentServerPolicy \
    --approve
```

其他資訊

如需詳細資訊，請參閱安裝 CloudWatch 代理程式。

AWS Kubernetes 專用私有 CA 連接器

AWS Private CA Connector for Kubernetes 是 cert-manager 的附加元件，可讓使用者從 AWS Private Certificate Authority (AWS Private CA) 取得憑證。

- Amazon EKS 附加元件名稱為 `aws-privateca-connector-for-kubernetes`。
- 附加元件命名空間為 `aws-privateca-issuer`。

此附加元件需要 `cert-manager`。`cert-manager`Amazon EKS 提供做為社群附加元件。如需此附加元件的詳細資訊，請參閱 the section called “Cert Manager”。如需安裝附加元件的詳細資訊，請參閱 the section called “建立附加元件”。

所需的 IAM 許可

此附加元件需要 IAM 許可。

使用 EKS Pod 身分將 `AWSPrivateCAConnectorForKubernetesPolicy` IAM 政策連接至 Kubernetes `aws-privateca-issuer` 服務帳戶。如需詳細資訊，請參閱the section called “使用 Pod 身分”。

如需必要許可的相關資訊，請參閱《 AWS 受管政策參考》中的 AWSPrivateCAConnectorForKubernetesPolicy。

其他資訊

如需詳細資訊，請參閱適用於 AWS Kubernetes GitHub 儲存庫的私有 CA 發行者。

如需設定附加元件的詳細資訊，請參閱 `aws-privateca-issuer` GitHub 儲存庫中的 values.yaml。確認 values.yaml 的版本與叢集上安裝的附加元件版本相符。

此附加元件可容忍 EKS Auto Mode `CriticalAddonsOnly` 的 `system` NodePool 所使用的污點。如需詳細資訊，請參閱the section called “執行關鍵附加元件”。

EKS Pod 身分識別代理程式

Amazon EKS Pod Identity Agent Amazon EKS 附加元件可讓您管理應用程式的登入資料，類似於 EC2 執行個體描述檔提供登入資料給 EC2 執行個體的方式。

> **Note**
>
> 您不需要在 Amazon EKS Auto Mode 叢集上安裝此附加元件。Amazon EKS Auto Mode 與 EKS Pod Identity 整合。如需詳細資訊，請參閱the section called “Amazon EKS Auto 模式的考量事項”。

Amazon EKS 附加元件名稱為 `eks-pod-identity-agent`。

所需的 IAM 許可

Pod Identity Agent 附加元件本身不需要 IAM 角色。它使用來自 Amazon EKS 節點 IAM 角色的許可來運作，但不需要用於附加元件的專用 IAM 角色。

更新資訊

您一次只能更新一個次要版本。例如，如果目前的版本是 `1.28.x-eksbuild.y`，並且您想要更新至 `1.30.x-eksbuild.y`，則您必須將目前的版本更新至 `1.29.x-eksbuild.y`，然後再更新至 `1.30.x-eksbuild.y`。如需更新附加元件的詳細資訊，請參閱 the section called “更新 附加元件”。

SR-IOV 網路指標匯出工具

SR-IOV 網路指標匯出工具 Amazon EKS 附加元件會以 Prometheus 格式收集和公開有關 SR-IOV 網路裝置的指標。它可在 EKS 裸機節點上監控 SR-IOV 網路效能。匯出工具會在具有支援 SR-IOV-capable介面的節點上執行為 DaemonSet，並匯出 Prometheus 可抓取的指標。

> **Note**
>
> 此附加元件需要具有SR-IOV-capable 網路介面的節點。

屬性	Value
附加元件名稱	`sriov-network-metrics-exporter`
命名空間	`monitoring`
文件	SR-IOV 網路指標匯出工具 GitHub 儲存庫
服務帳戶名稱	無
受管 IAM 政策	無
自訂 IAM 許可	無

社群附加元件

您可以使用 AWS APIs社群附加元件，例如 Kubernetes Metrics Server。您可以選擇將社群附加元件安裝為 Amazon EKS 附加元件，以減少在多個叢集上維護軟體的複雜性。

例如，您可以使用 AWS API、CLI 或 管理主控台來安裝社群附加元件。您可以在叢集建立期間安裝社群附加元件。

您可以像現有的 Amazon EKS 附加元件一樣管理社群附加元件。社群附加元件與現有的附加元件不同，因為它們具有唯一的支援範圍。

> **Note**
>
> 使用社群附加元件由您自行決定。作為您與 之間共同責任模型的一部分 AWS，您應該了解您在叢集中為這些第三方外掛程式安裝什麼。您也必須負責滿足叢集安全需求的社群附加元件。如需詳細資訊，請參閱the section called “支援部署到 EKS 的軟體”。

社群附加元件並非由 建置 AWS。 AWS 只會驗證社群附加元件的版本相容性。例如，如果您在叢集上安裝社群附加元件， 會 AWS 檢查它是否與叢集的 Kubernetes 版本相容。

重要的是， AWS 不提供社群附加元件的完整支援。 僅 AWS 支援使用 AWS APIs 完成的生命週期操作，例如安裝附加元件或刪除附加元件。

如果您需要社群附加元件的支援，請利用現有的專案資源。例如，您可以在專案的儲存庫上建立 GitHub 問題。

判斷附加元件類型

您可以使用 AWS CLI 來判斷 Amazon EKS 附加元件的類型。

使用下列 CLI 命令來擷取附加元件的相關資訊。您可以使用任何附加元件的名稱`metrics-server`來取代 。

```
aws eks describe-addon-versions --addon-name metrics-server
```

檢閱 `owner` 欄位的 CLI 輸出。

```
{
    "addons": [
        {
            "addonName": "metrics-server",
            "type": "observability",
            "owner": "community",
            "addonVersions": [
```

如果 的值`owner`是 `community`，則附加元件是社群附加元件。 AWS 僅支援安裝、更新和移除附加元件。如果您對附加元件本身的功能和操作有任何疑問，請使用社群資源，例如 GitHub 問題。

安裝或更新社群附加元件

您安裝或更新社群附加元件的方式與其他 Amazon EKS 附加元件相同。

- the section called “建立附加元件”
- the section called “更新 附加元件”
- the section called “移除附加元件”

可用的社群附加元件

下列社群附加元件可從 Amazon EKS 取得。

- the section called “Kubernetes 指標伺服器”
- the section called “kube-state-metrics”
- the section called “Prometheus Node Exporter”
- the section called “Cert Manager”
- the section called “外部 DNS”
- the section called “Fluent Bit”

Kubernetes 指標伺服器

Kubernetes Metrics Server 是 Kubernetes 內建自動擴展管道的可擴展且有效率的容器資源指標來源。它從 Kubelets 收集資源指標，並透過指標 API 在 Kubernetes apiserver 中公開，以供 Horizontal Pod Autoscaler 和 Vertical Pod Autoscaler 使用。

屬性	Value
附加元件名稱	`metrics-server`
命名空間	`kube-system`
文件	GitHub 讀我檔案
服務帳戶名稱	無
受管 IAM 政策	無

屬性	Value
自訂 IAM 許可	無

kube-state-metrics

用於產生和公開叢集層級指標的附加元件代理程式。

Kubernetes API 中的 Kubernetes 物件狀態可以公開為指標。稱為 kube-state-metrics 的附加元件代理程式可以連線至 Kubernetes API 伺服器，並使用從叢集中個別物件狀態產生的指標公開 HTTP 端點。它會公開物件狀態的各種資訊，例如標籤和註釋、啟動和終止時間、狀態或物件目前所在的階段。

屬性	Value
附加元件名稱	`kube-state-metrics`
命名空間	`kube-state-metrics`
文件	Kubernetes 文件中 Kubernetes 物件狀態的指標
服務帳戶名稱	無
受管 IAM 政策	無
自訂 IAM 許可	無

Prometheus Node Exporter

適用於 *NIX 核心公開之硬體和作業系統指標的 Prometheus 匯出工具，使用可插入指標收集器以 Go 撰寫。Prometheus Node Exporter 會公開各種硬體和核心相關指標。

屬性	Value
附加元件名稱	`prometheus-node-exporter`
命名空間	`prometheus-node-exporter`
文件	使用 Prometheus 文件中的 Node Exporter 監控 Linux 主機指標

屬性	Value
服務帳戶名稱	無
受管 IAM 政策	無
自訂 IAM 許可	無

Cert Manager

Cert Manager 可用來管理憑證的建立和續約。

屬性	Value
附加元件名稱	`cert-manager`
命名空間	`cert-manager`
文件	Cert Manager 文件
服務帳戶名稱	無
受管 IAM 政策	無
自訂 IAM 許可	無

外部 DNS

外部 DNS EKS 附加元件可用來透過 Kubernetes 資源管理 Route53 DNS 記錄。

在您要管理的託管區域`route53:ListResourceRecordSets`上，外部 DNS 許可可以減少為`route53:ChangeResourceRecordSets`、`route53:ListHostedZones` 和 。

屬性	Value
附加元件名稱	`external-dns`
命名空間	`external-dns`

屬性	Value
文件	GitHub 讀我檔案
服務帳戶名稱	external-dns
受管 IAM 政策	arn:aws: iam::aws:policy/AmazonRoute53FullAccess
自訂 IAM 許可	無

Fluent Bit

Fluent Bit 是輕量且高效能的日誌處理器和轉寄站。它可讓您從不同來源收集資料/日誌，統一它們，並將其傳送至多個目的地，包括 Amazon CloudWatch Logs、Amazon S3 和 Amazon Kinesis Data Firehose。Fluent Bit 的設計考量了效能和資源效率，因此非常適合 Kubernetes 環境。

此附加元件不需要預設組態中的 IAM 許可。不過，如果您設定 AWS 輸出位置，您可能需要授予此附加元件 IAM 許可。如需詳細資訊，請參閱the section called “使用 Pod 身分”。

屬性	Value
附加元件名稱	fluent-bit
命名空間	fluent-bit
文件	Fluent 位元文件
服務帳戶名稱	fluent-bit
受管 IAM 政策	無
自訂 IAM 許可	無

檢視屬性

您可以下載社群附加元件的開放原始碼屬性和授權資訊。

1. 決定您要下載屬性的附加元件名稱和版本。

2. 使用名稱和版本更新下列命令：

```
curl -O https://amazon-eks-docs.s3.amazonaws.com/attributions/<add-on-name>/<add-on-version>/attributions.zip
```

例如：

```
curl -O https://amazon-eks-docs.s3.amazonaws.com/attributions/kube-state-metrics/v2.14.0-eksbuild.1/attributions.zip
```

3. 使用 命令下載 檔案。

使用此 zip 檔案來檢視授權屬性的相關資訊。

AWS Marketplace 附加元件

除了先前的 Amazon EKS 附加元件清單之外，您還可以新增來自獨立軟體廠商的各種操作軟體 Amazon EKS 附加元件。選擇附加元件以進一步了解它及其安裝需求。

Accuknox

附加元件名稱為 `accuknox_kubearmor`，命名空間為 `kubearmor`。Accuknox 發佈附加元件。

如需附加元件的資訊，請參閱 KubeArmor 文件中的 KubeArmor 入門。 KubeArmor

服務帳戶名稱

服務帳戶不會與此附加元件搭配使用。

AWS 受管 IAM 政策

受管政策不會與此附加元件搭配使用。

自訂 IAM 許可

自訂許可不會與此附加元件搭配使用。

Akuity

附加元件名稱為 `akuity_agent`，命名空間為 `akuity`。Akuity 會發佈附加元件。

如需如何附加元件的資訊，請參閱 Akuity 平台文件中使用 Akuity EKS 附加元件在 Amazon EKS 上安裝 Akuity 代理程式。

服務帳戶名稱

服務帳戶不會與此附加元件搭配使用。

AWS 受管 IAM 政策

受管政策不會與此附加元件搭配使用。

自訂 IAM 許可

自訂許可不會與此附加元件搭配使用。

卡利普里亞

附加元件名稱為 `calyptia_fluent-bit`，命名空間為 `calytia-fluentbit`。Calyptia 會發佈附加元件。

如需附加元件的詳細資訊，請參閱 Calyptia 文件網站上的 Calyptia Core Agent 入門。

服務帳戶名稱

服務帳戶名稱為 `clyptia-fluentbit`。

AWS 受管 IAM 政策

此附加元件使用 `AWSMarketplaceMeteringRegisterUsage`受管政策。如需詳細資訊，請參閱《 AWS 受管政策參考指南》中的 AWSMarketplaceMeteringRegisterUsage。

建立必要 IAM 角色的命令

下列命令需要您擁有叢集的 IAM OpenID Connect (OIDC) 提供者。若要判斷您是否已經擁有一個，或是需要建立一個，請參閱 the section called “IAM OIDC 供應商”。將 *`my-cluster`* 取代為您的叢集名稱，並將 *`my-calyptia-role`* 取代為您的角色名稱。此命令需要您在裝置上安裝 eksctl。如果您需要使用不同的工具來建立角色並註釋 Kubernetes 服務帳戶，請參閱 the section called “指派 IAM 角色”。

```
eksctl create iamserviceaccount --name service-account-name  --namespace calyptia-
fluentbit --cluster my-cluster --role-name my-calyptia-role \
    --role-only --attach-policy-arn arn:aws: iam::aws:policy/
AWSMarketplaceMeteringRegisterUsage --approve
```

Cisco 可觀測性收集器

附加元件名稱為 `cisco_cisco-cloud-observability-collectors`，命名空間為 `appdynamics`。Cisco 會懲罰附加元件。

如需附加元件的資訊，請參閱 Cisco AppDynamics 文件中的使用 Cisco Cloud Observability AWS Marketplace 附加元件。 AppDynamics

服務帳戶名稱

服務帳戶不會與此附加元件搭配使用。

AWS 受管 IAM 政策

受管政策不會與此附加元件搭配使用。

自訂 IAM 許可

自訂許可不會與此附加元件搭配使用。

Cisco 可觀測性運算子

附加元件名稱為 `cisco_cisco-cloud-observability-operators`，命名空間為 `appdynamics`。Cisco 發佈附加元件。

如需附加元件的相關資訊，請參閱 Cisco AppDynamics 文件中的使用 Cisco Cloud Observability AWS Marketplace 附加元件。 AppDynamics

服務帳戶名稱

服務帳戶不會與此附加元件搭配使用。

AWS 受管 IAM 政策

受管政策不會與此附加元件搭配使用。

自訂 IAM 許可

自訂許可不會與此附加元件搭配使用。

CLOUDSOFT

附加元件名稱為 `cloudsoft_cloudsoft-amp`，命名空間為 `cloudsoft-amp`。CLOUDSOFT 會發佈附加元件。

如需附加元件的相關資訊，請參閱 CLOUDSOFT 文件中的 Amazon EKS ADDON。

服務帳戶名稱

服務帳戶不會與此附加元件搭配使用。

AWS 受管 IAM 政策

受管政策不會與此附加元件搭配使用。

自訂 IAM 許可

自訂許可不會與此附加元件搭配使用。

Cribl

附加元件名稱為 `cribl_cribledge`，命名空間為 `cribledge`。Cribl 發佈附加元件。

如需附加元件的資訊，請參閱 Cribl 文件中的安裝適用於 Edge 的 Cribl Amazon EKS 附加元件

服務帳戶名稱

服務帳戶不會與此附加元件搭配使用。

AWS 受管 IAM 政策

受管政策不會與此附加元件搭配使用。

自訂 IAM 許可

自訂許可不會與此附加元件搭配使用。

Dynatrace

附加元件名稱為 `dynatrace_dynatrace-operator`，命名空間為 `dynatrace`。Dynatrace 會發佈附加元件。

如需有關附加元件的資訊，請參閱 dynatrace 文件中的 Kubernetes 監控。

服務帳戶名稱

服務帳戶不會與此附加元件搭配使用。

AWS 受管 IAM 政策

受管政策不會與此附加元件搭配使用。

自訂 IAM 許可

自訂許可不會與此附加元件搭配使用。

大樹

附加元件名稱為 `datree_engine-pro`，命名空間為 `datree`。Datree 發佈附加元件。

如需附加元件的詳細資訊，請參閱 Datree 文件中的 Amazon EKS-intergration。

服務帳戶名稱

服務帳戶名稱為 datree-webhook-server-awsmp。

AWS 受管 IAM 政策

受管政策是 AWSLicenseManagerConsumptionPolicy。如需詳細資訊，請參閱《 AWS 受管政策參考指南》中的 AWSLicenseManagerConsumptionPolicy。

建立必要 IAM 角色的命令

下列命令需要您擁有叢集的 IAM OpenID Connect (OIDC) 提供者。若要判斷您是否已經擁有一個，或是需要建立一個，請參閱 the section called “IAM OIDC 供應商”。將 *my-cluster* 取代為您的叢集名稱，並將 *my-datree-role* 取代為您的角色名稱。此命令需要您在裝置上安裝 eksctl。如果您需要使用不同的工具來建立角色並註釋 Kubernetes 服務帳戶，請參閱 the section called “指派 IAM 角色”。

```
eksctl create iamserviceaccount --name datree-webhook-server-awsmp --namespace datree
 --cluster my-cluster --role-name my-datree-role \
    --role-only --attach-policy-arn arn:aws: iam::aws:policy/service-role/
AWSLicenseManagerConsumptionPolicy --approve
```

自訂許可

自訂許可不會與此附加元件搭配使用。

Datadog

附加元件名稱為 `datadog_operator`，命名空間為 `datadog-agent`。Datadog 會發佈附加元件。

如需附加元件的相關資訊，請參閱 Datadog 文件中的使用 Datadog Operator Add-on 在 Amazon EKS 上安裝 Datadog Agent。

服務帳戶名稱

服務帳戶不會與此附加元件搭配使用。

AWS 受管 IAM 政策

受管政策不會與此附加元件搭配使用。

自訂 IAM 許可

自訂許可不會與此附加元件搭配使用。

Groundcover

附加元件名稱為 `groundcover_agent`，命名空間為 `groundcover`。 Groundcover 會發佈附加元件。

如需附加元件的資訊，請參閱 Groundcover 文件中的安裝 Groundcover Amazon EKS 附加元件。

服務帳戶名稱

服務帳戶不會與此附加元件搭配使用。

AWS 受管 IAM 政策

受管政策不會與此附加元件搭配使用。

自訂 IAM 許可

自訂許可不會與此附加元件搭配使用。

Grafana 實驗室

附加元件名稱為 `grafana-labs_kubernetes-monitoring`，命名空間為 `monitoring`。Grafana 實驗室發佈 附加元件。

如需附加元件的資訊，請參閱 Grafana Labs 文件中的使用 Amazon EKS 將 Kubernetes 監控設定為附加元件。

服務帳戶名稱

服務帳戶不會與此附加元件搭配使用。

AWS 受管 IAM 政策

受管政策不會與此附加元件搭配使用。

自訂 IAM 許可

自訂許可不會與此附加元件搭配使用。

Guance

- 發佈者 – GUANCE
- 名稱 – `guance_datakit`
- 命名空間：`datakit`
- 服務帳戶名稱 – 服務帳戶不會與此附加元件搭配使用。
- AWS 受管 IAM 政策 – 受管政策不會與此附加元件搭配使用。
- 自訂 IAM 許可 – 自訂許可不會與此附加元件搭配使用。
- 設定和使用說明 – 請參閱 Guance 文件中的使用 Amazon EKS 附加元件。

HA 代理

名稱為 `haproxy-technologies_kubernetes-ingress-ee`，命名空間為 `haproxy-controller`。HA Proxy 會發佈附加元件。

如需附加元件的詳細資訊，請參閱 Datree 文件中的 Amazon EKS-intergration。

服務帳戶名稱

服務帳戶名稱為 `customer defined`。

AWS 受管 IAM 政策

受管政策是 AWSLicenseManagerConsumptionPolicy。如需詳細資訊，請參閱《 AWS 受管政策參考指南》中的 AWSLicenseManagerConsumptionPolicy。

建立必要 IAM 角色的命令

下列命令需要您擁有叢集的 IAM OpenID Connect (OIDC) 提供者。若要判斷您是否已經擁有一個，或是需要建立一個，請參閱 the section called “IAM OIDC 供應商”。將 *`my-cluster`* 取代為您的叢集名

稱，並將 *my-haproxy-role* 取代為您的角色名稱。此命令需要您在裝置上安裝 eksctl。如果您需要使用不同的工具來建立角色並註釋 Kubernetes 服務帳戶，請參閱 the section called “指派 IAM 角色”。

```
eksctl create iamserviceaccount --name service-account-name  --namespace haproxy-
controller --cluster my-cluster --role-name my-haproxy-role \
    --role-only --attach-policy-arn arn:aws: iam::aws:policy/service-role/
AWSLicenseManagerConsumptionPolicy --approve
```

自訂許可

自訂許可不會與此附加元件搭配使用。

Kpow

附加元件名稱為 `factorhouse_kpow`，命名空間為 `factorhouse`。Factorhouse 會發佈附加元件。

如需附加元件的詳細資訊，請參閱 Kpow 文件中的 AWS Marketplace LM。

服務帳戶名稱

服務帳戶名稱為 kpow。

AWS 受管 IAM 政策

受管政策是 AWSLicenseManagerConsumptionPolicy。如需詳細資訊，請參閱《 AWS 受管政策參考指南》中的 AWSLicenseManagerConsumptionPolicy。

建立必要 IAM 角色的命令

下列命令需要您擁有叢集的 IAM OpenID Connect (OIDC) 提供者。若要判斷您是否已經擁有一個，或是需要建立一個，請參閱 the section called “IAM OIDC 供應商”。將 *my-cluster* 取代為您的叢集名稱，並將 *my-role-name* 取代為您的角色名稱。此命令需要您在裝置上安裝 eksctl。如果您需要使用不同的工具來建立角色並註釋 Kubernetes 服務帳戶，請參閱 the section called “指派 IAM 角色”。

```
eksctl create iamserviceaccount --name kpow --namespace factorhouse --cluster my-
cluster --role-name my-kpow-role \
    --role-only --attach-policy-arn arn:aws: iam::aws:policy/service-role/
AWSLicenseManagerConsumptionPolicy --approve
```

自訂許可

自訂許可不會與此附加元件搭配使用。

Kubecost

附加元件名稱為 `kubecost_kubecost`，命名空間為 `kubecost`。Kubecost 會發佈附加元件。

如需附加元件的詳細資訊，請參閱 Kubecost 文件中的AWS 雲端帳單整合。

您必須在叢集上安裝具有 Amazon EBS 的 Store Kubernetes 磁碟區。否則，您會收到錯誤。

服務帳戶名稱

服務帳戶不會與此附加元件搭配使用。

AWS 受管 IAM 政策

受管政策不會與此附加元件搭配使用。

自訂 IAM 許可

自訂許可不會與此附加元件搭配使用。

Kasten

附加元件名稱為 `kasten_k10`，命名空間為 `kasten-io`。Kasten by Veeam 會發佈附加元件。

如需有關附加元件的資訊，請參閱 Kasten 文件中的AWS 使用 Amazon EKS 附加元件在 上安裝 K10。

您必須在叢集上安裝預設 的 Amazon EBS CSI 驅動程式`StorageClass`。

服務帳戶名稱

服務帳戶名稱為 `k10-k10`。

AWS 受管 IAM 政策

受管政策是 AWSLicenseManagerConsumptionPolicy。如需詳細資訊，請參閱《 AWS 受管政策參考指南》中的 AWSLicenseManagerConsumptionPolicy。

建立必要 IAM 角色的命令

下列命令需要您擁有叢集的 IAM OpenID Connect (OIDC) 提供者。若要判斷您是否已經擁有一個，或是需要建立一個，請參閱 the section called “IAM OIDC 供應商”。將 *`my-cluster`* 取代為您的叢集名

稱，並將 *my-kasten-role* 取代為您的角色名稱。此命令需要您在裝置上安裝 eksctl。如果您需要使用不同的工具來建立角色並註釋 Kubernetes 服務帳戶，請參閱 the section called “指派 IAM 角色”。

```
eksctl create iamserviceaccount --name k10-k10 --namespace kasten-io --cluster my-
cluster --role-name my-kasten-role \
    --role-only --attach-policy-arn arn:aws: iam::aws:policy/service-role/
AWSLicenseManagerConsumptionPolicy --approve
```

自訂許可

自訂許可不會與此附加元件搭配使用。

香港

附加元件名稱為 kong_konnect-ri，命名空間為 kong。Kong 發佈 附加元件。

如需附加元件的資訊，請參閱 Kong 文件中的安裝 Kong Gateway EKS 附加元件。

您必須在叢集上安裝具有 Amazon EBS 的 Store Kubernetes 磁碟區。否則，您會收到錯誤。

服務帳戶名稱

服務帳戶不會與此附加元件搭配使用。

AWS 受管 IAM 政策

受管政策不會與此附加元件搭配使用。

自訂 IAM 許可

自訂許可不會與此附加元件搭配使用。

LeakSignal

附加元件名稱為 leaksignal_leakagent，命名空間為 leakagent。LeakSignal 會發佈附加元件。

如需有關附加元件的資訊，請參閱 LeakSignal 文件中的 https://www.leaksignal.com/docs/LeakAgent/Deployment/ AWS%20EKS%20Addon/【Install the LeakAgent add-on】

您必須在叢集上安裝具有 Amazon EBS 的 Store Kubernetes 磁碟區。否則，您會收到錯誤。

服務帳戶名稱

服務帳戶不會與此附加元件搭配使用。

AWS 受管 IAM 政策

受管政策不會與此附加元件搭配使用。

自訂 IAM 許可

自訂許可不會與此附加元件搭配使用。

NetApp

附加元件名稱為 `netapp_trident-operator`，命名空間為 `trident`。NetApp 會發佈附加元件。

如需附加元件的資訊，請參閱 NetApp 文件中的設定 Trident EKS 附加元件。

服務帳戶名稱

服務帳戶不會與此附加元件搭配使用。

AWS 受管 IAM 政策

受管政策不會與此附加元件搭配使用。

自訂 IAM 許可

自訂許可不會與此附加元件搭配使用。

New Relic

附加元件名稱為 `new-relic_kubernetes-operator`，命名空間為 `newrelic`。New Relic 會發佈附加元件。

如需有關附加元件的資訊，請參閱 New Relic 文件中的安裝適用於 EKS 的新複本附加元件。

服務帳戶名稱

服務帳戶不會與此附加元件搭配使用。

AWS 受管 IAM 政策

受管政策不會與此附加元件搭配使用。

自訂 IAM 許可

自訂許可不會與此附加元件搭配使用。

Rafay

附加元件名稱為 `rafay-systems_rafay-operator`，命名空間為 `rafay-system`。Rafay 發佈附加元件。

如需附加元件的詳細資訊，請參閱 Rafay 文件中的安裝 Rafay Amazon EKS 附加元件。

服務帳戶名稱

服務帳戶不會與此附加元件搭配使用。

AWS 受管 IAM 政策

受管政策不會與此附加元件搭配使用。

自訂 IAM 許可

自訂許可不會與此附加元件搭配使用。

Rad 安全性

- 發佈者 – RAD SECURITY
- 名稱 – `rad-security_rad-security`
- 命名空間：`ksoc`
- 服務帳戶名稱 – 服務帳戶不會與此附加元件搭配使用。
- AWS 受管 IAM 政策 – 受管政策不會與此附加元件搭配使用。
- 自訂 IAM 許可 – 自訂許可不會與此附加元件搭配使用。
- 設定和使用說明 – 請參閱 Rad Security 文件中的透過 AWS Marketplace 安裝 Rad。

SolarWinds

- 發佈者 – SOLARWINDS
- 名稱 – `solarwinds_swo-k8s-collector-addon`
- 命名空間：`solarwinds`

- 服務帳戶名稱 – 服務帳戶不會與此附加元件搭配使用。
- AWS 受管 IAM 政策 – 受管政策不會與此附加元件搭配使用。
- 自訂 IAM 許可 – 自訂許可不會與此附加元件搭配使用。
- 設定和使用說明 – 請參閱 SolarWinds 文件中的監控 Amazon EKS 叢集。

Solo

附加元件名稱為 `solo-io_istio-distro`，命名空間為 `istio-system`。Solo 會發佈附加元件。

如需附加元件的詳細資訊，請參閱 https：//Solo.io 文件中的安裝 Istio。

服務帳戶名稱

服務帳戶不會與此附加元件搭配使用。

AWS 受管 IAM 政策

受管政策不會與此附加元件搭配使用。

自訂 IAM 許可

自訂許可不會與此附加元件搭配使用。

斯尼克

- 發佈者 – SNYK
- 名稱 – `snyk_runtime-sensor`
- 命名空間：`snyk_runtime-sensor`
- 服務帳戶名稱 – 服務帳戶不會與此附加元件搭配使用。
- AWS 受管 IAM 政策 – 受管政策不會與此附加元件搭配使用。
- 自訂 IAM 許可 – 自訂許可不會與此附加元件搭配使用。
- 設定和使用說明 – 請參閱 Snyk 使用者文件中的 Snyk 執行期感應器。

Stormforge

附加元件名稱為 `stormforge_optimize-Live`，命名空間為 `stormforge-system`。Stormforge 會發佈附加元件。

如需附加元件的資訊，請參閱 StormForge 文件中的安裝 StormForge 代理程式。 StormForge

服務帳戶名稱

服務帳戶不會與此附加元件搭配使用。

AWS 受管 IAM 政策

受管政策不會與此附加元件搭配使用。

自訂 IAM 許可

自訂許可不會與此附加元件搭配使用。

Splunk

附加元件名稱為 `splunk_splunk-otel-collector-chart`，命名空間為 `splunk-monitoring`。Splunk 會發佈附加元件。

如需有關附加元件的資訊，請參閱 Splunk 文件中的安裝 Amazon EKS 的 Splunk 附加元件。

服務帳戶名稱

服務帳戶不會與此附加元件搭配使用。

AWS 受管 IAM 政策

受管政策不會與此附加元件搭配使用。

自訂 IAM 許可

自訂許可不會與此附加元件搭配使用。

Teleport

附加元件名稱為 `teleport_teleport`，命名空間為 `teleport`。Teleport 會發佈附加元件。

如需附加元件的資訊，請參閱 Teleport 文件中的 Teleport 如何運作。

服務帳戶名稱

服務帳戶不會與此附加元件搭配使用。

AWS 受管 IAM 政策

受管政策不會與此附加元件搭配使用。

自訂 IAM 許可

自訂許可不會與此附加元件搭配使用。

滴定

附加元件名稱為 `tetrate-io_istio-distro`，命名空間為 `istio-system`。Tetrate Io 會發佈附加元件。

如需附加元件的詳細資訊，請參閱 Tetrate Istio Distro 網站。

服務帳戶名稱

服務帳戶不會與此附加元件搭配使用。

AWS 受管 IAM 政策

受管政策不會與此附加元件搭配使用。

自訂 IAM 許可

自訂許可不會與此附加元件搭配使用。

上行通用跨平面

附加元件名稱為 `upbound_universal-crossplane`，命名空間為 `upbound-system`。向上發佈附加元件。

如需有關附加元件的資訊，請參閱上行文件中的上行通用跨平面 (UXP)。

服務帳戶名稱

服務帳戶不會與此附加元件搭配使用。

AWS 受管 IAM 政策

受管政策不會與此附加元件搭配使用。

自訂 IAM 許可

自訂許可不會與此附加元件搭配使用。

上風

附加元件名稱為 upwind，命名空間為 upwind。倒轉會發佈附加元件。

如需附加元件的詳細資訊，請參閱上風文件。

服務帳戶名稱

服務帳戶不會與此附加元件搭配使用。

AWS 受管 IAM 政策

受管政策不會與此附加元件搭配使用。

自訂 IAM 許可

自訂許可不會與此附加元件搭配使用。

建立 Amazon EKS 附加元件

Amazon EKS 附加元件是 Amazon EKS 叢集的附加元件軟體。所有 Amazon EKS 附加元件：

- 包含最新的安全修補程式和錯誤修正。
- 經過 驗證 AWS ，可使用 Amazon EKS。
- 減少管理附加元件軟體所需的工作量。

您可以使用 eksctl、 AWS Management Console或 CLI 建立 Amazon EKS AWS 附加元件。如果附加元件需要 IAM 角色，請參閱 Amazon EKS 附加元件中特定附加元件的詳細資訊，以取得建立角色的詳細資訊。

先決條件

在建立附加元件之前，請先完成下列動作：

- 在您為其建立附加元件之前，叢集必須存在。如需詳細資訊，請參閱the section called “建立叢集”。
- 檢查您的附加元件是否需要 IAM 角色。如需詳細資訊，請參閱the section called “驗證相容性”。

- 確認 Amazon EKS 附加元件版本與叢集相容。如需詳細資訊，請參閱the section called “驗證相容性”。
- 確認您的電腦或 AWS CloudShell 上安裝了 0.190.0 版或更新版本的eksctl命令列工具。如需詳細資訊，請參閱 eksctl 網站文章安裝。

程序

您可以使用 eksctl、 AWS Management Console或 CLI 建立 Amazon EKS AWS 附加元件。如果附加元件需要 IAM 角色，請參閱 的可用 Amazon EKS 附加元件中特定附加元件 AWS的詳細資訊，以取得建立角色的詳細資訊。

建立附加元件 (eksctl)

1. 檢視叢集版本可用附加元件的名稱。將 *1.33* 取代為您的叢集版本。

```
eksctl utils describe-addon-versions --kubernetes-version 1.33 | grep AddonName
```

範例輸出如下。

```
"AddonName": "aws-ebs-csi-driver",
                        "AddonName": "coredns",
                        "AddonName": "kube-proxy",
                        "AddonName": "vpc-cni",
                        "AddonName": "adot",
                        "AddonName": "dynatrace_dynatrace-operator",
                        "AddonName": "upbound_universal-crossplane",
                        "AddonName": "teleport_teleport",
                        "AddonName": "factorhouse_kpow",
                        [...]
```

2. 檢視您想要建立之附加元件的可用版本。將 *1.33* 取代為您的叢集版本。以您要檢視版本之附加元件的名稱取代*name-of-addon*。名稱必須是上一個步驟中傳回的其中一個名稱。

```
eksctl utils describe-addon-versions --kubernetes-version 1.33 --name name-of-addon |
 grep AddonVersion
```

下列輸出是針對名為 vpc-cni 的附加元件傳回的範例。您可以看到附加元件有數個可用版本。

```
"AddonVersions": [
```

```
    "AddonVersion": "v1.12.0-eksbuild.1",
    "AddonVersion": "v1.11.4-eksbuild.1",
    "AddonVersion": "v1.10.4-eksbuild.1",
    "AddonVersion": "v1.9.3-eksbuild.1",
```

a. 決定您要建立的附加元件是 Amazon EKS 還是 AWS Marketplace 附加元件。 AWS Marketplace 具有第三方附加元件，要求您完成建立附加元件的額外步驟。

```
eksctl utils describe-addon-versions --kubernetes-version 1.33 --name name-of-addon | grep ProductUrl
```

如果沒有傳回輸出，則該附加元件是 Amazon EKS。如果傳回輸出，則附加元件是 AWS Marketplace 附加元件。下列輸出適用於名為 `teleport_teleport` 的附加元件。

```
"ProductUrl": "https://aws.amazon.com/marketplace/pp?sku=3bda70bb-566f-4976-806c-f96faef18b26"
```

您可以使用傳回的 URL，進一步了解 AWS Marketplace 中的附加元件。如果附加元件需要訂閱，您可以透過 AWS Marketplace 訂閱附加元件。如果您要從 AWS Marketplace 建立附加元件，則您用來建立附加元件的 IAM 主體必須具有建立 AWSServiceRoleForAWSLicenseManagerRole 服務連結角色的許可。如需有關將許可指派給 IAM 實體的詳細資訊，請參閱《IAM 使用者指南》中的新增和移除 IAM 身分許可。

3. 建立 Amazon EKS 附加元件。複製 命令並取代 *user-data*，如下所示：

- 使用您叢集的名稱取代 *my-cluster*。
- 將*name-of-addon*取代為您要建立的附加元件名稱。
- 如果您想要的附加元件版本早於最新版本，請將*####*取代為您想要使用之上一個步驟輸出中傳回的版本編號。
- 如果附加元件使用服務帳戶角色，請將 *111122223333* 取代為您的帳戶 ID，並將 *role-name* 取代為角色名稱。如需為您的服務帳戶建立角色的說明，請參閱您要建立之附加元件的文件。如需附加元件的清單，請參閱 the section called “ AWS 附加元件”。指定服務帳戶角色需要您擁有叢集的 IAM OpenID Connect (OIDC) 提供者。若要判定您的叢集是否已經擁有一個提供者，或是要建立一個提供者，請參閱 the section called “IAM OIDC 供應商”。

  如果附加元件不使用服務帳戶角色，請刪除 `--service-account-role-arn arn:aws:iam::111122223333:role/role-name`。
- 此範例命令會覆寫任何現有自我管理附加元件版本的組態 (如果有的話)。如果您不想覆寫現有自我管理附加元件的組態，請移除 *--force* 選項。如果您移除該選項，並且 Amazon EKS 附加元

件需要覆蓋現有自我管理附加元件的組態，則建立 Amazon EKS 附加元件的操作會失敗並會出現一條錯誤訊息，以幫助您解決衝突。指定此選項之前，請確定 Amazon EKS 附加元件不會管理您需要管理的設定，因為這些設定會以此選項覆寫。

```
eksctl create addon --cluster my-cluster --name name-of-addon --version latest \
    --service-account-role-arn arn:aws: iam::111122223333:role/role-name --force
```

您可以查看該命令所有可用選項的清單。

```
eksctl create addon --help
```

如需有關可用選項的詳細資訊，請參閱 `eksctl` 文件中的 Addons (附加元件)。

建立附加元件AWS （主控台）

1. 開啟 Amazon EKS 主控台。
2. 在左側導覽窗格中選擇叢集。
3. 選擇您要為其建立附加元件的叢集名稱。
4. 選擇附加元件索引標籤。
5. 選擇取得更多附加元件。
6. 在 Select add-ons (選取附加元件) 頁面上，選擇您要新增至叢集的附加元件。您可以視需要新增任意數量的 Amazon EKS 附加元件和 AWS Marketplace 附加元件。

   對於 AWS Marketplace 附加元件，您用來建立附加元件的 IAM 主體必須具有從 AWS LicenseManager 讀取附加元件權利的許可。 AWS LicenseManager 需要 AWSServiceRoleForAWSLicenseManagerRole 服務連結角色 (SLR)，以允許 AWS 資源代表您管理授權。SLR 是每個帳戶的一次性要求，您不需要為每個附加元件或每個叢集建立單獨的 SLR。如需有關將許可指派給 IAM 主體的詳細資訊，請參閱《IAM 使用者指南》中的新增和移除 IAM 身分許可。

   如果您想要安裝的 AWS Marketplace 附加元件未列出，您可以按一下頁面編號以檢視其他頁面結果或在搜尋方塊中搜尋。您也可以在篩選選項中，依類別、廠商或定價模式搜尋，然後從搜尋結果中選擇附加元件。選取要安裝的附加元件後，請選擇下一步。
7. 在設定選取的附加元件設定頁面上，執行以下操作：
   a. 選擇檢視訂閱選項即可開啟訂閱選項表單。檢閱定價詳細資料和法律區段，然後選擇訂閱按鈕以繼續。

b. 針對版本，選擇您要安裝的版本。建議您使用標記為最新的版本，除非您要建立的個別附加元件建議不同的版本。若要判斷附加元件是否有建議的版本，請參閱您要建立之附加元件的文件。如需附加元件的清單，請參閱 the section called " AWS 附加元件"。

c. 您有兩個設定附加元件角色的選項：EKS Pod 身分 服務帳戶 (IRSA) 的 IAM 角色和 IAM 角色。針對您偏好的選項，請遵循以下適當的步驟。如果您選取的所有附加元件在狀態下都需要訂閱，請選擇下一步。在叢集建立之後，您才能進一步設定這些附加元件。對於狀態下沒有需要訂閱的附加元件，請執行下列動作：

   i. 對於服務帳戶的 Pod Identity IAM 角色，您可以使用現有的 EKS Pod Identity IAM 角色，或使用建立建議的角色按鈕建立角色。此欄位只會提供具有適當信任政策的選項。如果沒有要選取的角色，則您沒有具有相符信任政策的現有角色。若要為所選附加元件的服務帳戶設定 EKS Pod Identity IAM 角色，請選擇建立建議的角色。角色建立精靈會在個別視窗中開啟。精靈會自動填入角色資訊，如下所示。對於您要建立 EKS Pod Identity IAM 角色的每個附加元件，請完成 IAM 精靈中的步驟，如下所示。

      - 在選取信任實體步驟中，EKS AWS 的服務選項和 EKS - Pod Identity 的使用案例會預先選取，且會自動填入附加元件的適當信任政策。例如，將使用包含 pods.eks.amazonaws.com IAM Principal 的適當信任政策來建立角色，如 中所述the section called "EKS Pod 身分識別的優勢"。選擇下一步。
      - 在新增許可步驟中，針對 附加元件預先選取角色政策的適當受管政策。例如，對於 Amazon VPC CNI 附加元件，將使用 `AmazonEKS_CNI_Policy` 中詳述的受管政策建立角色the section called "Kubernetes 專用 Amazon VPC CNI 外掛程式"。選擇下一步。
      - 在名稱、檢閱和建立步驟中，在角色名稱中，會自動填入附加元件的預設角色名稱。例如，對於 Amazon VPC CNI 附加元件，將使用名稱 AmazonEKSPodIdentityAmazonVPCCNIRole 建立角色。在描述中，預設描述會自動填入附加元件的適當描述。例如，對於 Amazon VPC CNI 附加元件，將使用描述建立角色 允許在 Amazon EKS 叢集中執行的 Pod 存取 AWS 資源。在信任政策中，檢視附加元件的填入信任政策。選擇建立角色。

        注意：保留預設角色名稱可讓 EKS 在新叢集中或將附加元件新增至現有叢集時，預先選取附加元件的角色。您仍然可以覆寫此名稱，該角色將可用於叢集的附加元件，但需要從下拉式清單中手動選取該角色。

   ii. 如需狀態下不需要訂閱的附加元件，以及您要使用 IRSA 設定角色的位置，請參閱您要建立的附加元件文件，以建立 IAM 政策並將其連接至角色。如需附加元件的清單，請參閱 the section called " AWS 附加元件"。選取 IAM 角色需要您擁有叢集的 IAM OpenID Connect (OIDC) 提供者。若要判定您的叢集是否已經擁有一個提供者，或是要建立一個提供者，請參閱 the section called "IAM OIDC 供應商"。

iii. 選擇 Optional configuration settings (選用組態設定)。

iv. 如果附加元件需要組態，請在 Configuration values (組態值) 方塊中輸入。若要判斷附加元件是否需要組態資訊，請參閱您要建立之附加元件的文件。如需附加元件的清單，請參閱 the section called “ AWS 附加元件”。

v. 選擇衝突解決方法的其中一個可用選項。如果您選擇覆寫衝突解決方法，可以使用 Amazon EKS 附加元件設定覆寫現有附加元件的一個或多個設定。如果您未啟用此選項，且與現有設定發生衝突，則操作會失敗。您可以使用產生的錯誤訊息對此衝突進行疑難排解。選擇此選項之前，請確定 Amazon EKS 附加元件不會管理自我管理所需的設定。

vi. 選擇下一步。

8. 在檢閱並新增頁面上，選擇建立。附加元件安裝完成後，您會看到已安裝的附加元件。
9. 如果您安裝的任何附加元件需要訂閱，請完成下列步驟：
   a. 選擇附加元件右下角的 Subscribe (訂閱) 按鈕。系統會將您導向 AWS Marketplace 中附加元件的頁面。請閱讀附加元件的相關資訊，例如其產品概觀和定價資訊。
   b. 選取附加元件頁面右上方的 Continue to Subscribe (繼續訂閱) 按鈕。
   c. 請完整閱讀條款與條件。如果您同意，請選擇 Accept Terms (接受條款)。處理訂閱可能需要幾分鐘的時間。訂閱正在處理時，Return to Amazon EKS Console (返回至 Amazon EKS 主控台) 按鈕會呈現灰色。
   d. 訂閱完成處理後，Return to Amazon EKS Console (返回至 Amazon EKS 主控台) 按鈕將不再呈現灰色。選擇按鈕以返回叢集的 Amazon EKS 主控台 Add-ons (附加元件) 索引標籤。
   e. 針對您訂閱的附加元件，請選擇 Remove and reinstall (移除後重新安裝)，然後選擇 Reinstall add-on (重新安裝附加元件)。安裝附加元件可能需要幾分鐘時間。安裝完成後，您可以設定附加元件。

建立附加元件 (AWS CLI)

1. 您需要在裝置或 AWS CloudShell 上安裝和設定版本 `2.12.3`或更新版本或版本 `1.27.160`或更新版本的 AWS 命令列界面 (AWS CLI)。若要檢查您目前的版本，請使用 `aws --version | cut -d / -f2 | cut -d ' ' -f1`。適用於 macOS 的 `yum`、 `apt-get`或 Homebrew 等套件管理員通常是最新版本 CLI AWS 後面的幾個版本。若要安裝最新版本，請參閱《 AWS 命令列界面使用者指南》中的使用 aws 設定安裝 和快速組態。 https://docs.aws.amazon.com/cli/latest/userguide/cli-configure-quickstart.html#cli-configure-quickstart-config安裝在 AWS CloudShell 中的 AWS CLI 版本也可能是最新版本後面的幾個版本。若要更新它，請參閱《CloudShell AWS 使用者指南》中的將 CLI 安裝到您的主目錄。 AWS CloudShell

2. 判斷有哪些附加元件可用。您可以查看所有可用的附加元件、其類型及其發佈者。您也可以查看可透過 AWS Marketplace 取得的附加元件 URL。將 *1.33* 取代為您的叢集版本。

```
aws eks describe-addon-versions --kubernetes-version 1.33 \
    --query 'addons[].{MarketplaceProductUrl: marketplaceInformation.productUrl,
 Name: addonName, Owner: owner Publisher: publisher, Type: type}' --output table
```

範例輸出如下。

```
----------------------------------------------------------------------------------------------------
|
 DescribeAddonVersions
  |
+-----------------------------------------------------------------
+------------------------------+------------------+--------------
+---------------------+
|                     MarketplaceProductUrl                      |             Name
          |      Owner       |  Publisher   |        Type         |
+-----------------------------------------------------------------
+------------------------------+------------------+--------------
+---------------------+
|  None                                                          |  aws-ebs-csi-driver
          |  aws             |  eks         |  storage            |
|  None                                                          |  coredns
          |  aws             |  eks         |  networking         |
|  None                                                          |  kube-proxy
          |  aws             |  eks         |  networking         |
|  None                                                          |  vpc-cni
          |  aws             |  eks         |  networking         |
|  None                                                          |  adot
          |  aws             |  eks         |  observability      |
| https://aws.amazon.com/marketplace/pp/prodview-brb73nceicv7u |
 dynatrace_dynatrace-operator |  aws-marketplace |  dynatrace   |  monitoring
 |
| https://aws.amazon.com/marketplace/pp/prodview-uhc2iwi5xysoc |  upbound_universal-
crossplane |  aws-marketplace |  upbound     |  infra-management   |
| https://aws.amazon.com/marketplace/pp/prodview-hd2ydsrgqy4li |  teleport_teleport
          |  aws-marketplace |  teleport    |  policy-management  |
| https://aws.amazon.com/marketplace/pp/prodview-vgghgqdsplhvc |  factorhouse_kpow
          |  aws-marketplace |  factorhouse |  monitoring         |
|  [...]                                                         |  [...]
          |  [...]           |  [...]       |  [...]              |
```

```
+-------------------------------------------------------------
+------------------------------+-----------------+-------------
+---------------------+
```

您的輸出可能不同。在此範例輸出中，有三種不同 networking 類型的附加元件可用，以及五個具有 eks 類型發佈者的附加元件。Owner 欄中具有 aws-marketplace的附加元件可能需要訂閱，才能進行安裝。您可以造訪 URL 以深入了解附加元件並訂閱該附加元件。

3. 您可以查看每個附加元件可用的版本。將 *1.33* 取代為您的叢集版本，並將 *vpc-cni* 取代為上一個步驟中傳回的附加元件名稱。

```
aws eks describe-addon-versions --kubernetes-version 1.33 --addon-name vpc-cni \
    --query 'addons[].addonVersions[].{Version: addonVersion, Defaultversion:
 compatibilities[0].defaultVersion}' --output table
```

範例輸出如下。

```
-----------------------------------------
|          DescribeAddonVersions         |
+-----------------+----------------------+
| Defaultversion  |       Version        |
+-----------------+----------------------+
|  False          |  v1.12.0-eksbuild.1  |
|  True           |  v1.11.4-eksbuild.1  |
|  False          |  v1.10.4-eksbuild.1  |
|  False          |  v1.9.3-eksbuild.1   |
+-----------------+----------------------+
```

Defaultversion 欄位中具有 True 的版本在預設情況下是用來建立附加元件的版本。

4. (選用) 執行下列命令，尋找您所選附加元件的組態選項：

```
aws eks describe-addon-configuration --addon-name vpc-cni --addon-version v1.12.0-
eksbuild.1
```

```
{
    "addonName": "vpc-cni",
    "addonVersion": "v1.12.0-eksbuild.1",
    "configurationSchema": "{\"$ref\":\"#/definitions/VpcCni\",\"$schema\":\"http://
json-schema.org/draft-06/schema#\",\"definitions\":{\"Cri\":{\"additionalProperties
\":false,\"properties\":{\"hostPath\":{\"$ref\":\"#/definitions/HostPath\"}},\"title
```

```
\":\"Cri\",\"type\":\"object\"},\"Env\":{\"additionalProperties\":false,\"properties
\":{\"ADDITIONAL_ENI_TAGS\":{\"type\":\"string\"},\"AWS_VPC_CNI_NODE_PORT_SUPPORT
\":{\"format\":\"boolean\",\"type\":\"string\"},\"AWS_VPC_ENI_MTU\":{\"format\":
\"integer\",\"type\":\"string\"},\"AWS_VPC_K8S_CNI_CONFIGURE_RPFILTER\":{\"format
\":\"boolean\",\"type\":\"string\"},\"AWS_VPC_K8S_CNI_CUSTOM_NETWORK_CFG\":{\"format
\":\"boolean\",\"type\":\"string\"},\"AWS_VPC_K8S_CNI_EXTERNALSNAT\":{\"format\":
\"boolean\",\"type\":\"string\"},\"AWS_VPC_K8S_CNI_LOGLEVEL\":{\"type\":\"string\"},
\"AWS_VPC_K8S_CNI_LOG_FILE\":{\"type\":\"string\"},\"AWS_VPC_K8S_CNI_RANDOMIZESNAT
\":{\"type\":\"string\"},\"AWS_VPC_K8S_CNI_VETHPREFIX\":{\"type\":\"string\"},
\"AWS_VPC_K8S_PLUGIN_LOG_FILE\":{\"type\":\"string\"},\"AWS_VPC_K8S_PLUGIN_LOG_LEVEL
\":{\"type\":\"string\"},\"DISABLE_INTROSPECTION\":{\"format\":\"boolean\",\"type
\":\"string\"},\"DISABLE_METRICS\":{\"format\":\"boolean\",\"type\":\"string
\"},\"DISABLE_NETWORK_RESOURCE_PROVISIONING\":{\"format\":\"boolean\",\"type
\":\"string\"},\"ENABLE_POD_ENI\":{\"format\":\"boolean\",\"type\":\"string
\"},\"ENABLE_PREFIX_DELEGATION\":{\"format\":\"boolean\",\"type\":\"string\"},
\"WARM_ENI_TARGET\":{\"format\":\"integer\",\"type\":\"string\"},\"WARM_PREFIX_TARGET
\":{\"format\":\"integer\",\"type\":\"string\"}},\"title\":\"Env\",\"type\":
\"object\"},\"HostPath\":{\"additionalProperties\":false,\"properties\":{\"path
\":{\"type\":\"string\"}},\"title\":\"HostPath\",\"type\":\"object\"},\"Limits
\":{\"additionalProperties\":false,\"properties\":{\"cpu\":{\"type\":\"string
\"},\"memory\":{\"type\":\"string\"}},\"title\":\"Limits\",\"type\":\"object\"},
\"Resources\":{\"additionalProperties\":false,\"properties\":{\"limits\":{\"$ref\":
\"#/definitions/Limits\"},\"requests\":{\"$ref\":\"#/definitions/Limits\"}},\"title
\":\"Resources\",\"type\":\"object\"},\"VpcCni\":{\"additionalProperties\":false,
\"properties\":{\"cri\":{\"$ref\":\"#/definitions/Cri\"},\"env\":{\"$ref\":\"#/
definitions/Env\"},\"resources\":{\"$ref\":\"#/definitions/Resources\"}},\"title\":
\"VpcCni\",\"type\":\"object\"}}}"
}
```

輸出是標準的 JSON 結構描述。

以下是適用於上述結構描述的有效組態值 (以 JSON 格式) 範例。

```
{
  "resources": {
    "limits": {
      "cpu": "100m"
    }
  }
}
```

以下是適用於上述結構描述的有效組態值 (以 YAML 格式) 範例。

```
resources:
  limits:
    cpu: 100m
```

5. 判斷附加元件是否需要 IAM 許可。若是如此，您需要 (1) 判斷您是否要針對服務帳戶 (IRSA) 使用 EKS Pod 身分或 IAM 角色，(2) 判斷要與附加元件搭配使用的 IAM 角色 ARN，以及 (3) 判斷附加元件使用的 Kubernetes 服務帳戶名稱。如需詳細資訊，請參閱the section called “擷取 IAM 資訊”。
   - 如果附加元件支援，Amazon EKS 建議使用 EKS Pod 身分。這需要在叢集上安裝 Pod Identity Agent。如需搭配附加元件使用 Pod 身分的詳細資訊，請參閱 the section called “IAM 角色”。
   - 如果未設定 EKS Pod 身分的附加元件或叢集，請使用 IRSA。確認您的叢集上已設定 IRSA。
   - 檢閱 Amazon EKS 附加元件文件，以判斷附加元件是否需要 IAM 許可和相關聯 Kubernetes 服務帳戶的名稱。

     a. 建立 Amazon EKS 附加元件。將隨後的命令複製到您的裝置。視需要對命令進行下列修改，然後執行修改後的命令：
   - 使用您叢集的名稱取代 *my-cluster*。
   - 將 *vpc-cni* 取代為您要建立之上一個步驟的輸出中傳回的附加元件名稱。
   - 將 *version-number* 取代為您要使用之上一個步驟輸出中傳回的版本。
   - 如果附加元件不需要 IAM 許可，請刪除 *<service-account-configuration>*。
   - 執行以下任意一項：
     - 如果附加元件 (1) 需要 IAM 許可，且 (2) 您的叢集使用 EKS Pod 身分，請將 *<service-account-configuration>* 取代為下列 Pod 身分關聯。以附加元件使用的服務帳戶名稱取代 *<service-account-name>*。將 *<role-arn>* 取代為 IAM 角色的 ARN。角色必須具有 EKS Pod 身分所需的信任政策。

       ```
       --pod-identity-associations 'serviceAccount=<service-account-name>,roleArn=<role-arn>'
       ```
     - 如果附加元件 (1) 需要 IAM 許可，且 (2) 您的叢集使用 IRSA，請使用下列 IRSA 組態取代 *<service-account-configuration>*。將 *111122223333* 取代為您的帳戶 ID，並將 *role-name* 取代為您已建立的現有 IAM 角色名稱。如需建立角色的說明，請參閱您要建立之附加元件的文件。如需附加元件的清單，請參閱 the section called “ AWS 附加元件”。指定服務帳戶角色需要您擁有叢集的 IAM OpenID Connect (OIDC) 提供者。若要判定您的叢集是否已經擁有一個提供者，或是要建立一個提供者，請參閱 the section called “IAM OIDC 供應商”。

       ```
       --service-account-role-arn arn:aws::iam::111122223333:role/role-name
       ```

- 這些範例命令會覆寫任何現有自我管理附加元件版本的 `--configuration-values` 選項 (如果有的話)。將其取代為所需的組態值，例如字串或檔案輸入。如果您不想提供組態值，請刪除 `--configuration-values`選項。如果您不希望 AWS CLI 覆寫現有自我管理附加元件的組態，請移除 *`--resolve-conflicts OVERWRITE`* 選項。如果您移除該選項，並且 Amazon EKS 附加元件需要覆蓋現有自我管理附加元件的組態，則建立 Amazon EKS 附加元件的操作會失敗並會出現一條錯誤訊息，以幫助您解決衝突。指定此選項之前，請確定 Amazon EKS 附加元件不會管理您需要管理的設定，因為這些設定會以此選項覆寫。

```
aws eks create-addon --cluster-name my-cluster --addon-name vpc-cni --addon-version
 version-number \
    <service-account-configuration> --configuration-values '{"resources":
{"limits":{"cpu":"100m"}}}' --resolve-conflicts OVERWRITE
```

```
aws eks create-addon --cluster-name my-cluster --addon-name vpc-cni --addon-version
 version-number \
   <service-account-configuration> --configuration-values 'file://example.yaml' --
resolve-conflicts OVERWRITE
```

如需可用選項的完整清單，請參閱 Amazon EKS Command Line Reference (《Amazon EKS 命令列參考》) 中的 `create-addon` 。如果您建立的附加元件已aws-marketplace列在上一個步驟的 Owner 欄中，則建立可能會失敗，而且您可能會收到類似下列錯誤的錯誤訊息。

```
{
    "addon": {
        "addonName": "addon-name",
        "clusterName": "my-cluster",
        "status": "CREATE_FAILED",
        "addonVersion": "version",
        "health": {
            "issues": [
                {
                    "code": "AddonSubscriptionNeeded",
                    "message": "You are currently not subscribed to this add-on. To
 subscribe, visit the AWS Marketplace console, agree to the seller EULA, select the
 pricing type if required, then re-install the add-on"
                }
            ]
        }
    }
```

```
}
```

如果您收到類似上一個輸出中之錯誤的錯誤，請造訪上一步輸出中的 URL，以訂閱附加元件。訂閱後，再次執行 `create-addon` 命令。

更新 Amazon EKS 附加元件

發行新版本或將叢集更新為新的 Kubernetes 次要版本後，Amazon EKS 不會自動更新附加元件。若要更新現有叢集的附加元件，您必須啟動更新。在您啟動更新後，Amazon EKS 會為您更新附加元件。在更新附加元件之前，請檢閱附加元件目前的文件。如需可用附加元件的清單，請參閱 the section called " AWS 附加元件"。如果附加元件需要 IAM 角色，請參閱 的可用 Amazon EKS 附加元件中特定附加元件 AWS的詳細資訊，以取得建立角色的詳細資訊。

先決條件

建立附加元件之前，請先完成下列動作：

- 檢查您的附加元件是否需要 IAM 角色。如需詳細資訊，請參閱the section called "Amazon EKS 附加元件"。
- 確認 Amazon EKS 附加元件版本與您的叢集相容。如需詳細資訊，請參閱the section called "驗證相容性"。

程序

您可以使用 `eksctl`、 AWS Management Console或 CLI 更新 Amazon EKS AWS 附加元件。

更新附加元件 (eksctl)

1. 判斷叢集上安裝的目前附加元件和附加元件版本。使用您叢集的名稱取代 *`my-cluster`*。

```
eksctl get addon --cluster my-cluster
```

範例輸出如下。

```
NAME        VERSION              STATUS  ISSUES  IAMROLE  UPDATE AVAILABLE
coredns     v1.8.7-eksbuild.2    ACTIVE  0
kube-proxy  v1.23.7-eksbuild.1   ACTIVE  0                v1.23.8-eksbuild.2
```

```
vpc-cni     v1.10.4-eksbuild.1   ACTIVE  0                 v1.12.0-eksbuild.1,v1.11.4-
eksbuild.1,v1.11.3-eksbuild.1,v1.11.2-eksbuild.1,v1.11.0-eksbuild.1
```

您的輸出可能看起來有所不同，具體取決於您的叢集上有哪些附加元件和版本。您可以看到，在前面的範例輸出中，叢集上的兩個現有附加元件在 UPDATE AVAILABLE 欄位中具有較新的版本。

2. 更新附加元件。

   a. 將隨後的命令複製到您的裝置。視需要對命令進行下列修改：

   - 使用您叢集的名稱取代 *my-cluster*。
   - 將 *region-code* 取代為您的叢集所在的 AWS 區域。
   - 將 *vpc-cni* 取代為您要更新之上一個步驟輸出中傳回的附加元件名稱。
   - 如果您想要更新至比最新可用版本更舊的版本，請將*##*版本取代為上一個步驟的輸出中傳回的版本編號。某些附加元件有建議的版本。如需詳細資訊，請參閱您要更新之附加元件的文件。如需附加元件的清單，請參閱 the section called “ AWS 附加元件”。* 如果附加元件使用 Kubernetes 服務帳戶和 IAM 角色，請將 *111122223333* 取代為您建立的現有 IAM 角色名稱的帳戶 ID 和 *role-name*。如需建立角色的說明，請參閱您要建立之附加元件的文件。如需附加元件的清單，請參閱 the section called “ AWS 附加元件”。指定服務帳戶角色需要您擁有叢集的 IAM OpenID Connect (OIDC) 提供者。若要判定您的叢集是否已經擁有一個提供者，或是要建立一個提供者，請參閱 the section called “IAM OIDC 供應商”。

     如果附加元件不使用 Kubernetes 服務帳戶和 IAM 角色，請刪除該serviceAccountRoleARN: arn:aws: iam::*111122223333*:role/*role-name* 行。

   - *preserve* 選項會保留附加元件的現有值。如果您已設定附加元件設定的自訂值，但未使用此選項，Amazon EKS 會以其預設值覆寫您的值。如果您使用此選項，建議您在更新生產叢集上的附加元件之前，測試非生產叢集上的任何欄位和值變更。如果您將此值變更為 overwrite，則所有設定都會變更為 Amazon EKS 預設值。如果您已為任何設定設定自訂值，這些值可能會以 Amazon EKS 預設值覆寫。如果您將此值變更為 none，Amazon EKS 不會變更任何設定的值，但更新可能會失敗。若更新失敗，您會收到錯誤訊息，以協助您解決衝突。

     ```
     cat >update-addon.yaml <<EOF
     apiVersion: eksctl.io/v1alpha5
     kind: ClusterConfig
     metadata:
       name: my-cluster
       region: region-code
     ```

```
addons:
- name: vpc-cni
  version: latest
  serviceAccountRoleARN: arn:aws: iam::111122223333:role/role-name
  resolveConflicts: preserve
EOF
```

b. 執行修改後的命令來建立 update-addon.yaml 檔案。

c. 將組態檔案套用至叢集。

```
eksctl update addon -f update-addon.yaml
```

如需更新附加元件的詳細資訊，請參閱 eksctl 文件中的更新附加元件。

更新附加元件AWS （主控台）

1. 開啟 Amazon EKS 主控台。
2. 在左側導覽窗格中選擇叢集。
3. 選擇您要更新附加元件的叢集名稱。
4. 選擇附加元件索引標籤。
5. 選擇您要更新的附加元件。
6. 選擇編輯。
7. 在設定######頁面上，執行下列動作：

   a. 選擇您要使用的版本。附加元件可能有建議的版本。如需詳細資訊，請參閱您要更新之附加元件的文件。如需附加元件的清單，請參閱 the section called “ AWS 附加元件”。

   b. 您有兩個設定附加元件角色的選項：EKS Pod 身分 服務帳戶 (IRSA) 的 IAM 角色和 IAM 角色。對於您偏好的選項，請遵循以下適當的步驟。如果您選取的所有附加元件在狀態下都需要訂閱，請選擇下一步。對於狀態下沒有需要訂閱的附加元件，請執行下列動作：

      i. 對於服務帳戶的 Pod Identity IAM 角色，您可以使用現有的 EKS Pod Identity IAM 角色，或使用建立建議的角色按鈕建立角色。此欄位只會提供具有適當信任政策的選項。如果沒有要選取的角色，則表示您沒有具有相符信任政策的現有角色。若要為所選附加元件的服務帳戶設定 EKS Pod Identity IAM 角色，請選擇建立建議的角色。角色建立精靈會在個別視窗中開啟。精靈會自動填入角色資訊，如下所示。對於您要建立 EKS Pod Identity IAM 角色的每個附加元件，請完成 IAM 精靈中的步驟，如下所示。

- 在選取信任實體步驟中，EKS AWS 的服務選項和 EKS - Pod Identity 的使用案例會預先選取，且會自動填入附加元件的適當信任政策。例如，將使用包含 pods.eks.amazonaws.com IAM Principal 的適當信任政策來建立角色，如 中所述the section called “EKS Pod 身分識別的優勢”。選擇下一步。
- 在新增許可步驟中，針對 附加元件預先選取角色政策的適當受管政策。例如，對於 Amazon VPC CNI 附加元件，將使用 中AmazonEKS_CNI_Policy詳述的受管政策來建立角色the section called “Kubernetes 專用 Amazon VPC CNI 外掛程式”。選擇下一步。
- 在名稱、檢閱和建立步驟中，在角色名稱中，會自動填入附加元件的預設角色名稱。例如，對於 Amazon VPC CNI 附加元件，將使用名稱 AmazonEKSPodIdentityAmazonVPCCNIRole 建立角色。在描述中，預設描述會自動填入附加元件的適當描述。例如，對於 Amazon VPC CNI 附加元件，將使用描述建立角色 允許在 Amazon EKS 叢集中執行的 Pod 存取 AWS 資源。在信任政策中，檢視附加元件的填入信任政策。選擇建立角色。

  > **Note**
  >
  > 保留預設角色名稱可讓 EKS 在新叢集中或將附加元件新增至現有叢集時，預先選取附加元件的角色。您仍然可以覆寫此名稱，該角色將可用於叢集的附加元件，但需要從下拉式清單中手動選取該角色。

ii. 對於狀態下不需要訂閱的附加元件，以及您想要使用 IRSA 設定角色的位置，請參閱您要建立的附加元件文件，以建立 IAM 政策並將其連接至角色。如需附加元件的清單，請參閱 the section called “ AWS 附加元件”。選取 IAM 角色需要您擁有叢集的 IAM OpenID Connect (OIDC) 提供者。若要判定您的叢集是否已經擁有一個提供者，或是要建立一個提供者，請參閱 the section called “IAM OIDC 供應商”。

c. 展開選用組態設定。

d. 在組態值中，輸入任何附加元件特定的組態資訊。如需詳細資訊，請參閱您要更新之附加元件的文件。如需附加元件清單，請參閱 the section called “ AWS 附加元件”... 如需衝突解決方法，請選取其中一個選項。如果您已設定附加元件設定的自訂值，我們建議您使用 Preserve (保留) 選項。如果您未選擇此選項，Amazon EKS 會使用其預設值覆寫您的值。如果您使用此選項，建議您在更新生產叢集上的附加元件之前，測試非生產叢集上的任何欄位和值變更。如果您將此值變更為覆寫，所有設定都會變更為 Amazon EKS 預設值。如果您已為任何設定設定自訂值，這些值可能會以 Amazon EKS 預設值覆寫。如果您將此值變更為無，Amazon EKS 不會變更任何設定的值，但更新可能會失敗。若更新失敗，您會收到錯誤訊息，以協助您解決衝突。

8. 選擇儲存變更。

更新附加元件 (AWS CLI)

1. 您需要在裝置或 AWS CloudShell 上安裝和設定版本 2.12.3或更新版本或版本 1.27.160或更新版本的 AWS 命令列界面 (AWS CLI)。若要檢查您目前的版本，請使用 aws --version | cut -d / -f2 | cut -d ' ' -f1。適用於 macOS 的 yum、 apt-get或 Homebrew 等套件管理員通常是最新版本 CLI AWS 後面的幾個版本。若要安裝最新版本，請參閱《 AWS 命令列界面使用者指南》中的使用 aws 設定安裝 和快速組態。 https://docs.aws.amazon.com/cli/latest/userguide/cli-configure-quickstart.html#cli-configure-quickstart-config在 AWS CloudShell 中安裝的 AWS CLI 版本也可能是最新版本後面的幾個版本。若要更新它，請參閱《CloudShell AWS 使用者指南》中的將 CLI 安裝到您的主目錄。 AWS CloudShell
2. 請參閱已安裝附加元件的清單。使用您叢集的名稱取代 *my-cluster*。

   ```
   aws eks list-addons --cluster-name my-cluster
   ```

   範例輸出如下。

   ```
   {
       "addons": [
           "coredns",
           "kube-proxy",
           "vpc-cni"
       ]
   }
   ```

3. 檢視您想要更新之附加元件的目前版本。將 *my-cluster* 取代為您的叢集名稱，並將 *vpc-cni* 取代為您要更新的附加元件名稱。

   ```
   aws eks describe-addon --cluster-name my-cluster --addon-name vpc-cni --query
    "addon.addonVersion" --output text
   ```

   範例輸出如下。

   ```
   v1.10.4-eksbuild.1
   ```

4. 決定哪些版本的附加元件可用於叢集的版本。將 *1.33* 取代為您的叢集版本，並將 *vpc-cni* 取代為您要更新的附加元件名稱。

   ```
   aws eks describe-addon-versions --kubernetes-version 1.33 --addon-name vpc-cni \
   ```

```
    --query 'addons[].addonVersions[].{Version: addonVersion, Defaultversion:
compatibilities[0].defaultVersion}' --output table
```

範例輸出如下。

```
-----------------------------------------
|          DescribeAddonVersions         |
+-----------------+----------------------+
| Defaultversion  |       Version        |
+-----------------+----------------------+
|  False          |  v1.12.0-eksbuild.1  |
|  True           |  v1.11.4-eksbuild.1  |
|  False          |  v1.10.4-eksbuild.1  |
|  False          |  v1.9.3-eksbuild.1   |
+-----------------+----------------------+
```

`Defaultversion` 欄位中具有 `True` 的版本在預設情況下是用來建立附加元件的版本。

5. 更新您的附加元件。將隨後的命令複製到您的裝置。視需要對命令進行下列修改，然後執行修改後的命令。如需此命令的詳細資訊，請參閱《Amazon EKS 命令列參考》中的 update-addon。
   - 使用您叢集的名稱取代 *my-cluster*。
   - 將 *vpc-cni* 取代為您要更新的附加元件名稱，該附加元件在上一個步驟的輸出中傳回。
   - 將 *version-number* 取代為您要更新之上一個步驟輸出中傳回的版本。某些附加元件有建議的版本。如需詳細資訊，請參閱您要更新之附加元件的文件。如需附加元件的清單，請參閱 the section called “ AWS 附加元件”。* 如果附加元件使用 Kubernetes 服務帳戶和 IAM 角色，請將 *111122223333* 取代為您建立的現有 IAM 角色名稱的帳戶 ID 和 *role-name*。如需建立角色的說明，請參閱您要建立之附加元件的文件。如需附加元件的清單，請參閱 the section called “ AWS 附加元件”。指定服務帳戶角色需要您擁有叢集的 IAM OpenID Connect (OIDC) 提供者。若要判定您的叢集是否已經擁有一個提供者，或是要建立一個提供者，請參閱 the section called “IAM OIDC 供應商”。

     如果附加元件不使用 Kubernetes 服務帳戶和 IAM 角色，請刪除該`serviceAccountRoleARN: arn:aws: iam::`*111122223333*`:role/`*role-name* 行。
   - `--resolve-conflicts PRESERVE` 選項會保留附加元件的現有值。如果您已設定附加元件設定的自訂值，但未使用此選項，Amazon EKS 會以其預設值覆寫您的值。如果您使用此選項，建議您在更新生產叢集上的附加元件之前，測試非生產叢集上的任何欄位和值變更。如果您將此值變更為 `OVERWRITE`，則所有設定都會變更為 Amazon EKS 預設值。如果您已為任何設定設定自訂值，這些值可能會以 Amazon EKS 預設值覆寫。如果您將此值變更為 `NONE`，Amazon EKS

不會變更任何設定的值，但更新可能會失敗。若更新失敗，您會收到錯誤訊息，以協助您解決衝突。

- 如果您想要移除所有自訂組態，請使用 *--configuration-values '{}'* 選項執行更新。這會將所有自訂組態設定回預設值。如果您不想變更自訂組態，請不要提供 *--configuration-values* 旗標。如果您想要調整自訂組態，請以新參數取代 *{}*。

```
aws eks update-addon --cluster-name my-cluster --addon-name vpc-cni --addon-version
 version-number \
    --service-account-role-arn arn:aws: iam::111122223333:role/role-name --
configuration-values '{}' --resolve-conflicts PRESERVE
```

6. 查看更新狀態。將 *my-cluster* 取代為叢集的名稱，並將 *vpc-cni* 取代為您要更新的附加元件名稱。

```
aws eks describe-addon --cluster-name my-cluster --addon-name vpc-cni
```

範例輸出如下。

```
{
    "addon": {
        "addonName": "vpc-cni",
        "clusterName": "my-cluster",
        "status": "UPDATING",
    }
}
```

當狀態為 ACTIVE 時，即表示更新已完成。

驗證 Amazon EKS 附加元件版本與叢集的相容性

在建立 Amazon EKS 附加元件之前，您需要驗證 Amazon EKS 附加元件版本是否與您的叢集相容。

使用 describe-addon-versions API 列出可用的 EKS 附加元件版本，以及每個附加元件版本支援的 Kubernetes 版本。

1. 確認已安裝 AWS CLI 並使用 `aws sts get-caller-identity`。如果此命令無法運作，請了解如何開始使用 AWS CLI。
2. 決定您要擷取版本相容性資訊的附加元件名稱，例如 `amazon-cloudwatch-observability`。

3. 判斷叢集的 Kubernetes 版本，例如 1.33。
4. 使用 AWS CLI 擷取與叢集的 Kubernetes 版本相容的附加元件版本。

```
aws eks describe-addon-versions --addon-name amazon-cloudwatch-observability --
kubernetes-version 1.33
```

範例輸出如下。

```
{
    "addons": [
        {
            "addonName": "amazon-cloudwatch-observability",
            "type": "observability",
            "addonVersions": [
                {
                    "addonVersion": "vX.X.X-eksbuild.X",
                    "architecture": [
                        "amd64",
                        "arm64"
                    ],
                    "computeTypes": [
                        "ec2",
                        "auto",
                        "hybrid"
                    ],
                    "compatibilities": [
                        {
                            "clusterVersion": "1.33",
                            "platformVersions": [
                                "*"
                            ],
                            "defaultVersion": true
                        }
                    ],
                }
            ]
        }
    ]
}
```

此輸出顯示附加元件版本與 Kubernetes 叢集版本 vX.X.X-eksbuild.X 相容1.33。

與運算類型的附加元件相容性

`describe-addon-versions` 輸出中的 `computeTypes` 欄位表示附加元件與 EKS Auto Mode 受管節點或混合節點的相容性。標記的附加元件`auto`適用於 EKS Auto Mode 的雲端型 AWS受管基礎設施，而標記的附加元件`hybrid`可在連接到 EKS 雲端控制平面的內部部署節點上執行。

如需詳細資訊，請參閱the section called “Amazon EKS Auto 模式的考量事項”。

從叢集移除 Amazon EKS 附加元件

您可以使用 `eksctl`、 AWS Management Console或 CLI 從叢集中移除 Amazon EKS AWS 附加元件。

當您從叢集移除 Amazon EKS 附加元件時：

- 此附加元件提供的功能沒有任何停機時間。
- 如果您使用服務帳戶 (IRSA) 的 IAM 角色，且附加元件具有與其相關聯的 IAM 角色，則不會移除 IAM 角色。
- 如果您使用的是 Pod 身分，則會移除附加元件擁有的任何 Pod 身分關聯。如果您指定 CLI AWS `--preserve`的選項，則會保留關聯。
- Amazon EKS 停止管理附加元件的設定。
- 當有新版本可用時，主控台會停止通知您。
- 您無法使用任何 AWS 工具或 APIs更新附加元件。
- 您可以選擇將附加元件軟體保留在叢集上，以便您可以自我管理附加元件軟體，或從叢集中移除附加元件軟體。如果您的叢集上的任何資源都不依賴於附加元件提供的功能，您應該僅從叢集中移除附加元件軟體。

先決條件

在建立附加元件之前，請先完成下列操作：

- 現有 Amazon EKS 叢集。若要部署叢集，請參閱 開始使用。
- 檢查您的附加元件是否需要 IAM 角色。如需詳細資訊，請參閱
- 裝置或 AWS CloudShell 上安裝的`eksctl`命令列工具版本 `0.210.0`或更新版本。若要安裝或更新 `eksctl`，請參閱 `eksctl` 文件中的安裝。

程序

移除 Amazon EKS 附加元件時有兩個選項。

- Preserve add-on software on your cluster (在叢集上保留附加元件軟體)：此選項會移除任何設定的 Amazon EKS 管理。其也會移除 Amazon EKS 通知您更新的功能，並在您啟動更新後自動更新 Amazon EKS 附加元件。不過，該選項會保留您叢集上的附加元件軟體。此選項會使附加元件成為自我管理安裝，而不是 Amazon EKS 附加元件。使用此選項時，附加元件不會停機。
- Remove add-on software entirely from your cluster (從叢集中完全移除附加元件軟體)：只有叢集上沒有資源依賴於附加元件提供的功能時，我們才會建議您將 Amazon EKS 附加元件從叢集中移除。

您可以使用 `eksctl`、 AWS Management Console或 CLI 移除 Amazon EKS AWS 附加元件。

移除附加元件 (eksctl)

1. 判斷叢集上目前安裝的附加元件。使用您叢集的名稱取代 *my-cluster*。

   ```
   eksctl get addon --cluster my-cluster
   ```

   範例輸出如下。

   ```
   NAME        VERSION              STATUS  ISSUES  IAMROLE  UPDATE AVAILABLE
   coredns     v1.8.7-eksbuild.2    ACTIVE  0
   kube-proxy  v1.23.7-eksbuild.1   ACTIVE  0
   vpc-cni     v1.10.4-eksbuild.1   ACTIVE  0
   [...]
   ```

   您的輸出可能看起來有所不同，具體取決於您的叢集上有哪些附加元件和版本。

2. 移除附加元件。將 *my-cluster* 取代為您的叢集名稱，並將*name-of-add-on*取代為您要移除之上一個步驟輸出中傳回的附加元件名稱。如果您移除 *--preserve* 選項，除了 Amazon EKS 不再管理附加元件之外，附加元件軟體也會從您的叢集中刪除。

   ```
   eksctl delete addon --cluster my-cluster --name name-of-addon --preserve
   ```

   如需移除附加元件的詳細資訊，請參閱 `eksctl` 文件中的刪除附加元件。

移除附加元件AWS （主控台）

1. 開啟 Amazon EKS 主控台。
2. 在左側導覽窗格中選擇叢集。
3. 選擇您要移除 Amazon EKS 附加元件的叢集名稱。
4. 選擇附加元件索引標籤。
5. 選擇您要移除的附加元件。
6. 選擇移除。
7. 在移除：########對話方塊中，執行下列動作：
   a. 若希望 Amazon EKS 停止管理附加元件的設定，請選取在叢集上保留。若要在叢集上保留附加元件軟體，請執行此動作。如此一來，您就可以自行管理附加元件的所有設定。
   b. 輸入附加元件名稱。
   c. 選擇移除。

移除附加元件 (AWS CLI)

1. 您需要在裝置0.210.0或 AWS CloudShell 上安裝版本 或更新版本的eksctl命令列工具。如需有關安裝或更新 eksctl 的指示，請參閱 eksctl 文件中的安裝一節。
2. 請參閱已安裝附加元件的清單。使用您叢集的名稱取代 *my-cluster*。

```
aws eks list-addons --cluster-name my-cluster
```

   範例輸出如下。

```
{
    "addons": [
        "coredns",
        "kube-proxy",
        "vpc-cni",
        "name-of-addon"
    ]
}
```

3. 移除已安裝的附加元件。將 *my-cluster* 取代為您的叢集名稱，並將*name-of-add-on*取代為您要移除的附加元件名稱。移除 *--preserve* 會從叢集中刪除附加元件軟體。

```
aws eks delete-addon --cluster-name my-cluster --addon-name name-of-addon --preserve
```

縮寫的範例輸出如下所示。

```
{
    "addon": {
        "addonName": "name-of-add-on",
        "clusterName": "my-cluster",
        "status": "DELETING",
    }
}
```

4. 檢查移除的狀態。將 *my-cluster* 取代為您的叢集名稱，並將*name-of-addon*取代為您移除的附加元件名稱。

```
aws eks describe-addon --cluster-name my-cluster --addon-name name-of-addon
```

移除附加元件後，範例輸出如下所示。

```
An error occurred (ResourceNotFoundException) when calling the DescribeAddon
 operation: No addon: name-of-addon found in cluster: my-cluster
```

Amazon EKS 附加元件的 IAM 角色

某些 Amazon EKS 附加元件需要 IAM 角色和許可才能呼叫 AWS APIs。例如，Amazon VPC CNI 附加元件會呼叫特定 AWS APIs來設定帳戶中的網路資源。這些附加元件需要使用 IAM 授予許可。更具體地說，執行附加元件的 Pod 服務帳戶必須與具有特定 IAM 政策的 IAM 角色相關聯。

授予叢集工作負載 AWS 許可的建議方法是使用 Amazon EKS 功能 Pod 身分。您可以使用 Pod Identity Association 將附加元件的服務帳戶映射至 IAM 角色。如果 Pod 使用具有關聯的服務帳戶，則 Amazon EKS 會在 Pod 的容器中設定環境變數。環境變數設定 AWS SDKs，包括 AWS CLI，以使用 EKS Pod Identity 憑證。如需詳細資訊，請參閱 the section called “Pod 身分”

Amazon EKS 附加元件可協助管理對應至附加元件之 Pod 身分關聯的生命週期。例如，您可以在單一 API 呼叫中建立或更新 Amazon EKS 附加元件和必要的 Pod 身分關聯。Amazon EKS 也提供 API 來擷取建議的 IAM 政策。

1. 確認您的叢集上已設定 Amazon EKS Pod 身分代理程式。

2. 判斷您要安裝的附加元件是否需要使用 AWS CLI 操作的 IAM `describe-addon-versions` 許可。如果`requiresIamPermissions`旗標為 `true`，則您應該使用 `describe-addon-configurations`操作來判斷附加元件所需的許可。回應包含建議的受管 IAM 政策清單。
3. 使用 CLI 操作擷取 Kubernetes 服務帳戶的名稱和 IAM `describe-addon-configuration` 政策。根據您的安全要求評估建議政策的範圍。
4. 使用建議的許可政策和 Pod Identity 所需的信任政策來建立 IAM 角色。如需詳細資訊，請參閱the section called “建立 Pod Identity 關聯AWS （主控台）”。
5. 使用 CLI 建立或更新 Amazon EKS 附加元件。指定至少一個 Pod 身分關聯。Pod 身分關聯是 Kubernetes 服務帳戶的名稱，以及 IAM 角色的 ARN。
   - 使用附加元件 APIs 建立的 Pod 身分關聯由個別附加元件擁有。如果您刪除附加元件，也會刪除 Pod 身分關聯。使用 CLI 或 API 刪除附加元件時，您可以使用 AWS `preserve`選項來防止此串聯刪除。您也可以視需要直接更新或刪除 Pod 身分關聯。附加元件無法擔任現有 Pod 身分關聯的擁有權。您必須刪除現有的關聯，並使用附加元件建立或更新操作重新建立關聯。
   - Amazon EKS 建議使用 Pod 身分關聯來管理附加元件的 IAM 許可。服務帳戶 (IRSA) 的 IAM 角色仍然支援先前的方法。您可以為附加元件指定 IRSA `serviceAccountRoleArn`和 Pod 身分關聯。如果叢集上安裝 EKS Pod 身分代理程式，`serviceAccountRoleArn`則會忽略 ，EKS 將使用提供的 Pod 身分關聯。如果未啟用 Pod Identity，`serviceAccountRoleArn`則會使用 。
   - 如果您更新現有附加元件的 Pod 身分關聯，Amazon EKS 會啟動附加元件 Pod 的滾動重新啟動。

擷取 Amazon EKS 附加元件的 IAM 資訊

建立附加元件之前，請使用 AWS CLI 判斷：

- 如果附加元件需要 IAM 許可
- 要使用的建議 IAM 政策

程序

1. 決定您要安裝的附加元件名稱，以及叢集的 Kubernetes 版本。如需附加元件的詳細資訊，請參閱the section called “Amazon EKS 附加元件”。
2. 使用 AWS CLI 判斷附加元件是否需要 IAM 許可。

```
aws eks describe-addon-versions \
--addon-name <addon-name> \
```

```
--kubernetes-version <kubernetes-version>
```

例如：

```
aws eks describe-addon-versions \
--addon-name aws-ebs-csi-driver \
--kubernetes-version 1.30
```

檢閱下列範例輸出。請注意， `requiresIamPermissions`為 `true`，且預設附加元件版本為 。擷取建議的 IAM 政策時，您需要指定附加元件版本。

```
{
    "addons": [
        {
            "addonName": "aws-ebs-csi-driver",
            "type": "storage",
            "addonVersions": [
                {
                    "addonVersion": "v1.31.0-eksbuild.1",
                    "architecture": [
                        "amd64",
                        "arm64"
                    ],
                    "compatibilities": [
                        {
                            "clusterVersion": "1.30",
                            "platformVersions": [
                                "*"
                            ],
                            "defaultVersion": true
                        }
                    ],
                    "requiresConfiguration": false,
                    "requiresIamPermissions": true
                },
[...]
```

3. 如果附加元件需要 IAM 許可，請使用 AWS CLI 擷取建議的 IAM 政策。

```
aws eks describe-addon-configuration \
--query podIdentityConfiguration \
--addon-name <addon-name> \
```

```
--addon-version <addon-version>
```

例如：

```
aws eks describe-addon-configuration \
--query podIdentityConfiguration \
--addon-name aws-ebs-csi-driver \
--addon-version v1.31.0-eksbuild.1
```

檢閱下列輸出。請記下 `recommendedManagedPolicies`。

```
[
    {
        "serviceAccount": "ebs-csi-controller-sa",
        "recommendedManagedPolicies": [
            "arn:aws: iam::aws:policy/service-role/AmazonEBSCSIDriverPolicy"
        ]
    }
]
```

4. 建立 IAM 角色並連接建議的 受管政策。或者，檢閱受管政策，並視需要縮小許可範圍。如需更多資訊，請參閱the section called “建立 Pod Identity 關聯AWS （主控台）”。

Pod Identity Support 參考

下表指出特定 Amazon EKS 附加元件是否支援 EKS Pod Identity。

附加元件名稱	Pod 身分支援	所需的最低版本
Amazon EBS CSI 驅動程式	是	v1.26.0-eksbuild.1
Amazon VPC CNI	是	v1.15.5-eksbuild.1
Amazon EFS CSI 驅動程式	是	v2.0.5-eksbuild.1
AWS Distro for OpenTelemetry	是	v0.94.1-eksbuild.1
Amazon S3 CSI 驅動程式掛載點	否	N/A

附加元件名稱	Pod 身分支援	所需的最低版本
Amazon CloudWatch 可觀測性代理程式	是	v3.1.0-eksbuild.1

此資料表上次更新日期為 2024 年 10 月 28 日。

使用 Pod 身分將 IAM 角色指派給 Amazon EKS 附加元件

某些 Amazon EKS 附加元件需要 IAM 角色和許可。新增 Amazon EKS 附加元件以使用 Pod Identity 關聯之前，請先驗證要使用的角色和政策。如需詳細資訊，請參閱the section called “擷取 IAM 資訊”。

1. 判斷：
   - `cluster-name` – 要安裝附加元件的叢集名稱。
   - `addon-name` – 要安裝的附加元件名稱。
   - `service-account-name` – 附加元件使用的 Kubernetes 服務帳戶名稱。
   - `iam-role-arn` – IAM 角色的 ARN，具有足夠附加元件的許可。角色必須具有 EKS Pod Identity 所需的信任政策。如需詳細資訊，請參閱 the section called “建立 Pod Identity 關聯AWS （主控台）”。
2. 使用 CLI AWS 更新附加元件。您也可以在建立附加元件時使用相同的`--pod-identity-assocations`語法來指定 Pod Identity 關聯。請注意，當您在更新附加元件時指定 Pod 身分關聯時，所有先前的 Pod 身分關聯都會遭到覆寫。

   ```
   aws eks update-addon --cluster-name <cluster-name> \
   --addon-name <addon-name> \
   --pod-identity-associations 'serviceAccount=<service-account-name>,roleArn=<role-arn>'
   ```

   例如：

   ```
   aws eks update-addon --cluster-name mycluster \
   --addon-name aws-ebs-csi-driver \
   --pod-identity-associations 'serviceAccount=ebs-csi-controller-sa,roleArn=arn:aws:iam::123456789012:role/StorageDriver'
   ```

3. 驗證是否已建立 Pod Identity 關聯：

```
aws eks list-pod-identity-associations --cluster-name <cluster-name>
```

如果執行成功，您應該會看到類似下列的輸出。請注意 EKS 附加元件的 OwnerARN。

```
{
    "associations": [
        {
            "clusterName": "mycluster",
            "namespace": "kube-system",
            "serviceAccount": "ebs-csi-controller-sa",
            "associationArn": "arn:aws: eks:us-
west-2:123456789012:podidentityassociation/mycluster/a-4wvljrezsukshq1bv",
            "associationId": "a-4wvljrezsukshq1bv",
            "ownerArn": "arn:aws: eks:us-west-2:123456789012:addon/mycluster/aws-ebs-
csi-driver/9cc7ce8c-2e15-b0a7-f311-426691cd8546"
        }
    ]
}
```

從 Amazon EKS 附加元件移除 Pod Identity 關聯

從 Amazon EKS 附加元件移除 Pod Identity 關聯。

1. 判斷：
   - `cluster-name` - 安裝附加元件的 EKS 叢集名稱。
   - `addon-name` - 要安裝的 Amazon EKS 附加元件名稱。
2. 更新附加元件以指定空的 Pod 身分關聯陣列。

```
aws eks update-addon --cluster-name <cluster-name> \
--addon-name <addon-name> \
--pod-identity-associations "[]"
```

針對 EKS 附加元件的 Pod 身分進行故障診斷

如果您的附加元件在嘗試 AWS API、 SDK 或 CLI 操作時遇到錯誤，請確認下列事項：

- Pod Identity Agent 已安裝在您的叢集中。

  - 如需如何安裝 Pod Identity Agent 的資訊，請參閱 the section called “設定 代理程式”。
- 附加元件具有有效的 Pod Identity 關聯。
  - 使用 AWS CLI 擷取附加元件使用之服務帳戶名稱的關聯。

```
aws eks list-pod-identity-associations --cluster-name <cluster-name>
```

- IAM 角色具有 Pod 身分所需的信任政策。
  - 使用 AWS CLI 擷取附加元件的信任政策。

```
aws iam get-role --role-name <role-name> --query Role.AssumeRolePolicyDocument
```

- IAM 角色具有附加元件的必要許可。
  - 使用 AWS CloudTrail 來檢閱 `AccessDenied`或 `UnauthorizedOperation`事件。
- Pod 身分關聯中的服務帳戶名稱符合附加元件所使用的服務帳戶名稱。
  - 如需可用附加元件的資訊，請參閱 the section called “ AWS 附加元件”。
- 檢查名為 的 MutatingWebhookConfiguration 組態 `pod-identity-webhook`
  - `admissionReviewVersions` Webhook 的 必須是 `v1beta1`，且不適用於 `v1`。

決定您可以為 Amazon EKS 附加元件自訂的欄位

使用標準的最佳實務組態將 Amazon EKS 附加元件安裝到您的叢集。如需將 Amazon EKS 附加元件新增至叢集的詳細資訊，請參閱 the section called “Amazon EKS 附加元件”。

您可能想要自訂 Amazon EKS 附加元件的組態，以啟用進階功能。Amazon EKS 使用 Kubernetes 伺服器端套用功能來啟用由 Amazon EKS 管理附加元件，而不會覆寫非由 Amazon EKS 管理之設定的組態。如需詳細資訊，請參閱 Kubernetes 文件中的伺服器端套用。為了達到此目的，Amazon EKS 會為其安裝的每個附加元件管理一組數量最少的欄位。您可以修改所有非由 Amazon EKS 管理的欄位，或另一個 Kubernetes 控制平面程序，例如 `kube-controller-manager`，不會發生問題。

> **Important**
>
> 修改 Amazon EKS 管理的欄位會阻止 Amazon EKS 管理附加元件，並且可能會導致在更新附加元件時覆寫您的變更。

欄位管理語法

檢視 Kubernetes 物件的詳細資訊時，輸出中會傳回受管和未受管的欄位。受管欄位可以是以下類型之一：

- 全受管：欄位的所有索引鍵均由 Amazon EKS 管理。修改任何值皆會導致衝突。
- 部分受管：欄位的某些索引鍵由 Amazon EKS 管理。只有對明確由 Amazon EKS 管理的索引鍵進行修改才會造成衝突。

這兩種類型的欄位均使用 `manager: eks` 標記。

每個索引鍵可以是 `.` 表示欄位本身，其一律會映射到一個空集，或映射至表示子欄位或項目的字串。欄位管理的輸出包含下列類型的宣告：

- `f:`*`name`*，其中 *`name`* 是清單中欄位的名稱。
- `k:`*`keys`*，其中*###*是清單項目欄位的映射。
- `v:`*`value`*，其中 *#*是清單項目的確切 JSON 格式值。
- `i:`*`index`*，其中*##*是清單中項目的位置。

CoreDNS 附加元件的下列輸出部分會說明先前的宣告：

- 全受管欄位：如果受管欄位具有指定的 `f:` (欄位)，但沒有 `k:` (索引鍵)，則會管理整個欄位。修改此欄位中的任何值皆會導致衝突。

  在下列輸出中，您可以看到名為 `coredns` 的容器由 `eks` 管理。`args`、`image` 及 `imagePullPolicy` 子欄位也由 `eks` 管理。修改這些欄位中的任何值皆會導致衝突。

```
[...]
f:containers:
  k:{"name":"coredns"}:
  .: {}
  f:args: {}
  f:image: {}
  f:imagePullPolicy: {}
[...]
manager: eks
[...]
```

- 部分受管欄位：如果受管索引鍵具有指定的值，則會針對該欄位管理宣告的索引鍵。修改指定的索引鍵會導致衝突。

  在下列輸出中，您可以看到 eks 管理 config-volume 和使用 name 索引鍵設定的 tmp 磁碟區。

  ```
  [...]
  f:volumes:
    k:{"name":"config-volume"}:
      .: {}
      f:configMap:
        f:items: {}
        f:name: {}
      f:name: {}
    k:{"name":"tmp"}:
      .: {}
      f:name: {}
  [...]
  manager: eks
  [...]
  ```

- 將索引鍵新增至部分受管欄位：如果只管理特定鍵值，您可以安全地將其他索引鍵 (例如引數) 新增至欄位，而不會造成衝突。如果您新增其他金鑰，請確定欄位未先受管。新增或修改受管的任何值會導致衝突。

  在下列輸出中，您可以看到 name 索引鍵和 name 欄位皆為受管項目。新增或修改任何容器名稱會導致與此受管索引鍵發生衝突。

  ```
  [...]
  f:containers:
    k:{"name":"coredns"}:
  [...]
      f:name: {}
  [...]
  manager: eks
  [...]
  ```

程序

您可以使用 kubectl，查看哪些欄位由 Amazon EKS 管理，適用於任何 Amazon EKS 附加元件。

您可以修改所有非由 Amazon EKS 管理的欄位，或另一個 Kubernetes 控制平面程序，例如 kube-controller-manager，不會發生問題。

1. 判定要檢查的附加元件。若要查看部署到叢集的所有 deployments和 DaemonSets，請參閱 the section called “存取叢集資源”。
2. 執行下列命令以檢視附加元件的受管欄位：

```
kubectl get type/add-on-name -n add-on-namespace -o yaml
```

例如，您可以使用下列命令查看 CoreDNS 附加元件的受管欄位。

```
kubectl get deployment/coredns -n kube-system -o yaml
```

欄位管理會列在傳回輸出的下一區段中。

```
[...]
managedFields:
  - apiVersion: apps/v1
    fieldsType: FieldsV1
    fieldsV1:
[...]
```

> **Note**
>
> 如果您在輸出managedFields中沒有看到 ，請將 --show-managed-fields新增至命令，然後再次執行。您使用kubectl的 版本會決定預設是否傳回受管欄位。

後續步驟

為您自訂 未擁有的欄位 AWS 。

在部署期間驗證容器映像簽章

如果您使用 AWS Signer 並想要在部署時驗證已簽章的容器映像，您可以使用下列其中一個解決方案：

- Gatekeeper 和 Ratify – 使用 Gatekeeper 做為許可控制器，並將 AWS Signer 外掛程式設定為 Web 掛鉤以驗證簽章。

- Kyverno – 使用 AWS Signer 外掛程式設定的 Kubernetes 政策引擎，用於驗證簽章。

> **Note**
>
> 在驗證容器映像簽章之前，請依照您選取的許可控制器的要求，設定 Notation 信任存放區和信任政策。

組織和監控叢集資源

本章節包括可協助您管理叢集的以下主題。您也可以使用 檢視 Kubernetes 資源的相關資訊 AWS Management Console。

- Kubernetes Dashboard 是 Kubernetes 叢集的一般用途、以 Web 為基礎的 UI。它可讓使用者管理叢集中執行的應用程式並對其進行疑難排解，以及管理叢集本身。如需詳細資訊，請參閱 Kubernetes Dashboard GitHub 儲存庫。
- the section called “指標伺服器” – Kubernetes 指標伺服器是叢集中資源用量資料的彙總工具。預設不會部署在您的叢集中，但會由 Kubernetes 附加元件使用，例如 Kubernetes Dashboard 和 the section called “Horizontal Pod Autoscaler”。在本主題中，您將了解如何安裝指標伺服器。
- the section called “使用 Helm 部署應用程式” – Kubernetes 的 Helm 套件管理工具可協助您安裝和管理 Kubernetes 叢集上的應用程式。此主題協助您安裝和執行 Helm 二進位檔，讓您可以在本機電腦上使用 Helm CLI 安裝和管理圖表。
- the section called “標記您的 資源” – 為協助您管理 Amazon EKS 資源，您可以用標籤形式將您自己的中繼資料指派給每個資源。本主題說明標籤並示範如何建立它們。
- the section called “Service Quotas” – AWS 您的帳戶具有每個 AWS 服務的預設配額，先前稱為限制。了解 Amazon EKS 的配額，以及如何提高配額。

監控和最佳化 Amazon EKS 叢集成本

成本監控是在 Amazon EKS 上管理 Kubernetes 叢集的重要層面。透過掌握叢集成本，您可以最佳化資源使用率、設定預算，並針對部署做出資料驅動型決策。Amazon EKS 提供兩種成本監控解決方案，每個解決方案都有自己的獨特優勢，可協助您有效追蹤和分配成本：

AWS Amazon EKS 的帳單分割成本分配資料 — 此原生功能與 AWS 帳單主控台無縫整合，可讓您使用與其他 AWS 服務相同的熟悉界面和工作流程來分析和分配成本。透過分割成本分配，您可以直接與其他 AWS 支出一起深入了解 Kubernetes 成本，讓您更輕鬆地在整個 AWS 環境中全面最佳化成本。您也可以利用現有的 AWS 帳單功能，例如Cost Categories和成本異常偵測，進一步增強您的成本管理功能。如需詳細資訊，請參閱《 AWS 帳單使用者指南》中的了解分割成本分配資料。

Kubecost – Amazon EKS 支援 Kubecost，這是一種 Kubernetes 成本監控工具。Kubecost 提供功能豐富的 Kubernetes 原生成本監控方法，依 Kubernetes 資源、成本最佳化建議以及out-of-the-box儀表板和報告提供精細的成本明細。Kubecost 還透過與 AWS 成本和用量報告整合來擷取準確的定價資料，確保您準確檢視 Amazon EKS 成本。了解如何安裝 Kubecost。如需取得免費 Kubecost 訂閱的相關資訊，請參閱 Kubecost AWS Marketplace 頁面。

使用分割成本分配依 Pod 在 AWS 帳單中檢視成本

使用 Amazon EKS 的 AWS 分割成本分配資料進行成本監控

您可以使用 Amazon EKS 的 AWS 分割成本分配資料，取得 Amazon EKS 叢集的精細成本可見性。這可讓您分析、最佳化和計費 Kubernetes 應用程式的成本和用量。您可以根據 Kubernetes 應用程式使用的 Amazon EC2 CPU 和記憶體資源，將應用程式成本分配給個別業務單位和團隊。Amazon EKS 的分割成本分配資料可讓您查看每個 Pod 的成本，並可讓您使用命名空間、叢集和其他 Kubernetes 基本概念來彙總每個 Pod 的成本資料。以下是可用於分析 Amazon EKS 成本分配資料的 Kubernetes 基本概念範例。

- 叢集名稱
- 部署
- 命名空間
- 節點
- 工作負載名稱
- 工作負載類型

也支援使用者定義的成本分配標籤。如需使用分割成本分配資料的詳細資訊，請參閱《 AWS 帳單使用者指南》中的了解分割成本分配資料。

設定成本和用量報告

您可以在成本管理主控台、 AWS 命令列界面或 AWS SDKs 中開啟 EKS 的分割成本分配資料。

將下列項目用於分割成本分配資料：

1. 選擇加入分割成本分配資料。如需詳細資訊，請參閱《 成本和用量報告使用者指南》中的啟用分割成本分配資料。 AWS
2. 在新的或現有的報告中包含資料。
3. 檢視報告。您可以使用帳單和成本管理主控台，或在 Amazon Simple Storage Service 中檢視報告檔案。

安裝 Kubecost

Amazon EKS 支援 Kubecost，您可以使用 Kubecost 來監控依 Kubernetes 資源細分的成本，包括 Pod、節點、命名空間和標籤。本主題涵蓋安裝 Kubecost，以及存取 Kubecost 儀表板。

Amazon EKS 提供 AWS 最佳化的 Kubecost 套件，以實現叢集成本可見性。您可以使用現有的 AWS 支援協議來取得支援。如需 Kubecost 可用版本的詳細資訊，請參閱 the section called “進一步了解 Kubecost”。

> **Note**
>
> Kubecost v2 推出數個主要的新功能。進一步了解 Kubecost v2。

如需 Kubecost 的詳細資訊，請參閱 Kubecost 文件和常見問答集。

安裝 Amazon EKS 最佳化 Kubecost 套件

您可以使用下列其中一個程序來安裝 Amazon EKS 最佳化 Kubecost 套件：

- 開始之前，建議您檢閱 Kubecost - 架構概觀，以了解 Kubecost 如何在 Amazon EKS 上運作。
- 如果您是初次使用 Amazon EKS，我們建議您使用 Amazon EKS 附加元件進行安裝，因為它可簡化 Amazon EKS 最佳化 Kubecost 套件安裝。如需詳細資訊，請參閱使用 Amazon EKS 附加元件在 Amazon EKS 叢集上部署 Kubecost。
- 若要自訂安裝，您可以使用 Helm 設定 Amazon EKS 最佳化 Kubecost 套件。如需詳細資訊，請參閱 Kubecost 文件中的使用 Helm 在 Amazon EKS 叢集上部署 Kubecost。

存取 Kubecost 儀表板

Amazon EKS 最佳化 Kubecost 套件設定完成後，您應該可以存取 Kubecost 儀表板。如需詳細資訊，請參閱the section called “存取 Kubecost 儀表板”。

存取 Kubecost 儀表板

先決條件

1. 確定 kubecost 相關的 Pod 狀態為「執行中」。

```
kubectl get pods --namespace kubecost
```

存取 Kubecost 儀表板

1. 在您的裝置上，啟用連接埠轉送以公開 Kubecost 儀表板。

```
kubectl port-forward deployment/kubecost-cost-analyzer 9090 --namespace kubecost
```

或者，您可以使用AWS Load Balancer控制器公開 Kubecost，並使用 Amazon Cognito 進行身分驗證、授權和使用者管理。如需詳細資訊，請參閱如何使用 Application Load Balancer 和 Amazon Cognito 來驗證 Kubernetes Web 應用程式的使用者。

2. 在您完成上一個步驟的同一部裝置上，開啟 Web 瀏覽器並輸入下列位址。

```
http://localhost:9090
```

您可以在瀏覽器中看到 Kubecost 概觀頁面。Kubecost 可能需要 5-10 分鐘（或更多）才能收集指標，取決於您的叢集大小。您可以查看您的 Amazon EKS 支出，包括累積叢集成本、相關聯的 Kubernetes 資產成本，以及每月彙總支出。

3. 要在叢集層級追蹤成本，請標記您的 Amazon EKS 資源以進行計費。如需詳細資訊，請參閱the section called “標記您的資源以便計費”。
   - Cost allocation (成本分配) – 檢視過去七天內每個命名空間和其他維度的每月 Amazon EKS 成本和累計成本。這有助於了解應用程式的哪些部分產生了 Amazon EKS 支出。
   - 資產 – 檢視與您的 Amazon EKS 資源相關聯的 AWS 基礎設施資產成本。

進一步了解 Kubecost

Amazon EKS 提供 AWS 最佳化的 Kubecost 套件，以實現叢集成本可見性。Amazon EKS 支援 Kubecost，您可以使用 Kubecost 來監控依 Kubernetes 資源細分的成本，包括 Pod、節點、命名空間和標籤。

本主題涵蓋 Kubecost 的可用版本，以及可用層之間的差異。EKS 支援 Kubecost 第 1 版和第 2 版。每個版本都在不同層中提供。您可以為 Amazon EKS 叢集使用 Amazon EKS 最佳化 Kubecost 套件，無需額外費用。您可能需要支付使用相關 AWS 服務的費用，例如 Amazon Managed Service for Prometheus。此外，您可以使用現有的 AWS 支援協議來取得支援。

身為 Kubernetes 平台管理員和財務主管，您可以使用 Kubecost 視覺化 Amazon EKS 費用明細、分配成本，以及向應用程式團隊等組織單位收費。您可以根據內部團隊和業務單位的實際 AWS 帳單，提供透明且準確的成本資料。此外，您還可以根據他們的基礎設施環境和叢集內的使用模式，獲得客製化的成本最佳化建議。如需 Kubecost 的詳細資訊，請參閱 Kubecost 文件。

Kubecost 的自訂套件與 Kubecost 的免費版本（也稱為 OpenCost) 有何不同？

AWS 和 Kubecost 合作提供自訂版本的 Kubecost。此版本提供商業功能的子集，不收取額外費用。如需自訂 Kubecost 套件中包含的 功能，請參閱下表。

Kubecost v2

Kubecost v1 和 v2 之間的差異是什麼？

Kubecost 2.0 是先前版本的主要升級，並包含主要的新功能，包括全新的 API 後端。請注意，配置和資產 API 完全回溯相容。 APIs 請檢閱 Kubecost 文件，以確保順利轉換。如需增強功能的完整清單，請參閱 Kubecost v2.0 公告和完整版本備註。

> **Important**
>
> 升級之前，請檢閱 Kubecost 文件。升級可能會影響報告的可用性。

核心功能比較：

功能	Kubecost 免費方案 2.0	Amazon EKS 最佳化 Kubecost 套件 2.0	Kubecost Enterprise 2.0
叢集成本能見度	無限制叢集，最多 250 個核心	與 Amazon Managed Service for Prometheus 整合時，沒有核心限制的統一多叢集	跨無限數量環境（即多雲端）的統一和無限數量的叢集
部署	使用者託管	使用者託管	使用者託管、Kubecost 託管（專用租用戶）、SaaS
支援的資料庫	本機 Prometheus	Amazon Managed Service for Prometheus 或本機 Prometheus	任何 prometheus 版本和自訂資料庫
資料庫保留支援（原始指標）	15 天	無限的歷史記錄資料	無限的歷史記錄資料
Kubecost API 和 UI 保留 (ETL)	15 天	15 天	無限制

功能	Kubecost 免費方案 2.0	Amazon EKS 最佳化 Kubecost 套件 2.0	Kubecost Enterprise 2.0
混合雲端能見度	-	Amazon EKS 和 Amazon EKS Anywhere 叢集	多雲端和混合雲端
警示和週期性報告	僅主要叢集支援，僅限 250 個核心	在所有叢集中支援效率提醒、預算提醒、支出變更提醒等功能	在所有叢集中支援效率提醒、預算提醒、支出變更提醒等功能
已儲存的報告	-	使用 15 天指標的報告	使用無限制歷史資料和指標的報告
雲端計費整合	僅主要叢集支援，僅限 250 個核心	的自訂定價支援 AWS（包括多個叢集和多個帳戶）	任何雲端的自訂定價支援
Savings 建議	僅主要叢集支援，僅限 250 個核心	主要叢集洞察，但沒有 250 個核心限制	多重叢集深入解析
治理：稽核	-	-	稽核歷史記錄成本事件
單一登入 (SSO) 支援	-	支援 Amazon Cognito	Okta、Auth 0、PingID、KeyCloak 和其他自訂項目
使用 SAML 2.0 的角色型存取控制 (RBAC)	-	-	Okta、Auth 0、PingID、KeyCloak 和其他自訂項目
企業培訓和入門	-	-	完整服務訓練和 FinOps 加入
團隊	-	-	是

新功能：

下列功能具有指標限制：

- Kubecost 彙總工具
- 網路監控
- Kubecost 動作
- 集合
- 異常偵測
- 容器請求適當調整大小
- Kubecost 預測
- 自動完成以篩選和彙總

指標限制：

指標	Kubecost 免費方案 2.0	Amazon EKS 最佳化 Kubecost 套件 2.0	Kubecost Enterprise 2.0
叢集大小	無限制叢集，最多 250 個核心	無限制	無限制
指標保留	15 天	15 天	無限制
多叢集支援	無	可用性	可用性
核心限制	每個叢集 250 個核心	無核心限制	無核心限制

Kubecost v1

功能	Kubecost 免費方案	Amazon EKS 最佳化 Kubecost 套件	Kubecost Enterprise
部署	使用者託管	使用者託管	使用者託管或 Kubecost 託管 (SaaS)
支援的叢集數目	無限制	無限制	無限制

功能	Kubecost 免費方案	Amazon EKS 最佳化 Kubecost 套件	Kubecost Enterprise
支援的資料庫	本機 Prometheus	Local Prometheus 或 Amazon Managed Service for Prometheus	Prometheus、Amazon Managed Service for Prometheus、Cortex 或 Thanos
資料庫保留支援	15 天	無限的歷史記錄資料	無限的歷史記錄資料
Kubecost API 保留 (ETL)	15 天	15 天	無限的歷史記錄資料
叢集成本能見度	單一叢集	統一的多重叢集	統一的多重叢集
混合雲端能見度	-	Amazon EKS 和 Amazon EKS Anywhere 叢集	多重雲端和混合雲端支援
警示和週期性報告	-	效率警示、預算警示、支出變更警示，以及更多支援的功能	效率警示、預算警示、支出變更警示，以及更多支援的功能
已儲存的報告	-	使用 15 天資料的報告	使用無限期歷史記錄資料的報告
雲端計費整合	每一個叢集都需要	的自訂定價支援 AWS（包括多個叢集和多個帳戶）	的自訂定價支援 AWS（包括多個叢集和多個帳戶）
Savings 建議	單一叢集深入解析	單一叢集深入解析	多重叢集深入解析
治理：稽核	-	-	稽核歷史記錄成本事件
單一登入 (SSO) 支援	-	支援 Amazon Cognito	Okta、Auth0、PingID、KeyCloak

功能	Kubecost 免費方案	Amazon EKS 最佳化 Kubecost 套件	Kubecost Enterprise
使用 SAML 的角色型存取控制 (RBAC) **2.0**	-	-	Okta、Auth 0、PingID、Keycloak
企業培訓和入門	-	-	完整服務訓練和 FinOps 加入

常見問答集

請參閱下列有關將 Kubecost 與 Amazon EKS 搭配使用的常見問題和解答。

什麼是 Kubecost API 保留 (ETL) 功能？

Kubecost ETL 功能會彙總和組織指標，以在各種精細程度 （例如 `namespace-level`、 `pod-level`和) 中呈現成本可見性`deployment-level`。對於 Amazon EKS 最佳化 Kubecost 套件，客戶會從過去 15 天的指標中取得資料和洞見。

什麼是提醒和週期性報告功能？ 它包含哪些提醒和報告？

Kubecost 提醒可讓團隊接收即時 Kubernetes 支出和雲端支出的更新。週期性報告可讓團隊接收歷史 Kubernetes 和雲端支出的自訂檢視。兩者皆可使用 Kubecost UI 或 Helm 值進行設定。他們支援電子郵件、Slack 和 Microsoft Teams。

儲存的報告包含哪些項目？

Kubecost 儲存的報告是成本和效率指標的預先定義檢視。它們包含依叢集、命名空間、標籤以及更多類別來區分的成本。

什麼是雲端帳單整合？

與 AWS 帳單 APIs整合可讓 Kubecost out-of-cluster (例如 Amazon S3)。此外，它允許 Kubecost 將 Kubecost 的叢集內預測與實際帳單資料進行協調，以考慮 Spot 用量、節省計劃和企業折扣。

節省成本的建議包括什麼？

Kubecost 提供洞見和自動化，協助使用者最佳化其 Kubernetes 基礎設施和支出。

此功能是否需要付費？

否。您可以免費使用 Amazon EKS 最佳化 Kubecost 套件。如果您想要不包含其他 Kubecost 功能，您可以透過 AWS Marketplace 或直接從 Kubecost 購買 Kubecost 的企業授權。

Amazon EKS 最佳化 Kubecost 套件是否提供支援？

是，只有在您使用 Amazon EKS 最佳化 Kubecost 套件時。

如何取得 Amazon EKS 最佳化 Kubecost 套件的支援？

您可以在聯絡 AWS支援團隊開啟 AWS 支援案例。

我是否需要授權才能使用 Amazon EKS 整合提供的 Kubecost 功能？

否。

我可以將 Kubecost 與 AWS 成本和用量報告整合，以獲得更準確的報告嗎？

是。您可以設定 Kubecost 從 AWS 成本和用量報告中擷取資料，以取得準確的成本可見性，包括折扣、Spot 定價、預留執行個體定價等。如需詳細資訊，請參閱 Kubecost 文件中的AWS 雲端帳單整合。

此版本是否支援 Amazon EC2 上自我管理 Kubernetes 叢集的成本管理？

否。Amazon EKS 最佳化 Kubecost 套件僅與 Amazon EKS 叢集相容。

Kubecost 是否可以追蹤 AWS Fargate 上 Amazon EKS 的成本？

Kubecost 會盡最大努力顯示 Amazon EKS on Fargate 的叢集成本可見性，但準確度低於 Amazon EC2 上的 Amazon EKS。這主要是因為您的用量計費方式不同。使用 Fargate 上的 Amazon EKS，您需要支付耗用資源的費用。使用 Amazon EC2 節點上的 Amazon EKS，您需要支付佈建資源的費用。Kubecost 會根據節點規格計算 Amazon EC2 節點的成本，其中包括 CPU、RAM 和暫時性儲存。使用 Fargate，成本是根據 Fargate Pod 的請求資源計算。

如何取得 Kubecost 的更新和新版本？

您可以使用標準 Helm 升級程序來升級 Kubecost 版本。最新版本位於 Amazon ECR Public Gallery 中。

是否支援 **`kubectl-cost`** CLI？ 如何安裝？

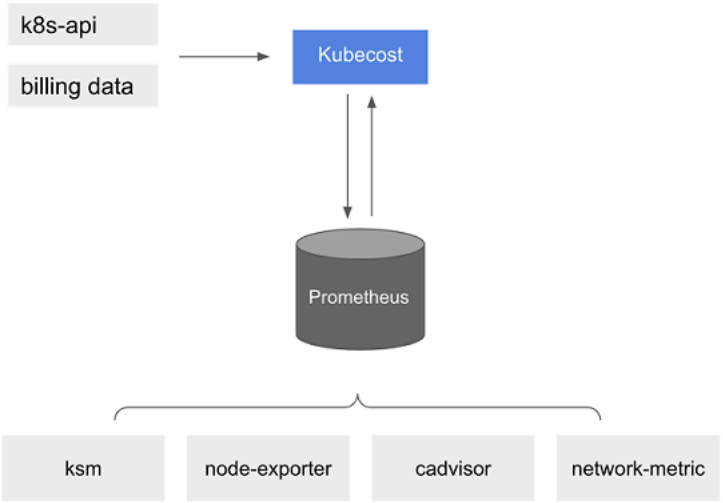
是。 `Kubectl-cost` 是 Kubecost (Apache 2.0 License) 的開放原始碼工具，可提供 Kubernetes 成本分配指標的 CLI 存取權。若要安裝 `kubectl-cost`，請參閱 GitHub 上的 Installation (安裝)。

是否支援 Kubecost 使用者介面？ 如何存取它？

Kubecost 提供 Web 儀表板，您可以透過`kubectl`連接埠轉送、輸入或負載平衡器存取。您也可以使用 AWS Load Balancer控制器公開 Kubecost，並使用 Amazon Cognito 進行身分驗證、授權和使用者管理。如需詳細資訊，請參閱 AWS 部落格上的如何使用 Application Load Balancer 和 Amazon Cognito 來驗證 Kubernetes Web 應用程式的使用者。

其他 Kubecost 功能

- 下列功能可在 Kubecost v1 和 v2 中使用。
  - 匯出成本指標 – Amazon EKS 最佳化成本監控是使用 Kubecost 和 Prometheus 部署，這是開放原始碼監控系統和時間序列資料庫。Kubecost 從 Prometheus 讀取指標，然後執行成本分配計算，並將指標寫回 Prometheus。Kubecost 前端會從 Prometheus 讀取指標，並在 Kubecost 使用者介面上顯示它們。下圖說明此架構。



    預先安裝 Prometheus 後，您可以撰寫查詢，將 Kubecost 資料擷取到您目前的商業智慧系統中，以進行進一步分析。您也可以將其用作目前 Grafana 儀表板的資料來源，以顯示內部團隊熟悉的 Amazon EKS 叢集成本。若要進一步了解如何撰寫 Prometheus 查詢，請參閱 GitHub 上的 https://opencost.io/docs/installation/prometheus/ `readme` 檔案，或使用 Kubecost Github 儲存庫中的範例 Grafana JSON 模型做為參考。

  - AWS 成本和用量報告整合 – 為了執行 Amazon EKS 叢集的成本分配計算，Kubecost 會從 AWS 價格清單 API 擷取 AWS 服務 AWS 和資源的公有定價資訊。您也可以將 Kubecost 與 AWS 成本和用量報告整合，以提升 AWS 您帳戶特定定價資訊的準確性。這類資訊包括企業折扣方案、預留執行個體使用量、Savings Plans 和 Spot 使用量。若要進一步了解 AWS 成本和用量報告整合的運作方式，請參閱 Kubecost 文件中的AWS 雲端帳單整合。

使用 Kubernetes Metrics Server 檢視資源用量

Kubernetes Metrics Server 是叢集中資源用量資料的彙整工具，預設不會部署在 Amazon EKS 叢集中。如需詳細資訊，請參閱 GitHub 上的 Kubernetes 指標伺服器。Metrics Server 通常由其他 Kubernetes 附加元件使用，例如使用 Horizontal Pod Autoscaler 擴展 Pod 部署或 Kubernetes Dashboard。如需詳細資訊，請參閱 Kubernetes 文件中的資源指標管道。此主題說明如何在 Amazon EKS 叢集上部署 Kubernetes 指標伺服器。

> **Important**
>
> 這些指標適用於point-in-time分析，並非歷史分析的準確來源。它們不能用作監控解決方案或用於其他非自動擴展目的。如需監控工具的相關資訊，請參閱 監控叢集。

考量事項

- 如果使用資訊清單將 Kubernetes Metrics Server 手動部署到 Fargate 節點，請將`metrics-server`部署設定為使用預設 以外的連接埠`10250`。此連接埠保留給 Fargate。Metrics Server 的 Amazon EKS 附加元件版本已預先設定為使用連接埠 `10251`。
- 確保安全群組和網路 ACLs 允許 Pod `metrics-server` 與所有其他節點和 Pod `10250`之間的連接埠。Kubernetes Metrics Server 仍會使用連接埠從叢集中的其他端點`10250`收集指標。如果您在 Fargate 節點上部署 ，請同時允許設定的替代指標伺服器連接埠和連接埠 `10250`。

使用 Amazon EKS 附加元件部署為社群附加元件

新功能：您現在可以使用 AWS 主控台或 Amazon EKS APIs，將 Metrics Server 部署為社群附加元件。

使用 AWS 主控台部署

1. 在 AWS 主控台中開啟您的 EKS 叢集
2. 從「附加元件」索引標籤中，選取取得更多附加元件。
3. 從「社群附加元件」區段中，選取指標伺服器，然後選取下一步
4. EKS 會為您的叢集決定適當的附加元件版本。您可以使用版本下拉式選單來變更版本。
5. 選取下一步，然後選取建立以安裝附加元件。

其他資源

進一步了解 the section called “社群附加元件”。

您安裝或更新社群附加元件的方式與其他 Amazon EKS 附加元件相同。

- the section called “建立附加元件”
- the section called “更新 附加元件”
- the section called “移除附加元件”

使用資訊清單部署

新功能：您現在可以使用 AWS 主控台或 Amazon EKS APIs，將 Metrics Server 部署為社群附加元件。這些資訊清單安裝指示將封存。

1. 使用下列命令部署指標伺服器：

   ```
   kubectl apply -f https://github.com/kubernetes-sigs/metrics-server/releases/latest/download/components.yaml
   ```

   如果您使用的是 Fargate，則需要變更此檔案。在預設組態中，指標伺服器使用連接埠 10250。此連接埠保留在 Fargate 上。將 components.yaml 中連接埠 10250 的參考取代為另一個連接埠，例如 10251。

2. 使用以下命令，確認`metrics-server`部署正在執行所需的 Pod 數量。

   ```
   kubectl get deployment metrics-server -n kube-system
   ```

   範例輸出如下。

   ```
   NAME             READY   UP-TO-DATE   AVAILABLE   AGE
   metrics-server   1/1     1            1           6m
   ```

3. 透過顯示節點的資源 (CPU/記憶體） 用量來測試指標伺服器運作中。

   ```
   kubectl top nodes
   ```

4. 如果您收到錯誤訊息 `Error from server (Forbidden)`，則需要更新 Kubernetes RBAC 組態。您的 Kubernetes RBAC 身分需要足夠的許可才能讀取叢集指標。檢閱在 GitHub 上讀取指標所需的最低 Kubernetes API 許可。了解如何授予 AWS IAM 身分，例如角色存取 Kubernetes APIs

在 Amazon EKS 上使用 Helm 部署應用程式

Kubernetes 的 Helm 套件管理工具可協助您安裝和管理 Kubernetes 叢集上的應用程式。如需詳細資訊，請參閱 Helm 文件。此主題協助您安裝和執行 Helm 二進位檔，讓您可以在本機系統上使用 Helm CLI 安裝和管理圖表。

> **Important**
>
> 將 Helm 圖表安裝在 Amazon EKS 叢集上之前，務必設定 `kubectl` 以搭配 Amazon EKS 來運作。若您尚未完成此動作，請參閱the section called “使用 kubectl 存取叢集” 之後再繼續。如果叢集的下列命令成功，表示您已正確設定。
>
> ```
> kubectl get svc
> ```

1. 為您的用戶端作業系統執行適當的命令。
   - 如果您使用 macOS 搭配 Homebrew，請使用下列命令安裝二進位檔。

     ```
     brew install helm
     ```

   - 如需更多安裝選項，請參閱在 Helm 文件中安裝 Helm。

     > **Note**
     >
     > 如果您收到必須先安裝 `openssl` 的訊息，您可以使用下列命令進行安裝。

     ```
     sudo yum install openssl
     ```

1. 若要在 PATH 中取得新的二進位檔，請關閉目前的終端機視窗並開啟新的視窗。
2. 查看您安裝的 Helm 版本。

```
helm version --template='{{ .Version }}{{ "\n" }}'
```

範例輸出如下。

```
v3.17.2
```

3. 請確定安裝的版本與您的叢集版本相容。檢查支援的版本扭曲以進一步了解。例如，如果您使用 執行 `3.17.x`，支援的 Kubernetes 版本不應超出 `1.29.x` ~ 的範圍`1.32.x`。
4. 此時，您可以執行任何 Helm 命令 (例如 `helm install` *`chart-name`*)，在叢集中安裝、修改、刪除或查詢 Helm 圖表。如果您是 Helm 的新手，而且沒有要安裝的特定圖表，您可以：
   - 透過安裝範例圖表進行實驗。請參閱 Helm 快速入門指南中的安裝範例圖表。
   - 建立範例圖表並將其推送到 Amazon ECR。如需詳細資訊，請參閱 Amazon Elastic Container Registry 使用者指南中的推送 Helm Chart。
   - 從 eks-chartsGitHub GitHub儲存庫或從 ArtifactHub 安裝 Amazon EKS 圖表。

使用標籤組織 Amazon EKS 資源

您可以使用標籤協助您管理 Amazon EKS 資源。本主題提供標籤功能的概觀，並展示您如何建立標籤。

主題

- 標籤基本概念
- 標記您的 資源
- 標籤限制
- 標記您的資源以便計費
- 透過主控台使用標籤
- 透過 CLI、API 或 eksctl 使用標籤

> **Note**
>
> 標籤是與 Kubernetes 標籤和註釋分開的一種中繼資料。如需有關這些其他中繼資料類型的詳細資訊，請參閱 Kubernetes 文件中的下列章節：
>
> - 標籤和選取器

- 註釋

標籤基本概念

標籤是您指派給 AWS 資源的標籤。每個標籤皆包含索引鍵與選用值。

您可以使用標籤來分類 AWS 資源。例如，您可以依用途、擁有者或環境來分類資源。當您有許多相同類型的資源時，您可以依據先前指派給特定資源的標籤來快速識別該資源。例如，您可以為 Amazon EKS 叢集定義一組標籤，以協助您追蹤每個叢集的擁有者和堆疊層級。建議您為每個資源類型設計一組一致的標籤金鑰。然後，就可以根據您新增的標籤來搜尋和篩選資源。

新增標籤後，您可以隨時編輯標籤索引鍵和值，或從資源移除標籤。如果您刪除資源，也會刪除任何該資源的標籤。

標籤對 Amazon EKS 沒有任何語意意義，並嚴格解譯為字元字串。您可以將標籤的值設為空白字串。不過，您無法將標籤的值設定為 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。

標記您的 資源

以下 Amazon EKS 資源支援標籤：

- 叢集
- 受管節點群組
- Fargate 描述檔

您可以使用以下項目標記這些資源：

- 如果您使用的是 Amazon EKS 主控台，您可以隨時將標籤套用至新的或現有的資源。您可以在相關資源頁面使用 Tags (標籤) 索引標籤進行此操作。如需詳細資訊，請參閱the section called “透過主控台使用標籤”。
- 如果您使用的是 `eksctl`，則可以在使用 `--tags`選項建立資源時將標籤套用至資源。

- 如果您使用 AWS CLI、Amazon EKS API 或 AWS SDK，您可以使用相關 API 動作上的 `tags` 參數，將標籤套用至新資源。您也可以使用 `TagResource` API 動作將標籤套用到現有資源。如需詳細資訊，請參閱 TagResource。

當您使用某些資源建立動作時，您也可以在建立資源的同時為資源指定標籤。如果無法在建立資源時套用標籤，則無法建立資源。此機制可確保您要標記的資源是以您指定的標籤建立，不然就根本不會建立。如果您在建立資源時標記資源，則不需要在建立資源之後執行自訂標記指令碼。

標籤不會傳播到與您建立的資源相關聯的其他資源。例如，Fargate 設定檔標籤不會傳播到與 Fargate 設定檔相關聯的其他資源，例如與其排程的 Pod。

標籤限制

以下限制適用於標籤：

- 資源最多可與 50 個標籤建立關聯。
- 標籤索引鍵無法針對一個資源重複。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。
- 請勿使用 `aws:`、 `AWS:`或任何大寫或小寫的 組合，例如索引鍵或值的字首。這些僅供保留 AWS 使用。您無法編輯或刪除具有此字首的標籤索引鍵或值。使用此字首的標籤不會計入tags-per-resource限制。

標記您的資源以便計費

當您將標籤套用到 Amazon EKS 叢集時，您可以在成本與用量報告中使用標籤來分配成本。成本與用量報告中的計量資料會顯示所有 Amazon ECS 叢集的用量。如需詳細資訊，請參閱《 AWS 帳單使用者指南》中的AWS 成本和用量報告。

AWS 產生的成本分配標籤，特別是 ，`aws:eks:cluster-name`可讓您依 Cost Explorer 中的個別 Amazon EKS 叢集細分 Amazon EC2 執行個體成本。不過，此標籤不會擷取控制平面費用。標籤會自動新增至參與 Amazon EKS 叢集的 Amazon EC2 執行個體中。無論執行個體是使用 Amazon EKS 受管節點群組、Karpenter 還是直接使用 Amazon EC2 佈建，都會發生此行為。此特定標籤不會計入

50 個標籤限制。若要使用 標籤，帳戶擁有者必須在 AWS 帳單主控台中或使用 API 來啟用該標籤。當 AWS Organizations 管理帳戶擁有者啟用標籤時，也會為所有組織成員帳戶啟用該標籤。

您也可以根據具有相同標籤索引鍵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。這樣一來，您就可以查看該應用程式跨數項服務的總成本。如需使用標籤設定成本分配報告的詳細資訊，請參閱 帳單使用者指南中的每月成本分配報告。 AWS

> **Note**
>
> 如果您剛啟用報告，當月資料會在 24 小時之後提供檢視。

Cost Explorer 是一種報告工具，可作為 AWS 免費方案的一部分使用。您可以使用 Cost Explorer 檢視過去 13 個月的 Amazon EKS 資源圖表。您也可以預測未來三個月的可能花費。您可以查看一段時間內在 AWS 資源上的花費模式。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。

透過主控台使用標籤

您可以使用 Amazon EKS 主控台，管理與新的或現有的叢集和受管節點群組相關聯的標籤。

當您在 Amazon EKS 主控台中選取資源限定頁面時，該頁面會顯示那些資源的清單。例如，若您從左側導航窗格選取 Clusters (叢集)，主控台會顯示 Amazon EKS 叢集的清單。當您從支援標籤的其中一個清單選取資源時 (例如，特定的叢集)，您便可以在 Tags (標籤) 索引標籤上檢視和管理其標籤。

您也可以在 中使用標籤編輯器 AWS Management Console，提供統一的方式來管理您的標籤。如需詳細資訊，請參閱《標籤編輯器使用者指南》中的使用標籤編輯器標記您的 AWS 資源。 AWS

在建立資源時新增標籤

在建立 Amazon EKS 叢集、受管節點群組和 Fargate 描述檔時，您可以將標籤新增至其中。如需詳細資訊，請參閱the section called “建立叢集”。

在資源上新增和刪除標籤

您可以直接從資源的頁面新增或刪除與叢集相關聯的標籤。

1. 開啟 Amazon EKS 主控台。
2. 在導覽列上，選取要使用的 AWS 區域。
3. 在左側導覽窗格中選擇叢集。

4. 選擇特定叢集。
5. 選擇 Tags (標籤) 索引標籤，然後選擇 Manage tags (管理標籤)。
6. 在 Manage tags (管理標籤) 頁面上，視需要新增或刪除標籤。
   - 若要新增標籤，請選擇 Add tag (新增標籤)。然後指定每個標籤的索引鍵和值。
   - 若要移除標籤，請選擇 Remove tag (移除標籤)。
7. 針對您要新增或刪除的每個標籤重複此程序。
8. 選擇 Update (更新) 以完成操作。

透過 CLI、API 或 `eksctl` 使用標籤

使用下列 AWS CLI 命令或 Amazon EKS API 操作來新增、更新、列出和刪除資源的標籤。您只能使用 `eksctl` 新增標籤，同時使用一個命令建立新資源。

任務	AWS CLI	AWS 適用於 Windows PowerShell 的工具	API 動作
新增或覆寫一或多個標籤。	tag-resource	Add-EKSResourceTag	TagResource
刪除一或多個標籤。	untag-resource	Remove-EKSResourceTag	UntagResource

下列範例示範如何使用 CLI AWS 標記或取消標記資源。

範例 1：標記現有的叢集

以下命令標記現有的叢集。

```
aws eks tag-resource --resource-arn resource_ARN --tags team=devs
```

範例 2：取消標記現有的叢集

以下命令從現有的叢集刪除標籤。

```
aws eks untag-resource --resource-arn resource_ARN --tag-keys tag_key
```

範例 3：列出資源的標籤

以下命令列出與現有資源相關聯的標籤。

```
aws eks list-tags-for-resource --resource-arn resource_ARN
```

當您使用某些資源建立動作時，您可以在建立資源的同時指定標籤。下列動作支援在建立資源時指定標籤。

任務	AWS CLI	AWS 適用於 Windows PowerShell 的工具	API 動作	eksctl
建立叢集	create-cluster	New-EKSCluster	CreateCluster	`create cluster`
建立受管節點群組*	create-nodegroup	New-EKSNodegroup	CreateNodegroup	`create nodegroup`
建立 Fargate 設定檔	create-fargate-profile	New-EKSFargateProfile	CreateFargateProfile.html	`create fargatepr ofile`

- 如果您想要在建立受管節點群組時標記 Amazon EC2 執行個體，請使用啟動範本建立受管節點群組。如需詳細資訊，請參閱the section called “標記 Amazon EC2 執行個體”。如果您的執行個體已經存在，則可以手動為執行個體加上標籤。如需詳細資訊，請參閱《Amazon EC2 使用者指南》中的標記您的 資源。

檢視和管理 Amazon EKS 和 Fargate 服務配額

Amazon EKS 已與 Service Quotas 整合，此服務 AWS 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？。透過 Service Quotas 整合，您可以使用 和 CLI 快速查詢 Amazon EKS AWS Management Console 和 AWS Fargate AWS 服務配額的值。

在 中檢視 EKS 服務配額 AWS Management Console

1. 開啟 Service Quotas 主控台。

2. 在左側導覽窗格中，選擇 AWS 服務。
3. 從 AWS 服務清單中，搜尋並選取 Amazon Elastic Kubernetes Service (Amazon EKS) 或 AWS Fargate。

   在服務配額清單中，您可以看到服務配額名稱、套用值（如果有的話）、 AWS 預設配額，以及配額值是否可以調整。
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 Management Console，請參閱 Service Quotas 使用者指南。若要請求提升配額，請參閱《Service Quotas 使用者指南》中的請求提升配額。

使用 CLI 檢視 EKS AWS 服務配額

執行下列命令，以檢視您的 Amazon EKS 配額。

```
aws service-quotas list-aws-default-service-quotas \
    --query 'Quotas[*].
{Adjustable:Adjustable,Name:QuotaName,Value:Value,Code:QuotaCode}' \
    --service-code eks \
    --output table
```

執行下列命令，以檢視您的 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
```

> **Note**
>
> 傳回的配額是可在目前 AWS 區域中此帳戶中的 Fargate 上同時執行的 Amazon ECS 任務或 Amazon EKS Pod 數量。

若要使用 CLI AWS 使用更多服務配額，請參閱 AWS CLI 命令參考中的服務配額。若要請求提高配額，請參閱 CLI 命令參考中的 request-service-quota-increase 命令。 AWS

Amazon EKS 服務配額

AWS 建議使用 AWS Management Console 來檢視您目前的配額。如需詳細資訊，請參閱the section called “在 中檢視 EKS 服務配額 AWS Management Console”。

若要檢視預設 EKS 服務配額，請參閱《 AWS 一般參考》中的 Amazon Elastic Kubernetes Service 端點和配額。

在 Service Quotas 主控台中，Amazon Elastic Kubernetes Service (Amazon EKS) 之下會列出這些服務配額。若要針對顯示為可調整的值申請增加配額，請參閱《Service Quotas 使用指南》中的請求提高配額。

> **Note**
>
> Service Quotas 不支援對下列元件進行調整：* 每個叢集的 Pod 身分關聯。如需限制，請參閱 the section called “Pod 身分”。* 遠端節點網路CIDRs 或混合節點的遠端 Pod 網路。如需限制，請參閱 the section called “混合節點”。

AWS Fargate 服務配額

Service Quotas 主控台中的 AWS Fargate 服務會列出數個服務配額。您可以設定警示，在您的用量接近服務配額時發出警示。如需詳細資訊，請參閱the section called “建立 CloudWatch 警示來監控 Fargate 資源用量指標”。

新 AWS 帳戶的初始配額可能較低，可能會隨著時間增加。Fargate 會持續監控每個 AWS 區域中的帳戶用量，然後根據用量自動增加配額。您也可以針對顯示為可調整的值申請增加配額。如需詳細資訊，請參閱「Service Quotas 使用者指南」中的請求提高配額。

AWS 建議使用 AWS Management Console 來檢視您目前的配額。如需詳細資訊，請參閱the section called “在 中檢視 EKS 服務配額 AWS Management Console”。

若要檢視 EKS 服務配額上的預設 AWS Fargate，請參閱《 AWS 一般參考》中的 Fargate 服務配額。

> **Note**
>
> Fargate 還會強制執行 Amazon ECS 任務和 Amazon EKS Pods 啟動速率配額。如需詳細資訊，請參閱《Amazon ECS 指南》中的 AWS Fargate 限流配額。

Amazon EKS 中的安全

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。

安全是 AWS 與您之間的共同責任。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 AWS 雲端中執行 AWS 服務的基礎設施。對於 Amazon EKS，AWS 負責 Kubernetes 控制平面，其中包含控制平面節點和etcd資料庫。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 EKS 的合規計劃，請參閱合規計劃範圍內的AWS 服務。
- 雲端內部的安全 – 您的責任包含下列領域：
  - 資料平面的安全組態，包括允許流量從 Amazon EKS 控制平面傳遞至客戶 VPC 的安全群組各項組態
  - 節點和容器本身的組態
  - 節點的作業系統（包括更新和安全性修補程式）
  - 其他相關的應用程式軟體：
    - 設定及管理網路控制，例如防火牆規則
    - 搭配 IAM 或另以其他方式管理平台層級身分與存取管理
  - 資料的機密性、公司的要求，以及適用法律和法規

Amazon EKS 經過多個合規計劃認證，適用於受管制和敏感的應用程式。Amazon EKS 符合 SOC、PCI、ISO、FedRAMP-Moderate、IRAP、C5、K-ISMS、ENS High、OSPAR、HITRUST CSF 規範，並且是符合 HIPAA 資格的服務。如需詳細資訊，請參閱管理存取。

本文件有助於您了解如何在使用 Amazon EKS 時套用共同責任模型。下列主題說明如何將 Amazon EKS 設定為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 EKS 資源。

> **Note**
>
> Linux 容器由控制群組 (cgroup) 和命名空間組成，這些命名空間有助於限制容器可存取的內容，但是所有容器皆會與主機 Amazon EC2 執行個體共用相同的 Linux 核心。非常不建議您以根使用者身分執行容器 (UID 0)，或授予容器存取主機資源或命名空間 (例如主機網路或主機 PID 命名空間)，因為這樣做會降低容器所提供之隔離的有效性。

主題

- 使用最佳實務保護 Amazon EKS 叢集
- 分析 Amazon EKS 中的漏洞
- Amazon EKS 叢集的合規驗證
- Amazon Elastic Kubernetes Service 的安全考量
- Kubernetes 的安全考量
- Amazon EKS Auto Mode 的安全考量
- Amazon EKS 的 Identity and Access Management

使用最佳實務保護 Amazon EKS 叢集

Amazon EKS 安全最佳實務位於 Amazon EKS 最佳實務指南中的安全最佳實務中。

分析 Amazon EKS 中的漏洞

安全性是設定和維護 Kubernetes 叢集和應用程式的重要考量因素。以下列出可供您分析 EKS 叢集安全組態的資源、可供您檢查漏洞的資源，以及可為您執行該分析 AWS 的服務整合。

Amazon EKS 的網際網路安全中心 (CIS) 基準測試

Center for Internet Security (CIS) Kubernetes Benchmark 提供 Amazon EKS 安全組態的指引。基準化分析：

- 適用於您負責 Kubernetes 元件的安全組態的 Amazon EC2 節點 (受管和自我管理)。
- 提供經社群核准的標準方式，以確保您在使用 Amazon EKS 時能安全地設定 Kubernetes 叢集和節點。
- 由四個部分組成：控制平面記錄設定、節點安全設定、政策和受管服務。
- 支援 Amazon EKS 目前可用的所有 Kubernetes 版本，並且可以使用 kube-bench 予以執行，這是一個標準開源工具，可用於在 Kubernetes 叢集上使用 CIS 基準檢查組態。

若要進一步了解，請參閱 CIS Amazon EKS 基準簡介。

如需使用 CIS 基準 AMI 更新節點群組的自動化`aws-sample`管道，請參閱 EKS 最佳化 AMI 強化管道。

Amazon EKS 平台版本

Amazon EKS 平台版本代表叢集控制平面的功能，包括啟用哪些 Kubernetes API 伺服器旗標和目前的 Kubernetes 修補程式版本。新叢集是使用最新的平台版本部署。如需詳細資訊，請參閱 eks/latest/userguide/platform-versions.html【EKS platform-versions，type="documentation"】。

您可以將 Amazon EKS 叢集更新為較新的 Kubernetes 版本。由於 Amazon EKS 會提供新的 Kubernetes 版本，所以我們建議您主動更新您的叢集，以便使用最新的可用版本。如需 EKS 中 Kubernetes 版本的詳細資訊，請參閱 eks/latest/userguide/kubernetes-versions.html【Amazon EKS 支援的版本，type="documentation"】。

作業系統漏洞清單

AL2023 漏洞清單

在 Amazon Linux 安全中心追蹤 Amazon Linux 2023 的安全或隱私權事件，或訂閱相關聯的 RSS 摘要。安全與隱私權事件包括影響問題的概觀和套件，並說明如何更新執行個體以修正問題。

Amazon Linux 2 漏洞清單

可以透過 Amazon Linux 安全中心或是訂閱關聯的 RSS 摘要，追蹤 Amazon Linux 2 的安全或隱私權事件。安全與隱私權事件包括影響問題的概觀和套件，並說明如何更新執行個體以修正問題。

使用 Amazon Inspector 進行節點偵測

您可以使用 Amazon Inspector 檢查節點是否有意外的網路存取問題，以及 Amazon EC2 執行個體是否有漏洞。

使用 Amazon GuardDuty 進行叢集和節點偵測

Amazon GuardDuty 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 提供下列兩種功能，可偵測對 EKS 叢集的潛在威脅：EKS 保護和執行期監控。

如需詳細資訊，請參閱the section called “Amazon GuardDuty”。

Amazon EKS 叢集的合規驗證

若要了解 AWS 服務是否在特定合規計劃範圍內，請參閱AWS 合規計劃範圍內的服務，並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 AWS Artifact 下載第三方稽核報告。如需詳細資訊，請參閱下載 AWS Artifact 中的報告。

您使用 AWS 服務的合規責任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。AWS 提供下列資源以協助合規：

- 安全合規與治理 - 這些解決方案實作指南內容討論了架構考量，並提供部署安全與合規功能的步驟。
- HIPAA 合格服務參考 – 列出 HIPAA 合格服務。並非所有 AWS 服務都符合 HIPAA 資格。
- AWS 合規資源 – 此工作手冊和指南集合可能適用於您的產業和位置。
- AWS 客戶合規指南 – 透過合規的角度了解共同責任模型。本指南摘要說明保護 AWS 服務的最佳實務，並將指引映射至跨多個架構的安全控制（包括國家標準技術研究所 (NIST)、支付卡產業安全標準委員會 (PCI) 和國際標準組織 (ISO)）。
- AWS 《Config 開發人員指南》中的使用規則評估資源 – Config AWS 服務會評估資源組態符合內部實務、產業準則和法規的程度。
- AWS Security Hub – AWS 此服務提供安全狀態的完整檢視 AWS。Security Hub 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清單，請參閱「Security Hub 控制參考」。
- Amazon GuardDuty – AWS 此服務透過監控您的環境是否有可疑和惡意活動，偵測對 AWS 您的帳戶、工作負載、容器和資料的潛在威脅。GuardDuty 可滿足特定合規架構所規定的入侵偵測需求，以協助您因應 PCI DSS 等各種不同的合規需求。
- AWS Audit Manager – AWS 此服務可協助您持續稽核 AWS 用量，以簡化您管理風險和符合法規和業界標準的方式。

Amazon Elastic Kubernetes Service 的安全考量

以下是雲端安全性的考量，因為它們會影響 Amazon EKS。

主題

- Amazon EKS 中的基礎設施安全
- 了解 Amazon EKS 叢集中的彈性

Amazon EKS 中的基礎設施安全

Amazon Elastic Kubernetes 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 保護基礎設施的相關資訊，請參閱AWS 雲端安全。若要使用基礎設施安全的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 已發佈的 API 呼叫，透過網路存取 Amazon EKS。使用者端必須支援下列專案：

- Transport Layer Security (TLS)。我們需要 TLS 1.2 並建議使用 TLS 1.3。
- 具備完美轉送私密(PFS)的密碼套件，例如 DHE (Ephemeral Diffie-Hellman)或 ECDHE (Elliptic Curve Ephemeral Diffie-Hellman)。現代系統(如 Java 7 和更新版本)大多會支援這些模式。

此外，請求必須使用存取金鑰 ID 和與 IAM 主體相關聯的私密存取金鑰來簽署。或者，您可以使用 AWS Security Token Service (AWS STS) 產生臨時安全登入資料來簽署請求。

建立 Amazon EKS 叢集時，為要使用的叢集指定 VPC 子網路。Amazon EKS 需要至少兩個處於可用區域的子網路。我們建議使用具有公有和私有子網路的 VPC，以便 Kubernetes 可以在公有子網路中建立公有負載平衡器，以平衡私有子網路中節點上執行之 Pod 的流量。

如需 VPC 考量的詳細資訊，請參閱 the section called “VPC 和子網要求”。

如果您使用 Amazon EKS 入門演練中提供的 AWS CloudFormation 範本建立 VPC 和節點群組，則您的控制平面和節點安全群組會使用我們建議的設定進行設定。

如需安全群組考量的詳細資訊，請參閱 the section called “安全群組要求”。

建立新的叢集時，Amazon EKS 會為您用來與叢集通訊的受管 Kubernetes API 伺服器建立端點 (使用 Kubernetes 管理工具，例如 `kubectl`)。根據預設，此 API 伺服器端點對網際網路是公有的，而且使用 AWS Identity and Access Management (IAM) 和原生 Kubernetes 角色型存取控制 (RBAC) 的組合來保護對 API 伺服器的存取。

您可啟用 Kubernetes API 伺服器的私有存取，讓節點和 API 伺服器間的所有通訊都不會離開 VPC。您可以限制可以從網際網路存取 API 伺服器的 IP 地址，或完全停用對 API 伺服器的網際網路存取。

如需修改叢集端點存取的詳細資訊，請參閱 the section called “修改叢集端點存取”。

您可以使用 Amazon VPC CNI 或第三方工具實作 Kubernetes 網路政策，例如 Project Calico。如需有關使用適用於網路政策的 Amazon VPC CNI 的詳細資訊，請參閱 the section called “Kubernetes 政策”。Project Calico 是第三方開放原始碼專案。如需詳細資訊，請參閱 Project Calico 文件。

使用 AWS PrivateLink 存取 Amazon EKS

您可以使用 AWS PrivateLink 在 VPC 和 Amazon Elastic Kubernetes Service 之間建立私有連線。您可以像在 VPC 中一樣存取 Amazon EKS，無需使用網際網路閘道、NAT 裝置、VPN 連接或 AWS Direct Connect 連接。VPC 中的執行個體不需要公有 IP 地址即可存取 Amazon EKS。

您可以透過建立採用 AWS PrivateLink 技術的介面端點來建立此私有連線。我們會在您為介面端點啟用的每個子網中建立端點網路介面。這些是請求者管理的網路介面，可作為目的地為 Amazon EKS 之流量的進入點。

如需詳細資訊，請參閱《 AWS PrivateLink 指南》中的透過 PrivateLink 存取 AWS 服務。 AWS PrivateLink

開始之前

開始之前，請確定您已執行下列任務：

- 《 AWS PrivateLink 指南》中的檢閱使用介面 VPC 端點存取 AWS 服務

考量事項

- 支援和限制：Amazon EKS 介面端點可讓您從 VPC 安全存取所有 Amazon EKS API 動作，但具有特定限制：它們不支援存取 Kubernetes APIs，因為這些端點具有單獨的私有端點，因此您無法將 Amazon EKS 設定為只能透過介面端點存取。
- 定價：使用 Amazon EKS 的界面端點會產生 standard AWS PrivateLink 費用：每個可用區域中佈建的每個端點的每小時費用、透過端點的流量的資料處理費用。若要進一步了解，請參閱 AWS PrivateLink 定價。
- 安全與存取控制：建議使用這些額外組態增強安全性並控制存取：使用 VPC 端點政策透過介面端點控制對 Amazon EKS 的存取、將安全群組與端點網路介面建立關聯以管理流量、使用 VPC 流量日誌擷取和監控進出介面端點的 IP 流量，以及可發佈至 Amazon CloudWatch 或 Amazon S3 的日誌。若要進一步了解，請參閱使用端點政策控制對 VPC 端點的存取，以及使用 VPC 流程日誌記錄 IP 流量。
- 連線選項：界面端點使用內部部署存取（使用 AWS Direct Connect 或 AWS Site-to-Site VPN 將內部部署資料中心連接至 VPC 與介面端點）或透過跨 VPC 連線（使用 AWS Transit Gateway 或 VPC 對等互連將其他 VPCs 與介面端點連接至 VPC，將流量保留在 AWS 網路內）提供彈性的連線選項。
- IP 版本支援：2024 年 8 月之前建立的端點僅支援使用 eks.region.amazonaws.com 的 IPv4。在 2024 年 8 月之後建立的新端點支援雙堆疊 IPv4 和 IPv6（例如 eks.region.amazonaws.com、eks.region.api.aws)。
- 區域可用性：EKS API 的 AWS PrivateLink不適用於亞太區域（馬來西亞）(ap-southeast-5)、亞太區域（泰國）(ap-southeast-7)、墨西哥（中部）(mx-central-1) 和亞太區域（台北）(ap-east-2) 區域。亞太區域（馬來西亞）(ap-southeast-5) 區域提供 eks-auth (EKS Pod Identity) 的 AWS PrivateLink支援。

建立 Amazon VPC 的介面端點

您可以使用 Amazon VPC 主控台或 AWS 命令列界面 (AWS CLI) 來建立 Amazon EKS 的界面端點。如需詳細資訊，請參閱 AWS PrivateLink 指南中的建立 VPC 端點。

使用下列服務名稱建立 Amazon EKS 的介面端點：

EKS API

- com.amazonaws.region-code.eks
- com.amazonaws.region-code.eks-fips （適用於 FIPS 相容端點）

EKS 身分驗證 API (EKS Pod 身分）

- com.amazonaws.region-code.eks-auth

Amazon EKS 介面端點的私有 DNS 功能

預設針對 Amazon EKS 和其他 AWS 服務的介面端點啟用的私有 DNS 功能，可使用預設的區域 DNS 名稱來促進安全且私有的 API 請求。此功能可確保透過私有 AWS 網路的界面端點路由 API 呼叫，從而增強安全性和效能。

當您為 Amazon EKS 或其他 AWS 服務建立介面端點時，私有 DNS 功能會自動啟用。若要啟用 ，您需要透過設定特定屬性來正確設定 VPC：

- enableDnsHostnames：允許 VPC 內的執行個體具有 DNS 主機名稱。
- enableDnsSupport：啟用整個 VPC 的 DNS 解析。

如需檢查或修改這些設定的step-by-step說明，請參閱檢視和更新 VPC 的 DNS 屬性。

DNS 名稱和 IP 地址類型

啟用私有 DNS 功能後，您可以使用特定 DNS 名稱來連線至 Amazon EKS，而且這些選項會隨著時間演進：

- eks.region.amazonaws.com：傳統 DNS 名稱，僅在 2024 年 8 月之前解析為 IPv4 地址。對於更新為雙堆疊的現有端點，此名稱會解析為 IPv4 和 IPv6 地址。
- eks.region.api.aws：適用於 2024 年 8 月之後建立的新端點，此雙堆疊 DNS 名稱同時解析為 IPv4 和 IPv6 地址。

2024 年 8 月之後，新的介面端點會有兩個 DNS 名稱，您可以選擇雙堆疊 IP 地址類型。對於現有端點，更新至雙堆疊會修改 eks.region.amazonaws.com 以支援 IPv4 和 IPv6。

使用私有 DNS 功能

設定完成後，即可將私有 DNS 功能整合至您的工作流程，提供下列功能：

- API 請求：根據您端點的設定`eks.region.api.aws`，使用預設的區域 DNS 名稱 `eks.region.amazonaws.com`或 向 Amazon EKS 提出 API 請求。
- 應用程式相容性：呼叫 EKS APIs現有應用程式不需要變更即可利用此功能。
- 具有雙堆疊的 AWS CLI：若要搭配 CLI AWS 使用雙堆疊端點，請參閱 AWS SDKs和工具參考指南中的雙堆疊和 FIPS 端點組態。
- 自動轉接：對 Amazon EKS 預設服務端點的任何呼叫都會自動導向介面端點，以確保私有且安全的連線。

了解 Amazon EKS 叢集中的彈性

AWS 全域基礎設施是以 AWS 區域和可用區域為基礎建置。 AWS 區域提供多個實體隔離和隔離的可用區域，這些區域與低延遲、高輸送量和高度冗餘聯網連接。透過可用區域，您所設計與操作的應用程式和資料庫，就能夠在可用區域之間自動容錯移轉，而不會發生中斷。可用區域的可用性、容錯能力和擴充能力，均較單一或多個資料中心的傳統基礎設施還高。

Amazon EKS 會跨多個 AWS 可用區域執行和擴展 Kubernetes 控制平面，以確保高可用性。Amazon EKS 會根據負載來自動擴展控制平面執行個體，且會偵測及取代狀態不佳的控制平面執行個體，還會對控制平面進行自動化程式修補。啟動版本更新後，Amazon EKS 會更新控制平面，並在更新期間維持控制平面的高可用性。

此控制平面包含至少兩個 API 伺服器執行個體，以及三個執行個體，這些`etcd`執行個體在 AWS 區域內的三個可用區域之間執行。Amazon EKS：

- 主動監控控制平面執行個體上的負載，並自動進行擴展以確保高效能。
- 自動偵測並取代運作狀態不佳的控制平面執行個體，並視需要在 AWS 區域內的可用區域重新啟動執行個體。
- 利用 AWS 區域的架構來維持高可用性。因此，Amazon EKS 能夠提供 API 伺服器端點可用性的 SLA。

如需 AWS 區域和可用區域的詳細資訊，請參閱AWS 全域基礎設施。

Kubernetes 的安全考量

以下是雲端安全性的考量，因為它們會影響 Amazon EKS 叢集中的 Kubernetes。如需深入了解 Kubernetes 中的安全控制和實務，請參閱 Kubernetes 文件中的 Cloud Native Security 和 Kubernetes。

主題

- 使用 Kubernetes 憑證保護工作負載
- 了解 Amazon EKS 建立的 RBAC 角色和使用者
- 在現有叢集上使用 KMS 加密 Kubernetes 秘密
- 搭配 Amazon EKS Pod 使用 AWS Secrets Manager 秘密
- 所有 Kubernetes API 資料的預設信封加密

使用 Kubernetes 憑證保護工作負載

Kubernetes Certificates API 會自動佈建 X.509 登入資料。API 具有命令列界面，可讓 Kubernetes API 用戶端向憑證授權機構 (CA) 請求和取得 X.509 憑證。您可以使用 `CertificateSigningRequest` (CSR) 資源來請求受指示的簽署者簽署憑證。您的請求在簽署之前獲得核准或拒絕。Kubernetes 支援具有明確定義行為的內建簽署者和自訂簽署者。透過這種方式，客戶可以預測其 CSR 會發生什麼情況。如需憑證簽署的相關資訊，請參閱 Certificate Signing Requests (憑證簽署請求)。

其中一個內建簽署者是 `kubernetes.io/legacy-unknown`。CSR 資源的 `v1beta1` API 接受此傳統未知的簽署者。不過，CSR 的穩定 `v1` API 不允許 `signerName` 設定為 `kubernetes.io/legacy-unknown`。

如果您想使用 Amazon EKS CA 在叢集上產生憑證，則必須使用自訂簽署者。若要使用 CSR `v1` API 版本並產生新憑證，您必須遷移任何現有清單檔案和 API 用戶端。使用現有 `v1beta1` API 建立的現有憑證有效且可正常運作，直到憑證過期為止。這包含下列項目：

- 信任分佈：無。Kubernetes 叢集中沒有此簽署者的標準信任或分佈。
- 允許的主體：任何
- 允許的 x509 延伸項目：接受 subjectAltName 和金鑰使用方式延伸項目，並放棄其他延伸項目
- 允許的金鑰使用方式：不得包括 [「金鑰加密」、「數位簽章」、「伺服器身分驗證」] 以外的用法

> **Note**
>
> 不支援用戶端憑證簽署。

- 到期/憑證生命週期：1 年 (預設和最長)
- 允許/不允許 CA 位元：不允許

使用 signerName 產生 CSR 的範例

以下步驟將展示如何使用 `signerName: beta.eks.amazonaws.com/app-serving` 為 DNS 名稱 `myserver.default.svc` 產生服務憑證。將此作為您自己環境的指南。

1. 執行 `openssl genrsa -out myserver.key 2048` 命令以產生 RSA 私有金鑰。

```
openssl genrsa -out myserver.key 2048
```

2. 執行下列命令來產生憑證請求。

```
openssl req -new -key myserver.key -out myserver.csr -subj "/CN=myserver.default.svc"
```

3. 產生 CSR 請求的 base64 值，並將其存放於變數中以供稍後步驟中使用。

```
base_64=$(cat myserver.csr | base64 -w 0 | tr -d "
")
```

4. 執行下列命令以建立名為 `mycsr.yaml` 的檔案。在下列範例中，`beta.eks.amazonaws.com/app-serving` 是 `signerName`。

```
cat >mycsr.yaml <<EOF
apiVersion: certificates.k8s.io/v1
kind: CertificateSigningRequest
metadata:
  name: myserver
spec:
  request: $base_64
  signerName: beta.eks.amazonaws.com/app-serving
  usages:
    - digital signature
    - key encipherment
    - server auth
```

```
EOF
```

5. 提交 CSR。

```
kubectl apply -f mycsr.yaml
```

6. 核准服務憑證。

```
kubectl certificate approve myserver
```

7. 驗證憑證已核發。

```
kubectl get csr myserver
```

範例輸出如下。

```
NAME       AGE     SIGNERNAME                              REQUESTOR          CONDITION
myserver   3m20s   beta.eks.amazonaws.com/app-serving   kubernetes-admin
 Approved,Issued
```

8. 匯出已核發的憑證。

```
kubectl get csr myserver -o jsonpath='{.status.certificate}'| base64 -d >
 myserver.crt
```

了解 Amazon EKS 建立的 RBAC 角色和使用者

當您建立 Kubernetes 叢集時，會在該叢集上建立數個預設 Kubernetes 身分，以正常運作 Kubernetes。Amazon EKS 會為其每個預設元件建立 Kubernetes 身分。身分為叢集元件提供 Kubernetes 角色型授權控制 (RBAC)。如需詳細資訊，請參閱 Kubernetes 文件中的使用 RBAC 授權。

當您將選用的附加元件安裝到叢集時，可能會將其他 Kubernetes 身分新增至叢集。如需有關本主題未討論之身分的詳細資訊，請參閱附加元件的文件。

您可以使用 AWS Management Console 或 `kubectl`命令列工具，在叢集上檢視 Amazon EKS 建立的 Kubernetes 身分清單。所有的使用者身分都會出現在 kube 稽核日誌中，可供您透過 Amazon CloudWatch 加以利用。

AWS Management Console

先決條件

您使用的 IAM 主體必須具有必要許可中所述的許可。

使用 檢視 Amazon EKS 建立的身分 AWS Management Console

1. 開啟 Amazon EKS 主控台。
2. 在 Clusters (叢集) 清單中，選擇包含您要檢視之身分的叢集。
3. 選擇 Resources (資源) 標籤。
4. 在 Resource types (資源類型) 下選擇 Authorization (授權)。
5. 選擇 ClusterRoles、ClusterRoleBindings、Roles (角色) 或 RoleBindings。所有前面帶有 eks 的資源皆是由 Amazon EKS 所建立。其他由 Amazon EKS 建立的身分資源包含：
   - 名為 aws-node 的 ClusterRole 和 ClusterRoleBinding。aws-node 資源支援 Kubernetes 的 Amazon VPC CNI 外掛程式，Amazon EKS 會將其安裝在所有叢集上。
   - 名為 vpc-resource-controller-role 的 ClusterRole，以及名為 vpc-resource-controller-rolebinding 的 ClusterRoleBinding。這些資源支援 Amazon VPC 資源控制器，Amazon EKS 會在所有叢集上安裝該控制器。

   除了您在 主控台中看到的資源之外，叢集上還存在下列特殊使用者身分，但叢集的組態中看不到這些身分：
   - **`eks:cluster-bootstrap`** – 用於叢集引導期間的 `kubectl` 操作。
   - **`eks:support-engineer`** – 用於叢集管理操作。
6. 選擇特定資源以檢視其相關詳細資訊。根據預設，您會在結構化檢視中看到資訊。您可以在詳細資訊頁面的右上角，選擇 Raw view (原始檢視) 以查看資源的所有資訊。

Kubectl

先決條件

您用來列出叢集上 Kubernetes 資源的實體 (AWS Identity and Access Management (IAM) 或 OpenID Connect (OIDC)) 必須由 IAM 或 OIDC 身分提供者進行驗證。必須授予實體許可，才能針對您希望實體使用之叢集上的 `Role`、`RoleBinding`、 `ClusterRole`和 `ClusterRoleBinding` 資源使用 Kubernetes 和`getlist`動詞。如需有關授予 IAM 實體叢集存取權的詳細資訊，請參閱the section called “Kubernetes API 存取”。如需有關授予由您自己的 OIDC 提供者驗證的實體存取叢集的詳細資訊，請參閱 the section called “連結 OIDC 供應商”。

使用 **kubectl** 檢視 Amazon EKS 所建立的身分

執行您要查看的資源類型的命令。所有前面帶有 eks 的傳回資源皆是由 Amazon EKS 所建立。除了從命令輸出中傳回的資源之外，叢集上還存在下列特殊使用者身分，但叢集的組態中看不到這些身分：

- **eks:cluster-bootstrap** – 用於叢集引導期間的 kubectl 操作。
- **eks:support-engineer** – 用於叢集管理操作。

ClusterRoles – ClusterRoles 範圍限定於您的叢集，因此授予角色的任何許可都適用於叢集上任何 Kubernetes 命名空間中的資源。

下列命令會傳回叢集ClusterRoles上建立的所有 Amazon EKS Kubernetes。

```
kubectl get clusterroles | grep eks
```

除了前面帶有 的輸出中傳回的 ClusterRoles 之外，還存在以下 ClusterRoles。

- **aws-node** – 這ClusterRole支援 Kubernetes 的 Amazon VPC CNI 外掛程式，Amazon EKS 會安裝在所有叢集上。
- **vpc-resource-controller-role** – 這ClusterRole支援 Amazon VPC 資源控制器，Amazon EKS 會安裝在所有叢集上。

若要查看 ClusterRole 的規格，請將以下命令中的 *ek: k8s-metrics* 取代為先前命令輸出中傳回的 ClusterRole。下列範例會傳回 *eks:k8s-metrics* ClusterRole 的規格。

```
kubectl describe clusterrole eks:k8s-metrics
```

範例輸出如下。

```
Name:         eks:k8s-metrics
Labels:       <none>
Annotations:  <none>
PolicyRule:
  Resources         Non-Resource URLs  Resource Names  Verbs
  ---------         -----------------  --------------  -----
                    [/metrics]         []              [get]
  endpoints         []                 []              [list]
  nodes             []                 []              [list]
```

```
  pods              []                []              [list]
  deployments.apps  []                []              [list]
```

ClusterRoleBindings：ClusterRoleBindings 設定範圍為您的叢集內。

下列命令會傳回叢集ClusterRoleBindings上建立的所有 Amazon EKS Kubernetes。

```
kubectl get clusterrolebindings | grep eks
```

除了輸出中傳回的 ClusterRoleBindings 之外，還存在以下 ClusterRoleBindings。

- **aws-node** – 這ClusterRoleBinding支援 Kubernetes 的 Amazon VPC CNI 外掛程式，Amazon EKS 會安裝在所有叢集上。
- **vpc-resource-controller-rolebinding** – 這ClusterRoleBinding支援 Amazon VPC 資源控制器，Amazon EKS 會安裝在所有叢集上。

若要查看 ClusterRoleBinding 的規格，請將以下命令中的 *ek: k8s-metrics* 取代為先前命令輸出中傳回的 ClusterRoleBinding。下列範例會傳回 *eks:k8s-metrics* ClusterRoleBinding 的規格。

```
kubectl describe clusterrolebinding eks:k8s-metrics
```

範例輸出如下。

```
Name:         eks:k8s-metrics
Labels:       <none>
Annotations:  <none>
Role:
  Kind:  ClusterRole
  Name:  eks:k8s-metrics
Subjects:
  Kind  Name             Namespace
  ----  ----             ---------
  User  eks:k8s-metrics
```

角色 – Roles 範圍限定於 Kubernetes 命名空間。所有 Amazon EKS 所建立的 Roles 設定範圍皆為 kube-system 命名空間內。

下列命令會傳回叢集Roles上所有建立的 Amazon EKS Kubernetes。

```
kubectl get roles -n kube-system | grep eks
```

若要查看 Role 的規格，請將以下命令中的 *ek: k8s-metrics* 取代為先前命令輸出中傳回之 Role 的名稱。下列範例會傳回 *eks:k8s-metrics* Role 的規格。

```
kubectl describe role eks:k8s-metrics -n kube-system
```

範例輸出如下。

```
Name:         eks:k8s-metrics
Labels:       <none>
Annotations:  <none>
PolicyRule:
  Resources         Non-Resource URLs  Resource Names              Verbs
  ---------         -----------------  --------------              -----
  daemonsets.apps   []                 [aws-node]                  [get]
  deployments.apps  []                 [vpc-resource-controller]  [get]
```

RoleBindings – RoleBindings 範圍限定於 Kubernetes 命名空間。所有 Amazon EKS 所建立的 RoleBindings 設定範圍皆為 kube-system 命名空間內。

下列命令會傳回叢集RoleBindings上建立的所有 Amazon EKS Kubernetes。

```
kubectl get rolebindings -n kube-system | grep eks
```

若要查看 RoleBinding 的規格，請將以下命令中的 *ek: k8s-metrics* 取代為先前命令輸出中傳回的 RoleBinding。下列範例會傳回 *eks:k8s-metrics* RoleBinding 的規格。

```
kubectl describe rolebinding eks:k8s-metrics -n kube-system
```

範例輸出如下。

```
Name:         eks:k8s-metrics
Labels:       <none>
Annotations:  <none>
Role:
  Kind:  Role
  Name:  eks:k8s-metrics
Subjects:
  Kind  Name             Namespace
```

```
----  ----              ---------
User  eks:k8s-metrics
```

在現有叢集上使用 KMS 加密 Kubernetes 秘密

> **Important**
>
> 此程序僅適用於執行 Kubernetes 1.27 版或更舊版本的 EKS 叢集。如果您執行的是 Kubernetes 1.28 版或更新版本，根據預設，您的 Kubernetes 秘密會受到信封加密保護。如需詳細資訊，請參閱the section called “所有 Kubernetes API 資料的預設信封加密”。

如果您啟用秘密加密，則會使用您選取的 AWS KMS 金鑰來加密 Kubernetes 秘密。KMS 金鑰必須滿足以下條件：

- 對稱
- 可加密和解密資料
- 在與叢集相同的 AWS 區域中建立
- 如果 KMS 金鑰是在其他帳戶中建立，則 IAM p主體必須具有 KMS 金鑰的存取權。

如需詳細資訊，請參閱 AWS Key Management Service 開發人員指南中的允許其他帳戶中的 IAM 主體使用 KMS 金鑰。

> **Warning**
>
> 您無法在啟用秘密加密之後停用秘密加密。此動作不可復原。

eksctl

此程序僅適用於執行 Kubernetes 1.27 版或更舊版本的 EKS 叢集。如需詳細資訊，請參閱the section called “所有 Kubernetes API 資料的預設信封加密”。

您可以在以兩種方式啟用加密：

- 使用單一命令將加密新增至叢集。

  若要自動重新加密您的秘密，請執行下列命令。

```
eksctl utils enable-secrets-encryption \
    --cluster my-cluster \
    --key-arn arn:aws: kms:region-code:account:key/key
```

若要退出自動重新加密您的密碼，請執行下列命令。

```
eksctl utils enable-secrets-encryption
    --cluster my-cluster \
    --key-arn arn:aws: kms:region-code:account:key/key \
    --encrypt-existing-secrets=false
```

- 使用 `kms-cluster.yaml` 檔案將加密新增到您的叢集中。

```
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: my-cluster
  region: region-code

secretsEncryption:
  keyARN: arn:aws: kms:region-code:account:key/key
```

若要讓您的秘密自動重新加密，請執行下列命令。

```
eksctl utils enable-secrets-encryption -f kms-cluster.yaml
```

若要退出自動重新加密您的密碼，請執行下列命令。

```
eksctl utils enable-secrets-encryption -f kms-cluster.yaml --encrypt-existing-
secrets=false
```

AWS Management Console

a. 此程序僅適用於執行 Kubernetes 1.27 版或更舊版本的 EKS 叢集。如需詳細資訊，請參閱the section called “所有 Kubernetes API 資料的預設信封加密”。

b. 開啟 Amazon EKS 主控台。

c. 選擇您要向其新增 KMS 加密的叢集。

d. 選擇 Overview (概觀) 索引標籤 (預設選取此項)。

e. 向下捲動到 Secrets encryption (私密加密) 區段，然後選擇 Enable (啟用)。

f. 從下拉式清單中選取金鑰，然後選擇 Enable (啟用) 按鈕。如果未列出任何金鑰，您必須先建立一個。如需詳細資訊，請參閱建立金鑰。

g. 選擇 Confirm (確認) 按鈕以使用選擇的金鑰。

AWS CLI

a. 此程序僅適用於執行 Kubernetes 1.27 版或更舊版本的 EKS 叢集。如需詳細資訊，請參閱the section called “所有 Kubernetes API 資料的預設信封加密”。

b. 使用下列 CLI AWS 命令，將秘密加密組態與叢集建立關聯。使用自己的取代###。

```
aws eks associate-encryption-config \
    --cluster-name my-cluster \
    --encryption-config '[{"resources":["secrets"],"provider":
{"keyArn":"arn:aws: kms:region-code:account:key/key"}}]'
```

範例輸出如下。

```
{
  "update": {
    "id": "3141b835-8103-423a-8e68-12c2521ffa4d",
    "status": "InProgress",
    "type": "AssociateEncryptionConfig",
    "params": [
      {
        "type": "EncryptionConfig",
        "value": "[{\"resources\":[\"secrets\"],\"provider\":{\"keyArn\":
\"arn:aws: kms:region-code:account:key/key\"}}]"
      }
    ],
    "createdAt": 1613754188.734,
    "errors": []
  }
}
```

c. 您可以使用下列命令來監控您的加密更新的狀態。使用之前輸出傳回的特定 `cluster name` 和 `update ID`。顯示 `Successful` 狀態時，即表示更新已完成。

```
aws eks describe-update \
    --region region-code \
    --name my-cluster \
```

```
    --update-id 3141b835-8103-423a-8e68-12c2521ffa4d
```

範例輸出如下。

```
{
  "update": {
    "id": "3141b835-8103-423a-8e68-12c2521ffa4d",
    "status": "Successful",
    "type": "AssociateEncryptionConfig",
    "params": [
      {
        "type": "EncryptionConfig",
        "value": "[{\"resources\":[\"secrets\"],\"provider\":{\"keyArn\":
\"arn:aws: kms:region-code:account:key/key\"}}]"
      }
    ],
    "createdAt": 1613754188.734>,
    "errors": []
  }
}
```

d. 若要驗證您的叢集中是否已啟用加密，請執行 `describe-cluster` 命令。回應包含 `EncryptionConfig` 字串。

```
aws eks describe-cluster --region region-code --name my-cluster
```

在叢集上啟用加密之後，您須使用新金鑰來加密所有現有的密碼：

> **Note**
>
> 如果您使用 `eksctl`，則僅當您選擇退出自動重新加密秘密時，才需要執行以下命令。

```
kubectl get secrets --all-namespaces -o json | kubectl annotate --overwrite -f - kms-
encryption-timestamp="time value"
```

> **Warning**
>
> 如果您為現有叢集啟用秘密加密，且您使用的 KMS 金鑰曾經遭到刪除，則無法復原叢集。若刪除 KMS 金鑰，即會永久使叢集處於降級狀態。如需詳細資訊，請參閱刪除 AWS KMS 金鑰。

> **Note**
>
> 根據預設，`create-key`命令會建立對稱加密 KMS 金鑰，其中包含金鑰政策，該政策可讓帳戶根管理員存取 AWS KMS 動作和資源。如果您想要縮小許可範圍，請確保要呼叫 `create-cluster` API 之主體的政策上允許 `kms:DescribeKey` 和 `kms:CreateGrant` 動作。對於使用 KMS 封套加密的叢集，需要 `kms:CreateGrant` 許可。CreateCluster 動作不支援條件 `kms:GrantIsForAWSResource`，並且不應在 KMS 政策中使用該條件來控制執行 CreateCluster 的使用者的 `kms:CreateGrant` 許可。

搭配 Amazon EKS Pod 使用 AWS Secrets Manager 秘密

若要將 Secrets Manager 的秘密和參數存放區的參數顯示為掛載在 Amazon EKS Pod 中的檔案，您可以使用 Kubernetes Secrets Store CSI 驅動程式的 AWS Secrets and Configuration Provider (ASCP)。

透過 ASCP，您可以在 Secrets Manager 中存放和管理您的秘密，然後透過在 Amazon EKS 上執行的工作負載擷取這些秘密。您可以使用 IAM 角色和政策來限制對叢集中特定 Kubernetes Pod 秘密的存取。ASCP 會擷取 Pod 身分，並交換 IAM 角色的身分。ASCP 會擔任 Pod 的 IAM 角色，然後可以從已授權該角色的 Secrets Manager 擷取秘密。

如果您使用 Secrets Manager 自動輪換您的秘密，您也可以使用 Secrets Store CSI Driver 輪換調解器功能，以確保您從 Secrets Manager 擷取最新的秘密。

> **Note**
>
> AWS 不支援 Fargate (Fargate) 節點群組。

如需詳細資訊，請參閱《Secrets Manager 使用者指南》中的在 Amazon EKS 中使用 AWS Secrets Manager 秘密。

所有 Kubernetes API 資料的預設信封加密

Amazon Elastic Kubernetes Service (Amazon EKS) 為執行 Kubernetes 1.28 版或更新版本的 EKS 叢集中的所有 Kubernetes API 資料提供預設信封加密。

信封加密可保護您使用 Kubernetes API 伺服器存放的資料。例如，信封加密適用於 Kubernetes 叢集的組態，例如 `ConfigMaps`。信封加密不適用於節點或 EBS 磁碟區上的資料。EKS 先前支援加密 Kubernetes 秘密，現在此信封加密會延伸至所有 Kubernetes API 資料。

這可提供受管的預設體驗，為您的 Kubernetes 應用程式實作defense-in-depth，而且不需要您採取任何動作。

Amazon EKS 使用 AWS Key Management Service (KMS) 搭配 Kubernetes KMS 提供者 v2 搭配 Amazon Web Services 擁有的金鑰，提供這一層額外的安全保護，以及讓您從 AWS KMS 自攜客戶受管金鑰 (CMK) 的選項。

了解信封加密

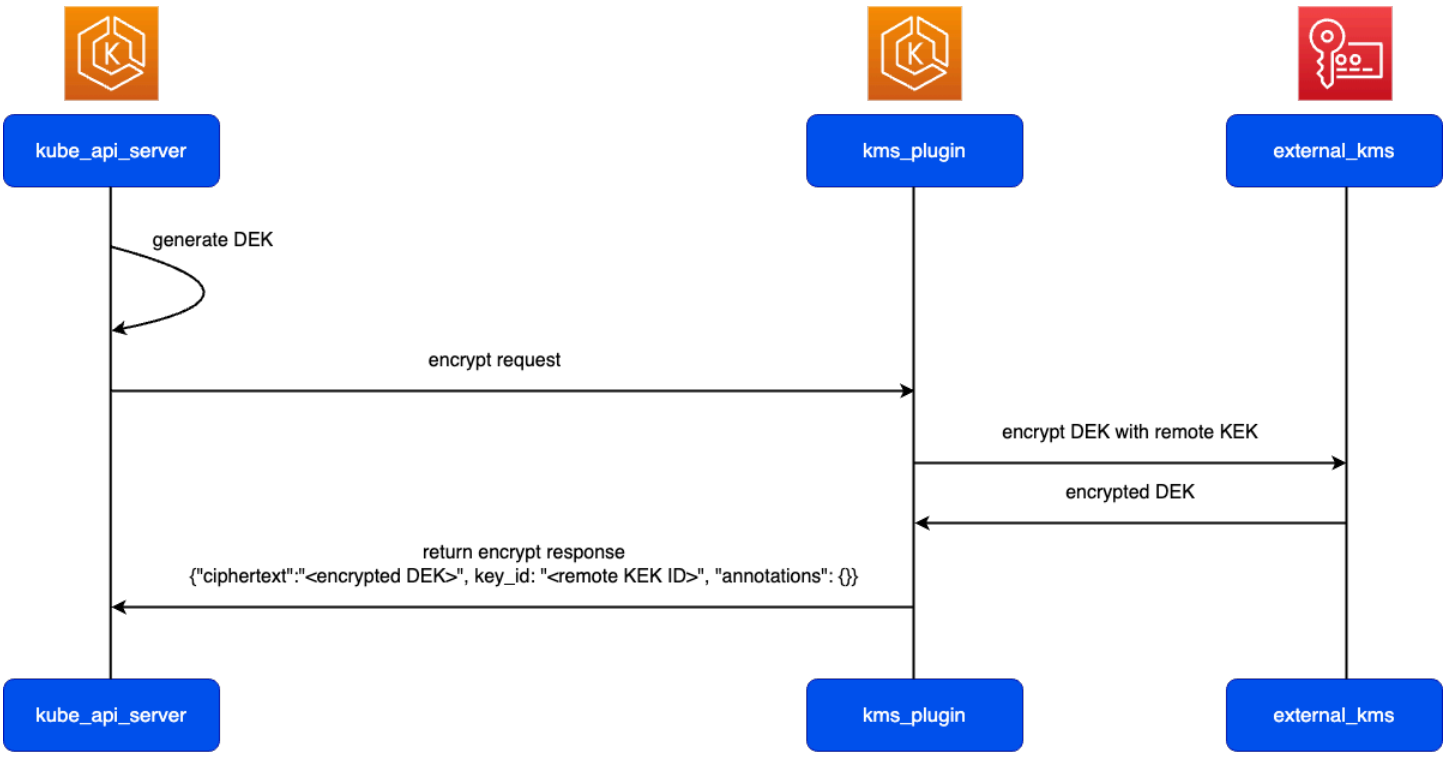
信封加密是使用資料加密金鑰 (DEK) 加密純文字資料的程序，然後再傳送到資料存放區 （ 等 ），然後使用存放在遠端、集中受管 KMS 系統 (AWS KMS) 中的根 KMS 金鑰加密 DEK。這是defense-in-depth策略，因為它會使用加密金鑰 (DEK) 保護資料，然後使用稱為金鑰加密金鑰 (KEK) 的個別安全儲存加密金鑰來保護該 DEK，以新增另一個安全層。

Amazon EKS 如何使用 KMS v2 和 AWS KMS 啟用預設信封加密

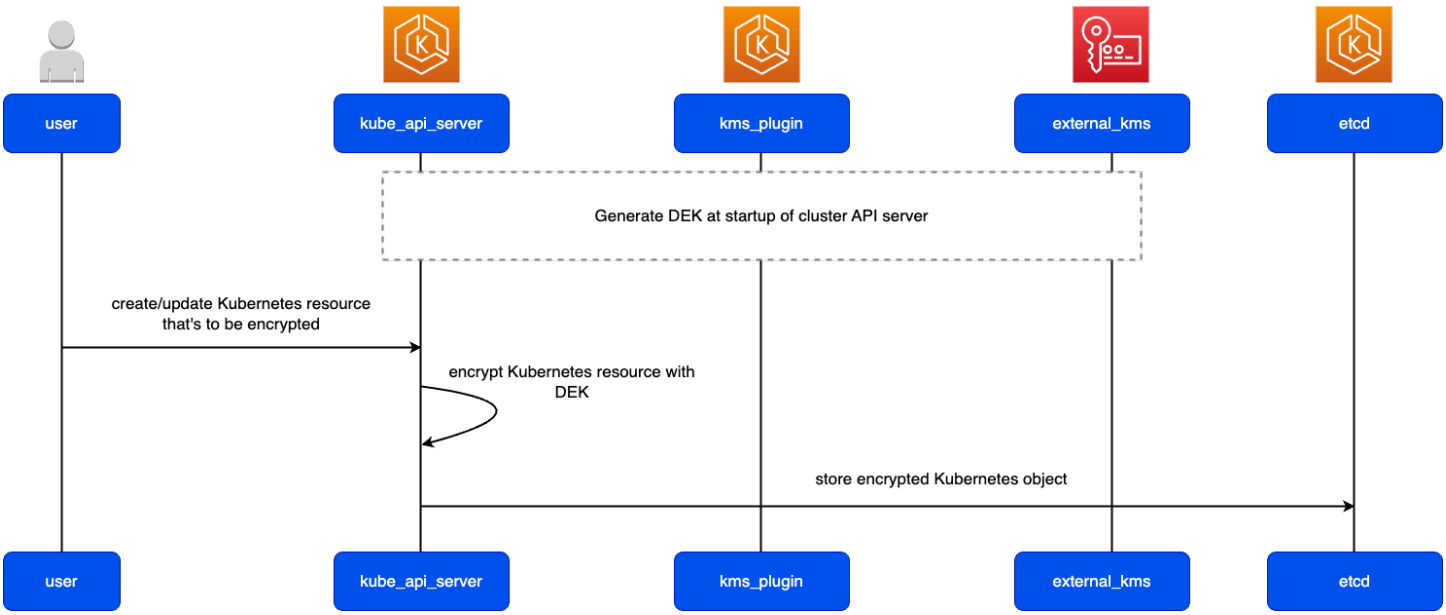
Amazon EKS 使用 KMS v2 來實作受管 Kubernetes 控制平面中所有 API 資料的預設信封加密，然後再保留在已加密的資料庫中。啟動時，叢集 API 伺服器會從秘密種子產生資料加密金鑰 (DEK)，並結合隨機產生的資料。此外，在啟動時，API 伺服器會呼叫 KMS 外掛程式，使用來自 KMS 的遠端金鑰加密金鑰 (KEK) 來加密 AWS DEK 種子。這是在 API 伺服器啟動和 KEK 輪換時執行的一次性呼叫。API 伺服器接著會快取加密的 DEK 種子。之後，API 伺服器會使用快取的 DEK 種子，根據金鑰衍生函數 (KDF) 產生其他單次使用 DEKs。然後，這些產生的每個 DEKs都只會用來加密單一 Kubernetes 資源一次，然後再存放在 等。在 KMS v2 中使用加密的快取 DEK 種子，在 API 伺服器上加密 Kubernetes 資源的程序不僅效能更高，而且更具成本效益。

根據預設，此 KEK 由 擁有 AWS，但您可以選擇從 AWS KMS 自攜。

下圖說明在 API 伺服器啟動時產生和加密 DEK。



以下高階圖表說明 Kubernetes 資源在存放於 等環境之前加密的方式。



常見問答集

預設信封加密如何改善 EKS 叢集的安全狀態？

此功能可減少未加密中繼資料和客戶內容的表面積和期間。使用預設信封加密時，中繼資料和客戶內容只會在 kube-apiserver 的記憶體中處於暫時未加密狀態，然後再存放在 等。kube-apiserver 的記憶體

是透過 Nitro 系統進行保護。Amazon EKS 僅針對受管 Kubernetes 控制平面使用 Nitro 型 EC2 執行個體。這些執行個體具有安全控制設計，可防止任何系統或人員存取其記憶體。

我需要執行哪個版本的 Kubernetes，才能擁有此功能？

若要啟用預設信封加密，Amazon EKS 叢集必須執行 Kubernetes 1.28 版或更新版本。

如果我執行的是不支援此功能的 Kubernetes 叢集版本，我的資料是否仍然安全？

是。 AWS安全是我們的最高優先順序。我們以最高安全營運實務為基礎進行所有數位轉型和創新，並致力於提高該標準。

無論執行的 Kubernetes 版本為何，儲存在 等中的所有資料都會在每個 EKS 叢集的磁碟層級加密。EKS 使用根金鑰來產生由 EKS 服務管理的磁碟區加密金鑰。此外，每個 Amazon EKS 叢集都是使用叢集特定的虛擬機器在隔離的 VPC 中執行。由於此架構，以及我們對營運安全性的實務，Amazon EKS 已達成多個合規評分和標準，包括 SOC 1、2、3、PCI-DSS、ISO 和 HIPAA 資格。這些合規評分和標準會針對所有具有或沒有預設信封加密的 EKS 叢集進行維護。

信封加密如何在 Amazon EKS 中運作？

啟動時，叢集 API 伺服器會從秘密種子產生資料加密金鑰 (DEK)，並結合隨機產生的資料。此外，在啟動時，API 伺服器會呼叫 KMS 外掛程式，以使用來自 KMS 的遠端金鑰加密金鑰 (KEK) 來加密 AWS DEK。這是在 API 伺服器啟動和 KEK 輪換時執行的一次性呼叫。API 伺服器接著會快取加密的 DEK 種子。之後，API 伺服器會使用快取的 DEK 種子，根據金鑰衍生函數 (KDF) 產生其他單次使用 DEKs。然後，這些產生的每個 DEKs都只會用來加密單一 Kubernetes 資源一次，然後再存放在 等。

請務必注意，API 伺服器還有其他呼叫，以驗證 AWS KMS 整合的運作狀態和正常功能。這些額外的運作狀態檢查會顯示在您的 AWS CloudTrail 中。

我是否必須執行任何動作或變更此功能的任何許可，才能在 EKS 叢集中運作？

否，您不需要採取任何動作。Amazon EKS 中的信封加密現在是在執行 Kubernetes 1.28 版或更新版本的所有叢集中啟用的預設組態。 AWS KMS 整合是由 管理的 Kubernetes API 伺服器建立 AWS。這表示您不需要設定任何許可，即可開始為叢集使用 KMS 加密。

如何知道我的叢集是否已啟用預設信封加密？

如果您遷移至使用自己的 CMK，則會看到與叢集相關聯的 KMS 金鑰 ARN。此外，您可以檢視與使用叢集的 CMK 相關聯的 AWS CloudTrail 事件日誌。

如果您的叢集使用 AWS 擁有的金鑰，則會在 EKS 主控台中詳細說明 （金鑰的 ARN 除外）。

是否可以 AWS 存取 Amazon EKS 中用於預設信封加密的 AWS 擁有金鑰？

否。Amazon EKS 中 AWS 具有嚴格的安全控制，可防止任何人存取用於保護已加密資料庫中資料的任何純文字加密金鑰。這些安全措施也會套用至 AWS 擁有的 KMS 金鑰。

我現有的 EKS 叢集是否已啟用預設信封加密？

如果您使用 Kubernetes 1.28 版或更新版本執行 Amazon EKS 叢集，則會啟用所有 Kubernetes API 資料的信封加密。對於現有的叢集，Amazon EKS 使用 `eks:kms-storage-migrator` RBAC ClusterRole 將先前未以 等式加密的資料遷移至此新加密狀態。

如果我已在 EKS 叢集中啟用秘密的信封加密，這代表什麼意義？

如果您在 KMS 中已有用於對 Kubernetes 秘密進行信封加密的客戶受管金鑰 (CMK)，則該金鑰將用作叢集中所有 Kubernetes API 資料類型的信封加密 KEK。

使用預設信封加密執行 EKS 叢集是否有任何額外費用？

如果您使用 Amazon Web Services 擁有的金鑰進行預設信封加密，則無需支付與受管 Kubernetes 控制平面相關的額外費用。根據預設，每個執行 Kubernetes 1.28 版或更新版本的 EKS 叢集都會使用 Amazon Web Service 擁有的金鑰。不過，如果您使用自己的 AWS KMS 金鑰，則會套用一般 KMS 定價。

使用我自己的 AWS KMS 金鑰來加密叢集中的 Kubernetes API 資料需要多少成本？

您每月需支付 1 USD 來存放您建立或匯入 KMS 的任何自訂金鑰。加密和解密請求的 KMS 費用。每個帳戶每月有 20，000 個請求的免費方案，每月每 10，000 個請求支付 0.03 USD。這適用於帳戶的所有 KMS 用量，因此在叢集上使用您自己的 AWS KMS 金鑰的成本將受到您帳戶中其他叢集或 AWS 資源使用此金鑰的影響。

現在我的客戶受管金鑰 (CMK) 用於信封加密所有 Kubernetes API 資料，而不只是秘密，我的 KMS 費用是否會更高？

否。我們使用 KMS v2 實作可大幅減少對 AWS KMS 發出的呼叫次數。無論 EKS 叢集中的其他 Kubernetes 資料是加密或解密，這都會降低與您的 CMK 相關的成本。

如上所述，用於加密 Kubernetes 資源的產生 DEK 種子會在使用遠端 KEK 加密之後，儲存在本機的 Kubernetes API 伺服器快取中。如果加密的 DEK 種子不在 API 伺服器的快取中，API 伺服器將呼叫 AWS KMS 來加密 DEK 種子。然後，API 伺服器會快取加密的 DEK 種子，以供未來在叢集中使用，而無需呼叫 KMS。同樣地，對於解密請求，API 伺服器將呼叫 AWS KMS 進行第一個解密請求，之後將快取解密的 DEK 種子並用於未來的解密操作。

如需詳細資訊，請參閱 GitHub 上的 Kubernetes 增強功能中的 KEP-3299：KMS v2 改進。

我可以將相同的 CMK 金鑰用於多個 Amazon EKS 叢集嗎？

是。若要再次使用金鑰，您可以在建立期間將 ARN 與叢集建立關聯，以將其連結至相同區域中的叢集。不過，如果您針對多個 EKS 叢集使用相同的 CMK，您應該採取必要的措施，以防止 CMK 的任意停用。否則，與多個 EKS 叢集相關聯的停用 CMK 將對叢集產生更廣泛的影響，具體取決於該金鑰。

如果啟用預設信封加密後我的 CMK 無法使用，我的 EKS 叢集會發生什麼情況？

如果您停用 KMS 金鑰，則無法用於任何密碼編譯操作。如果無法存取現有的 CMK，API 伺服器將無法加密和保留任何新建立的 Kubernetes 物件，以及解密存放在 等項目中先前加密的任何 Kubernetes 物件。如果停用 CMK，叢集將立即處於運作狀態不佳/降級狀態，此時我們將無法履行服務承諾，直到您重新啟用相關聯的 CMK 為止。

停用 CMK 時，您將收到有關 EKS 叢集運作狀態降低的通知，以及在停用 CMK 後 30 天內需要重新啟用 CMK，以確保成功還原 Kubernetes 控制平面資源。

如何保護 EKS 叢集免受已停用/已刪除 CMK 的影響？

為了保護您的 EKS 叢集免於發生這種情況，您的金鑰管理員應使用最低權限原則的 IAM 政策來管理對 KMS 金鑰操作的存取，以降低任何與 EKS 叢集相關聯的任意停用或刪除金鑰的風險。此外，您可以設定 CloudWatch 警示，以接收 CMK 狀態的通知。

如果我重新啟用 CMK，是否會還原我的 EKS 叢集？

為了確保成功還原 EKS 叢集，強烈建議在停用 CMK 後的前 30 天內重新啟用 CMK。不過，成功還原 EKS 叢集也會取決於它是否因為叢集處於運作狀態不佳/降級狀態時可能發生的自動 Kubernetes 升級而經歷任何 API 中斷變更。

停用 CMK 後，為什麼我的 EKS 叢集處於運作狀態不佳/降級狀態？

EKS 控制平面的 API 伺服器使用 DEK 金鑰，該金鑰會加密並快取在 API 伺服器的記憶體中，以在建立/更新操作期間加密所有物件，然後再存放在 等。從 等推擷取現有物件時，API 伺服器會使用相同的快取 DEK 金鑰並解密 Kubernetes 資源物件。如果您停用 CMK，API 伺服器將不會因為 API 伺服器記憶體中的快取 DEK 金鑰而看到任何立即影響。不過，當 API 伺服器執行個體重新啟動時，它不會有快取的 DEK，而且需要呼叫 AWS KMS 以進行加密和解密操作。如果沒有 CMK，此程序將會失敗，並顯示 KMS_KEY_DISABLED 錯誤代碼，導致 API 伺服器無法成功開機。

如果我刪除 CMK，我的 EKS 叢集會發生什麼情況？

刪除與 EKS 叢集相關聯的 CMK 金鑰，將會降低其復原後的運作狀態。如果沒有叢集的 CMK，API 伺服器將無法再加密和保留任何新的 Kubernetes 物件，也無法解密存放在已加密資料庫中的任何先前加

密的 Kubernetes 物件。只有在您確定不再需要使用 EKS 叢集時，才應該繼續刪除 EKS 叢集的 CMK 金鑰。

請注意，如果找不到 CMK (KMS_KEY_NOT_FOUND) 或已撤銷與叢集相關聯的 CMK 授予 (KMS_GRANT_REVOKED)，您的叢集將無法復原。如需叢集運作狀態和錯誤代碼的詳細資訊，請參閱叢集運作狀態FAQs和具有解析路徑的錯誤代碼。

我是否仍會因為停用或刪除 CMK 而需要為降級/運作狀態不佳的 EKS 叢集付費？

是。雖然在停用 CMK 的情況下，EKS 控制平面將無法使用，但 AWS 仍會執行配置給 EKS 叢集的專用基礎設施資源，直到客戶將其刪除為止。此外，在這種情況下，我們的服務承諾將不適用，因為它將是客戶自願採取或不採取的動作，以防止您的 EKS 叢集正常運作。

當 EKS 叢集因為停用 CMK 而處於運作狀態不佳/降級狀態時，是否可以自動升級？

是。不過，如果您的叢集有已停用的 CMK，您將有 30 天的期間可重新啟用它。在這 30 天期間，不會自動升級您的 Kubernetes 叢集。不過，如果超過此期間，而且您尚未重新啟用 CMK，則叢集將依照 EKS 中的 Kubernetes 版本生命週期，自動升級至標準支援的下一個版本 (n+1)。

當您發現受影響的叢集時，強烈建議快速重新啟用已停用的 CMK。請務必注意，雖然 EKS 會自動升級這些受影響的叢集，但無法保證它們會成功復原，尤其是如果叢集經歷多次自動升級，因為這可能包括 Kubernetes API 的變更，以及 API 伺服器引導程序中的意外行為。

我可以使用 KMS 金鑰別名嗎？

是。Amazon EKS 支援使用 KMS 金鑰別名。別名是 Amazon Web Service KMS 金鑰的易記名稱。例如，別名可讓您將 KMS 金鑰稱為 my-key，而不是 **1234abcd-12ab-34cd-56ef-1234567890ab**。

我是否仍然可以使用自己的 Kubernetes 備份解決方案來備份和還原叢集資源？

是。您可以使用 Kubernetes 備份解決方案（例如 Velero) 進行 Kubernetes 叢集災難復原、資料遷移和資料保護。如果您執行透過 API 伺服器存取叢集資源的 Kubernetes 備份解決方案，應用程式擷取的任何資料都會在到達用戶端之前解密。這可讓您復原另一個 Kubernetes 叢集中的叢集資源。

Amazon EKS Auto Mode 的安全考量

本主題說明 Amazon EKS Auto Mode 的安全架構、控制項和最佳實務。隨著組織大規模部署容器化應用程式，維護強大的安全狀態變得越來越複雜。EKS Auto Mode 實作自動化安全控制，並與 AWS 安全服務整合，以協助您保護叢集基礎設施、工作負載和資料。透過強制執行節點生命週期管理和自動修補程式部署等內建安全功能，EKS Auto Mode 可協助您維護安全最佳實務，同時降低營運開銷。

繼續本主題之前，請確定您已熟悉基本的 EKS Auto Mode 概念，並已檢閱在叢集上啟用 EKS Auto Mode 的先決條件。如需 Amazon EKS 安全性的一般資訊，請參閱 安全。

Amazon EKS Auto Mode 以 Amazon EKS 的現有安全基礎為基礎，同時為 EC2 受管執行個體引入額外的自動化安全控制。

API 安全性和身分驗證

Amazon EKS Auto Mode 使用 AWS 平台安全機制來保護和驗證對 Amazon EKS API 的呼叫。

- 對 Kubernetes API 的存取是透過 EKS 存取項目進行保護，這些項目與 AWS IAM 身分整合。
  - 如需詳細資訊，請參閱the section called “授予許可”。
- 客戶可以透過 EKS 存取項目的組態，對 Kubernetes API 端點實作精細的存取控制。

網路安全

Amazon EKS Auto Mode 支援多層網路安全：

- VPC 整合
  - 在您的 Amazon Virtual Private Cloud (VPC) 中操作
  - 支援自訂 VPC 組態和子網路配置
  - 啟用叢集元件之間的私有聯網
  - 如需詳細資訊，請參閱管理 Amazon Virtual Private Cloud 的安全責任
- 網路政策
  - Kubernetes 網路政策的原生支援
  - 定義精細網路流量規則的能力
  - 如需詳細資訊，請參閱 the section called “Kubernetes 政策”

EC2 受管執行個體安全性

Amazon EKS Auto Mode 使用下列安全控制操作 EC2 受管執行個體：

EC2 安全性

- EC2 受管執行個體維護 Amazon EC2 的安全功能。
- 如需 EC2 受管執行個體的詳細資訊，請參閱 Amazon EC2 中的安全性。

執行個體生命週期管理

由 EKS Auto Mode 操作的 EC2 受管執行個體的生命週期上限為 21 天。Amazon EKS Auto Mode 會自動終止超過此生命週期的執行個體。此生命週期限制有助於防止組態偏離並維持安全狀態。

資料保護

- Amazon EC2 執行個體儲存體已加密，這是直接連接到執行個體的儲存體。如需詳細資訊，請參閱 Amazon EC2 中的資料保護。
- EKS Auto Mode 會在建立時管理連接至 EC2 執行個體的磁碟區，包括根磁碟區和資料磁碟區。EKS Auto Mode 不會完全管理使用 Kubernetes 持久性儲存功能建立的 EBS 磁碟區。

修補管理

- Amazon EKS Auto Mode 會自動將修補程式套用至受管執行個體。
- 修補程式包括：
  - 作業系統更新
  - 安全性修補程式
  - Amazon EKS Auto Mode 元件

> **Note**
>
> 客戶仍需負責保護和更新在這些執行個體上執行的工作負載。

存取控制

- 直接執行個體存取受到限制：
  - 無法使用 SSH 存取。
  - AWS 無法使用 Systems Manager Session Manager (SSM) 存取。
- 透過 Amazon EKS API 和 Kubernetes API 執行管理操作。

自動化資源管理

Amazon EKS Auto Mode 不會完整管理使用 Kubernetes 持久性儲存功能建立的 Amazon Elastic Block Store (Amazon EBS) 磁碟區。EKS Auto Mode 也不會管理 Elastic Load Balancer (ELB)。Amazon EKS Auto Mode 會自動執行這些資源的例行任務。

儲存安全

- AWS 建議您為 Kubernetes 持久性儲存功能佈建的 EBS 磁碟區啟用加密。如需詳細資訊，請參閱the section called “建立 StorageClass”。
- 使用 AWS KMS 進行靜態加密
- 您可以設定 AWS 帳戶，強制加密您建立的新 EBS 磁碟區和快照複本。如需詳細資訊，請參閱《Amazon EBS 使用者指南》中的預設啟用 Amazon EBS 加密。
- 如需詳細資訊，請參閱 Amazon EBS 中的安全性。

負載平衡器安全性

- Elastic Load Balancer 的自動化組態
- 透過 AWS Certificate Manager 整合進行 SSL/TLS 憑證管理
- 負載平衡器存取控制的安全群組自動化
- 如需詳細資訊，請參閱 Elastic Load Balancing 中的安全性。

安全最佳實務

下節說明 Amazon EKS Auto Mode 的安全最佳實務。

- 定期檢閱 AWS IAM 政策和 EKS 存取項目。
- 實作工作負載的最低權限存取模式。
- 透過 AWS CloudTrail 和 Amazon CloudWatch 監控叢集活動。如需詳細資訊，請參閱 the section called “ AWS CloudTrail” 和 the section called “Amazon CloudWatch”。
- 使用 AWS Security Hub 進行安全狀態評估。
- 實作適合您工作負載的 Pod 安全標準。

Amazon EKS 的 Identity and Access Management

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 管理員可以控制驗證 (已登入) 和授權 (具有許可) 來使用 Amazon EKS 資源。IAM 是一項服務 AWS ，您可以免費使用。

目標對象

使用 AWS Identity and Access Management (IAM) 的方式會有所不同，取決於您在 Amazon EKS 中執行的工作。

服務使用者 – 如果您使用 Amazon EKS 執行任務，您的管理員會為您提供您需要的憑證和許可。隨著您為了執行作業而使用的 Amazon EKS 功能數量變多，您可能會需要額外的許可。了解存取許可的管理方式可協助您向管理員請求正確的許可。如果您無法存取 Amazon EKS 中的功能，請參閱 the section called “故障診斷”。

服務管理員 – 如果您在公司負責管理 Amazon EKS 資源，您可能擁有 Amazon EKS 的完整存取權。您的任務是判斷服務使用者應存取的 Amazon EKS 功能和資源。接著，您必須將請求提交給您的 IAM 管理員，來變更您服務使用者的許可。檢閱此頁面上的資訊，了解 IAM 的基本概念。若要進一步了解貴公司可搭配 Amazon EKS 使用 IAM 的方式，請參閱 the section called “Amazon EKS 和 IAM”。

IAM 管理員 – 如果您是 IAM 管理員，建議您了解如何撰寫政策以管理 Amazon EKS 存取權的詳細資訊。若要檢視您可以在 IAM 中使用 Amazon EKS 身分型政策範例，請參閱 the section called “身分型政策”。

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 （ 登入 AWS)。

您可以使用透過身分來源提供的登入資料，以聯合身分 AWS 身分身分身分登入。 AWS IAM Identity Center (IAM Identity Center) 使用者、您公司的單一登入身分驗證，以及您的 Google 或 Facebook 登入資料，都是聯合身分的範例。您以聯合身分登入時，您的管理員先前已設定使用 IAM 角色的聯合身分。當您使用聯合 AWS 身分存取 時，您會間接擔任角色。

根據您身分的使用者類型，您可以登入 AWS Management Console 或 AWS 存取入口網站。如需登入的詳細資訊 AWS，請參閱 AWS 登入使用者指南中的如何登入 AWS 您的帳戶。

如果您以 AWS 程式設計方式存取 ， AWS 會提供軟體開發套件 (SDK) 和命令列界面 (CLI)，以使用您的 憑證以密碼編譯方式簽署您的請求。如果您不使用 AWS 工具，則必須自行簽署請求。如需使用建議方法自行簽署請求的詳細資訊，請參閱《IAM 使用者指南》中的簽署 AWS API 請求。

無論您使用何種身分驗證方法，您可能都需要提供額外的安全性資訊。例如， AWS 建議您使用多重要素驗證 (MFA) 來提高帳戶的安全性。若要進一步了解，請參閱《 AWS IAM Identity Center 使用者指南》中的多重要素驗證和《IAM 使用者指南》中的在 中使用多重要素驗證 (MFA) AWS。

AWS 帳戶根使用者

建立 AWS 帳戶時，您會從一個登入身分開始，該身分可完整存取帳戶中的所有 AWS 服務和資源。此身分稱為 AWS 帳戶根使用者，可透過使用您用來建立帳戶的電子郵件地址和密碼登入來存取。強烈建議您不要以根使用者處理日常作業。保護您的根使用者憑證，並將其用來執行只能由根使用者執行的任務。如需這些任務的完整清單，了解需以根使用者登入的任務，請參閱《IAM 使用者指南》中的需要根使用者憑證的任務。

IAM 使用者和群組

IAM 使用者是您 AWS 帳戶中的身分，具有單一人員或應用程式的特定許可。建議您盡可能依賴臨時憑證，而不是擁有建立長期憑證 (例如密碼和存取金鑰) 的 IAM 使用者。但是如果特定使用案例需要擁有長期憑證的 IAM 使用者，建議您輪換存取金鑰。如需更多資訊，請參閱 IAM 使用者指南中的為需要長期憑證的使用案例定期輪換存取金鑰。

IAM 群組是一種指定 IAM 使用者集合的身分。您無法以群組身分登入。您可以使用群組來一次為多名使用者指定許可。群組可讓管理大量使用者許可的程序變得更為容易。例如，您可以擁有一個名為 IAMAdmins 的群組，並給予該群組管理 IAM 資源的許可。

使用者與角色不同。使用者只會與單一人員或應用程式建立關聯，但角色的目的是在由任何需要它的人員取得。使用者擁有永久的長期憑證，但角色僅提供暫時憑證。如需進一步了解，請參閱IAM 使用者指南中的建立 IAM 使用者 (而非角色) 的時機。

IAM 角色

IAM 角色是您 AWS 帳戶中具有特定許可的身分。它類似 IAM 使用者，但不與特定的人員相關聯。您可以透過切換角色暫時在 中擔任 IAM AWS Management Console 角色。 https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles_use_switch-role-console.html您可以透過呼叫 CLI AWS 或 AWS API 操作或使用自訂 URL 來擔任角色。如需使用角色的方法詳細資訊，請參閱 IAM 使用者指南中的使用 IAM 角色。

使用暫時憑證的 IAM 角色在下列情況中非常有用：

- 聯合身分使用者存取 — 如需向聯合身分指派許可，請建立角色，並為角色定義許可。當聯合身分進行身分驗證時，該身分會與角色建立關聯，並獲授予由角色定義的許可。如需有關聯合角色的相關資訊，請參閱 IAM 使用者指南中的為第三方身分提供者建立角色。如果您使用 IAM Identity Center，則需要設定許可集。為控制身分驗證後可以存取的內容，IAM Identity Center 將許可集與 IAM 中的角色相關聯。如需許可集的資訊，請參閱《 AWS IAM Identity Center 使用者指南》中的許可集。
- 暫時 IAM 使用者許可 – IAM 使用者或角色可以擔任 IAM 角色來暫時針對特定任務採用不同的許可。
- 跨帳戶存取權：您可以使用 IAM 角色，允許不同帳戶中的某人 (信任的主體) 存取您帳戶的資源。角色是授予跨帳戶存取權的主要方式。不過，對於某些 AWS 服務，您可以將政策直接連接到資源（而不是使用角色做為代理）。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。
- 跨服務存取 – 有些 AWS 服務使用其他服務中的功能 AWS 。例如，當您在服務中呼叫 時，該服務通常會在 Amazon EC2 中執行應用程式，或在 Amazon S3 中存放物件。服務可能會使用呼叫主體的許可、使用服務角色或使用服務連結角色來執行此作業。
  - 轉送存取工作階段 (FAS) – 當您使用 IAM 使用者或角色在其中執行動作時 AWS，您被視為委託人。使用某些服務時，您可能會執行某個動作，進而在不同服務中啟動另一個動作。FAS 使用呼叫 AWS 服務的委託人許可，結合請求 AWS 服務，向下游服務提出請求。只有當服務收到需要與其他 AWS 服務或資源互動才能完成的請求時，才會提出 FAS 請求。在此情況下，您必須具有執行這兩個動作的許可。如需提出 FAS 請求時的政策詳細資訊，請參閱《轉發存取工作階段》。
  - 服務角色 – 服務角色是服務擔任的 IAM 角色，可代表您執行動作。IAM 管理員可以從 IAM 內建立、修改和刪除服務角色。如需詳細資訊，請參閱《IAM 使用者指南》中的建立角色以將許可委派給 AWS 服務。
  - 服務連結角色 – 服務連結角色是連結至 服務的一種 AWS 服務角色。服務可以擔任代表您執行動作的角色。服務連結角色會顯示在您的帳戶中 AWS ，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。
- 在 Amazon EC2 上執行的應用程式 – 您可以使用 IAM 角色來管理在 EC2 執行個體上執行之應用程式的臨時登入資料，以及提出 AWS CLI 或 AWS API 請求。這是在 EC2 執行個體內儲存存取金鑰的較好方式。若要將 AWS 角色指派給 EC2 執行個體，並將其提供給其所有應用程式，您可以建立連接至執行個體的執行個體描述檔。執行個體設定檔包含該角色，並且可讓 EC2 執行個體上執行的程式取得暫時憑證。如需詳細資訊，請參閱 IAM 使用者指南中的利用 IAM 角色來授予許可給 Amazon EC2 執行個體上執行的應用程式。

如需了解是否要使用 IAM 角色或 IAM 使用者，請參閱 IAM 使用者指南中的建立 IAM 角色 (而非使用者) 的時機。

使用政策管理存取權

您可以透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。政策是 中的物件， AWS 當與身分或資源相關聯時，會定義其許可。當委託人（使用者、根使用者或角色工作階段）發出請求時， 會 AWS 評估這些政策。政策中的許可決定是否允許或拒絕請求。大多數政策會以 JSON 文件 AWS 形式存放在 中。如需 JSON 政策文件結構和內容的詳細資訊，請參閱 IAM 使用者指南中的 JSON 政策概觀。

管理員可以使用 AWS JSON 政策來指定誰可以存取內容。也就是說，哪個主體在什麼條件下可以對什麼資源執行哪些動作。

預設情況下，使用者和角色沒有許可。若要授予使用者對其所需資源執行動作的許可，IAM 管理員可以建立 IAM 政策。然後，管理員可以將 IAM 政策新增至角色，使用者便能擔任這些角色。

IAM 政策定義該動作的許可，無論您使用何種方法來執行操作。例如，假設您有一個允許 `iam:GetRole` 動作的政策。具有該政策的使用者可以從 AWS Management Console、 CLI 或 AWS API AWS 取得角色資訊。

身分型政策

身分型政策是可以附加到身分 (例如 IAM 使用者、使用者群組或角色) 的 JSON 許可政策文件。這些政策可控制身分在何種條件下能對哪些資源執行哪些動作。若要了解如何建立身分類型政策，請參閱 IAM 使用者指南中的建立 IAM 政策。

身分型政策可進一步分類成內嵌政策或受管政策。內嵌政策會直接內嵌到單一使用者、群組或角色。受管政策是獨立的政策，您可以連接到 AWS 帳戶中的多個使用者、群組和角色。受管政策包括 AWS 受管政策和客戶受管政策。如需了解如何在受管政策及內嵌政策間選擇，請參閱 IAM 使用者指南中的在受管政策和內嵌政策間選擇。

資源型政策

資源型政策是連接到資源的 JSON 政策文件。資源型政策的最常見範例是 IAM 角色信任政策和 Amazon S3 儲存貯體政策。在支援資源型政策的服務中，服務管理員可以使用它們來控制對特定資源的存取權限。對於附加政策的資源，政策會定義指定的主體可以對該資源執行的動作以及在何種條件下執行的動作。您必須在資源型政策中指定主體。委託人可以包括帳戶、使用者、角色、聯合身分使用者 AWS 或服務。

資源型政策是位於該服務中的內嵌政策。您無法在以資源為基礎的政策中使用來自 IAM 的 AWS 受管政策。

存取控制清單 (ACL)

存取控制清單 (ACL) 可控制哪些主體 (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。

Amazon S3、 AWS WAF 和 Amazon VPC 是支援 ACLs 的服務範例。如需進一步了解 ACL，請參閱 Amazon Simple Storage Service 開發人員指南中的存取控制清單 (ACL) 概觀。

其他政策類型

AWS 支援其他較不常見的政策類型。這些政策類型可設定較常見政策類型授予您的最大許可。

- 許可界限 – 許可範圍是一種進階功能，可供您設定身分型政策能授予 IAM 實體 (IAM 使用者或角色) 的最大許可。您可以為實體設定許可界限。產生的許可是實體身分型政策及其許可界限的交集。會在 `Principal` 欄位中指定使用者或角色的資源型政策則不會受到許可界限限制。所有這類政策中的明確拒絕都會覆寫該允許。如需許可界限的詳細資訊，請參閱 IAM 使用者指南中的 IAM 實體許可界限。
- 服務控制政策 (SCPs) – SCPs是 JSON 政策，可指定 AWS Organizations 中組織或組織單位 (OU) 的最大許可。 AWS Organizations 是一項服務，用於分組和集中管理您企業擁有的多個 AWS 帳戶。若您啟用組織中的所有功能，您可以將服務控制政策 (SCP) 套用到任何或所有帳戶。SCP 會限制成員帳戶中實體的許可，包括每個 AWS 帳戶根使用者。如需 Organizations 和 SCPs的詳細資訊，請參閱 AWS 《 Organizations 使用者指南》中的服務控制政策。
- 工作階段政策 – 工作階段政策是一種進階政策，您可以在透過撰寫程式的方式建立角色或聯合使用者的暫時工作階段時，做為參數傳遞。所產生工作階段的許可會是使用者或角色的身分型政策和工作階段政策的交集。許可也可以來自資源型政策。所有這類政策中的明確拒絕都會覆寫該允許。如需詳細資訊，請參閱 IAM 使用者指南中的工作階段政策。

多種政策類型

將多種政策類型套用到請求時，其結果形成的許可會更為複雜、更加難以理解。若要了解如何在涉及多種政策類型時 AWS 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 EKS 如何搭配 IAM 運作

在您使用 IAM 管理對 Amazon EKS 的存取權之前，您應該了解哪些 IAM 功能可以與 Amazon EKS 搭配使用。若要全面了解 Amazon EKS 和其他 AWS 服務如何與 IAM 搭配使用，請參閱《IAM 使用者指南》中的AWS 與 IAM 搭配使用的 服務。

主題

- Amazon EKS 身分型政策
- Amazon EKS 資源型政策
- 以 Amazon EKS 標籤為基礎的授權
- Amazon EKS IAM 角色

Amazon EKS 身分型政策

使用 IAM 身分型政策，您可以指定允許或拒絕的動作和資源，以及在何種條件下允許或拒絕動作。Amazon EKS 支援特定動作、資源和條件索引鍵。若要了解您在 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。

動作

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。政策動作通常具有與相關聯 AWS API 操作相同的名稱。有一些例外狀況，例如沒有相符 API 操作的僅限許可動作。也有一些作業需要政策中的多個動作。這些額外的動作稱為相依動作。

政策會使用動作來授予執行相關聯動作的許可。

Amazon EKS 中的政策動作會在動作之前使用下列字首：`eks:`。例如，若要授予某人取得 Amazon EKS 叢集描述性資訊的許可，請在其政策中加入 `DescribeCluster` 動作。政策陳述式必須包含 `Action` 或 `NotAction` 元素。

若要在單一陳述式中指定多個動作，請用逗號分隔，如下所示：

```
"Action": ["eks:action1", "eks: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": "eks:Describe*"
```

若要查看 Amazon EKS 動作的清單，請參閱服務授權參考中的 Amazon Elastic Kubernetes Service 定義的動作。

資源

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。陳述式必須包含 Resource 或 NotResource 元素。最佳實務是使用其 Amazon Resource Name (ARN) 來指定資源。您可以針對支援特定資源類型的動作 (稱為資源層級許可) 來這麼做。

對於不支援資源層級許可的動作，例如列出操作，請使用萬用字元 (*) 來表示陳述式適用於所有資源。

```
"Resource": "*"
```

Amazon EKS 叢集資源有以下 ARN。

```
 arn:aws: eks:region-code:account-id:cluster/cluster-name
```

如需 ARNs 格式的詳細資訊，請參閱 Amazon 資源名稱 (ARNs) AWS 和服務命名空間。

例如，若要在您的陳述式中指定具有 *my-cluster* 名稱的叢集，請使用以下 ARN：

```
"Resource": "arn:aws: eks:region-code:111122223333:cluster/my-cluster"
```

若要指定屬於特定帳戶和 AWS 區域的所有叢集，請使用萬用字元 (*)：

```
"Resource": "arn:aws: eks:region-code:111122223333:cluster/*"
```

有些 Amazon EKS 動作無法在特定資源上執行，例如用於建立資源的動作。在這些情況下，您必須使用萬用字元 (*)。

```
"Resource": "*"
```

若要查看 Amazon EKS 資源類型及其 ARN 的詳細資訊，請參閱服務授權參考中的 Amazon Elastic Kubernetes Service 定義的資源。若要了解您可以使用哪些動作指定每個資源的 ARN，請參閱 Amazon Elastic Kubernetes Service 定義的動作。

條件索引鍵

Amazon EKS 會定義自己的一組條件索引鍵，也支援使用一些全域條件金鑰。若要查看所有 AWS 全域條件金鑰，請參閱《IAM 使用者指南》中的AWS 全域條件內容金鑰。

將 OpenID Connect 提供商關聯到您的叢集時，您可以設定條件索引鍵。如需詳細資訊，請參閱the section called “範例 IAM 政策”。

所有 Amazon EC2 操作都支援 aws:RequestedRegion 和 ec2:Region 條件索引鍵。如需詳細資訊，請參閱範例：限制對特定 AWS 區域的存取。

如需 Amazon EKS 條件金鑰的清單，請參閱服務授權參考中的 Amazon Elastic Kubernetes Service 定義的條件。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 Kubernetes Service 定義的動作。

範例

若要檢視 Amazon EKS 身分型政策的範例，請參閱 the section called “身分型政策”。

當您建立 Amazon EKS 叢集時，會自動在 Amazon EKS 控制平面中叢集的角色型存取控制 (RBAC) 組態中授予建立叢集的 IAM 主體system:masters許可。此主體不會出現在任何可見的組態中，因此請務必追蹤最初建立叢集的主體。若要授予其他 IAM 主體與叢集互動的能力，請在 Kubernetes aws-auth ConfigMap中編輯 ，並使用group您在 中指定的 clusterrolebinding名稱建立 Kubernetes rolebinding或 aws-auth ConfigMap。

如需使用 ConfigMap 代理程式組態檔案的詳細資訊，請參閱 the section called “Kubernetes API 存取”。

Amazon EKS 資源型政策

Amazon EKS 不支援資源型政策。

以 Amazon EKS 標籤為基礎的授權

您可以將標籤連接至 Amazon EKS 資源，或是在請求中將標籤傳遞至 Amazon EKS。若要根據標籤控制存取，請使用 aws:ResourceTag/*key-name* 、aws:RequestTag/*key-name* 或 aws:TagKeys 條件索引鍵，在政策的條件元素中，提供標籤資訊。如需標記 Amazon EKS 資源的詳細資訊，請參閱 the section called “標記您的 資源”。如需您在條件索引鍵下可使用哪些標籤動作的詳細資訊，請參閱《服務授權參考》中的 Amazon EKS 定義的動作。

Amazon EKS IAM 角色

IAM 角色是您 AWS 帳戶中具有特定許可的實體。

將暫時憑證與 Amazon EKS 搭配使用

您可以搭配聯合使用暫時憑證、擔任 IAM 角色，或是擔任跨帳戶角色。您可以透過呼叫 AssumeRole 或 GetFederationToken 等 AWS STS API 操作來取得臨時安全登入資料。

Amazon EKS 支援使用臨時憑證。

服務連結角色

服務連結角色可讓 AWS 服務存取其他服務中的資源，以代表您完成 動作。服務連結角色會顯示在您的 IAM 帳戶中，並由該服務所擁有。管理員可以檢視，但無法編輯服務連結角色的許可。

Amazon EKS 支援服務連結角色。如需建立或管理 Amazon EKS 服務連結角色的詳細資訊，請參閱 the section called “服務連結角色”。

服務角色

此功能可讓服務代表您擔任服務角色。此角色可讓服務存取其他服務中的資源，以代表您完成動作。服務角色會出現在您的 IAM 帳戶中，且由該帳戶所擁有。這表示 IAM 管理員可以變更此角色的許可。不過，這樣可能會破壞此服務的功能。

Amazon EKS 支援服務角色。如需詳細資訊，請參閱the section called “叢集 IAM 角色”及the section called “節點 IAM 角色”。

在 Amazon EKS 中選擇 IAM 角色

當您在 Amazon EKS 中建立叢集資源時，您必須選擇角色，以允許 Amazon EKS 代表您存取數個其他 AWS 資源。如果您之前已建立服務角色，則 Amazon EKS 會提供一個角色清單供您選擇。請務必選擇已連接 Amazon EKS 受管政策的角色。如需詳細資訊，請參閱the section called “檢查現有的叢集角色”及the section called “檢查現有的節點角色”。

Amazon EKS 身分型政策範例

根據預設，IAM 使用者和角色沒有建立或修改 Amazon EKS 資源的許可。他們也無法使用 、 AWS CLI AWS Management Console或 AWS API 執行任務。IAM 管理員必須建立 IAM 政策，授予使用者和角色在指定資源上執行特定 API 作業的所需許可。管理員接著必須將這些政策連接至需要這些許可的 IAM 使用者或群組。

若要了解如何使用這些範例 JSON 政策文件建立 IAM 身分型政策，請參閱《IAM 使用者指南》中的在 JSON 標籤上建立政策。

當您建立 Amazon EKS 叢集時，會自動授予建立叢集的 IAM 主體在 Amazon EKS 控制平面中叢集的角色型存取控制 (RBAC) 組態中的`system:masters`許可。此主體不會出現在任何可見的組態中，因此請務必追蹤最初建立叢集的主體。若要授予其他 IAM 主體與叢集互動的能力，請在 Kubernetes `aws-auth ConfigMap`中編輯 ，並使用`group`您在 中指定的 `clusterrolebinding`名稱建立 Kubernetes `rolebinding`或 `aws-auth ConfigMap`。

如需使用 ConfigMap 代理程式組態檔案的詳細資訊，請參閱 the section called “Kubernetes API 存取”。

主題

- 政策最佳實務
- 使用 Amazon EKS 主控台
- 允許 IAM 使用者檢視他們自己的許可
- 在 AWS 雲端上建立 Kubernetes 叢集
- 在 Outpost 上建立本機 Kubernetes 叢集
- 更新 Kubernetes 叢集
- 所有叢集的清單或描述

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 EKS 資源。這些動作可能會對您的帳戶產生成本 AWS 。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 受管政策並邁向最低權限許可 – 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。它們可在您的帳戶中使用 AWS 。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 AWS CloudFormation 等特定服務使用服務動作，您也可以使用條件來授予存取 AWS 服務動作的權限。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。如需更多資訊，請參閱 IAM 使用者指南中的 IAM Access Analyzer 政策驗證。

- 需要多重要素驗證 (MFA) – 如果您的案例需要 IAM 使用者或 AWS 帳戶中的根使用者，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。如需更多資訊，請參閱 IAM 使用者指南中的設定 MFA 保護的 API 存取。

如需 IAM 中最佳實務的相關資訊，請參閱 IAM 使用者指南中的 IAM 安全最佳實務。

使用 Amazon EKS 主控台

若要存取 Amazon EKS 主控台，IAM 主體必須擁有最基本的一組許可。這些許可允許委託人列出和檢視您 AWS 帳戶中 Amazon EKS 資源的詳細資訊。如果您建立比最低必要許可更嚴格的身分型政策，則主控台對於附加該政策的主體將無法如預期運作。

為了確保您的 IAM 主體仍然可以使用 Amazon EKS 主控台，建立擁有唯一名稱的政策，例如 `AmazonEKSAdminPolicy`。將政策連接至主體。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。

> **Important**
>
> 下列範例政策可讓主體檢視主控台內的組態標籤上的資訊。若要檢視 中概觀和資源索引標籤的相關資訊 AWS Management Console，委託人也需要 Kubernetes 許可。如需詳細資訊，請參閱the section called “所需的許可”。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "eks:*"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "*",
```

```
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": "eks.amazonaws.com"
                }
            }
        }
    ]
}
```

對於僅呼叫 CLI 或 AWS API AWS 的主體，您不需要允許最低主控台許可。相反地，僅允許存取與您嘗試執行的 API 操作相符的動作。

允許 IAM 使用者檢視他們自己的許可

此範例會示範如何建立政策，允許 IAM 使用者檢視連接到他們使用者身分的內嵌及受管政策。此政策包含在主控台上完成此動作的許可，或使用 CLI 或 AWS API AWS 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": "*"
        }
    ]
}
```

在 AWS 雲端上建立 Kubernetes 叢集

此範例政策包含在 *us-west-2 ###### my-cluster* 的 Amazon EKS 叢集所需的最低許可。AWS 您可以將 AWS 區域取代為您要建立叢集 AWS 的區域。如果您看到警告，指出政策中的動作不支援資源層級許可，並要求您在 **All resources**中選擇 AWS Management Console，則可以安全地忽略它。如果您的帳戶已具有 *AWSServiceRoleForAmazonEKS* 角色，您可以移除來自政策的 `iam:CreateServiceLinkedRole` 動作。如果您已在帳戶中建立 Amazon EKS 叢集，則此角色已存在，除非您將其刪除。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "eks:CreateCluster",
            "Resource": "arn:aws: eks:us-west-2:111122223333:cluster/my-cluster"
        },
        {
            "Effect": "Allow",
            "Action": "iam:CreateServiceLinkedRole",
            "Resource": "arn:aws: iam::111122223333:role/aws-service-role/
eks.amazonaws.com/AWSServiceRoleForAmazonEKS",
            "Condition": {
                "ForAnyValue:StringEquals": {
                    "iam:AWSServiceName": "eks"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "arn:aws: iam::111122223333:role/cluster-role-name"
        }
    ]
```

```
}
```

在 Outpost 上建立本機 Kubernetes 叢集

此範例政策包含在 us*-west-2 ### Outpost ##### my-cluster* 的 Amazon EKS 本機叢集所需的最低許可。 AWS 您可以將 AWS 區域取代為您要建立叢集 AWS 的區域。如果您看到警告，指出政策中的動作不支援資源層級許可，並要求您在 **All resources**中選擇 AWS Management Console，則可以安全地忽略它。如果您的帳戶已具有 AWSServiceRoleForAmazonEKSLocalOutpost 角色，您可以移除來自政策的 iam:CreateServiceLinkedRole 動作。如果您已在帳戶中的 Outpost 上建立 Amazon EKS 本機叢集，則此角色已存在，除非您將其刪除。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "eks:CreateCluster",
            "Resource": "arn:aws: eks:us-west-2:111122223333:cluster/my-cluster"
        },
        {
            "Action": [
                "ec2:DescribeSubnets",
                "ec2:DescribeVpcs",
                "iam:GetRole"
            ],
            "Resource": "*",
            "Effect": "Allow"
        },
        {
            "Effect": "Allow",
            "Action": "iam:CreateServiceLinkedRole",
            "Resource": "arn:aws: iam::111122223333:role/aws-service-role/outposts.eks-
local.amazonaws.com/AWSServiceRoleForAmazonEKSLocalOutpost"
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:PassRole",
                "iam:ListAttachedRolePolicies"
            ]
            "Resource": "arn:aws: iam::111122223333:role/cluster-role-name"
        },
```

```
        {
            "Action": [
                "iam:CreateInstanceProfile",
                "iam:TagInstanceProfile",
                "iam:AddRoleToInstanceProfile",
                "iam:GetInstanceProfile",
                "iam:DeleteInstanceProfile",
                "iam:RemoveRoleFromInstanceProfile"
            ],
            "Resource": "arn:aws: iam::*:instance-profile/eks-local-*",
            "Effect": "Allow"
        },
    ]
}
```

更新 Kubernetes 叢集

此範例政策包含更新 us*-west-2Region ### my-cluster* 的叢集所需的最低許可。 AWS

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "eks:UpdateClusterVersion",
            "Resource": "arn:aws: eks:us-west-2:111122223333:cluster/my-cluster"
        }
    ]
}
```

所有叢集的清單或描述

此範例政策包含列出及說明帳戶中所有叢集所需的最低許可。IAM 主體必須能夠列出和描述叢集，才能使用 AWS CLI `update-kubeconfig` 命令。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "eks:DescribeCluster",
```

```
                "eks:ListClusters"
            ],
            "Resource": "*"
        }
    ]
}
```

使用 Amazon EKS 的服務連結角色

Amazon Elastic Kubernetes 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 EKS 的一種特殊 IAM 角色類型。服務連結角色是由 Amazon EKS 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 Amazon EKS 叢集的角色
- 使用 Amazon EKS 節點群組的角色
- 使用 Amazon EKS Fargate 描述檔的角色
- 使用角色將 Kubernetes 叢集連線至 Amazon EKS
- 使用位於 Outpost 上的 Amazon EKS 本機叢集的角色
- 使用 Amazon EKS 儀表板的角色

使用 Amazon EKS 叢集的角色

Amazon Elastic Kubernetes 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 EKS 的一種特殊 IAM 角色類型。服務連結角色是由 Amazon EKS 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。

服務連結角色可讓您更輕鬆地設定 Amazon EKS，因為您不必手動新增必要的許可。Amazon EKS 定義其服務連結角色的許可，除非另有定義，否則僅有 Amazon EKS 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。這可保護您的 Amazon EKS 資源，因為您不會不小心移除存取資源的許可。

如需關於支援服務連結角色的其他服務的資訊，請參閱可搭配 IAM 運作的AWS 服務，尋找 Service-Linked Role (服務連結角色) 欄中顯示為 Yes (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 EKS 的服務連結角色許可

Amazon EKS 使用名為 的服務連結角色AWSServiceRoleForAmazonEKS。角色允許 Amazon EKS 管理您帳戶中的叢集。連接的政策允許角色管理下列資源：網路介面、安全群組、記錄和 VPC。

> **Note**
>
> AWSServiceRoleForAmazonEKS 服務連結角色並非叢集建立所需的角色。如需詳細資訊，請參閱the section called “叢集 IAM 角色”。

AWSServiceRoleForAmazonEKS 服務連結角色信任下列服務擔任角色：

- eks.amazonaws.com

此角色許可政策允許 Amazon EKS 對指定資源完成下列動作：

- AmazonEKSServiceRolePolicy

您必須設定許可，IAM 實體 (如使用者、群組或角色) 才可建立、編輯或刪除服務連結角色。如需詳細資訊，請參閱《IAM 使用者指南》中的服務連結角色許可。

建立 Amazon EKS 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 Management Console、 CLI 或 AWS API AWS 中建立叢集時，Amazon EKS 會為您建立服務連結角色。

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。當您建立叢集時，Amazon EKS 會再次為您建立服務連結角色。

編輯 Amazon EKS 的服務連結角色

Amazon EKS 不允許您編輯 AWSServiceRoleForAmazonEKS 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 EKS 的服務連結角色

若您不再使用需要服務連結角色的功能或服務，我們建議您刪除該角色。如此一來，您就沒有未主動監控或維護的未使用實體。然而，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 EKS 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。

1. 開啟 Amazon EKS 主控台。
2. 在左側導覽窗格中選擇叢集。
3. 如果您的叢集具有任何節點群組或 Fargate 描述檔，則必須先將其刪除，才能刪除叢集。如需詳細資訊，請參閱 the section called “Delete” 及 the section called “刪除設定檔”。
4. 在 Clusters (叢集) 頁面上，選擇您要刪除的叢集，然後選擇 Delete (刪除)。
5. 在刪除確認視窗中輸入叢集的名稱，然後選擇 Delete (刪除)。
6. 重複對您帳戶中的任何其他叢集執行此程序。等候所有刪除作業完成。

手動刪除服務連結角色

使用 IAM 主控台、CLI AWS 或 AWS API 來刪除`AWSServiceRoleForAmazonEKS`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 EKS 服務連結角色的支援區域

Amazon EKS 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 Amazon EKS endpoints and quotas (Amazon EKS 端點和配額)。

使用 Amazon EKS 節點群組的角色

Amazon EKS 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 EKS 的一種特殊 IAM 角色類型。服務連結角色是由 Amazon EKS 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。

服務連結角色可讓您更輕鬆地設定 Amazon EKS，因為您不必手動新增必要的許可。Amazon EKS 定義其服務連結角色的許可，除非另有定義，否則僅有 Amazon EKS 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。這可保護您的 Amazon EKS 資源，因為您不會不小心移除存取資源的許可。

如需關於支援服務連結角色的其他服務的資訊，請參閱可搭配 IAM 運作的AWS 服務，尋找 Service-Linked Role (服務連結角色) 欄中顯示為 Yes (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 EKS 的服務連結角色許可

Amazon EKS 使用名為 的服務連結角色AWSServiceRoleForAmazonEKSNodegroup。角色允許 Amazon EKS 管理您帳戶中的節點群組。連接AWSServiceRoleForAmazonEKSNodegroup的政策允許角色管理下列資源：Auto Scaling 群組、安全群組、啟動範本和 IAM 執行個體描述檔。如需詳細資訊，請參閱the section called “AWS 受管政策：AWSServiceRoleForAmazonEKSNodegroup”。

AWSServiceRoleForAmazonEKSNodegroup 服務連結角色信任下列服務擔任角色：

- eks-nodegroup.amazonaws.com

此角色許可政策允許 Amazon EKS 對指定資源完成下列動作：

- AWSServiceRoleForAmazonEKSNodegroup

您必須設定許可，IAM 實體 (如使用者、群組或角色) 才可建立、編輯或刪除服務連結角色。如需詳細資訊，請參閱《IAM 使用者指南》中的服務連結角色許可。

建立 Amazon EKS 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 Management Console、 CLI AWS 或 AWS API 中 CreateNodegroup 時，Amazon EKS 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 EKS 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 EKS 會在帳戶中建立 AWSServiceRoleForAmazonEKSNodegroup 角色。若要進一步了解，請參閱顯示在我的 IAM 帳戶中的新角色。

在 Amazon EKS (AWS API) 中建立服務連結角色

您不需要手動建立服務連結角色。當您在 AWS Management Console、 CLI 或 AWS API AWS 中建立受管節點群組時，Amazon EKS 會為您建立服務連結角色。

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。當您建立另一個受管節點群組時，Amazon EKS 會再次為您建立服務連結角色。

編輯 Amazon EKS 的服務連結角色

Amazon EKS 不允許您編輯 AWSServiceRoleForAmazonEKSNodegroup 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 EKS 的服務連結角色

若您不再使用需要服務連結角色的功能或服務，我們建議您刪除該角色。如此一來，您就沒有未主動監控或維護的未使用實體。然而，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 EKS 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。

1. 開啟 Amazon EKS 主控台。
2. 在左側導覽窗格中選擇叢集。
3. 選取 Compute (運算) 標籤。
4. 在 Node Groups (節點群組) 區段中，選擇要刪除的節點群組。
5. 在刪除確認視窗中，輸入節點群組的名稱，然後選擇 Delete (刪除)。
6. 重複對叢集中的任何其他節點群組執行此程序。等候所有刪除作業完成。

手動刪除服務連結角色

使用 IAM 主控台、CLI AWS 或 AWS API 來刪除AWSServiceRoleForAmazonEKSNodegroup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 EKS 服務連結角色的支援區域

Amazon EKS 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 Amazon EKS endpoints and quotas (Amazon EKS 端點和配額)。

使用 Amazon EKS Fargate 描述檔的角色

Amazon EKS 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 EKS 的一種特殊 IAM 角色類型。服務連結角色是由 Amazon EKS 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。

服務連結角色可讓您更輕鬆地設定 Amazon EKS，因為您不必手動新增必要的許可。Amazon EKS 定義其服務連結角色的許可，除非另有定義，否則僅有 Amazon EKS 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。這可保護您的 Amazon EKS 資源，因為您不會不小心移除存取資源的許可。

如需關於支援服務連結角色的其他服務的資訊，請參閱可搭配 IAM 運作的AWS 服務，尋找 Service-Linked Role (服務連結角色) 欄中顯示為 Yes (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 EKS 的服務連結角色許可

Amazon EKS 使用名為 的服務連結角色`AWSServiceRoleForAmazonEKSForFargate`。此角色允許 Amazon EKS Fargate 設定 Fargate Pod 所需的 VPC 聯網。已連接的政策允許角色建立和刪除彈性網路介面，並描述彈性網路介面和資源。

`AWSServiceRoleForAmazonEKSForFargate` 服務連結角色信任下列服務以擔任角色：

- `eks-fargate.amazonaws.com`

此角色許可政策允許 Amazon EKS 對指定資源完成下列動作：

- AmazonEKSForFargateServiceRolePolicy

您必須設定許可，IAM 實體 (如使用者、群組或角色) 才可建立、編輯或刪除服務連結角色。如需詳細資訊，請參閱《IAM 使用者指南》中的服務連結角色許可。

建立 Amazon EKS 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 Management Console、 CLI 或 AWS API AWS 中建立 Fargate 設定檔時，Amazon EKS 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 EKS 自 2019 年 12 月 13 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 EKS 會在帳戶中建立 AWSServiceRoleForAmazonEKSForFargate 角色。若要進一步了解，請參閱顯示在我的 IAM 帳戶中的新角色。

在 Amazon EKS (AWS API) 中建立服務連結角色

您不需要手動建立服務連結角色。當您在 AWS Management Console、 CLI 或 AWS API AWS 中建立 Fargate 設定檔時，Amazon EKS 會為您建立服務連結角色。

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。當您建立另一個受管節點群組時，Amazon EKS 會再次為您建立服務連結角色。

編輯 Amazon EKS 的服務連結角色

Amazon EKS 不允許您編輯 `AWSServiceRoleForAmazonEKSForFargate` 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 EKS 的服務連結角色

若您不再使用需要服務連結角色的功能或服務，我們建議您刪除該角色。如此一來，您就沒有未主動監控或維護的未使用實體。然而，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 EKS 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。

1. 開啟 Amazon EKS 主控台。
2. 在左側導覽窗格中選擇叢集。
3. 在 Clusters (叢集) 頁面上，選取您的叢集。
4. 選取 Compute (運算) 標籤。
5. 如果 Fargate Profiles (Fargate 描述檔) 區段中有任何 Fargate 描述檔，請個別選取每個描述檔，然後選擇 Delete (刪除)。
6. 在刪除確認視窗中輸入描述檔的名稱，然後選擇 Delete (刪除)。
7. 針對叢集中的任何其他 Fargate 描述檔，以及您帳戶中的任何其他叢集，重複此程序。

手動刪除服務連結角色

使用 IAM 主控台、CLI AWS 或 AWS API 來刪除 AWSServiceRoleForAmazonEKSForFargate 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 EKS 服務連結角色的支援區域

Amazon EKS 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 Amazon EKS endpoints and quotas (Amazon EKS 端點和配額)。

使用角色將 Kubernetes 叢集連線至 Amazon EKS

Amazon EKS 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 EKS 的一種特殊 IAM 角色類型。服務連結角色是由 Amazon EKS 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。

服務連結角色可讓您更輕鬆地設定 Amazon EKS，因為您不必手動新增必要的許可。Amazon EKS 定義其服務連結角色的許可，除非另有定義，否則僅有 Amazon EKS 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。這可保護您的 Amazon EKS 資源，因為您不會不小心移除存取資源的許可。

如需關於支援服務連結角色的其他服務的資訊，請參閱可搭配 IAM 運作的AWS 服務，尋找 Service-Linked Role (服務連結角色) 欄中顯示為 Yes (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 EKS 的服務連結角色許可

Amazon EKS 使用名為 的服務連結角色AWSServiceRoleForAmazonEKSConnector。此角色允許 Amazon EKS 連接 Kubernetes 叢集。連接的政策允許 角色管理必要的資源，以連線至已註冊的 Kubernetes 叢集。

AWSServiceRoleForAmazonEKSConnector 服務連結角色信任下列服務以擔任角色：

- eks-connector.amazonaws.com

此角色許可政策允許 Amazon EKS 對指定資源完成下列動作：

- AmazonEKSConnectorServiceRolePolicy

您必須設定許可，IAM 實體 (如使用者、群組或角色) 才可建立、編輯或刪除服務連結角色。如需詳細資訊，請參閱《IAM 使用者指南》中的服務連結角色許可。

建立 Amazon EKS 的服務連結角色

您不需要手動建立服務連結角色即可連接叢集。當您在 AWS Management Console、 CLI、eksctl或 AWS API AWS 中連接叢集時，Amazon EKS 會為您建立服務連結角色。

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。在連線叢集時，Amazon EKS 會再次為您建立服務連結角色。

編輯 Amazon EKS 的服務連結角色

Amazon EKS 不允許您編輯 AWSServiceRoleForAmazonEKSConnector 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 EKS 的服務連結角色

若您不再使用需要服務連結角色的功能或服務，我們建議您刪除該角色。如此一來，您就沒有未主動監控或維護的未使用實體。然而，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 EKS 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。

1. 開啟 Amazon EKS 主控台。
2. 在左側導覽窗格中選擇叢集。
3. 在 Clusters (叢集) 頁面上，選取您的叢集。
4. 選取 Deregister (取消註冊) 索引標籤，然後選取 OK (確定) 索引標籤。

手動刪除服務連結角色

使用 IAM 主控台、CLI AWS 或 AWS API 來刪除 AWSServiceRoleForAmazonEKSConnector 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。

使用位於 Outpost 上的 Amazon EKS 本機叢集的角色

Amazon Elastic Kubernetes 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 EKS 的一種特殊 IAM 角色類型。服務連結角色是由 Amazon EKS 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。

服務連結角色可讓您更輕鬆地設定 Amazon EKS，因為您不必手動新增必要的許可。Amazon EKS 定義其服務連結角色的許可，除非另有定義，否則僅有 Amazon EKS 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。這可保護您的 Amazon EKS 資源，因為您不會不小心移除存取資源的許可。

如需關於支援服務連結角色的其他服務的資訊，請參閱可搭配 IAM 運作的AWS 服務，尋找 Service-Linked Role (服務連結角色) 欄中顯示為 Yes (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 EKS 的服務連結角色許可

Amazon EKS 使用名為 的服務連結角色`AWSServiceRoleForAmazonEKSLocalOutpost`。此角色允許 Amazon EKS 管理您帳戶中的本機叢集。已連結的政策允許角色管理下列資源：網路介面、安全群組、日誌和 Amazon EC2 執行個體。

> **Note**
>
> AWSServiceRoleForAmazonEKSLocalOutpost 服務連結角色並非叢集建立所需的角色。如需詳細資訊，請參閱the section called “叢集 IAM 角色”。

AWSServiceRoleForAmazonEKSLocalOutpost 服務連結角色信任下列服務擔任角色：

- outposts.eks-local.amazonaws.com

此角色許可政策允許 Amazon EKS 對指定資源完成下列動作：

- AmazonEKSServiceRolePolicy

您必須設定許可，IAM 實體 (如使用者、群組或角色) 才可建立、編輯或刪除服務連結角色。如需詳細資訊，請參閱《IAM 使用者指南》中的服務連結角色許可。

建立 Amazon EKS 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 Management Console、 CLI 或 AWS API AWS 中建立叢集時，Amazon EKS 會為您建立服務連結角色。

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。當您建立叢集時，Amazon EKS 會再次為您建立服務連結角色。

編輯 Amazon EKS 的服務連結角色

Amazon EKS 不允許您編輯 AWSServiceRoleForAmazonEKSLocalOutpost 服務連結角色。建立服務連結角色之後，您無法變更角色的名稱，因為各種實體可能會參考角色。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 EKS 的服務連結角色

若您不再使用需要服務連結角色的功能或服務，我們建議您刪除該角色。如此一來，您就沒有未主動監控或維護的未使用實體。然而，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 EKS 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。

1. 開啟 Amazon EKS 主控台。
2. 在左側導覽窗格中選擇 Amazon EKS Clusters (叢集)。
3. 如果您的叢集具有任何節點群組或 Fargate 描述檔，則必須先將其刪除，才能刪除叢集。如需詳細資訊，請參閱 the section called “Delete” 及 the section called “刪除設定檔”。
4. 在 Clusters (叢集) 頁面上，選擇您要刪除的叢集，然後選擇 Delete (刪除)。
5. 在刪除確認視窗中輸入叢集的名稱，然後選擇 Delete (刪除)。
6. 重複對您帳戶中的任何其他叢集執行此程序。等候所有刪除作業完成。

手動刪除服務連結角色

使用 IAM 主控台、CLI AWS 或 AWS API 來刪除`AWSServiceRoleForAmazonEKSLocalOutpost`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 EKS 服務連結角色的支援區域

Amazon EKS 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 Amazon EKS endpoints and quotas (Amazon EKS 端點和配額)。

使用 Amazon EKS 儀表板的角色

Amazon EKS 儀表板會使用此服務連結角色來彙總來自多個區域和帳戶的叢集資源相關資訊。儀表板會與 AWS Organizations 整合，以安全地讀取組織中多個帳戶的相關資訊。

若要進一步了解 Amazon EKS 儀表板，請參閱 the section called “Amazon EKS 儀表板”。

背景介紹

Amazon EKS 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 EKS 的一種特殊 IAM 角色類型。服務連結角色是由 Amazon EKS 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。

服務連結角色可讓您更輕鬆地設定 Amazon EKS，因為您不必手動新增必要的許可。Amazon EKS 定義其服務連結角色的許可，除非另有定義，否則僅有 Amazon EKS 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。這可保護您的 Amazon EKS 資源，因為您不會不小心移除存取資源的許可。

如需關於支援服務連結角色的其他服務資訊，請參閱《可搭配 IAM 運作的AWS 服務》，尋找 Service-linked roles (服務連結角色) 欄中顯示為 Yes (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 EKS 的服務連結角色許可

Amazon EKS 使用名為 的服務連結角色AWSServiceRoleForAmazonEKSDashboard。此角色允許 Amazon EKS 產生和顯示 EKS 儀表板，包括叢集資源的彙總資訊。連接AmazonEKSDashboardServiceRolePolicy的政策允許角色管理下列資源：Auto Scaling 群組、安全群組、啟動範本和 IAM 執行個體設定檔。如需詳細資訊，請參閱the section called “AWS 受管政策：AmazonEKSDashboardServiceRolePolicy”。

AWSServiceRoleForAmazonEKSDashboard 服務連結角色信任下列服務擔任角色：

- dashboard.eks.amazonaws.com

若要檢視最新版本的 JSON 政策文件，請參閱《 AWS 受管政策參考指南》中的AmazonEKSDashboardServiceRolePolicy。

您必須設定許可，IAM 實體 (如使用者、群組或角色) 才可建立、編輯或刪除服務連結角色。如需詳細資訊，請參閱《IAM 使用者指南》中的服務連結角色許可。

建立 Amazon EKS 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 主控台中啟用儀表板時，Amazon EKS 會為您建立服務連結角色。

如需啟用 EKS 儀表板的詳細資訊，請參閱 the section called “設定組織”。

> **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。

編輯 Amazon EKS 的服務連結角色

Amazon EKS 不允許您編輯 AWSServiceRoleForAmazonEKSDashboard 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 EKS 的服務連結角色

若您不再使用需要服務連結角色的功能或服務，我們建議您刪除該角色。如此一來，您就沒有未主動監控或維護的未使用實體。然而，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 EKS 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。

如需停用 EKS 儀表板的詳細資訊，請參閱 the section called “設定組織”。

手動刪除服務連結角色

使用 IAM 主控台、CLI AWS 或 AWS API 來刪除AWSServiceRoleForAmazonEKSDashboard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 EKS 服務連結角色的支援區域

Amazon EKS 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 Amazon EKS endpoints and quotas (Amazon EKS 端點和配額)。

Amazon EKS Pod 執行 IAM 角色

在 AWS Fargate 基礎設施上執行 Pod 需要 Amazon EKS Pod 執行角色。

當您的叢集在 AWS Fargate 基礎設施上建立 Pod 時，在 Fargate 基礎設施上執行的元件必須代表您呼叫 AWS APIs。如此一來，他們可以執行動作，例如從 Amazon ECR 提取容器映像，或將日誌路由到其他 AWS 服務。Amazon EKS Pod 執行角色提供執行此操作的 IAM 許可。

建立 Fargate 設定檔時，您必須為使用設定檔在 Fargate 基礎設施上執行的 Amazon EKS 元件指定 Pod 執行角色。此角色會新增至叢集的 Kubernetes 角色型存取控制 (RBAC) 以進行授權。這可讓在 Fargate 基礎設施上執行`kubelet`的 向 Amazon EKS 叢集註冊，使其可以作為節點出現在叢集中。

> **Note**
>
> Fargate 設定檔的 IAM 角色必須與 Amazon EC2 節點群組不同。

> **Important**
>
> 在 Fargate Pod 中執行的容器無法擔任與 Pod 執行角色相關聯的 IAM 許可。若要授予 Fargate Pod 中的容器存取其他服務的許可 AWS ，您必須為服務帳戶使用 IAM 角色。

建立 Fargate 設定檔之前，您必須使用 AmazonEKSFargatePodExecutionRolePolicy 建立 IAM 角色。

檢查是否已正確設定現有的 Pod 執行角色

您可以使用下列程序來檢查並查看您的帳戶是否已有正確設定的 Amazon EKS Pod 執行角色。為了避免混淆代理人安全問題，角色必須根據 限制存取`SourceArn`。您可以視需要修改執行角色，以納入對其他叢集上的 Fargate 描述檔的支援。

1. 開啟位於 https://console.aws.amazon.com/iam/ 的 IAM 主控台。
2. 在左側導覽窗格中，選擇 Roles (角色)。
3. 在 Roles (角色) 頁面上，搜尋 Amazon EKS Fargate Pod Execution Role (Amazon EKS Fargate Pod 執行角色) 的角色清單。如果角色不存在，請參閱 the section called “建立 Amazon EKS Pod 執行角色” 以建立角色。如果該角色存在，請選取角色。
4. 在 Amazon EKS Fargate Pod Execution Role (Amazon EKS Fargate Pod 執行角色) 頁面上，執行下列動作：
   a. 選擇許可。
   b. 確定 AmazonEKSFargatePodExecutionRolePolicy Amazon 受管政策已連接至角色。
   c. 選擇 Trust relationships (信任關係)。
   d. 選擇 Edit trust policy (編輯信任政策)。
5. 在 Edit trust policy (編輯信任政策)頁面上，確認信任關係包含以下政策，並且在叢集上具有 Fargate 描述檔的行。若是如此，請選擇 Cancel (取消)。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Condition": {
         "ArnLike": {
            "aws:SourceArn": "arn:aws: eks:region-code:111122223333:fargateprofile/
my-cluster/*"
         }
      },
      "Principal": {
        "Service": "eks-fargate-pods.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

如果政策相符，但沒有在叢集上指定 Fargate 設定檔的行，您可以在ArnLike物件頂端新增以下行。將 *region-code* 取代為您的叢集所在的 AWS 區域、將 *111122223333* 取代為您的 帳戶 ID，並將 *my-cluster* 取代為您的叢集名稱。

```
"aws:SourceArn": "arn:aws: eks:region-code:111122223333:fargateprofile/my-cluster/*",
```

如果政策不相符，請將完整的先前政策複製到表單，然後選擇更新政策。將 *region-code* 取代為您的叢集所在的 AWS 區域。如果您想要在帳戶的所有 AWS 區域中使用相同的角色，請以 取代 *region-code**。將 *111122223333* 取代為您的 帳戶 ID，並將 *my-cluster* 取代為您的叢集名稱。如果您想要對帳戶中的所有叢集使用相同的角色，請將 *my-cluster* 取代為 *。

建立 Amazon EKS Pod 執行角色

如果您還沒有叢集的 Amazon EKS Pod 執行角色，您可以使用 AWS Management Console 或 AWS CLI 來建立叢集。

AWS Management Console

a. 開啟位於 https://console.aws.amazon.com/iam/ 的 IAM 主控台。

b. 在左側導覽窗格中，選擇 Roles (角色)。

c. 在 Roles (角色) 頁面上，選擇 Create role (建立角色)。

d. 在 Select trusted entity (選取信任的實體) 頁面上，執行以下作業：

i. 在信任的實體類型區段中，選擇 AWS 服務。

ii. 從 AWS 其他服務的使用案例下拉式清單中，選擇 EKS。

iii. 選擇 EKS - Fargate Pod。

iv. 選擇 Next (下一步)。

e. 在 Add permissions (新增許可) 頁面上，選擇 Next (下一步)。

f. 在 Name, review, and create (命名、檢閱和建立) 頁面上，執行以下作業：

i. 針對 Role name (角色名稱)，為您的角色輸入唯一名稱 (例如 `AmazonEKSFargatePodExecutionRole`)。

ii. 藉由連接標籤做為鍵值對，在新增標籤 (選用) 下將中繼資料新增至角色。如需有關在 IAM 中使用標籤的詳細資訊，請參閱《IAM 使用者指南》中的標記 IAM 資源。

iii. 選擇建立角色。

g. 在 Roles (角色) 頁面上，搜尋 Amazon EKS Fargate Pod Execution Role (Amazon EKS Fargate Pod 執行角色) 的角色清單。選擇角色。

h. 在 Amazon EKS Fargate Pod Execution Role (Amazon EKS Fargate Pod 執行角色) 頁面上，執行下列動作：

i. 選擇 Trust relationships (信任關係)。

ii. 選擇 Edit trust policy (編輯信任政策)。

i. 在 Edit trust policy (編輯信任政策) 頁面上，執行下列動作：

i. 請複製以下內容，並在 Edit trust policy (編輯信任政策) 表單貼上。將 *region-code* 取代為您叢集所在的 AWS 區域。如果您想要在帳戶的所有 AWS 區域中使用相同的角色，請以 取代 *region-code**。將 *111122223333* 取代為您的 帳戶 ID，並將 *my-cluster* 取代為您的叢集名稱。如果您想要對帳戶中的所有叢集使用相同的角色，請將 *my-cluster* 取代為 *。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Condition": {
         "ArnLike": {
            "aws:SourceArn": "arn:aws: eks:region-
code:111122223333:fargateprofile/my-cluster/*"
         }
```

```
      },
      "Principal": {
        "Service": "eks-fargate-pods.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

ii. 選擇更新政策。

AWS CLI

a. 複製以下內容並在名為 pod-execution-role-trust-policy.json 的檔案貼上。將 *region-code* 取代為您的叢集所在的 AWS 區域。如果您想要在帳戶的所有 AWS 區域中使用相同的角色，請以 取代 *region-code**。將 *111122223333* 取代為您的 帳戶 ID，並將 *my-cluster* 取代為您的叢集名稱。如果您想要對帳戶中的所有叢集使用相同的角色，請將 *my-cluster* 取代為 *。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Condition": {
         "ArnLike": {
            "aws:SourceArn": "arn:aws: eks:region-
code:111122223333:fargateprofile/my-cluster/*"
         }
      },
      "Principal": {
        "Service": "eks-fargate-pods.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

b. 建立 Pod 執行 IAM 角色。

```
aws iam create-role \
  --role-name AmazonEKSFargatePodExecutionRole \
  --assume-role-policy-document file://"pod-execution-role-trust-policy.json"
```

c. 將必要的 Amazon EKS 受管 IAM 政策連接到角色。

```
aws iam attach-role-policy \
  --policy-arn arn:aws: iam::aws:policy/AmazonEKSFargatePodExecutionRolePolicy \
  --role-name AmazonEKSFargatePodExecutionRole
```

Amazon EKS 連接器 IAM 角色

您可以連接 Kubernetes 叢集，在 中檢視它們 AWS Management Console。若要連接至 Kubernetes 叢集，請建立 IAM 角色。

檢查是否有現有的 EKS 連接器角色

您可使用以下程序，檢查帳戶是否已有 Amazon EKS 連接器角色。

1. 開啟位於 https://console.aws.amazon.com/iam/ 的 IAM 主控台。
2. 在左側導覽窗格中，選擇 Roles (角色)。
3. 搜尋 `AmazonEKSConnectorAgentRole` 的角色清單。如果`AmazonEKSConnectorAgentRole`包含 的角色不存在，請參閱 the section called “建立 Amazon EKS 連接器代理程式角色” 以建立角色。如果包含 `AmazonEKSConnectorAgentRole` 的角色存在，請選取角色以檢視連接的政策。
4. 選擇許可。
5. 確定 AmazonEKSConnectorAgentPolicy 受管政策已附加到該角色。如果已連接政策，則您的 Amazon EKS 連接器角色應已設定妥當。
6. 選擇 Trust Relationships (信任關係)，然後選擇 Edit trust policy (編輯信任政策)。
7. 確認信任關係包含下列政策。如果信任關係符合下列政策，請選擇 Cancel (取消)。如果信任關係不相符，請將政策複製到編輯信任政策視窗，然後選擇更新政策。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "ssm.amazonaws.com"
                ]
            },
```

```
            "Action": "sts:AssumeRole"
        }
    ]
}
```

建立 Amazon EKS 連接器代理程式角色

您可以使用 AWS Management Console or AWS CloudFormation 來建立連接器代理程式角色。

AWS CLI

a. 建立名為 `eks-connector-agent-trust-policy.json` 的檔案，其中包含用於 IAM 角色的下列 JSON。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "ssm.amazonaws.com"
                ]
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

b. 建立名為 `eks-connector-agent-policy.json` 的檔案，其中包含用於 IAM 角色的下列 JSON。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "SsmControlChannel",
            "Effect": "Allow",
            "Action": [
                "ssmmessages:CreateControlChannel"
            ],
            "Resource": "arn:aws: eks:*:*:cluster/*"
```

```
        },
        {
            "Sid": "ssmDataplaneOperations",
            "Effect": "Allow",
            "Action": [
                "ssmmessages:CreateDataChannel",
                "ssmmessages:OpenDataChannel",
                "ssmmessages:OpenControlChannel"
            ],
            "Resource": "*"
        }
    ]
}
```

c. 使用信任政策和您在先前清單項目中建立的政策，建立 Amazon EKS 連接器代理程式角色。

```
aws iam create-role \
     --role-name AmazonEKSConnectorAgentRole \
     --assume-role-policy-document file://eks-connector-agent-trust-policy.json
```

d. 將政策連接到 Amazon EKS 連接器代理程式角色。

```
aws iam put-role-policy \
     --role-name AmazonEKSConnectorAgentRole \
     --policy-name AmazonEKSConnectorAgentPolicy \
     --policy-document file://eks-connector-agent-policy.json
```

AWS CloudFormation

a. 將下列 AWS CloudFormation 範本儲存至本機系統上的文字檔案。

> **Note**
>
> 此範本也會建立服務連結角色，否則會在呼叫 `registerCluster` API 時建立。如需詳細資訊，請參閱 the section called “叢集連接器角色”。

```
---
AWSTemplateFormatVersion: '2010-09-09'
Description: 'Provisions necessary resources needed to register clusters in EKS'
Parameters: {}
Resources:
```

```
  EKSConnectorSLR:
    Type: AWS::IAM::ServiceLinkedRole
    Properties:
      AWSServiceName: eks-connector.amazonaws.com

  EKSConnectorAgentRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: '2012-10-17'
        Statement:
          - Effect: Allow
            Action: [ 'sts:AssumeRole' ]
            Principal:
              Service: 'ssm.amazonaws.com'

  EKSConnectorAgentPolicy:
    Type: AWS::IAM::Policy
    Properties:
      PolicyName: EKSConnectorAgentPolicy
      Roles:
        - {Ref: 'EKSConnectorAgentRole'}
      PolicyDocument:
        Version: '2012-10-17'
        Statement:
          - Effect: 'Allow'
            Action: [ 'ssmmessages:CreateControlChannel' ]
            Resource:
            - Fn::Sub: 'arn:${AWS::Partition}:eks:*:*:cluster/*'
          - Effect: 'Allow'
            Action: [ 'ssmmessages:CreateDataChannel',
 'ssmmessages:OpenDataChannel', 'ssmmessages:OpenControlChannel' ]
            Resource: "*"
Outputs:
  EKSConnectorAgentRoleArn:
    Description: The agent role that EKS connector uses to communicate with AWS
 services.
    Value: !GetAtt EKSConnectorAgentRole.Arn
```

b. 開啟 AWS CloudFormation 主控台。

c. 選擇使用新資源建立堆疊 （標準）。

d. 針對 Specify template (指定範本)，選擇 Upload a template file (上傳範本檔案)，然後選擇 Choose file (選擇檔案)。

e. 選擇您剛建立的檔案，然後選擇 Next (下一步)。

f. 針對 Stack name (堆疊名稱)，輸入您角色的名稱 (例如 `eksConnectorAgentRole`)，然後選擇 Next (下一步)。

g. 在 Configure stack options (設定堆疊選項) 頁面，選擇 Next (下一步)。

h. 在 Review (檢閱) 頁面上，檢閱您的資訊，確認該堆疊可能會建立 IAM 資源，並選擇 Create stack (建立堆疊)。

AWS Amazon Elastic Kubernetes Service 的 受管政策

AWS 受管政策是由 AWS AWS 受管政策建立和管理的獨立政策旨在為許多常用案例提供許可，以便您可以開始將許可指派給使用者、群組和角色。

請記住， AWS 受管政策可能不會授予特定使用案例的最低權限許可，因為這些許可可供所有 AWS 客戶使用。我們建議您定義使用案例專屬的客戶管理政策，以便進一步減少許可。

您無法變更 AWS 受管政策中定義的許可。如果 AWS 更新 AWS 受管政策中定義的許可，則更新會影響政策連接的所有委託人身分 （使用者、群組和角色）。當新 AWS 服務啟動或新 API 操作可用於現有服務時， AWS 很可能更新 AWS 受管政策。

如需詳細資訊，請參閱《IAM 使用者指南》中的 AWS 受管政策。

AWS 受管政策：AmazonEKS_CNI_Policy

您可以將 `AmazonEKS_CNI_Policy` 連接到 IAM 實體。建立 Amazon EC2 節點群組之前，此政策必須連接至節點 IAM 角色，或 Amazon VPC CNI 外掛程式特別用於 Kubernetes 的 IAM 角色。這樣一來，可以代表您執行動作。建議您將政策連接至僅由外掛程式使用的角色。如需詳細資訊，請參閱the section called “Amazon VPC CNI”及the section called “為 IRSA 設定”。

許可詳細資訊

此政策包含下列許可，這些許可能讓 Amazon EKS 完成下列任務：

- **`ec2:*NetworkInterface`** 和 **`ec2:*PrivateIpAddresses`** – 允許 Amazon VPC CNI 外掛程式執行動作，例如為 Pod 佈建彈性網路介面和 IP 地址，為在 Amazon EKS 中執行的應用程式提供聯網。
- **`ec2`** 讀取動作 – 允許 Amazon VPC CNI 外掛程式執行動作，例如描述執行個體和子網路，以查看 Amazon VPC 子網路中的可用 IP 地址數量。VPC CNI 可以使用每個子網路中的可用 IP 地址，來挑選具有建立彈性網路介面時要使用之最可用 IP 地址的子網路。

若要檢視最新版本的 JSON 政策文件，請參閱《 受管政策參考指南》中的 AmazonEKS_CNI_Policy。AWS

AWS 受管政策：AmazonEKSClusterPolicy

您可以將 AmazonEKSClusterPolicy 連接到 IAM 實體。建立叢集之前，您必須具有連接了此政策的叢集 IAM 角色。由 Amazon EKS 管理的 Kubernetes 叢集會代表您呼叫其他 AWS 服務。他們會執行此作業，以管理您搭配本服務使用的資源。

此政策包含下列許可，這些許可能讓 Amazon EKS 完成下列任務：

- **autoscaling** – 讀取和更新 Auto Scaling 群組的組態。Amazon EKS 不會使用這些許可，但會保留在政策中，以確保回溯相容性。
- **ec2** – 使用與 Amazon EC2 節點相關聯的磁碟區和網路資源。這是必要項目，以便 Kubernetes 控制平面可以將執行個體加入叢集，並動態佈建和管理 Kubernetes 持久性磁碟區請求的 Amazon EBS 磁碟區。
- **ec2** - 刪除 VPC CNI 建立的彈性網路介面。這是必要的，以便 EKS 可以在 VPC CNI 意外退出時清除留下的彈性網路界面。
- **elasticloadbalancing** – 使用 Elastic Load Balancer，並將節點新增為目標。這是必要項目，以便 Kubernetes 控制平台可以動態佈建 Kubernetes 服務要求的 Elastic Load Balancer。
- **iam** – 建立服務連結角色。這是必要項目，以便 Kubernetes 控制平面可以動態佈建 Kubernetes 服務要求的 Elastic Load Balancer。
- **kms** – 從 AWS KMS 讀取金鑰。這是 Kubernetes 控制平面支援在 etcd 中存放的 Kubernetes 秘密之秘密加密的必要條件。

若要檢視最新版本的 JSON 政策文件，請參閱《 AWS 受管政策參考指南》中的 AmazonEKSClusterPolicy。

AWS 受管政策：AmazonEKSDashboardConsoleReadOnly

您可以將 AmazonEKSDashboardConsoleReadOnly 連接到 IAM 實體。

此政策包含下列許可，這些許可能讓 Amazon EKS 完成下列任務：

- **eks** - 唯讀存取 EKS 儀表板資料、資源和叢集版本資訊。這允許檢視 EKS 相關指標和叢集組態詳細資訊。
- **organizations** - AWS Organizations 資訊的唯讀存取權，包括：

- 檢視組織詳細資訊和服務存取
- 列出組織根目錄、帳戶和組織單位
- 檢視組織結構

若要檢視 JSON 政策文件的最新版本，請參閱《 AWS 受管政策參考指南》中的 AmazonEKSDashboardConsoleReadOnly。

AWS 受管政策：AmazonEKSFargatePodExecutionRolePolicy

您可以將 `AmazonEKSFargatePodExecutionRolePolicy` 連接到 IAM 實體。在建立 Fargate 設定檔之前，您必須建立 Fargate Pod 執行角色，並將此政策連接至該角色。如需詳細資訊，請參閱the section called “步驟 2：建立 Fargate Pod 執行角色”及the section called “定義設定檔”。

此政策授予該角色許可，該許可提供在 Fargate 上執行 Amazon EKS Pod 所需的其他 AWS 服務資源的存取權。

許可詳細資訊

此政策包含下列許可，這些許可能讓 Amazon EKS 完成下列任務：

- **`ecr`** – 允許在 Fargate 上執行的 Pod 提取存放在 Amazon ECR 中的容器映像。

若要檢視最新版本的 JSON 政策文件，請參閱《 AWS 受管政策參考指南》中的 AmazonEKSFargatePodExecutionRolePolicy。

AWS 受管政策：AmazonEKSForFargateServiceRolePolicy

您無法`AmazonEKSForFargateServiceRolePolicy`連接至 IAM 實體。此政策會連接到服務連結角色，而此角色可讓 Amazon EKS 代表您執行動作。如需詳細資訊，請參閱`AWSServiceRoleforAmazonEKSForFargate`。

此政策授予 Amazon EKS 執行 Fargate 任務的必要許可。只有在您擁有 Fargate 節點時才會使用此政策。

許可詳細資訊

此政策包含下列允許 Amazon EKS 完成下列任務的許可。

- **`ec2`** – 建立和刪除彈性網路界面，並描述彈性網路界面和資源。這是必要的，以便 Amazon EKS Fargate 服務可以設定 Fargate Pod 所需的 VPC 聯網。

若要檢視最新版本的 JSON 政策文件，請參閱《 AWS 受管政策參考指南》中的 AmazonEKSForFargateServiceRolePolicy。

AWS 受管政策：AmazonEKSComputePolicy

您可以將 AmazonEKSComputePolicy 連接到 IAM 實體。您可以將此政策連接至叢集 IAM 角色，以擴展 EKS 可在帳戶中管理的資源。

此政策授予 Amazon EKS 為 EKS 叢集建立和管理 EC2 執行個體所需的許可，以及設定 EC2 所需的 IAM 許可。此外，此政策會授予 Amazon EKS 代表您建立 EC2 Spot 服務連結角色的許可。

許可詳細資訊

此政策包含下列許可，這些許可能讓 Amazon EKS 完成下列任務：

- **ec2** 許可：
  - ec2:CreateFleet 和 ec2:RunInstances - 允許建立 EC2 執行個體，並為 EKS 叢集節點使用特定的 EC2 資源 （ 影像、安全群組、子網路 ）。
  - ec2:CreateLaunchTemplate - 允許為 EKS 叢集節點建立 EC2 啟動範本。
  - 此政策也包含限制使用這些 EC2 許可的條件，以使用 EKS 叢集名稱和其他相關標籤標記的資源。
  - ec2:CreateTags - 允許將標籤新增至 CreateFleet、 RunInstances和 CreateLaunchTemplate動作建立的 EC2 資源。
- **iam** 許可：
  - iam:AddRoleToInstanceProfile - 允許將 IAM 角色新增至 EKS 運算執行個體描述檔。
  - iam:PassRole - 允許將必要的 IAM 角色傳遞至 EC2 服務。

若要檢視最新版本的 JSON 政策文件，請參閱《 AWS 受管政策參考指南》中的 AmazonEKSComputePolicy。

AWS 受管政策：AmazonEKSNetworkingPolicy

您可以將 AmazonEKSNetworkingPolicy 連接到 IAM 實體。您可以將此政策連接至叢集 IAM 角色，以擴展 EKS 可在帳戶中管理的資源。

此政策旨在授予 Amazon EKS 建立和管理 EKS 叢集網路介面的必要許可，讓控制平面和工作者節點可以正常通訊和運作。

許可詳細資訊

此政策授予下列許可，以允許 Amazon EKS 管理叢集的網路介面：

- **ec2** 網路界面許可：
  - `ec2:CreateNetworkInterface` - 允許建立 EC2 網路介面。
  - 此政策包含限制將此許可用於以 EKS 叢集名稱和 Kubernetes CNI 節點名稱標記的網路介面的條件。
  - `ec2:CreateTags` - 允許將標籤新增至 `CreateNetworkInterface`動作建立的網路介面。
- **ec2** 網路介面管理許可：
  - `ec2:AttachNetworkInterface`、 `ec2:DetachNetworkInterface` - 允許將網路介面連接到 EC2 執行個體並分離。
  - `ec2:UnassignPrivateIpAddresses`、`ec2:UnassignIpv6Addresses`、`ec2:AssignPrivateIp`、`ec2:AssignIpv6Addresses` - 允許管理網路介面的 IP 地址指派。
  - 這些許可僅限於以 EKS 叢集名稱標記的網路介面。

若要檢視 JSON 政策文件的最新版本，請參閱《 AWS 受管政策參考指南》中的 AmazonEKSNetworkingPolicy。

AWS 受管政策：AmazonEKSBlockStoragePolicy

您可以將 `AmazonEKSBlockStoragePolicy` 連接到 IAM 實體。您可以將此政策連接至叢集 IAM 角色，以擴展 EKS 可在帳戶中管理的資源。

此政策授予 Amazon EKS 建立、管理和維護 EKS 叢集 EC2 磁碟區和快照的必要許可，讓控制平面和工作者節點能夠根據 Kubernetes 工作負載的需求佈建和使用持久性儲存。

許可詳細資訊

此 IAM 政策授予下列許可，以允許 Amazon EKS 管理 EC2 磁碟區和快照：

- **ec2** 磁碟區管理許可：
  - `ec2:AttachVolume`、`ec2:DetachVolume`、`ec2:ModifyVolume`、`ec2:EnableFastSnapshotRestores` - 允許連接、分離、修改和啟用 EC2 磁碟區的快速快照還原。
  - 這些許可僅限於以 EKS 叢集名稱標記的磁碟區。

  - `ec2:CreateTags` - 允許將標籤新增至 和 `CreateSnapshot`動作建立的 EC2 磁碟區`CreateVolume`和快照。
- **ec2** 磁碟區建立許可：
  - `ec2:CreateVolume` - 允許建立新的 EC2 磁碟區。
  - 此政策包含限制將此許可用於以 EKS 叢集名稱和其他相關標籤標記的磁碟區的條件。
  - `ec2:CreateSnapshot` - 允許建立新的 EC2 磁碟區快照。
  - 此政策包含限制將此許可用於以 EKS 叢集名稱和其他相關標籤標記的快照的條件。

若要檢視最新版本的 JSON 政策文件，請參閱《 AWS 受管政策參考指南》中的 AmazonEKSBlockStoragePolicy。

AWS 受管政策：AmazonEKSLoadBalancingPolicy

您可以將 `AmazonEKSLoadBalancingPolicy` 連接到 IAM 實體。您可以將此政策連接至叢集 IAM 角色，以擴展 EKS 可在帳戶中管理的資源。

此 IAM 政策授予 Amazon EKS 使用各種 AWS 服務的必要許可，以管理 Elastic Load Balancer (ELBs) 和相關資源。

許可詳細資訊

此政策授予的金鑰許可為：

- **elasticloadbalancing** ：允許建立、修改和管理 Elastic Load Balancer 和目標群組。這包括建立、更新和刪除負載平衡器、目標群組、接聽程式和規則的許可。
- **ec2** ：允許建立和管理 Kubernetes 控制平面將執行個體加入叢集和管理 Amazon EBS 磁碟區所需的安全群組。也允許描述和列出 EC2 資源，例如執行個體、VPCs、子網路、安全群組和其他聯網資源。
- **iam** ：允許為 Elastic Load Balancing 建立服務連結角色，這是 Kubernetes 控制平面動態佈建 ELBs 所需的角色。
- **kms** ：允許從 AWS KMS 讀取金鑰，這是 Kubernetes 控制平面支援加密存放在 等項目中的 Kubernetes 秘密所必需。
- **wafv2** 和 **shield** ：允許關聯和取消關聯 Web ACLs，以及建立/刪除 Elastic Load Balancer AWS 的 Shield 保護。
- **cognito-idp** 、 **acm** 和 **elasticloadbalancing** ：授予許可，以描述使用者集區用戶端、列出和描述憑證，以及描述 Kubernetes 控制平面管理 Elastic Load Balancer 所需的目標群組。

此政策也包含數個條件檢查，以確保使用 eks:eks-cluster-name標籤將許可範圍限定為要管理的特定 EKS 叢集。

若要檢視最新版本的 JSON 政策文件，請參閱《 AWS 受管政策參考指南》中的 AmazonEKSLoadBalancingPolicy。

AWS 受管政策：AmazonEKSServicePolicy

您可以將 AmazonEKSServicePolicy 連接到 IAM 實體。在 2020 年 4 月 16 日之前建立的叢集，需要您建立 IAM 角色並將此政策連接到該角色。在 2020 年 4 月 16 日或之後建立的叢集不需要您建立角色，也不需要您指派此政策。當您使用具有 iam:CreateServiceLinkedRole許可的 IAM 主體建立叢集時，會自動為您建立 AWSServiceRoleforAmazonEKS 服務連結角色。服務連結角色具有連接到它的受管政策：AmazonEKSServiceRolePolicy。

此政策允許 Amazon EKS 建立和管理操作 Amazon EKS 叢集所需的資源。

許可詳細資訊

此政策包含下列允許 Amazon EKS 完成下列任務的許可。

- **eks** – 在您啟動更新後更新叢集的 Kubernetes 版本。Amazon EKS 不會使用此許可，但會保留在政策中，以確保回溯相容性。
- **ec2** – 使用彈性網路介面和其他網路資源和標籤。Amazon EKS 要求這樣做，以設定可促進節點與 Kubernetes 控制平面之間的通訊的網路。讀取安全群組的相關資訊。更新安全群組上的標籤。
- **route53** – 將 VPC 與託管區域建立關聯。Amazon EKS 必須執行這項操作，才能為您的 Kubernetes 叢集 API 伺服器啟用私有端點聯網。
- **logs** – 記錄事件。這是必要項目，以便 Amazon EKS 可以將 Kubernetes 控制平面日誌運送到 CloudWatch。
- **iam** – 建立服務連結角色。這是必需的，以便 Amazon EKScan 代表您建立 the section called “Amazon EKS 的服務連結角色許可” 服務連結角色。

若要檢視最新版本的 JSON 政策文件，請參閱《 AWS 受管政策參考指南》中的 AmazonEKSServicePolicy。

AWS 受管政策：AmazonEKSServiceRolePolicy

您無法AmazonEKSServiceRolePolicy連接至 IAM 實體。此政策會連接到服務連結角色，而此角色可讓 Amazon EKS 代表您執行動作。如需詳細資訊，請參閱the section called “Amazon EKS 的服務

連結角色許可”。當您使用具有 iam:CreateServiceLinkedRole許可的 IAM 主體建立叢集時，系統會自動為您建立 AWSServiceRoleforAmazonEKS 服務連結角色，且此政策會連接至叢集。

此政策允許服務連結角色代表您呼叫 AWS 服務。

許可詳細資訊

此政策包含下列允許 Amazon EKS 完成下列任務的許可。

- **ec2** – 建立和描述建立叢集所需的彈性網路介面和 Amazon EC2 執行個體、叢集安全群組和 VPC。如需詳細資訊，請參閱the section called “安全群組要求”。讀取安全群組的相關資訊。更新安全群組上的標籤。閱讀有關隨需容量預留的資訊。
- **ec2** 自動模式 – 終止 EKS Auto Mode 建立的 EC2 執行個體。如需詳細資訊，請參閱EKS 自動模式。
- **iam** – 列出連接至 IAM 角色的所有受管政策。這是必要項目，以便 Amazon EKS 可以列出和驗證建立叢集所需的所有受管政策和許可。
- 將 VPC 與託管區域建立關聯 – Amazon EKS 必須執行這項操作，才能為您的 Kubernetes 叢集 API 伺服器啟用私有端點聯網。
- Log event (日誌事件) – 這是必要項目，以便 Amazon EKS 可以將 Kubernetes 控制平面日誌運送到 CloudWatch。
- 放置指標 – 這是必要的，以便 Amazon EKS 可以將 Kubernetes 控制平面日誌運送到 CloudWatch。
- **eks** - 管理叢集存取項目和政策，允許精細控制誰可以存取 EKS 資源，以及他們可以執行哪些動作。這包括關聯運算、聯網、負載平衡和儲存操作的標準存取政策。
- **elasticloadbalancing** - 建立、管理和刪除與 EKS 叢集相關聯的負載平衡器及其元件 （接聽程式、目標群組、憑證）。檢視負載平衡器屬性和運作狀態。
- **events** - 建立和管理 EventBridge 規則，以監控與 EKS 叢集相關的 EC2 和 AWS 運作狀態事件，從而自動回應基礎設施變更和運作狀態提醒。
- **iam** - 使用「eks」字首管理 EC2 執行個體描述檔，包括建立、刪除和角色關聯，這是 EKS 節點管理所需的。
- **pricing & shield** - 存取 AWS 定價資訊和 Shield 保護狀態，為 EKS 資源啟用成本管理和進階安全功能。
- 資源清除 - 在叢集清除操作期間安全地刪除 EKS 標記的資源，包括磁碟區、快照、啟動範本和網路介面。

若要檢視最新版本的 JSON 政策文件，請參閱《 AWS 受管政策參考指南》中的 AmazonEKSServiceRolePolicy。

AWS 受管政策：AmazonEKSVPCResourceController

您可將 AmazonEKSVPCResourceController 政策連接到 IAM 身分。如果您使用 Pod 的安全群組，您必須將此政策連接至 Amazon EKS 叢集 IAM 角色，才能代表您執行動作。

此政策會授予叢集角色許可，以管理節點的彈性網路介面和 IP 地址。

許可詳細資訊

此政策包含下列許可，這些許可能讓 Amazon EKS 完成下列任務：

- **ec2** – 管理彈性網路介面和 IP 地址，以支援 Pod 安全群組和 Windows 節點。

若要檢視最新版本的 JSON 政策文件，請參閱《AWS 受管政策參考指南》中的 AmazonEKSVPCResourceController。

AWS 受管政策：AmazonEKSWorkerNodePolicy

您可以將 AmazonEKSWorkerNodePolicy 連接到 IAM 實體。您必須將此政策連接至您建立 Amazon EC2 節點時指定的 IAM 角色，從而可讓 Amazon EKS 代表您執行動作。如果您使用 eksctl 建立節點群組，則其會建立節點 IAM 角色並自動將此政策連接至角色。

此政策授予 Amazon EKS Amazon EC2 節點許可，以連接到 Amazon EKS 叢集。

許可詳細資訊

此政策包含下列許可，這些許可能讓 Amazon EKS 完成下列任務：

- **ec2** – 讀取執行個體磁碟區和網路資訊。如此一來，Kubernetes 節點才能描述節點加入 Amazon EKS 叢集所需的 Amazon EC2 資源相關資訊，這是必要的。
- **eks** – 選擇性地將叢集描述為節點引導的一部分。
- **eks-auth:AssumeRoleForPodIdentity** – 允許擷取節點上 EKS 工作負載的登入資料。必須要有這項，EKS Pod 身分識別才能正常運作。

若要檢視 JSON 政策文件的最新版本，請參閱《AWS 受管政策參考指南》中的 AmazonEKSWorkerNodePolicy。

AWS 受管政策：AmazonEKSWorkerNodeMinimalPolicy

您可以將 AmazonEKSWorkerNodeMinimalPolicy 連接至您的 IAM 實體。您可以將此政策連接至您在建立允許 Amazon EKS 代表您執行動作的 Amazon EC2 節點時指定的節點 IAM 角色。

此政策授予 Amazon EKS Amazon EC2 節點許可，以連接到 Amazon EKS 叢集。相較於 AmazonEKSWorkerNodePolicy，此政策的許可較少。

許可詳細資訊

此政策包含下列許可，這些許可能讓 Amazon EKS 完成下列任務：

- `eks-auth:AssumeRoleForPodIdentity` - 允許在節點上擷取 EKS 工作負載的登入資料。必須要有這項，EKS Pod 身分識別才能正常運作。

若要檢視 JSON 政策文件的最新版本，請參閱《 AWS 受管政策參考指南》中的 AmazonEKSWorkerNodePolicy。

AWS 受管政策：AWSServiceRoleForAmazonEKSNodegroup

您無法`AWSServiceRoleForAmazonEKSNodegroup`連接至 IAM 實體。此政策會連接到服務連結角色，而此角色可讓 Amazon EKS 代表您執行動作。如需詳細資訊，請參閱the section called “Amazon EKS 的服務連結角色許可”。

此政策可授予 `AWSServiceRoleForAmazonEKSNodegroup` 角色許可，允許它在您的帳戶中建立和管理 Amazon EC2 節點群組。

許可詳細資訊

此政策包含下列許可，這些許可能讓 Amazon EKS 完成下列任務：

- **`ec2`** – 使用安全群組、標籤、容量保留和啟動範本。這是 Amazon EKS 受管節點群組啟用遠端存取組態，以及描述可用於受管節點群組的容量保留的必要項目。此外，Amazon EKS 受管節點群組代表您建立啟動範本。這是為了設定支援每個受管節點群組的 Amazon EC2 Auto Scaling 群組。
- **`iam`** – 建立服務連結角色並傳遞角色。Amazon EKS 受管節點群組必須執行這項操作，才能管理建立受管節點群組時所傳遞之角色的執行個體描述檔。此執行個體描述檔適用於作為受管節點群組一部分的 Amazon EC2 執行個體。Amazon EKS 需要為其他服務 (例如 Amazon EC2 Auto Scaling 群組) 建立服務連結角色。這些許可會用於建立受管節點群組。
- **`autoscaling`** – 使用安全 Auto Scaling 群組。Amazon EKS 受管節點群組必須執行這項操作，才能管理支援每個受管節點群組的 Amazon EC2 Auto Scaling 群組。它也用於支援功能，例如在節點群組更新期間終止或回收節點時移出 Pod。

若要檢視最新版本的 JSON 政策文件，請參閱《 AWS 受管政策參考指南》中的 AWSServiceRoleForAmazonEKSNodegroup。

AWS 受管政策：AmazonEKSDashboardServiceRolePolicy

您無法AmazonEKSDashboardServiceRolePolicy連接至 IAM 實體。此政策會連接到服務連結角色，而此角色可讓 Amazon EKS 代表您執行動作。如需詳細資訊，請參閱the section called “Amazon EKS 的服務連結角色許可”。

此政策可授予 AWSServiceRoleForAmazonEKSDashboard 角色許可，允許它在您的帳戶中建立和管理 Amazon EC2 節點群組。

許可詳細資訊

此政策包含下列許可，允許存取 來完成這些任務：

- **organizations** – 檢視 Organizations AWS 結構和帳戶的相關資訊。這包括列出組織中帳戶的許可、檢視組織單位和根目錄、列出委派管理員、檢視可存取您組織的服務，以及擷取組織和帳戶的詳細資訊。

若要檢視 JSON 政策文件的最新版本，請參閱《 AWS 受管政策參考指南》中的 AmazonEKSDashboardServiceRolePolicy。

AWS 受管政策：AmazonEBSCSIDriverPolicy

AmazonEBSCSIDriverPolicy 政策允許 Amazon EBS 容器儲存介面 (CSI) 驅動程式代表您建立、修改、連接、分離和刪除磁碟區。這包括修改現有磁碟區的標籤，並在 EBS 磁碟區上啟用快速快照還原 (FSR)。它還授予 EBS CSI 驅動程式許可，以建立、還原和刪除快照，以及列出您的執行個體、磁碟區和快照。

若要檢視 JSON 政策文件的最新版本，請參閱《 AWS 受管政策參考指南》中的 AmazonEBSCSIDriverServiceRolePolicy。

AWS 受管政策：AmazonEFSCSIDriverPolicy

AmazonEFSCSIDriverPolicy 政策可讓 Amazon EFS 容器儲存介面 (CSI) 代表您建立和刪除存取點。它還授予 Amazon EFS CSI 驅動程式許可，列出您的存取點檔案系統、掛載目標，以及 Amazon EC2 可用區域。

若要檢視最新版本的 JSON 政策文件，請參閱《 AWS 受管政策參考指南》中的 AmazonEFSCSIDriverServiceRolePolicy。

AWS 受管政策：AmazonEKSLocalOutpostClusterPolicy

您可將此政策連接至 IAM 實體。建立本機叢集之前，您必須將此政策連接至叢集角色。由 Amazon EKS 管理的 Kubernetes 叢集會代表您呼叫其他 AWS 服務。他們會執行此作業，以管理您搭配本服務使用的資源。

AmazonEKSLocalOutpostClusterPolicy 包含以下許可：

- **ec2** 讀取動作 – 允許控制平面執行個體描述可用區域、路由表、執行個體和網路介面屬性。Amazon EC2 執行個體成功加入叢集做為控制平面執行個體所需的許可。
- **ssm** – 允許 Amazon EC2 Systems Manager 連線至控制平面執行個體，Amazon EKS 會使用該執行個體來通訊和管理您帳戶中的本機叢集。
- **logs** – 允許執行個體將日誌推送至 Amazon CloudWatch。
- **secretsmanager** – 允許執行個體從 AWS Secrets Manager 安全地取得和刪除控制平面執行個體的引導資料。
- **ecr** – 允許在控制平面執行個體上執行的 Pod 和容器提取存放在 Amazon Elastic Container Registry 中的容器映像。

若要檢視 JSON 政策文件的最新版本，請參閱《 AWS 受管政策參考指南》中的 AmazonEKSLocalOutpostClusterPolicy。

AWS 受管政策：AmazonEKSLocalOutpostServiceRolePolicy

您無法將此政策連接至您的 IAM 實體。當您使用具有 iam:CreateServiceLinkedRole許可的 IAM 主體建立叢集時，Amazon EKS 會自動為您建立 AWSServiceRoleforAmazonEKSLocalOutpost 服務連結角色，並將此政策連接到該角色。此政策允許服務連結角色代表您為本機叢集呼叫 AWS 服務。

AmazonEKSLocalOutpostServiceRolePolicy 包含以下許可：

- **ec2** – 允許 Amazon EKS 使用安全、網路和其他資源，以成功啟動和管理您帳戶中的控制平面執行個體。
- **ssm**， **ssmmessages** – 允許 Amazon EC2 Systems Manager 連線到控制平面執行個體，Amazon EKS 會使用此執行個體來通訊和管理您帳戶中的本機叢集。
- **iam** – 允許 Amazon EKS 管理與控制平面執行個體相關聯的執行個體描述檔。
- **secretsmanager** - 允許 Amazon EKS 將控制平面執行個體的引導資料放入 AWS Secrets Manager，以便在執行個體引導期間安全地參考。

- **`outposts`** – 允許 Amazon EKS 從您的帳戶取得 Outpost 資訊，以在 Outpost 中成功啟動本機叢集。

若要檢視 JSON 政策文件的最新版本，請參閱《 AWS 受管政策參考指南》中的 AmazonEKSLocalOutpostServiceRolePolicy。

AWS 受管政策的 Amazon EKS 更新

檢視自此服務開始追蹤 Amazon EKS AWS 受管政策更新以來的詳細資訊。

若要接收此特定文件頁面所有來源檔案變更的通知，您可以使用 RSS 讀取器訂閱下列 URL：

```
https://github.com/awsdocs/amazon-eks-user-guide/commits/mainline/latest/ug/security/iam-reference/security-iam-awsmanpol.adoc.atom
```

變更	描述	日期
新增許可至 the section called “AWS 受管政策：AmazonEKSLocalOutpostServiceRolePolicy”	已新增`ssmmessages:OpenDataChannel` 許可至 `AmazonEKSLocalOutpostServiceRolePolicy` 。	2025 年 6 月 26 日
已新增許可至 the section called “AWS 受管政策：AmazonEKSComputePolicy”。	更新 `ec2:RunInstances` 和 `ec2:CreateFleet` 動作的資源許可，以包含容量保留 `arn:aws: ec2:*:*:capacity-reservation/*` 。這可讓 Amazon EKS Auto 模式使用帳戶中的 EC2 隨需容量預留來啟動執行個體。新增`iam:CreateServiceLinkedRole` 以允許 Amazon EKS Auto Mode `AWSServiceRoleForEC2Spot` 代表您建立 EC2 Spot 服務連結角色。	2025 年 6 月 20 日

變更	描述	日期
新增 AmazonEKSServiceRolePolicy 的許可。	新增允許 Amazon EKS Auto Mode 使用您帳戶中的 EC2 隨需容量保留啟動執行個體的`ec2:DescribeCapacityReservations` 許可。	2025 年 6 月 20 日
已推出 the section called “AWS 受管政策：AmazonEKSDashboardConsoleReadOnly”。	引進新`AmazonEKSDashboardConsoleReadOnly` 政策。	2025 年 6 月 19 日
已推出 the section called “AWS 受管政策：AmazonEKSDashboardServiceRolePolicy”。	引進新`AmazonEKSDashboardServiceRolePolicy` 政策。	2025 年 5 月 21 日
已新增許可至 AmazonEKSClusterPolicy。	新增`ec2:DeleteNetworkInterfaces` 許可，允許 Amazon EKS 在 VPC CNI 意外退出時刪除留下的彈性網路介面。	2025 年 4 月 16 日
新增 AmazonEKSServiceRolePolicy 的許可。	新增 `ec2:RevokeSecurityGroupEgress` 和 `ec2:AuthorizeSecurityGroupEgress` 許可，允許 EKS AI/ML 客戶將安全群組輸出規則新增至與 EFA 相容的預設 EKS 叢集 SG，作為 EKS 1.33 版本發行的一部分。	2025 年 4 月 14 日
新增 AmazonEKSServiceRolePolicy 的許可。	新增終止 EKS Auto Mode 建立之 EC2 執行個體的許可。	2025 年 2 月 28 日

變更	描述	日期
新增 AmazonEBSCSIDriverPolicy 的許可。	新增了授權 EBS CSI 驅動程式還原所有快照的新陳述式。現有政策先前允許這麼做，但由於 的 IAM 處理方式有所變更，因此需要新的明確陳述式`CreateVolume` 。 新增 EBS CSI 驅動程式在現有磁碟區上修改標籤的功能。EBS CSI 驅動程式可以透過 Kubernetes VolumeAttributesClasses 中的參數修改現有磁碟區的標籤。 新增 EBS CSI 驅動程式在 EBS 磁碟區上啟用快速快照還原 (FSR) 的功能。EBS CSI 驅動程式可以透過 Kubernetes 儲存類別中的參數，在新磁碟區上啟用 FSR。	2025 年 1 月 13 日
新增許可至 the section called “AWS 受管政策：AmazonEKSLoadBalancingPolicy”。	更新`AmazonEKSLoadBalancingPolicy` 以允許列出和描述聯網和 IP 地址資源。	2024 年 12 月 26 日
新增許可至 the section called “AWS 受管政策：AWSServiceRoleForAmazonEKSNodegroup”。	更新`AWSServiceRoleForAmazonEKSNodegroup` 以相容於中國區域。	2024 年 11 月 22 日

變更	描述	日期
新增許可至 the section called “AWS 受管政策：AmazonEKSLocalOutpostClusterPolicy”	新增 的ec2:DescribeAvailabilityZones 許可，AmazonEKSLocalOutpostClusterPolicy 讓叢集控制平面上的 AWS Cloud Controller Manager 可以識別每個節點所在的可用區域。	2024 年 11 月 21 日
新增許可至 the section called “AWS 受管政策：AWSServiceRoleForAmazonEKSNodegroup”。	更新AWSServiceRoleForAmazonEKSNodegroup 政策以允許 Amazon EKS 受管節點群組建立ec2:RebootInstances 的執行個體。限制 Amazon EC2 資源的ec2:CreateTags 許可。	2024 年 11 月 20 日
新增許可至 the section called “AWS 受管政策：AmazonEKSServiceRolePolicy”。	EKS 已更新 AWS 受管政策 AmazonEKSServiceRolePolicy 。新增 EKS 存取政策、負載平衡器管理和自動叢集資源清除的許可。	2024 年 11 月 16 日
已推出 the section called “AWS 受管政策：AmazonEKSComputePolicy”。	EKS 已更新 AWS 受管政策 AmazonEKSComputePolicy 。已更新 iam:AddRoleToInstanceProfile 動作的資源許可。	2024 年 11 月 7 日
已推出 the section called “AWS 受管政策：AmazonEKSComputePolicy”。	AWS 推出 AmazonEKSComputePolicy 。	2024 年 11 月 1 日

變更	描述	日期
新增許可至 AmazonEKS ClusterPolicy	新增允許 Amazon EKS 將拓撲資訊連接至節點做為標籤的ec2:DescribeInstanceTopology 許可。	2024 年 11 月 1 日
已推出 the section called “AWS 受管政策：AmazonEKSBlockStoragePolicy”。	AWS 推出 AmazonEKS BlockStoragePolicy 。	2024 年 10 月 30 日
已推出 the section called “AWS 受管政策：AmazonEKSLoadBalancingPolicy”。	AWS 推出 AmazonEKS LoadBalancingPolicy 。	2024 年 10 月 30 日
新增 AmazonEKSServiceRolePolicy 的許可。	新增允許 Amazon EKS 將指標發佈至 Amazon CloudWatch 的cloudwatch:PutMetricData 許可。	2024 年 10 月 29 日
已推出 the section called “AWS 受管政策：AmazonEKSNetworkingPolicy”。	AWS 推出 AmazonEKS NetworkingPolicy 。	2024 年 10 月 28 日
新增對 AmazonEKS ServicePolicy 和 的許可 AmazonEKSServiceRolePolicy	新增ec2:GetSecurityGroupsForVpc 和相關聯的標籤許可，以允許 EKS 讀取安全群組資訊和更新相關標籤。	2024 年 10 月 10 日
推出 AmazonEKSWorkerNodeMinimalPolicy。	AWS 推出 AmazonEKS WorkerNodeMinimalPolicy 。	2024 年 10 月 3 日

變更	描述	日期
已新增許可至 AWSServiceRoleForAmazonEKSNodegroup。	新增允許 Amazon EKS 在 Amazon EKS 受管 Auto Scaling 群組`AZRebalance` 中暫停和繼續的 `autoscaling:ResumeProcesses` 和 `autoscaling:SuspendProcesses` 許可。	2024 年 8 月 21 日
已新增許可至 AWSServiceRoleForAmazonEKSNodegroup。	新增允許 Amazon EKS 描述使用者帳戶中容量保留的`ec2:DescribeCapacityReservations` 許可。新增在`CAPACITY_BLOCK` 節點群組上啟用設定排程擴展的`autoscaling:PutScheduledUpdateGroupAction` 許可。	2024 年 6 月 27 日
AmazonEKS_CNI_Policy – 更新至現有政策	Amazon EKS 新增了新的`ec2:DescribeSubnets` 許可，允許 Kubernetes 的 Amazon VPC CNI 外掛程式查看 Amazon VPC 子網路中的可用 IP 地址數量。VPC CNI 可以使用每個子網路中的可用 IP 地址來挑選具有建立彈性網路介面時要使用之最可用 IP 地址的子網路。	2024 年 3 月 4 日
AmazonEKSWorkerNodePolicy – 更新至現有政策	Amazon EKS 新增了新的許可來允許 EKS Pod 身分識別。Amazon EKS Pod 身分識別代理程式使用節點角色。	2023 年 11 月 26 日

變更	描述	日期
引進 AmazonEFSCSIDriverPolicy。	AWS 推出 AmazonEFSCSIDriverPolicy 。	2023 年 7 月 26 日
已新增許可至 AmazonEKSClusterPolicy。	在建立負載平衡器時，新增允許 Amazon EKS 在子網路自動探索期間取得 AZ 詳細資訊的 ec2:DescribeAvailabilityZones 許可。	2023 年 2 月 7 日
已更新 AmazonEBSCSIDriverPolicy 中的政策條件。	已移除在 StringLike 金鑰欄位中使用萬用字元的無效政策條件。也已將新條件 ec2:ResourceTag/kubernetes.io/created-for/pvc/name: "*" 新增至 ec2:DeleteVolume ，允許 EBS CSI 驅動程式刪除由樹狀內外掛程式建立的磁碟區。	2022 年 11 月 17 日
已新增許可至 AmazonEKSLocalOutpostServiceRolePolicy。	已新增 ec2:DescribeVPCAttribute 、ec2:GetConsoleOutput 和 ec2:DescribeSecret ，以允許更好的先決條件驗證和受管生命週期控制。還將 ec2:DescribePlacementGroups 和 "arn:aws: ec2:*:*:placement-group/*" 新增至 ec2:RunInstances ，以支援 Outposts 上的控制平面 Amazon EC2 執行個體的置放控制。	2022 年 10 月 24 日

變更	描述	日期
更新 AmazonEKSLocalOutpostClusterPolicy 中的 Amazon Elastic Container Registry 許可。	已將動作 `ecr:GetDownloadUrlForLayer` 從所有資源區段移至限定範圍區段。已新增資源 `arn:aws:ecr:*:*:repository/eks/` 。已移除資源 `arn:aws: ecr:`。新增的 `arn:aws: ecr:*:*:repository/eks/*` 資源涵蓋此資源。	2022 年 10 月 20 日
已新增許可至 AmazonEKSLocalOutpostClusterPolicy。	已新增 `arn:aws:ecr:*:*:repository/kubelet-config-updater` Amazon Elastic Container Registry 儲存庫，以便叢集控制平面執行個體可以更新部分 `kubelet` 引數。	2022 年 8 月 31 日
已引進 AmazonEKSLocalOutpostClusterPolicy。	AWS 推出 `AmazonEKSLocalOutpostClusterPolicy` 。	2022 年 8 月 24 日
已引進 AmazonEKSLocalOutpostServiceRolePolicy.	AWS 推出 `AmazonEKSLocalOutpostServiceRolePolicy` 。	2022 年 8 月 23 日
引進 Amazon EBS CSI Driver Policy (Amazon EBS CSI 驅動程式政策)。	AWS 推出 `AmazonEBSCSIDriverPolicy` 。	2022 年 4 月 4 日

變更	描述	日期
已新增許可至 AmazonEKSWorkerNodePolicy。	已新增 `ec2:DescribeInstanceTypes` 以啟用可自動發現發現執行個體層級屬性的 Amazon EKS 最佳化 AMI。	2022 年 3 月 21 日
已新增許可至 AWSServiceRoleForAmazonEKSNodegroup。	已新增 `autoscaling:EnableMetricsCollection` 許可以允許 Amazon EKS 啟用指標收集。	2021 年 12 月 13 日
已新增許可至 AmazonEKSClusterPolicy。	已新增 `ec2:DescribeAccountAttributes` 、`ec2:DescribeAddresses` 以及 `ec2:DescribeInternetGateways` 許可，以允許 Amazon EKS 為 Network Load Balancer 建立服務連結角色。	2021 年 6 月 17 日
Amazon EKS 已開始追蹤變更。	Amazon EKS 開始追蹤其 AWS 受管政策的變更。	2021 年 6 月 17 日

疑難排解 IAM

本主題涵蓋一些搭配 Amazon EKS 使用 IAM 時可能遇到的常見錯誤，並提供解決方法。

AccessDeniedException

如果您在呼叫 AWS API 操作`AccessDeniedException`時收到 ，則您使用的 IAM 主體憑證沒有發出該呼叫所需的許可。

```
An error occurred (AccessDeniedException) when calling the DescribeCluster operation:
User: arn:aws: iam::111122223333:user/user_name is not authorized to perform:
eks:DescribeCluster on resource: arn:aws: eks:region:111122223333:cluster/my-cluster
```

在先前的範例訊息中，使用者不具備呼叫 Amazon EKS `DescribeCluster` API 操作的許可。若要將 Amazon EKS 管理員許可提供給 IAM 主體，請參閱 the section called “身分型政策”。

如需關於 IAM 的一般資訊，請參閱《IAM 使用者指南》的使用政策控制存取。

在運算索引標籤或資源索引標籤上看不到節點，且您在 中收到錯誤 AWS Management Console

您可能會看到內容為 `Your current user or role does not have access to Kubernetes objects on this EKS cluster` 的主控台錯誤訊息。請確定您 AWS Management Console 搭配 使用 的 IAM 主體使用者具有必要的許可。如需詳細資訊，請參閱the section called “所需的許可”。

aws-auth **`ConfigMap`** 不會授予對叢集的存取權限

AWS IAM Authenticator 不允許在 中使用的角色 ARN 中使用路徑`ConfigMap`。因此，在您指定 `rolearn` 之前，請移除該路徑。例如，請將 `arn:aws:iam::111122223333:role/team/developers/eks-admin` 變更為 `arn:aws:iam::111122223333:role/eks-admin` 。

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，您的政策必須更新，以允許您將角色傳遞給 Amazon EKS。

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 EKS 中執行動作時，發生下列範例錯誤。但是，動作請求服務具備服務角色授予的許可。Mary 沒有將角色傳遞至該服務的許可。

```
User: {arn-aws}iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

在這種情況下，Mary 的政策必須更新，以允許她執行 `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 AWS 帳戶外的人員存取我的 Amazon EKS 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。

若要進一步了解，請參閱以下內容：

- 若要了解 Amazon EKS 是否支援這些功能，請參閱 the section called “Amazon EKS 和 IAM”。
- 若要了解如何在您擁有的 AWS 帳戶中提供資源的存取權，請參閱《IAM 使用者指南》中的為您擁有的另一個 AWS 帳戶中的 IAM 使用者提供存取權。
- 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擁有 AWS 的帳戶。
- 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。

Pod 容器會接收到下列錯誤：**An error occurred (SignatureDoesNotMatch) when calling the GetCallerIdentity operation: Credential should be scoped to a valid region**

如果您的應用程式明確向 AWS STS 全域端點（ `https://sts.amazonaws` ）提出請求，且您的 Kubernetes 服務帳戶設定為使用區域端點，則您的容器會收到此錯誤。您可以使用下列其中一種選項來解決此問題：

- 更新您的應用程式程式碼，以移除對 AWS STS 全域端點的明確呼叫。
- 更新應用程式的程式碼以明確呼叫區域端點，例如 `https://sts.us-west-2.amazonaws.com` 。您的應用程式應內建備援，以便在 AWS 區域中的服務故障時挑選不同的 AWS 區域。如需詳細資訊，請參閱《IAM 使用者指南》中的在 AWS 區域中管理 AWS STS。
- 將服務帳戶設定為使用全域端點。根據預設，叢集會使用區域端點。如需詳細資訊，請參閱the section called “STS 端點”。

Amazon EKS 叢集 IAM 角色

每個叢集都需要 Amazon EKS 叢集 IAM 角色。由 Amazon EKS 管理的 Kubernetes 叢集使用此角色來管理節點，而舊版雲端提供者使用此角色來建立具有 Elastic Load Balancing for 服務的負載平衡器。

建立 Amazon EKS 叢集之前，您必須先建立具有以下任一 IAM 政策的 IAM 角色：

- AmazonEKSClusterPolicy
- 自訂 IAM 政策。以下的最低許可允許 Kubernetes 叢集管理節點，但不允許舊版雲端提供者使用 Elastic Load Balancing 建立負載平衡器。您的自訂 IAM 政策必須至少擁有下列許可：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ec2:CreateTags"
      ],
      "Resource": "arn:aws: ec2:*:*:instance/*",
      "Condition": {
        "ForAnyValue:StringLike": {
          "aws:TagKeys": "kubernetes.io/cluster/*"
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": [
        "ec2:DescribeInstances",
        "ec2:DescribeNetworkInterfaces",
        "ec2:DescribeVpcs",
        "ec2:DescribeDhcpOptions",
        "ec2:DescribeAvailabilityZones",
        "ec2:DescribeInstanceTopology",
        "kms:DescribeKey"
      ],
      "Resource": "*"
    }
  ]
}
```

> **Note**
>
> 在 2023 年 10 月 3 日之前，每個叢集的 IAM 角色都必須使用 AmazonEKSClusterPolicy。在 2020 年 4 月 16 日之前，需要 AmazonEKSServicePolicy 和 AmazonEKSClusterPolicy，且該角色的建議名稱為 eksServiceRole。使用 AWSServiceRoleForAmazonEKS服務連結角色時，在 2020 年 4 月 16 日或之後建立的叢集不再需要 AmazonEKSServicePolicy 政策。

檢查現有的叢集角色

您可使用以下程序，檢查您的帳戶是否已有 Amazon EKS 叢集角色。

1. 開啟位於 https://console.aws.amazon.com/iam/ 的 IAM 主控台。
2. 在左側導覽窗格中，選擇 Roles (角色)。
3. 搜尋 eksClusterRole 的角色清單。如果eksClusterRole包含 的角色不存在，請參閱 the section called “建立 Amazon EKS 叢集角色” 以建立角色。如果包含 eksClusterRole 的角色存在，請選取角色以檢視連接的政策。
4. 選擇許可。
5. 確定 AmazonEKSClusterPolicy 受管政策已附加到該角色。如果已連接政策，則您的 Amazon EKS 叢集角色應已設定妥當。
6. 選擇 Trust Relationships (信任關係)，然後選擇 Edit trust policy (編輯信任政策)。
7. 確認信任關係包含下列政策。如果信任關係符合下列政策，請選擇 Cancel (取消)。如果信任關係不相符，請將政策複製到編輯信任政策視窗，然後選擇更新政策。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "eks.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

建立 Amazon EKS 叢集角色

您可以使用 AWS Management Console 或 AWS CLI 來建立叢集角色。

AWS Management Console

a. 開啟位於 https://console.aws.amazon.com/iam/ 的 IAM 主控台。
b. 選擇 Roles (角色)，然後選擇 Create role (建立角色)。
c. 在信任的實體類型下，選取 AWS 服務。
d. 從 AWS 其他服務的使用案例下拉式清單中，選擇 EKS。
e. 針對您的使用案例選擇 EKS - Cluster (EKS - 叢集)，然後選擇 Next (下一步)。
f. 在 Add permissions (新增許可) 標籤上，選擇 Next (下一步)。
g. 針對 Role name (角色名稱)，為您的角色輸入唯一名稱 (例如 `eksClusterRole`)。
h. 針對 Description (描述)，輸入描述性文字，如 `Amazon EKS - Cluster role`。
i. 選擇建立角色。

AWS CLI

a. 將下列內容複製到名為 *`cluster-trust-policy.json`* 的檔案。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "eks.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

b. 建立角色。您可以將 *`eksClusterRole`* 取代為您選擇的任何名稱。

```
aws iam create-role \
  --role-name eksClusterRole \
  --assume-role-policy-document file://"cluster-trust-policy.json"
```

c. 將所需的 IAM 政策連接至角色。

```
aws iam attach-role-policy \
  --policy-arn arn:aws: iam::aws:policy/AmazonEKSClusterPolicy \
  --role-name eksClusterRole
```

Amazon EKS 節點 IAM 角色

Amazon EKS kubelet 節點協助程式會代表您呼叫 AWS APIs。節點透過 IAM 執行個體描述檔和關聯的政策，取得這些 API 呼叫的許可。啟動節點並在叢集中註冊之前，您必須先為那些節點建立啟動時要使用的 IAM 角色。這項要求會套用到運用 Amazon 提供的 Amazon EKS 最佳化 AMI，或是任何其他您打算使用的節點 AMI 來啟動的節點。此外，此要求皆適用於受管節點群組和自我管理節點。

> **Note**
>
> 您無法使用用來建立任何叢集的相同角色。

建立節點之前，您必須先建立具有以下許可的 IAM 角色：

- kubelet 描述 VPC 中 Amazon EC2 資源的許可，例如 AmazonEKSWorkerNodePolicy 政策提供的許可。此政策也為 Amazon EKS Pod 身分識別代理程式提供許可。
- kubelet 使用來自 Amazon Elastic Container Registry (Amazon ECR) 的容器映像的許可，例如 AmazonEC2ContainerRegistryPullOnly 政策所提供的許可。由於用於聯網的內建附加元件會執行使用 Amazon ECR 容器映像的 Pod，因此需要 Amazon Elastic Container Registry (Amazon ECR) 容器映像的使用許可。
- (選用) Amazon EKS Pod 身分識別代理程式使用 eks-auth:AssumeRoleForPodIdentity 動作擷取 Pod 憑證的許可。如果您不使用 AmazonEKSWorkerNodePolicy，除了使用 EKS Pod Identity 的 EC2 許可之外，還必須提供此許可。
- （選用）如果您不使用 IRSA 或 EKS Pod Identity 來授予 VPC CNI Pod 的許可，則必須在執行個體角色上提供 VPC CNI 的許可。您可以使用 AmazonEKS_CNI_Policy 受管政策 (如果您使用 IPv4 系列建立叢集) 或您建立的 IPv6 政策 (如果您使用 IPv6 系列建立叢集)。但是，我們建議您將政策連接至專門用於 Amazon VPC CNI 附加元件的單獨角色，而不是將政策連接至此角色。如需為 Amazon VPC CNI 附加元件建立單獨角色的更多資訊，請參閱 the section called “為 IRSA 設定”。

> **Note**
>
> Amazon EC2 節點群組的 IAM 角色必須與 Fargate 設定檔不同。如需詳細資訊，請參閱the section called “Pod 執行 IAM 角色”。

檢查現有的節點角色

您可使用以下程序，檢查您的帳戶是否已有 Amazon EKS 節點角色。

1. 開啟位於 https://console.aws.amazon.com/iam/ 的 IAM 主控台。
2. 在左側導覽窗格中，選擇 Roles (角色)。
3. 搜尋 `eksNodeRole`、`AmazonEKSNodeRole` 或 `NodeInstanceRole` 的角色清單。如果具有這些名稱之一的角色不存在，請參閱 the section called “建立 Amazon EKS 節點 IAM 角色” 以建立角色。如果包含 `eksNodeRole`、`AmazonEKSNodeRole` 或 `NodeInstanceRole` 的角色存在，請選取角色以檢視連接的政策。
4. 選擇許可。
5. 確定 AmazonEKSWorkerNodePolicy 和 AmazonEC2ContainerRegistryPullOnly 受管政策已連接至角色，或自訂政策已附加最少的許可。

   > **Note**
   >
   > 如果 AmazonEKS_CNI_Policy 政策已連接至角色，我們建議將其移除，並轉而將其連接到映射至 `aws-node` Kubernetes 服務帳戶的 IAM 角色。如需詳細資訊，請參閱the section called “為 IRSA 設定”。

6. 選擇 Trust Relationships (信任關係)，然後選擇 Edit trust policy (編輯信任政策)。
7. 確認信任關係包含下列政策。如果信任關係符合下列政策，請選擇 Cancel (取消)。如果信任關係不相符，請將政策複製到編輯信任政策視窗，然後選擇更新政策。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "sts:AssumeRole"
            ],
```

```
            "Principal": {
                "Service": [
                    "ec2.amazonaws.com"
                ]
            }
        }
    ]
}
```

建立 Amazon EKS 節點 IAM 角色

您可以使用 AWS Management Console 或 CLI 建立節點 IAM AWS 角色。

AWS Management Console

a. 開啟位於 https://console.aws.amazon.com/iam/ 的 IAM 主控台。
b. 在左側導覽窗格中，選擇 Roles (角色)。
c. 在 Roles (角色) 頁面上，選擇 Create role (建立角色)。
d. 在 Select trusted entity (選取信任的實體) 頁面上，執行以下作業：
   i. 在信任的實體類型區段中，選擇 AWS 服務。
   ii. 在 Use case (使用案例) 下，選擇 EC2。
   iii. 選擇 Next (下一步)。
e. 在新增許可頁面上，連接自訂政策或執行下列動作：
   i. 在 Filter policies (篩選政策) 方塊中，輸入 `AmazonEKSWorkerNodePolicy`。
   ii. 勾選搜尋結果中 AmazonEKSWorkerNodePolicy 左側的核取方塊。
   iii. 選擇 Clear filters (清除篩選條件)。
   iv. 在 Filter policies (篩選政策) 方塊中，輸入 `AmazonEC2ContainerRegistryPullOnly`。
   v. 在搜尋結果中選取 AmazonEC2ContainerRegistryPullOnly 左側的核取方塊。

      AmazonEKS_CNI_Policy 受管政策或您建立的 IPv6 政策也必須連接到此角色，或連接到映射到 Kubernetes `aws-node` 服務帳戶的不同角色。建議您將政策指派給與 Kubernetes 服務帳戶相關聯的角色，而不要將政策指派給這個角色。如需詳細資訊，請參閱the section called “為 IRSA 設定”。
   vi. 選擇 Next (下一步)。
f. 在 Name, review, and create (命名、檢閱和建立) 頁面上，執行以下作業：
   i. 針對 Role name (角色名稱)，為您的角色輸入唯一名稱 (例如 `AmazonEKSNodeRole`)。

ii. 針對 Description (描述)，請以描述性文字 (例如 `Amazon EKS - Node role`) 取代目前的文字。

iii. 藉由連接標籤做為鍵值對，在新增標籤 (選用) 下將中繼資料新增至角色。如需有關在 IAM 中使用標籤的詳細資訊，請參閱《IAM 使用者指南》中的標記 IAM 資源。

iv. 選擇建立角色。

AWS CLI

a. 執行下列命令以建立 `node-role-trust-relationship.json` 檔案。

```
cat >node-role-trust-relationship.json <<EOF
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "sts:AssumeRole"
            ],
            "Principal": {
                "Service": [
                    "ec2.amazonaws.com"
                ]
            }
        }
    ]
}
EOF
```

b. 建立 IAM 角色。

```
aws iam create-role \
  --role-name AmazonEKSNodeRole \
  --assume-role-policy-document file://"node-role-trust-relationship.json"
```

c. 將兩個所需的受管 IAM 政策連接到角色。

```
aws iam attach-role-policy \
  --policy-arn arn:aws: iam::aws:policy/AmazonEKSWorkerNodePolicy \
  --role-name AmazonEKSNodeRole
aws iam attach-role-policy \
  --policy-arn arn:aws: iam::aws:policy/AmazonEC2ContainerRegistryPullOnly \
```

```
  --role-name AmazonEKSNodeRole
```

d. 根據您建立叢集所使用的 IP 系列，將以下 IAM 政策之一連接至 IAM 角色。政策必須連接到此角色，或與用於 Kubernetes 的 Amazon VPC CNI 外掛程式的 Kubernetes `aws-node`服務帳戶相關聯的角色。建議您將政策指派給與 Kubernetes 服務帳戶相關聯的角色。要將政策指派給與 Kubernetes 服務帳戶相關聯的角色，請參閱 the section called “為 IRSA 設定”。

- IPv4

```
aws iam attach-role-policy \
  --policy-arn arn:aws: iam::aws:policy/AmazonEKS_CNI_Policy \
  --role-name AmazonEKSNodeRole
```

- IPv6

A. 複製下列文字並將它儲存至名為 `vpc-cni-ipv6-policy.json` 的檔案。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ec2:AssignIpv6Addresses",
                "ec2:DescribeInstances",
                "ec2:DescribeTags",
                "ec2:DescribeNetworkInterfaces",
                "ec2:DescribeInstanceTypes"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:CreateTags"
            ],
            "Resource": [
                "arn:aws: ec2:*:*:network-interface/*"
            ]
        }
    ]
}
```

B. 建立 IAM 政策。

```
aws iam create-policy --policy-name AmazonEKS_CNI_IPv6_Policy --policy-
document file://vpc-cni-ipv6-policy.json
```

C. 將 IAM 政策連接至 IAM 角色。使用您的帳戶 ID 取代 *111122223333*。

```
aws iam attach-role-policy \
  --policy-arn arn:aws: iam::111122223333:policy/AmazonEKS_CNI_IPv6_Policy \
  --role-name AmazonEKSNodeRole
```

Amazon EKS Auto Mode 叢集 IAM 角色

每個叢集都需要 Amazon EKS 叢集 IAM 角色。Amazon EKS 管理的 Kubernetes 叢集使用此角色來自動化儲存、聯網和運算自動擴展的例行任務。

您必須先使用 EKS Auto Mode 所需的政策建立 IAM 角色，才能建立 Amazon EKS 叢集。您可以連接建議的 AWS IAM 受管政策，或建立具有同等許可的自訂政策。

- AmazonEKSComputePolicy
- AmazonEKSBlockStoragePolicy
- AmazonEKSLoadBalancingPolicy
- AmazonEKSNetworkingPolicy
- AmazonEKSClusterPolicy

檢查現有的叢集角色

您可使用以下程序，檢查您的帳戶是否已有 Amazon EKS 叢集角色。

1. 前往 https://console.aws.amazon.com/iam/ 開啟 IAM 主控台。
2. 在左側導覽窗格中，選擇 Roles (角色)。
3. 搜尋 `AmazonEKSAutoClusterRole` 的角色清單。如果`AmazonEKSAutoClusterRole`包含 的角色不存在，請參閱下一節中的指示來建立角色。如果包含 `AmazonEKSAutoClusterRole` 的角色存在，請選取角色以檢視連接的政策。
4. 選擇許可。
5. 確定 AmazonEKSClusterPolicy 受管政策已附加到該角色。如果已連接政策，則您的 Amazon EKS 叢集角色應已設定妥當。

6. 選擇 Trust Relationships (信任關係)，然後選擇 Edit trust policy (編輯信任政策)。
7. 確認信任關係包含下列政策。如果信任關係符合下列政策，請選擇 Cancel (取消)。如果信任關係不相符，請將政策複製到編輯信任政策視窗，然後選擇更新政策。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "eks.amazonaws.com"
      },
      "Action": [
        "sts:AssumeRole",
        "sts:TagSession"
      ]
    }
  ]
}
```

> **Note**
>
> AWS 不需要`AmazonEKSAutoClusterRole`此角色的名稱。

建立 Amazon EKS 叢集角色

您可以使用 AWS Management Console 或 AWS CLI 來建立叢集角色。

AWS Management Console

1. 前往 https://console.aws.amazon.com/iam/ 開啟 IAM 主控台。
2. 選擇 Roles (角色)，然後選擇 Create role (建立角色)。
3. 在信任的實體類型下，選取 AWS 服務。
4. 從 AWS 其他服務的使用案例下拉式清單中，選擇 EKS。
5. 針對您的使用案例選擇 EKS - Cluster (EKS - 叢集)，然後選擇 Next (下一步)。
6. 在新增許可索引標籤上，選取政策，然後選擇下一步。
   - AmazonEKSComputePolicy

- AmazonEKSBlockStoragePolicy
- AmazonEKSLoadBalancingPolicy
- AmazonEKSNetworkingPolicy
- AmazonEKSClusterPolicy

7. 針對 Role name (角色名稱)，為您的角色輸入唯一名稱 (例如 `AmazonEKSAutoClusterRole`)。
8. 針對 Description (描述)，輸入描述性文字，如 `Amazon EKS - Cluster role`。
9. 選擇建立角色。

AWS CLI

1. 將下列內容複製到名為 *`cluster-trust-policy.json`* 的檔案。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "eks.amazonaws.com"
      },
      "Action": [
        "sts:AssumeRole",
        "sts:TagSession"
      ]
    }
  ]
}
```

2. 建立角色。您可以將 *`AmazonEKSAutoClusterRole`* 取代為您選擇的任何名稱。

```
aws iam create-role \
  --role-name AmazonEKSAutoClusterRole \
  --assume-role-policy-document file://"cluster-trust-policy.json"
```

3. 將必要的 IAM 政策連接至角色：

AmazonEKSClusterPolicy：

```
aws iam attach-role-policy \
```

```
    --role-name AmazonEKSAutoClusterRole \
    --policy-arn arn:aws: iam::aws:policy/AmazonEKSClusterPolicy
```

AmazonEKSComputePolicy：

```
aws iam attach-role-policy \
    --role-name AmazonEKSAutoClusterRole \
    --policy-arn arn:aws: iam::aws:policy/AmazonEKSComputePolicy
```

AmazonEKSBlockStoragePolicy：

```
aws iam attach-role-policy \
    --role-name AmazonEKSAutoClusterRole \
    --policy-arn arn:aws: iam::aws:policy/AmazonEKSBlockStoragePolicy
```

AmazonEKSLoadBalancingPolicy：

```
aws iam attach-role-policy \
    --role-name AmazonEKSAutoClusterRole \
    --policy-arn arn:aws: iam::aws:policy/AmazonEKSLoadBalancingPolicy
```

AmazonEKSNetworkingPolicy：

```
aws iam attach-role-policy \
    --role-name AmazonEKSAutoClusterRole \
    --policy-arn arn:aws: iam::aws:policy/AmazonEKSNetworkingPolicy
```

Amazon EKS Auto Mode 節點 IAM 角色

> **Note**
>
> 您無法使用用來建立任何叢集的相同角色。

建立節點之前，您必須使用下列政策或同等許可來建立 IAM 角色：

- AmazonEKSWorkerNodeMinimalPolicy
- AmazonEC2ContainerRegistryPullOnly

檢查現有的節點角色

您可使用以下程序，檢查您的帳戶是否已有 Amazon EKS 節點角色。

1. 前往 https://console.aws.amazon.com/iam/ 開啟 IAM 主控台。
2. 在左側導覽窗格中，選擇 Roles (角色)。
3. 搜尋 AmazonEKSAutoNodeRole 的角色清單。如果具有這些名稱之一的角色不存在，請參閱下一節中的指示來建立角色。如果包含 AmazonEKSAutoNodeRole 的角色存在，請選取角色以檢視連接的政策。
4. 選擇許可。
5. 確定已連接上述必要政策或同等自訂政策。
6. 選擇 Trust Relationships (信任關係)，然後選擇 Edit trust policy (編輯信任政策)。
7. 確認信任關係包含下列政策。如果信任關係符合下列政策，請選擇 Cancel (取消)。如果信任關係不相符，請將政策複製到編輯信任政策視窗，然後選擇更新政策。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "ec2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

建立 Amazon EKS 節點 IAM 角色

您可以使用 AWS Management Console 或 CLI 建立節點 IAM AWS 角色。

AWS Management Console

1. 前往 https://console.aws.amazon.com/iam/ 開啟 IAM 主控台。
2. 在左側導覽窗格中，選擇 Roles (角色)。
3. 在 Roles (角色) 頁面上，選擇 Create role (建立角色)。

4. 在 Select trusted entity (選取信任的實體) 頁面上，執行以下作業：
   a. 在信任的實體類型區段中，選擇 AWS 服務。
   b. 在 Use case (使用案例) 下，選擇 EC2。
   c. 選擇下一步。
5. 在新增許可頁面上，連接下列政策：
   - AmazonEKSWorkerNodeMinimalPolicy
   - AmazonEC2ContainerRegistryPullOnly
6. 在 Name, review, and create (命名、檢閱和建立) 頁面上，執行以下作業：
   a. 針對 Role name (角色名稱)，為您的角色輸入唯一名稱 (例如 `AmazonEKSAutoNodeRole`)。
   b. 針對 Description (描述)，請以描述性文字 (例如 `Amazon EKS - Node role`) 取代目前的文字。
   c. 藉由連接標籤做為鍵值對，在新增標籤 (選用) 下將中繼資料新增至角色。如需有關在 IAM 中使用標籤的詳細資訊，請參閱《IAM 使用者指南》中的標記 IAM 資源。
   d. 選擇建立角色。

AWS CLI

建立節點 IAM 角色

使用上一個步驟中的 node-trust-policy.json 檔案來定義哪些實體可以擔任該角色。執行下列命令來建立節點 IAM 角色：

```
aws iam create-role \
    --role-name AmazonEKSAutoNodeRole \
    --assume-role-policy-document file://node-trust-policy.json
```

記下角色 ARN

建立角色之後，請擷取並儲存節點 IAM 角色的 ARN。在後續步驟中，您將需要此 ARN。使用下列命令來取得 ARN：

```
aws iam get-role --role-name AmazonEKSAutoNodeRole --query "Role.Arn" --output text
```

連接必要的政策

將下列 AWS 受管政策連接至節點 IAM 角色，以提供必要的許可：

若要連接 AmazonEKSWorkerNodeMinimalPolicy：

```
aws iam attach-role-policy \
    --role-name AmazonEKSAutoNodeRole \
    --policy-arn arn:aws: iam::aws:policy/AmazonEKSWorkerNodeMinimalPolicy
```

若要連接 AmazonEC2ContainerRegistryPullOnly：

```
aws iam attach-role-policy \
    --role-name AmazonEKSAutoNodeRole \
    --policy-arn arn:aws: iam::aws:policy/AmazonEC2ContainerRegistryPullOnly
```

監控叢集效能並檢視日誌

您可以使用許多可用的監控或日誌記錄工具在 Amazon EKS 中觀察您的資料。您的 Amazon EKS 日誌資料可以串流到 AWS 服務或合作夥伴工具以進行資料分析。中有許多 服務可提供 AWS Management Console 資料，以針對 Amazon EKS 問題進行故障診斷。您也可以使用 AWS支援的開放原始碼解決方案來監控 Amazon EKS 基礎設施。

在 Amazon EKS 主控台的左側導覽窗格中選取叢集後，您可以選擇叢集的名稱並選擇可觀測性索引標籤，以檢視叢集運作狀態和詳細資訊。若要檢視有關部署到叢集的任何現有 Kubernetes 資源的詳細資訊，請參閱 the section called “存取叢集資源”。

監控是維護 Amazon EKS 和 AWS 解決方案可靠性、可用性和效能的重要部分。我們建議您從 AWS 解決方案的所有部分收集監控資料。這樣，如果出現多點故障，您可以更輕鬆地進行偵錯。開始監控 Amazon EKS 前，請確保您的監控計畫可以解決下列問題。

- 您的目標是什麼？ 如果叢集大幅擴展，您是否需要即時通知？
- 需要觀察哪些資源？
- 您需要多長時間觀察這些資源？ 貴公司是否希望快速應對風險？
- 您要使用哪些工具？ 如果您已在啟動時執行 AWS Fargate，則可以使用內建日誌路由器。
- 您打算由誰執行監控任務？
- 當出現問題時，您希望向誰傳送通知？

在 Amazon EKS 上監控和記錄

Amazon EKS 提供用於監控和記錄的內建工具。對於支援的版本，可觀測性儀表板可讓您了解叢集的效能。它可協助您快速偵測、疑難排解和修復問題。除了監控功能之外，還包含以控制平面稽核日誌為基礎的清單。Kubernetes 控制平面公開了許多也可以在主控台外部抓取的指標。

控制平面日誌記錄工具記錄對叢集的所有 API 呼叫、稽核資訊 (擷取哪些使用者對叢集執行哪些操作)，以及以角色為基礎的資訊。如需詳細資訊，請參閱 AWS 規範指引中的在 Amazon EKS 上記錄和監控。

Amazon EKS 控制平面記錄從 Amazon EKS 控制平面將稽核和診斷日誌直接提供至您帳戶中的 CloudWatch Logs。這些日誌可讓您輕鬆執行叢集並確保叢集的安全。您可以選取所需的確切日誌類型，且日誌將以日誌串流傳送至 CloudWatch 中各個 Amazon EKS 叢集的群組中。如需詳細資訊，請參閱the section called “控制平面日誌”。

> **Note**
>
> 當您檢查 Amazon CloudWatch 中的 Amazon EKS 驗證器日誌時，會顯示包含類似下列範例文字的項目。
>
> ```
> level=info msg="mapping IAM role" groups="[]" role="arn:aws:
>  iam::111122223333:role/XXXXXXXXXXXXXXXXXX-NodeManagerRole-XXXXXXXX"
>  username="eks:node-manager"
> ```
>
> 預期應包含此文字的項目。`username` 是 Amazon EKS 內部服務角色，可對受管節點群組和 Fargate 執行特定操作。
> 對於低層級、可自訂的日誌記錄，可以使用 Kubernetes 日誌記錄。

Amazon EKS 已與 AWS CloudTrail 整合，CloudTrail 是一種服務，可提供使用者、角色或 Amazon EKS 中 AWS 服務所採取動作的記錄。CloudTrail 會將 Amazon EKS 的所有 API 呼叫擷取為事件。擷取的呼叫包括從 Amazon EKS 主控台執行的呼叫，以及對 Amazon EKS API 作業發出的程式碼呼叫。如需詳細資訊，請參閱the section called “ AWS CloudTrail”。

Kubernetes API 伺服器公開多個可用於監控和分析的指標。如需詳細資訊，請參閱the section called “Prometheus 指標”。

若要為自訂 Amazon CloudWatch logs設定 Fluent Bit，請參閱《Amazon CloudWatch 使用者指南》中的設定 Fluent Bit。

Amazon EKS 監控和記錄工具

Amazon Web Services 提供各種工具讓您可用於監控 Amazon EKS。您可以設定某些工具來設定自動監控，但有些工具則需要手動呼叫。建議您在您的環境和現有工具集允許的範圍內自動執行監控任務。

下表說明各種監控工具選項。

區域	工具	描述	設定
控制平台	可觀測性儀表板	對於支援的版本，可觀測性儀表板可讓您了解叢集的效能。它可協助您快速偵測、	設定程序

區域	工具	描述	設定
		疑難排解和修復問題。	
應用程式/控制平面	Prometheus	Prometheus 可用來監控應用程式和控制平面的指標和提醒。	設定程序
應用程式	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。	設定程序
應用程式	AWS Distro for OpenTelemetry (ADOT)	ADOT 可以收集相關指標、追蹤資料和中繼資料，並將其傳送至 AWS 監控服務或合作夥伴。可以透過 CloudWatch Container Insights 進行設定。	設定程序
應用程式	Amazon DevOps Guru	Amazon DevOps Guru 會偵測節點層級的操作效能和可用性。	設定程序

區域	工具	描述	設定
應用程式	AWS X-Ray	AWS X-Ray 會接收有關應用程式的追蹤資料。此追蹤資料包括傳入和傳出請求以及有關請求的中繼資料。對於 Amazon EKS，實作需要 OpenTelemetry 附加元件。	設定程序
應用程式	Amazon CloudWatch	CloudWatch 在支援的版本上免費提供一些基本的 Amazon EKS 指標。您可以使用 CloudWatch 可觀測性運算子擴展此功能，以處理收集指標、日誌和追蹤資料。	設定程序

下表說明各種記錄工具選項。

區域	工具	描述	設定
控制平台	可觀測性儀表板	對於支援的版本，可觀測性儀表板會根據控制平面稽核日誌顯示清單。它還包含控制 Amazon CloudWatch 中平面日誌的連結。	設定程序
應用程式	Amazon CloudWatch Container Insights	Amazon CloudWatch Container Insights 會從容器化應用程式和	設定程序

區域	工具	描述	設定
		微服務收集、彙總和摘要指標和日誌。	
控制平台	Amazon CloudWatch Logs	您可以將稽核和診斷日誌直接從 Amazon EKS 控制平面傳送至您帳戶中的 CloudWatch Logs。	設定程序
控制平台	AWS CloudTrail	它記錄由使用者、角色或服務所進行的 API 呼叫。	設定程序
AWS Fargate 執行個體的多個區域	AWS Fargate 日誌路由器	對於 AWS Fargate 執行個體，日誌路由器會將日誌串流到 AWS 服務或合作夥伴工具。它使用 AWS 作為 Fluent Bit。日誌可以串流到其他 AWS 服務或合作夥伴工具。	設定程序

使用可觀測性儀表板監控您的叢集

Amazon EKS 主控台包含可觀測性儀表板，可讓您了解叢集的效能。它提供的資訊可協助您快速偵測、疑難排解和修復問題。您可以在運作狀態和效能摘要中選擇項目，開啟可觀測性儀表板的適用區段。此摘要包含在多個地方，包括可觀測性索引標籤。

可觀測性儀表板分為數個索引標籤。

Summary

運作狀態和效能摘要會列出各種類別中的項目數量。每個數字都充當可觀測性儀表板中位置的超連結，其中包含該類別的清單。

叢集運作狀態

叢集運作狀態提供需要注意的重要通知，其中一些可能需要儘快採取行動。透過此清單，您可以查看描述和受影響的資源。叢集運作狀態包含兩個資料表：運作狀態問題和組態洞察。若要重新整理運作狀態問題的狀態，請選擇重新整理按鈕 (↻)。組態洞見每 24 小時自動更新一次，無法手動重新整理。

如需運作狀態問題的詳細資訊，請參閱 the section called “叢集運作狀態FAQs和具有解析路徑的錯誤代碼”。如需組態洞見的詳細資訊，請參閱 the section called “叢集洞察”。

控制平面監控

控制平面監控索引標籤分為三個區段，每個區段都可協助您監控叢集的控制平面並進行疑難排解。

指標

對於 Kubernetes 版本 `1.28` 及更高版本的叢集，指標區段會顯示針對各種控制平面元件收集的數個指標圖表。

您可以在區段頂端進行選擇，以設定每個圖形的 X 軸所使用的時段。您可以使用重新整理按鈕重新整理資料 (↻)。對於每個單獨的圖形，垂直省略號按鈕 (⋮) 會開啟具有 CloudWatch 選項的功能表。

這些指標等等會自動做為 CloudWatch 中`AWS/EKS`命名空間下的基本監控指標。如需詳細資訊，請參閱《Amazon CloudWatch 使用者指南》中的基本監控和詳細監控。若要取得更詳細的指標、視覺化和洞見，請參閱《Amazon CloudWatch 使用者指南》中的容器洞見。或者，如果您偏好以 Prometheus 為基礎的監控，請參閱 the section called “Prometheus 指標”。

下表說明可用的指標。

指標	描述
APIServer 請求	對 API 伺服器發出的每分鐘請求數。
APIServer 請求總數 4XX	每分鐘具有 HTTP 4XX 回應碼（用戶端錯誤）的 API 伺服器請求計數。
APIServer 請求總數 5XX	每分鐘具有 HTTP 5XX 回應碼（伺服器端錯誤）的 API 伺服器請求計數。
APIServer 請求總數 429	每分鐘具有 HTTP 429 回應碼（太多請求）的 API 伺服器請求計數。

指標	描述
儲存體大小	儲存資料庫 (etcd) 大小。
排程器嘗試	依結果 "unschedulable" "error" 和 "scheduled" 排程 Pod 的嘗試次數。
待定 Pod	依「作用中」、「退避」、「不可排程」和「門控」佇列類型的待處理 Pod 數量。
API 伺服器請求延遲	API 伺服器請求的延遲。
API 伺服器目前的傳輸中請求	API 伺服器的目前進行中請求。
Webhook 請求	每分鐘的 Webhook 請求數。
Webhook 請求拒絕	已拒絕的 Webhook 請求計數。
Webhook 請求延遲 P99	外部第三方 Webhook 請求的第 99 個百分位數延遲。

CloudWatch Log Insights

CloudWatch Log Insights 區段會根據控制平面稽核日誌顯示各種清單。需要開啟 Amazon EKS 控制平面日誌才能使用此功能，您可以從 CloudWatch 中的檢視控制平面日誌區段執行此操作。

當已過足夠的時間收集資料時，您可以執行所有查詢，或一次為單一清單選擇執行查詢。每當您執行查詢時CloudWatch 都會產生額外費用。選擇您要在區段頂端檢視的結果時段。如果您想要對任何查詢進行更進階的控制，您可以選擇在 CloudWatch 中檢視。這可讓您更新 CloudWatch 中的查詢，以符合您的需求。

如需詳細資訊，請參閱《Amazon CloudWatch Logs 使用者指南》中的使用 CloudWatch Logs Insights 分析日誌資料。 Amazon CloudWatch

在 CloudWatch 中檢視控制平面日誌

選擇管理記錄以更新可用的日誌類型。啟用記錄後，日誌需要幾分鐘才會出現在 CloudWatch Logs 中。經過足夠的時間後，請選擇本節中的任何檢視連結，以導覽至適用的日誌。

如需詳細資訊，請參閱the section called “控制平面日誌”。

叢集洞察

升級洞見表同時顯示問題並建議修正動作，加速升級至新 Kubernetes 版本的驗證程序。Amazon EKS 會根據影響問題的潛在 Kubernetes 版本升級清單自動掃描叢集。升級洞見資料表列出 Amazon EKS 針對此叢集執行的洞見檢查，以及其相關聯的狀態。

Amazon EKS 會根據 Kubernetes 專案中的變更評估，以及與新版本相關的 Amazon EKS 服務變更，維護並定期重新整理要執行的洞見檢查清單。Amazon EKS 主控台會自動重新整理每個洞見的狀態，這可以在上次重新整理時間欄中看到。

如需詳細資訊，請參閱the section called “叢集洞察”。

節點運作狀態問題

Amazon EKS 節點監控代理程式會自動讀取節點日誌，以偵測運作狀態問題。無論自動修復設定為何，都會報告所有節點運作狀態問題，以便您可以視需要進行調查。如果列出沒有描述的問題類型，您可以讀取其彈出式元素中的描述。

當您重新整理頁面時，任何已解決的問題都會從清單中消失。如果啟用自動修復，您可以暫時看到一些可在不採取任何動作的情況下解決的安全問題。自動修復不支援的問題可能需要您手動動作，視類型而定。

若要報告節點運作狀態問題，您的叢集必須使用 Amazon EKS Auto 模式或具有節點監控代理程式附加元件。如需詳細資訊，請參閱the section called “節點運作狀態”。

使用 Prometheus 監控您的叢集指標

Prometheus 是一種監控和時間序列資料庫，可抓取端點。它提供查詢、彙總和儲存收集之資料的能力。您也可以將其用於警示和警示彙總。本主題說明如何將 Prometheus 設定為受管或開放原始碼選項。監控 Amazon EKS 控制平面指標是常見的使用案例。

Amazon Managed Service for Prometheus 是與 Prometheus 相容的監控和提示服務，可讓您輕鬆地大規模監控容器化應用程式和基礎設施。這是一項全受管服務，既可自動擴展指標的擷取、儲存、查詢和提醒，它還與 AWS 安全服務整合，以快速安全地存取您的資料。您可以使用開放原始碼 PromQL 查詢語言來查詢指標並根據指標發出提醒。此外，您可以使用 Amazon Managed Service for Prometheus 中的警示管理員來設定重要警示的警示規則。然後，您可以將這些重要提醒作為通知傳送至 Amazon SNS 主題。

搭配 Amazon EKS 使用 Prometheus 有幾種不同的選項：

- 您可以在第一次建立 Amazon EKS 叢集時開啟 Prometheus 指標，也可以為現有叢集建立自己的 Prometheus 湊集器。本主題涵蓋這兩個選項。
- 您可以使用 Helm 部署 Prometheus。如需詳細資訊，請參閱the section called “使用 Helm 部署”。
- 您可以檢視 Prometheus 格式的控制平面原始指標。如需詳細資訊，請參閱the section called “控制平台”。

步驟 1：開啟 Prometheus 指標

> **Important**
>
> Amazon Managed Service for Prometheus 資源不在叢集生命週期內，需要獨立於叢集進行維護。當您刪除叢集時，請務必同時刪除任何適用的抓取器，以停止適用的成本。如需詳細資訊，請參閱《Amazon Managed Service for Prometheus 使用者指南》中的尋找和刪除抓取器。

Prometheus 透過稱為抓取的提取型模型，從您的叢集探索和收集指標。湊集器的設定是為了從您的叢集基礎設施和容器化應用程式收集資料。當您開啟傳送 Prometheus 指標的選項時，Amazon Managed Service for Prometheus 會提供全受管的無代理程式湊集器。

如果您尚未建立叢集，您可以在第一次建立叢集時開啟傳送指標至 Prometheus 的選項。在 Amazon EKS 主控台中，此選項位於建立新叢集的設定可觀測性步驟中。如需詳細資訊，請參閱the section called “建立叢集”。

如果您已有現有的叢集，您可以建立自己的 Prometheus 湊集器。若要在 Amazon EKS 主控台中執行此操作，請導覽至叢集的可觀測性索引標籤，然後選擇新增湊集器按鈕。如果您想要使用 AWS API 或 AWS CLI 執行此操作，請參閱《Amazon Managed Service for Prometheus 使用者指南》中的建立抓取器。

使用 Amazon EKS 主控台建立抓取器時，可使用下列選項。

Scraper 別名

(選用) 輸入湊集器的唯一別名。

目的地

選擇 Amazon Managed Service for Prometheus 工作區。工作區是專門用於儲存和查詢 Prometheus 指標的邏輯空間。使用此工作區，您將能夠檢視可存取 Prometheus 指標之帳戶的

Prometheus 指標。建立新工作區選項會讓 Amazon EKS 知道可以使用您提供的工作區別名來代表您建立工作區。使用選取現有工作區選項，您可以從下拉式清單中選取現有的工作區。如需工作區的詳細資訊，請參閱《Amazon Managed Service for Prometheus 使用者指南》中的管理工作區。

服務存取

本節摘要說明您在傳送 Prometheus 指標時授予的許可：

- 允許 Amazon Managed Service for Prometheus 描述湊集的 Amazon EKS 叢集
- 允許遠端寫入 Amazon Managed Prometheus 工作區

如果 `AmazonManagedScraperRole` 已經存在，則湊集器會使用它。
選擇 `AmazonManagedScraperRole` 連結以查看許可詳細資訊。如果
尚`AmazonManagedScraperRole`不存在，請選擇檢視許可詳細資訊連結，透過傳送 Prometheus 指標來查看您授予的特定許可。

子網路

視需要修改抓取器將繼承的子網路。如果您需要新增灰色的子網路選項，請返回建立叢集 指定聯網步驟。

碎片組態

視需要修改 YAML 格式的湊集器組態。若要執行這項操作，請使用表單或上傳取代 YAML 檔案。如需詳細資訊，請參閱《Amazon Managed Service for Prometheus 使用者指南》中的湊集器組態。

Amazon Managed Service for Prometheus 是指與叢集一起建立的無代理程式湊集器，做為 AWS 受管收集器。如需 AWS 受管收集器的詳細資訊，請參閱《Amazon Managed Service for Prometheus 使用者指南》中的使用 AWS 受管收集器擷取指標。

Important

- 如果您使用 CLI AWS 或 AWS API 建立 Prometheus 湊集器，則需要調整其組態，以授予湊集器叢集內許可。如需詳細資訊，請參閱《Amazon Managed Service for Prometheus 使用者指南》中的設定 Amazon EKS 叢集。
- 如果您在 2024 年 11 月 11 日之前建立使用 `aws-authConfigMap`而非存取項目的 Prometheus 湊集器，則需要更新它以從 Amazon EKS 叢集控制平面存取其他指標。如需更新組態，請參閱《Amazon Managed Service for Prometheus 使用者指南》中的手動設定 Amazon EKS 以進行抓取器存取。

步驟 2：使用 Prometheus 指標

如需有關如何在叢集開啟 Prometheus 指標之後使用它們的詳細資訊，請參閱《Amazon Managed Service for Prometheus 使用者指南》。

步驟 3：管理 Prometheus 抓取器

若要管理抓取器，請選擇 Amazon EKS 主控台中的可觀測性索引標籤。表格會顯示叢集的湊集器清單，包含湊集器 ID、別名、狀態和建立日期等資訊。您可以新增更多抓取器、編輯抓取器、刪除抓取器，或檢視目前抓取器的詳細資訊。

若要查看抓取器的詳細資訊，請選擇抓取器 ID 連結。例如，您可以檢視 ARN、環境、工作區 ID、IAM 角色、組態和聯網資訊。您可以使用抓取器 ID 做為 `DescribeScraper`、 `UpdateScraper`和 等 Amazon Managed Service for Prometheus API 操作的輸入`DeleteScraper`。如需使用 Prometheus API 的詳細資訊，請參閱 Amazon Managed Service for Prometheus API 參考。

使用 Helm 部署 Prometheus

除了使用 Amazon Managed Service for Prometheus 之外，您還可以使用 Helm 將 Prometheus 部署到您的叢集。如果您已經安裝了 Helm，可以使用 `helm version` 命令檢查您的版本。Helm 是 Kubernetes 叢集的套件管理工具。如需有關 Helm 及如何安裝它的詳細資訊，請參閱 the section called “使用 Helm 部署應用程式”。

在您為 Amazon EKS 叢集設定 Helm 之後，即可用它來依照下列步驟部署 Prometheus。

1. 建立 Prometheus 命名空間。

```
kubectl create namespace prometheus
```

2. 新增 `prometheus-community` 圖表儲存庫。

```
helm repo add prometheus-community https://prometheus-community.github.io/helm-charts
```

3. 部署 Prometheus。

```
helm upgrade -i prometheus prometheus-community/prometheus \
    --namespace prometheus \
    --set alertmanager.persistence.storageClass="gp2" \
    --set server.persistentVolume.storageClass="gp2"
```

> **Note**
>
> 如果執行此命令時收到錯誤 `Error: failed to download "stable/prometheus" (hint: running helm repo update may help)`，請執行 `helm repo update prometheus-community`，然後嘗試再次執行步驟 2 命令。

如果收到錯誤 `Error: rendered manifests contain a resource that already exists`，請執行 `helm uninstall` *`your-release-name`* `-n` *`namespace`* ，然後嘗試再次執行步驟 3 命令。

4. 確認prometheus命名空間中的所有 Pod 都處於 READY 狀態。

```
kubectl get pods -n prometheus
```

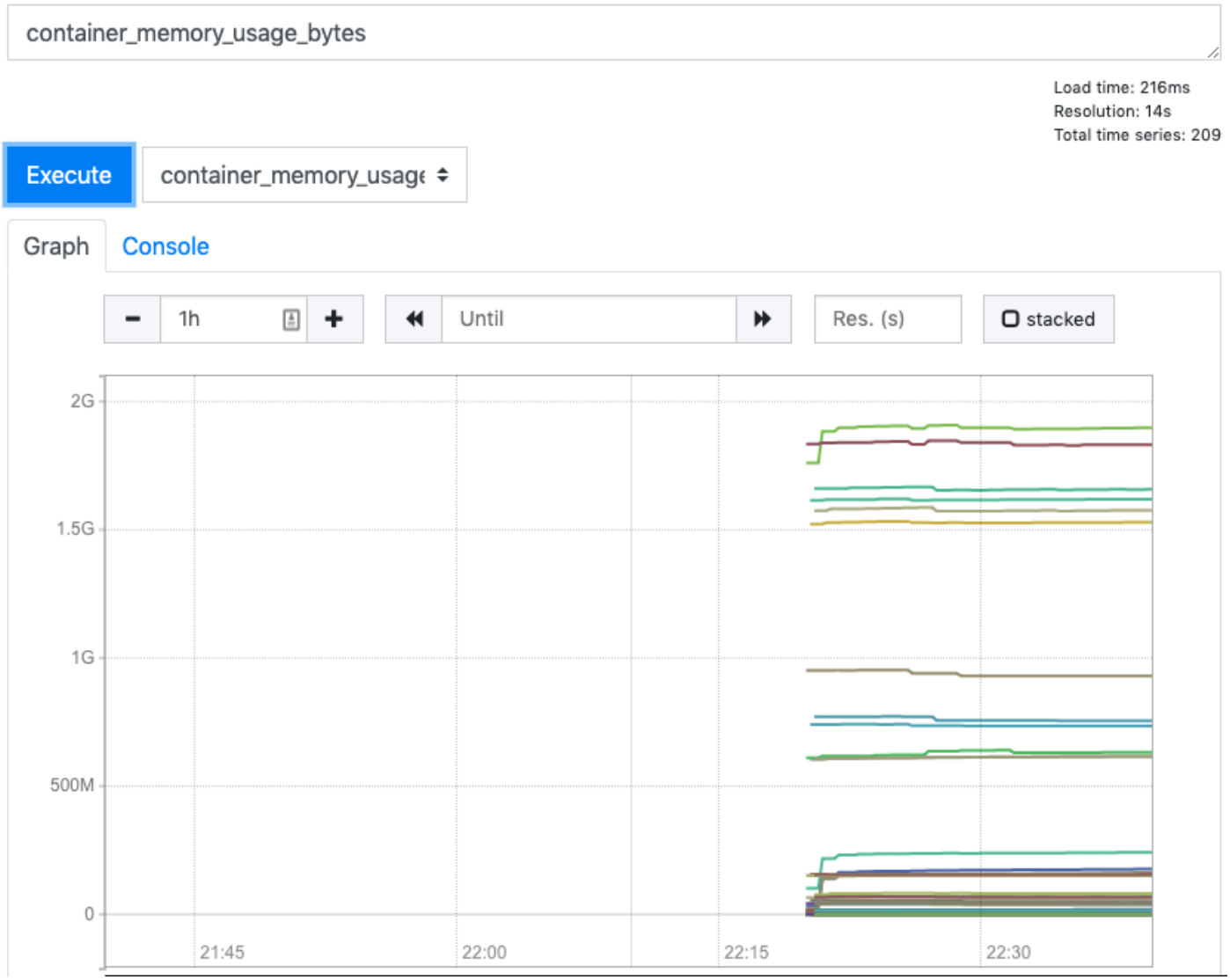
範例輸出如下。

```
NAME                                             READY   STATUS    RESTARTS   AGE
prometheus-alertmanager-59b4c8c744-r7bgp         1/2     Running   0          48s
prometheus-kube-state-metrics-7cfd87cf99-jkz2f   1/1     Running   0          48s
prometheus-node-exporter-jcjqz                   1/1     Running   0          48s
prometheus-node-exporter-jxv2h                   1/1     Running   0          48s
prometheus-node-exporter-vbdks                   1/1     Running   0          48s
prometheus-pushgateway-76c444b68c-82tnw          1/1     Running   0          48s
prometheus-server-775957f748-mmht9               1/2     Running   0          48s
```

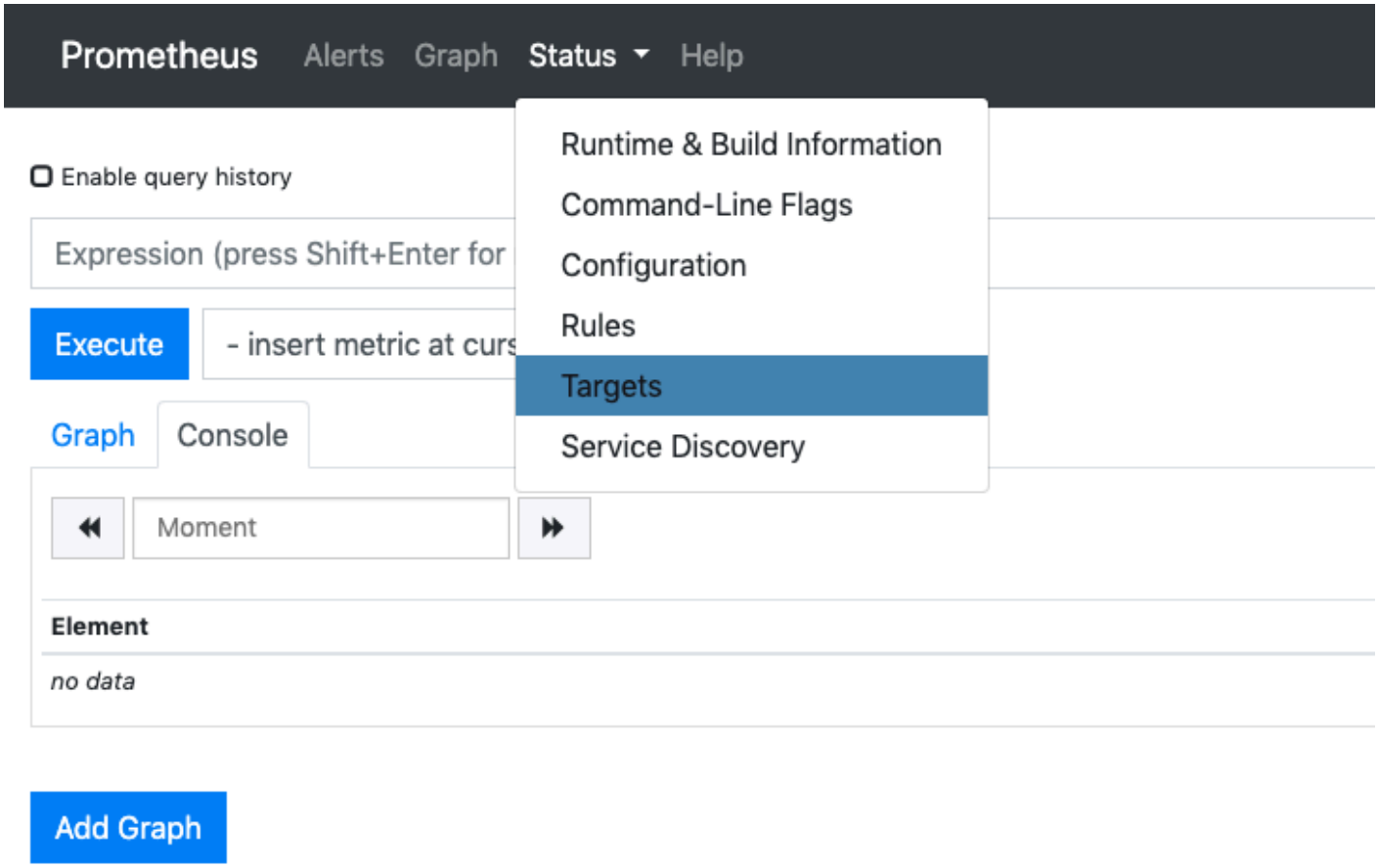
5. 使用 `kubectl` 將 Prometheus 主控台的連接埠轉送到本機機器。

```
kubectl --namespace=prometheus port-forward deploy/prometheus-server 9090
```

6. 將 Web 瀏覽器指向 `http://localhost:9090` 以檢視 Prometheus 主控台。
7. 從 – insert metric at cursor (- 在游標處插入指標) 功能表中選擇一個指標，然後選擇 Execute (執行)。選擇 Graph (圖表) 標籤以顯示一段時間內的指標。下圖顯示一段時間內的 `container_memory_usage_bytes`。



8. 選擇頂部導覽列中的 Status (狀態)，然後選擇 Targets (目標)。



將會顯示使用服務探索連接至 Prometheus 的所有 Kubernetes 端點。

擷取 Prometheus 格式的控制平面原始指標

Kubernetes 控制平面公開了以 Prometheus 格式表示的許多指標。這些指標對於監視和分析非常有用。它們會透過指標端點在內部公開，無需完全部署 Prometheus 即可存取。不過，部署 Prometheus 更輕鬆地允許分析一段時間內的指標。

若要檢視原始指標輸出，請取代 `endpoint`並執行下列命令。

```
kubectl get --raw endpoint
```

此命令可讓您傳遞任何端點路徑，並傳回原始回應。輸出line-by-line列出不同的指標，每一行都包含指標名稱、標籤和值。

```
metric_name{tag="value"[,...]} value
```

從 API 伺服器擷取指標

一般 API 伺服器端點會在 Amazon EKS 控制平面上公開。此端點主要適用於查看特定指標。

```
kubectl get --raw /metrics
```

範例輸出如下。

```
[...]
# HELP rest_client_requests_total Number of HTTP requests, partitioned by status code,
 method, and host.
# TYPE rest_client_requests_total counter
rest_client_requests_total{code="200",host="127.0.0.1:21362",method="POST"} 4994
rest_client_requests_total{code="200",host="127.0.0.1:443",method="DELETE"} 1
rest_client_requests_total{code="200",host="127.0.0.1:443",method="GET"} 1.326086e+06
rest_client_requests_total{code="200",host="127.0.0.1:443",method="PUT"} 862173
rest_client_requests_total{code="404",host="127.0.0.1:443",method="GET"} 2
rest_client_requests_total{code="409",host="127.0.0.1:443",method="POST"} 3
rest_client_requests_total{code="409",host="127.0.0.1:443",method="PUT"} 8
# HELP ssh_tunnel_open_count Counter of ssh tunnel total open attempts
# TYPE ssh_tunnel_open_count counter
ssh_tunnel_open_count 0
# HELP ssh_tunnel_open_fail_count Counter of ssh tunnel failed open attempts
# TYPE ssh_tunnel_open_fail_count counter
ssh_tunnel_open_fail_count 0
```

此原始輸出會逐字傳回 API 伺服器公開的內容。

使用 擷取控制平面指標 `metrics.eks.amazonaws.com`

對於 Kubernetes 版本 `1.28` 及更高版本的叢集，Amazon EKS 也會在 API 群組 下公開指標`metrics.eks.amazonaws.com`。這些指標包括控制平面元件，例如 `kube-scheduler`和 `kube-controller-manager`。

> **Note**
>
> 如果您的 Webhook 組態可能封鎖在叢集`v1.metrics.eks.amazonaws.com`上建立新`APIService`資源，則指標端點功能可能無法使用。您可以搜尋`v1.metrics.eks.amazonaws.com`關鍵字，在`kube-apiserver`稽核日誌中驗證 。

擷取**kube-scheduler**指標

若要擷取kube-scheduler指標，請使用下列命令。

```
kubectl get --raw "/apis/metrics.eks.amazonaws.com/v1/ksh/container/metrics"
```

範例輸出如下。

```
# TYPE scheduler_pending_pods gauge
scheduler_pending_pods{queue="active"} 0
scheduler_pending_pods{queue="backoff"} 0
scheduler_pending_pods{queue="gated"} 0
scheduler_pending_pods{queue="unschedulable"} 18
# HELP scheduler_pod_scheduling_attempts [STABLE] Number of attempts to successfully
 schedule a pod.
# TYPE scheduler_pod_scheduling_attempts histogram
scheduler_pod_scheduling_attempts_bucket{le="1"} 79
scheduler_pod_scheduling_attempts_bucket{le="2"} 79
scheduler_pod_scheduling_attempts_bucket{le="4"} 79
scheduler_pod_scheduling_attempts_bucket{le="8"} 79
scheduler_pod_scheduling_attempts_bucket{le="16"} 79
scheduler_pod_scheduling_attempts_bucket{le="+Inf"} 81
[...]
```

擷取**kube-controller-manager**指標

若要擷取kube-controller-manager指標，請使用下列命令。

```
kubectl get --raw "/apis/metrics.eks.amazonaws.com/v1/kcm/container/metrics"
```

範例輸出如下。

```
[...]
workqueue_work_duration_seconds_sum{name="pvprotection"} 0
workqueue_work_duration_seconds_count{name="pvprotection"} 0
workqueue_work_duration_seconds_bucket{name="replicaset",le="1e-08"} 0
workqueue_work_duration_seconds_bucket{name="replicaset",le="1e-07"} 0
workqueue_work_duration_seconds_bucket{name="replicaset",le="1e-06"} 0
workqueue_work_duration_seconds_bucket{name="replicaset",le="9.999999999999999e-06"} 0
workqueue_work_duration_seconds_bucket{name="replicaset",le="9.999999999999999e-05"} 19
workqueue_work_duration_seconds_bucket{name="replicaset",le="0.001"} 109
```

```
workqueue_work_duration_seconds_bucket{name="replicaset",le="0.01"} 139
workqueue_work_duration_seconds_bucket{name="replicaset",le="0.1"} 181
workqueue_work_duration_seconds_bucket{name="replicaset",le="1"} 191
workqueue_work_duration_seconds_bucket{name="replicaset",le="10"} 191
workqueue_work_duration_seconds_bucket{name="replicaset",le="+Inf"} 191
workqueue_work_duration_seconds_sum{name="replicaset"} 4.265655885000002
[...]
```

了解排程器和控制器管理員指標

下表說明可供 Prometheus 樣式抓取使用的排程器和控制器管理員指標。如需這些指標的詳細資訊，請參閱 Kubernetes 文件中的 Kubernetes 指標參考。

指標	控制平面元件	描述
scheduler_pending_pods	排程器	等待排程到節點執行的 Pod 數量。
scheduler_schedule_attempts_total	排程器	嘗試排程 Pod 的次數。
scheduler_preemption_attempts_total	排程器	排程器透過排除較低優先順序的 Pod 來排程較高優先順序的嘗試次數。
scheduler_preemption_victims	排程器	已選取要移出的 Pod 數量，以便為較高優先順序的 Pod 騰出空間。
scheduler_pod_scheduling_attempts	排程器	成功排程 Pod 的嘗試次數。
scheduler_scheduling_attempt_duration_seconds	排程器	指出排程器能夠根據資源可用性和排程規則等各種因素，找到適合 Pod 執行的位置。
scheduler_pod_scheduling_sli_duration_seconds	排程器	正在排程之 Pod end-to-end 延遲，從 Pod 進入排程佇列開始。這可能涉及多次排程嘗試。

指標	控制平面元件	描述
cronjob_controller_job_creation_skew_duration_seconds	控制器管理員	排定執行 cronjob 與建立對應任務之間的時間。
workqueue_depth	控制器管理員	佇列的目前深度。
workqueue_adds_total	控制器管理員	由 workqueue 處理的新增總數。
workqueue_queue_duration_seconds	控制器管理員	在請求之前，項目保持在工作佇列中的時間，以秒為單位。
workqueue_work_duration_seconds	控制器管理員	從工作佇列處理項目所需的秒數。

部署 Prometheus 抓取器以持續抓取指標

若要部署 Prometheus 湊集器以持續抓取指標，請使用下列組態：

```
---
apiVersion: v1
kind: ConfigMap
metadata:
  name: prometheus-conf
data:
  prometheus.yml: |-
    global:
      scrape_interval: 30s
    scrape_configs:
    # apiserver metrics
    - job_name: apiserver-metrics
      kubernetes_sd_configs:
      - role: endpoints
      scheme: https
      tls_config:
        ca_file: /var/run/secrets/kubernetes.io/serviceaccount/ca.crt
        insecure_skip_verify: true
      bearer_token_file: /var/run/secrets/kubernetes.io/serviceaccount/token
      relabel_configs:
      - source_labels:
```

```
        [
          __meta_kubernetes_namespace,
          __meta_kubernetes_service_name,
          __meta_kubernetes_endpoint_port_name,
        ]
      action: keep
      regex: default;kubernetes;https
  # Scheduler metrics
  - job_name: 'ksh-metrics'
    kubernetes_sd_configs:
    - role: endpoints
    metrics_path: /apis/metrics.eks.amazonaws.com/v1/ksh/container/metrics
    scheme: https
    tls_config:
      ca_file: /var/run/secrets/kubernetes.io/serviceaccount/ca.crt
      insecure_skip_verify: true
    bearer_token_file: /var/run/secrets/kubernetes.io/serviceaccount/token
    relabel_configs:
    - source_labels:
        [
          __meta_kubernetes_namespace,
          __meta_kubernetes_service_name,
          __meta_kubernetes_endpoint_port_name,
        ]
      action: keep
      regex: default;kubernetes;https
  # Controller Manager metrics
  - job_name: 'kcm-metrics'
    kubernetes_sd_configs:
    - role: endpoints
    metrics_path: /apis/metrics.eks.amazonaws.com/v1/kcm/container/metrics
    scheme: https
    tls_config:
      ca_file: /var/run/secrets/kubernetes.io/serviceaccount/ca.crt
      insecure_skip_verify: true
    bearer_token_file: /var/run/secrets/kubernetes.io/serviceaccount/token
    relabel_configs:
    - source_labels:
        [
          __meta_kubernetes_namespace,
          __meta_kubernetes_service_name,
          __meta_kubernetes_endpoint_port_name,
        ]
      action: keep
```

```
        regex: default;kubernetes;https
---
apiVersion: v1
kind: Pod
metadata:
  name: prom-pod
spec:
  containers:
  - name: prom-container
    image: prom/prometheus
    ports:
    - containerPort: 9090
    volumeMounts:
    - name: config-volume
      mountPath: /etc/prometheus/
  volumes:
  - name: config-volume
    configMap:
      name: prometheus-conf
```

Pod 存取新的指標端點需要以下許可。

```
{
  "effect": "allow",
  "apiGroups": [
    "metrics.eks.amazonaws.com"
  ],
  "resources": [
    "kcm/metrics",
    "ksh/metrics"
  ],
  "verbs": [
    "get"
  ] },
```

若要修補正在使用的角色，您可以使用下列命令。

```
kubectl patch clusterrole <role-name> --type=json -p='[
  {
    "op": "add",
    "path": "/rules/-",
    "value": {
      "verbs": ["get"],
```

```
      "apiGroups": ["metrics.eks.amazonaws.com"],
      "resources": ["kcm/metrics", "ksh/metrics"]
    }
  }
]'
```

然後，您可以將 Prometheus 湊集器的連接埠代理至本機連接埠，以檢視 Prometheus 儀表板。

```
kubectl port-forward pods/prom-pod 9090:9090
```

對於 Amazon EKS 叢集，核心 Kubernetes 控制平面指標也會擷取到 AWS/EKS 命名空間下的 Amazon CloudWatch 指標。若要檢視它們，請開啟 CloudWatch 主控台，然後從左側導覽窗格中選取所有指標。在指標選擇頁面上，選擇叢集的AWS/EKS命名空間和指標維度。

使用 Amazon CloudWatch 監控叢集資料

Amazon CloudWatch 是一種監控服務，可從雲端資源收集指標和日誌。CloudWatch 使用版本 1.28和更新版本的新叢集時，可免費提供一些基本的 Amazon EKS 指標。不過，使用 CloudWatch 可觀測性運算子做為 Amazon EKS 附加元件時，您可以取得增強的可觀測性功能。

Amazon CloudWatch 中的基本指標

對於 Kubernetes 版本 1.28 及更高版本的叢集，您可以在 AWS/EKS 命名空間中免費取得 CloudWatch 已販賣的指標。下表提供適用於支援版本的基本指標清單。列出的每個指標的頻率為一分鐘。

指標名稱	描述
scheduler_schedule_attempts_total	排程器在指定期間內排程叢集中 Pod 的嘗試次數。此指標有助於監控排程器的工作負載，並可能指出排程壓力或 Pod 放置的潛在問題。 單位：計數 有效統計資料：總和
scheduler_schedule_attempts_SCHEDULED	排程器在指定期間內將 Pod 排程到叢集中節點的成功嘗試次數。

指標名稱	描述
	單位：計數 有效統計資料：總和
scheduler_schedule_attempts_UNSCHEDULABLE	由於有效限制，例如節點上的 CPU 或記憶體不足，在指定期間內無法排程的 Pod 嘗試次數。 單位：計數 有效統計資料：總和
scheduler_schedule_attempts_ERROR	由於排程器本身的內部問題，例如 API Server 連線問題，在指定期間內失敗的 Pod 嘗試次數。 單位：計數 有效統計資料：總和
scheduler_pending_pods	叢集中排程器在指定期間內要排程的待處理 Pod 總數。 單位：計數 有效統計資料：總和
scheduler_pending_pods_ACTIVEQ	activeQ 中等待在叢集中排程一段指定期間的待處理 Pod 數量。 單位：計數 有效統計資料：總和
scheduler_pending_pods_UNSCHEDULABLE	排程器嘗試排程和失敗的待定 Pod 數量，並維持在無法排程的狀態以供重試。 單位：計數 有效統計資料：總和

指標名稱	描述
scheduler_pending_pods_BACKOFF	backoffQ 處於退避狀態且等待退避期間過期的中的待定 Pod 數量。 單位：計數 有效統計資料：總和
scheduler_pending_pods_GATED	目前等待處於鎖定狀態的待定 Pod 數量，因為在符合必要條件之前無法排程。 單位：計數 有效統計資料：總和
apiserver_request_total	叢集中所有 API 伺服器提出的 HTTP 請求數量。 單位：計數 有效統計資料：總和
apiserver_request_total_4XX	對叢集中產生 4XX（用戶端錯誤）狀態碼的所有 API 伺服器提出的 HTTP 請求數量。 單位：計數 有效統計資料：總和
apiserver_request_total_429	對叢集中所有 API 伺服器提出的 HTTP 請求數量，這些請求會產生429狀態碼，當用戶端超過速率限制閾值時就會發生。 單位：計數 有效統計資料：總和

指標名稱	描述
apiserver_request_total_5XX	對叢集中產生 5XX（伺服器錯誤）狀態碼的所有 API 伺服器提出的 HTTP 請求數量。 單位：計數 有效統計資料：總和
apiserver_request_total_LIST_PODS	對叢集中所有 API 伺服器提出的 LIST Pod 請求數量。 單位：計數 有效統計資料：總和
apiserver_request_duration_seconds_PUT_P99	從叢集中所有 API 伺服器的所有PUT請求計算的請求的第 99 個百分位數。代表低於 99% 的所有PUT請求完成的回應時間。 單位：秒 有效統計資料：平均
apiserver_request_duration_seconds_PATCH_P99	從叢集中所有 API 伺服器的所有PATCH請求計算的請求的第 99 個百分位數延遲。代表低於 99% 的所有PATCH請求完成的回應時間。 單位：秒 有效統計資料：平均
apiserver_request_duration_seconds_POST_P99	從叢集中所有 API 伺服器的所有POST請求計算的請求的第 99 個百分位數延遲。代表低於 99% 的所有POST請求完成的回應時間。 單位：秒 有效統計資料：平均

指標名稱	描述
apiserver_request_duration_seconds_GET_P99	從叢集中所有 API 伺服器的所有GET請求計算的請求的第 99 個百分位數延遲。代表低於 99% 的所有GET請求完成的回應時間。 單位：秒 有效統計資料：平均
apiserver_request_duration_seconds_LIST_P99	從叢集中所有 API 伺服器的所有LIST請求計算的請求的第 99 個百分位數延遲。代表低於 99% 的所有LIST請求完成的回應時間。 單位：秒 有效統計資料：平均
apiserver_request_duration_seconds_DELETE_P99	從叢集中所有 API 伺服器的所有DELETE請求計算的請求的第 99 個百分位數延遲。代表低於 99% 的所有DELETE請求完成的回應時間。 單位：秒 有效統計資料：平均
apiserver_current_inflight_requests_MUTATING	叢集中所有 API 伺服器目前正在處理的變動請求數量 (POST、PUTDELETE、、PATCH)。此指標代表正在進行中且尚未完成處理的請求。 單位：計數 有效統計資料：總和
apiserver_current_inflight_requests_READONLY	叢集中所有 API 伺服器目前正在處理的唯讀請求 (GET、LIST) 數量。此指標代表正在進行中且尚未完成處理的請求。 單位：計數 有效統計資料：總和

指標名稱	描述
apiserver_admission_webhook_request_total	叢集中所有 API 伺服器發出的許可 Webhook 請求數目。 單位：計數 有效統計資料：總和
apiserver_admission_webhook_request_total_ADMIT	叢集中所有 API 伺服器之間發出的變動許可 Webhook 請求數目。 單位：計數 有效統計資料：總和
apiserver_admission_webhook_request_total_VALIDATING	驗證叢集中所有 API 伺服器發出的許可 Webhook 請求數目。 單位：計數 有效統計資料：總和
apiserver_admission_webhook_rejection_count	在叢集中所有 API 伺服器之間提出的許可 Webhook 請求數量已被拒絕。 單位：計數 有效統計資料：總和
apiserver_admission_webhook_rejection_count_ADMIT	在叢集中所有 API 伺服器之間發出且已拒絕的變動許可 Webhook 請求數目。 單位：計數 有效統計資料：總和

指標名稱	描述
apiserver_admission_webhook_rejection_count_VALIDATING	驗證叢集中所有 API 伺服器之間所提出的許可 Webhook 請求數量，已遭到拒絕。 單位：計數 有效統計資料：總和
apiserver_admission_webhook_admission_duration_seconds	第三方許可 Webhook 請求的第 99 個百分位數，從叢集中所有 API 伺服器的所有請求計算得出。代表低於 99% 的第三方許可 Webhook 請求完成的回應時間。 單位：秒 有效統計資料：平均
apiserver_admission_webhook_admission_duration_seconds_ADMIT_P99	第三方變動許可 Webhook 請求的延遲第 99 個百分位數，從叢集中所有 API 伺服器的所有請求計算。代表回應時間低於 99% 的第三方變動許可 Webhook 請求完成的時間。 單位：秒 有效統計資料：平均
apiserver_admission_webhook_admission_duration_seconds_VALIDATING_P99	第三方驗證從叢集中所有 API 伺服器的所有請求計算的許可 Webhook 請求的延遲第 99 個百分位數。代表回應時間低於 99% 的所有驗證許可 Webhook 請求的第三方完成時間。 單位：秒 有效統計資料：平均

指標名稱	描述
apiserver_storage_size_bytes	叢集中 API 伺服器所使用的已壓縮儲存資料庫檔案的實體大小，以位元組為單位。此指標代表為儲存配置的實際磁碟空間。 單位：位元組 有效統計資料：上限

Amazon CloudWatch 可觀測性運算子

Amazon CloudWatch Observability 會收集即時日誌、指標和追蹤資料。它會將其傳送至 Amazon CloudWatch 和 AWS X-Ray。您可以安裝此附加元件來啟用 CloudWatch Application Signals 和 CloudWatch Container Insights，搭配 Amazon EKS 的增強可觀測性。這有助於監控基礎設施和容器化應用程式的運作狀態與效能。Amazon CloudWatch Observability Operator 旨在安裝和設定必要的元件。

Amazon EKS 支援 CloudWatch 可觀測性運算子做為 Amazon EKS 附加元件。附加元件可在叢集中的 Linux 和 Windows 工作者節點上允許 Container Insights。若要在 Windows 上啟用 Container Insights，Amazon EKS 附加元件版本必須為 `1.5.0`或更高版本。Amazon EKS Windows 目前不支援 CloudWatch Application Signals。

以下主題說明如何開始使用適用於 Amazon EKS 叢集的 CloudWatch Observability Operator。

- 如需安裝此附加元件的指示，請參閱《Amazon CloudWatch 使用者指南》中的使用 Amazon CloudWatch 可觀測性 EKS 附加元件安裝 CloudWatch 代理程式或 Helm Chart Amazon CloudWatch。 Amazon CloudWatch
- 如需 CloudWatch Application Signals 的詳細資訊，請參閱《Amazon CloudWatch 使用者指南》中的 Application Signals。
- 如需有關 Container Insights 的詳細資訊，請參閱《Amazon CloudWatch 使用者指南》中的使用 Container Insights。

將控制平面日誌傳送至 CloudWatch Logs

Amazon EKS 控制平面記錄從 Amazon EKS 控制平面將稽核和診斷日誌直接提供至您帳戶中的 CloudWatch Logs。這些日誌可讓您輕鬆執行叢集並確保叢集的安全。您可以選取所需的確切日誌

類型，且日誌將以日誌串流傳送至 CloudWatch 中各個 Amazon EKS 叢集的群組中。您可以使用 CloudWatch 訂閱篩選條件對日誌進行即時分析，或將日誌轉送至其他 服務 （ 日誌將以 Base64 編碼並以 gzip 格式壓縮 ）。如需詳細資訊，請參閱 Amazon CloudWatch logging (Amazon CloudWatch 日誌記錄)。

您可以選擇要為每個新的或現有的 Amazon EKS 叢集啟用哪些日誌類型，以開始使用 Amazon EKS 控制平面記錄。您可以使用 AWS Management Console、 AWS CLI （ 版本 `1.16.139`或更高版本 ） 或透過 Amazon EKS API，依叢集啟用或停用每個日誌類型。啟用之後，日誌會自動從 Amazon EKS 叢集傳送至相同帳戶中的 CloudWatch Logs。

當您使用 Amazon EKS 控制平面記錄時，系統會針對您執行的每個叢集向您收取標準 Amazon EKS 定價。您必須為從叢集傳送至 CloudWatch Logs 的任何日誌，支付標準 CloudWatch Logs 資料擷取和儲存成本。您也需要為作為叢集一部分佈建的任何 AWS 資源付費，例如 Amazon EC2 執行個體或 Amazon EBS 磁碟區。

以下叢集控制平面日誌類型可供使用。每個日誌類型皆對應 Kubernetes 控制平面的某個元件。如需進一步了解這些元件的詳細資訊，請參閱 Kubernetes 文件中的 Kubernetes 元件。

API 伺服器**`api`** ()

> 您叢集的 API 伺服器是公開 Kubernetes API 的控制平面元件。如果您在啟動叢集時或之後隨即啟用 API 伺服器日誌，則日誌中會包含用於啟動 API 伺服器的 API 伺服器旗標。如需詳細資訊，請參閱 Kubernetes 文件中的 kube-apiserver 和稽核政策。

稽核**`audit`** ()

> Kubernetes 稽核日誌提供個別使用者、管理員或系統元件的記錄，這些元件會影響您的叢集。如需詳細資訊，請參閱 Kubernetes 文件中的稽核。

驗證器**`authenticator`** ()

> 驗證器日誌是 Amazon EKS 獨有的。這些日誌代表 Amazon EKS 使用 IAM 憑證進行 Kubernetes 角色型存取控制 (RBAC) 身分驗證的控制平面元件。如需詳細資訊，請參閱叢集管理。

控制器管理員**`controllerManager`** ()

> 控制器管理員會管理 Kubernetes 隨附的核心控制迴圈。如需詳細資訊，請參閱 Kubernetes 文件中的 kube-controller-manager。

排程器**`scheduler`** ()

> 排程器元件會管理在叢集中執行 Pod 的時間和位置。如需詳細資訊，請參閱 Kubernetes 文件中的 kube-scheduler。

啟用或停用控制平面日誌

根據預設，叢集控制平面日誌不會傳送至 CloudWatch Logs。您必須個別啟用各個日誌類型，才能傳送您叢集的日誌。CloudWatch Logs 擷取、封存儲存和資料掃描速率會套用啟用的控制平面記錄。如需詳細資訊，請參閱 CloudWatch 定價。

若要更新控制平面日誌記錄組態，Amazon EKS 需要每個子網路中最多五個可用的 IP 地址。當您啟用日誌類型時，將會以日誌詳細資訊等級 2 傳送日誌。

您可以使用 或 AWS CLI AWS Management Console 啟用或停用控制平面日誌。

AWS Management Console

1. 開啟 Amazon EKS 主控台。
2. 選擇叢集名稱以顯示您叢集的資訊。
3. 選擇可觀測性索引標籤。
4. 在控制平面日誌記錄區段中，選擇管理日誌記錄。
5. 對於每個日誌類型，選擇其日誌類型為開啟或關閉。根據預設，系統會關閉每個日誌類型。
6. 選擇 Save changes (儲存變更) 以完成操作。

AWS CLI

1. 使用以下命令檢查您的 AWS CLI 版本。

```
aws --version
```

   如果您的 AWS CLI 版本早於 `1.16.139`，您必須先更新至最新版本。若要安裝或升級 AWS CLI，請參閱《AWS 命令列界面使用者指南》中的安裝 AWS 命令列界面。
2. 使用以下 AWS CLI 命令更新叢集的控制平面日誌匯出組態。使用叢集名稱取代 *`my-cluster`* 並指定所需的端點存取值。

> **Note**
>
> 以下命令會將所有可用的日誌類型傳送至 CloudWatch Logs。

```
aws eks update-cluster-config \
```

```
    --region region-code \
    --name my-cluster \
    --logging '{"clusterLogging":[{"types":
["api","audit","authenticator","controllerManager","scheduler"],"enabled":true}]}'
```

範例輸出如下。

```
{
    "update": {
        "id": "883405c8-65c6-4758-8cee-2a7c1340a6d9",
        "status": "InProgress",
        "type": "LoggingUpdate",
        "params": [
            {
                "type": "ClusterLogging",
                "value": "{\"clusterLogging\":[{\"types\":[\"api\",\"audit\",
\"authenticator\",\"controllerManager\",\"scheduler\"],\"enabled\":true}]}"
            }
        ],
        "createdAt": 1553271814.684,
        "errors": []
    }
}
```

3. 使用以下命令以及上個命令傳回的叢集名稱和更新 ID，監控日誌組態更新的狀態。當狀態出現 Successful 時，您的更新就完成了。

```
aws eks describe-update \
    --region region-code\
    --name my-cluster \
    --update-id 883405c8-65c6-4758-8cee-2a7c1340a6d9
```

範例輸出如下。

```
{
    "update": {
        "id": "883405c8-65c6-4758-8cee-2a7c1340a6d9",
        "status": "Successful",
        "type": "LoggingUpdate",
        "params": [
            {
                "type": "ClusterLogging",
```

```
                "value": "{\"clusterLogging\":[{\"types\":[\"api\",\"audit\",
\"authenticator\",\"controllerManager\",\"scheduler\"],\"enabled\":true}]}"
            }
        ],
        "createdAt": 1553271814.684,
        "errors": []
    }
}
```

檢視叢集控制平面日誌

當您為您的 Amazon EKS 叢集啟用任何控制平面日誌類型之後，即可在 CloudWatch 主控台上檢視它們。

若要進一步了解在 CloudWatch 中檢視、分析和管理日誌，請參閱 Amazon CloudWatch Logs 使用者指南。

1. 開啟 CloudWatch 主控台。此連結會開啟主控台及顯示目前可用的日誌群組，並以 /aws/eks 字首進行篩選。
2. 選擇您要檢視日誌的叢集。日誌群組的名稱格式是 /aws/eks/*my-cluster*/cluster。
3. 選擇要檢視的日誌串流。以下清單說明每個日誌類型的日誌串流名稱格式。

> **Note**
>
> 隨著日誌串流資料增加，日誌串流名稱將會輪換。當特定日誌類型有多個日誌串流時，您可以使用最新的 Last Event Time (上次事件時間) 來尋找日誌串流名稱，以檢視最新的日誌串流。

   - Kubernetes API 伺服器元件記錄 (**api**) – kube-apiserver-*1234567890abcdef01234567890abcde*
   - 稽核 (**audit**) – kube-apiserver-audit-*1234567890abcdef01234567890abcde*
   - 驗證器 (**authenticator**) – authenticator-*1234567890abcdef01234567890abcde*
   - 控制器管理員 (**controllerManager**) – kube-controller-manager-*1234567890abcdef01234567890abcde*
   - 排程器 (**scheduler**) – kube-scheduler-*1234567890abcdef01234567890abcde*
4. 查看日誌串流事件。

例如，在檢視 kube-apiserver-*1234567890abcdef01234567890abcde* 的頂端時，您應該會看到叢集的初始 API 伺服器旗標。

> **Note**
>
> 如果您在日誌串流開頭沒有看到 API 伺服器日誌，則 API 伺服器日誌檔案可能會在伺服器上輪換，然後再啟用伺服器上的 API 伺服器日誌記錄。在啟用 API 伺服器記錄之前輪換的任何日誌檔案都無法匯出至 CloudWatch。

不過，您可以使用相同的 Kubernetes 版本建立新的叢集，並在建立叢集時啟用 API 伺服器記錄。具有相同平台版本的叢集已啟用相同的旗標，因此您的旗標應與新叢集的旗標相符。當您檢視完 CloudWatch 中新叢集的旗標時，即可刪除新叢集。

將 API 呼叫記錄為 AWS CloudTrail 事件

Amazon EKS 已與 AWS CloudTrail 整合。CloudTrail 是一項服務，提供 Amazon EKS 中使用者、角色或 AWS 服務的動作記錄。CloudTrail 會將 Amazon EKS 的所有 API 呼叫擷取為事件。事件包括從 Amazon EKS 主控台的呼叫，以及對 Amazon EKS API 操作的程式碼呼叫。

若您建立追蹤，便可將 CloudTrail 事件持續交付至 Amazon S3 儲存貯體，包括 的事件。其中包括 Amazon EKS 的事件。如果您未設定線索，仍然可以在事件歷史記錄中檢視 CloudTrail 主控台中的最新事件。 使用 CloudTrail 收集的資訊，您可以判斷關於請求的數個詳細資訊。例如，您可以判斷向 Amazon EKS 提出請求的時間、提出請求的 IP 地址、以及提出請求的人員。

若要進一步了解 CloudTrail，請參閱 AWS CloudTrail 使用者指南。

主題

- 檢視 AWS CloudTrail 的實用參考
- Analyze AWS CloudTrail 日誌檔案項目
- 檢視 Amazon EC2 Auto Scaling 群組的指標

檢視 AWS CloudTrail 的實用參考

建立 AWS 帳戶時，也會在 AWS 帳戶中啟用 CloudTrail。當 Amazon EKS 中發生任何活動時，該活動會記錄在 CloudTrail 事件中，以及事件歷史記錄中的其他 AWS 服務事件。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。

若要持續記錄您 AWS 帳戶中的事件，包括 Amazon EKS 的事件，請建立追蹤。線索能讓 CloudTrail 將日誌檔案交付至 Amazon S3 儲存貯體。根據預設，當您在主控台中建立追蹤時，該追蹤會套用至所有 AWS 區域。線索會記錄 AWS 分割區中所有 AWS 區域的事件，並將日誌檔案交付至您指定的 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，請參閱下列資源。

- 建立追蹤的概觀
- CloudTrail 支援的服務和整合
- 設定 CloudTrail 的 Amazon SNS 通知
- 接收多個區域的 CloudTrail 日誌檔案和接收多個帳戶的 CloudTrail 日誌檔案

CloudTrail 會記錄所有 Amazon EKS 動作，並記錄在《Amazon EKS API 參考》中。例如，對 CreateCluster、ListClusters 和 DeleteCluster 區段的呼叫會在 CloudTrail 日誌檔案中產生項目。

每個事件或日誌項目包含提出請求之 IAM 身分類型及所使用之憑證的資訊。如果使用暫時性憑證，此項目會說明憑證的取得方式。

如需詳細資訊，請參閱 CloudTrail userIdentity 元素。

Analyze AWS CloudTrail 日誌檔案項目

追蹤是一種組態，能讓事件以日誌檔案的形式交付到您指定的 Amazon S3 儲存貯體。CloudTrail 日誌檔案包含一或多個日誌專案。事件代表來自任何來源的單一請求，其中包含請求動作的相關資訊。其中包含了動作的日期和時間，以及所使用的請求參數等資訊。CloudTrail 日誌檔並非依公有 API 呼叫的堆疊追蹤排序，因此不會以任何特定順序出現。

以下範例顯示的是展示 `CreateCluster` 動作的 CloudTrail 日誌項目。

```
{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "AKIAIOSFODNN7EXAMPLE",
    "arn": "arn:aws: iam::111122223333:user/username",
    "accountId": "111122223333",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
    "userName": "username"
  },
  "eventTime": "2018-05-28T19:16:43Z",
  "eventSource": "eks.amazonaws.com",
```

```
  "eventName": "CreateCluster",
  "awsRegion": "region-code",
  "sourceIPAddress": "205.251.233.178",
  "userAgent": "PostmanRuntime/6.4.0",
  "requestParameters": {
    "resourcesVpcConfig": {
      "subnetIds": [
        "subnet-a670c2df",
        "subnet-4f8c5004"
      ]
    },
    "roleArn": "arn:aws: iam::111122223333:role/AWSServiceRoleForAmazonEKS-
CAC1G1VH3ZKZ",
    "clusterName": "test"
  },
  "responseElements": {
    "cluster": {
      "clusterName": "test",
      "status": "CREATING",
      "createdAt": 1527535003.208,
      "certificateAuthority": {},
      "arn": "arn:aws: eks:region-code:111122223333:cluster/test",
      "roleArn": "arn:aws: iam::111122223333:role/AWSServiceRoleForAmazonEKS-
CAC1G1VH3ZKZ",
      "version": "1.10",
      "resourcesVpcConfig": {
        "securityGroupIds": [],
        "vpcId": "vpc-21277358",
        "subnetIds": [
          "subnet-a670c2df",
          "subnet-4f8c5004"
        ]
      }
    }
  },
  "requestID": "a7a0735d-62ab-11e8-9f79-81ce5b2b7d37",
  "eventID": "eab22523-174a-499c-9dd6-91e7be3ff8e3",
  "readOnly": false,
  "eventType": "AwsApiCall",
  "recipientAccountId": "111122223333"
}
```

Amazon EKS 服務連結角色的日誌項目

Amazon EKS 服務連結角色會對 AWS 資源進行 API 呼叫。呼叫如果是由 Amazon EKS 服務連結角色進行，會顯示搭配 username: AWSServiceRoleForAmazonEKS 和 username: AWSServiceRoleForAmazonEKSNodegroup 的 CloudTrail 日誌項目。如需 Amazon EKS 和服務連結角色的詳細資訊，請參閱 the section called “服務連結角色”。

下列範例顯示 CloudTrail 日誌項目，示範由 AWSServiceRoleForAmazonEKSNodegroup服務連結角色所做的DeleteInstanceProfile動作，如 中所述sessionContext。

```
{
    "eventVersion": "1.05",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AROA3WHGPEZ7SJ2CW55C5:EKS",
        "arn": "arn:aws: sts::111122223333:assumed-role/
AWSServiceRoleForAmazonEKSNodegroup/EKS",
        "accountId": "111122223333",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AROA3WHGPEZ7SJ2CW55C5",
                "arn": "arn:aws: iam::111122223333:role/aws-service-role/eks-
nodegroup.amazonaws.com/AWSServiceRoleForAmazonEKSNodegroup",
                "accountId": "111122223333",
                "userName": "AWSServiceRoleForAmazonEKSNodegroup"
            },
            "webIdFederationData": {},
            "attributes": {
                "mfaAuthenticated": "false",
                "creationDate": "2020-02-26T00:56:33Z"
            }
        },
        "invokedBy": "eks-nodegroup.amazonaws.com"
    },
    "eventTime": "2020-02-26T00:56:34Z",
    "eventSource": "iam.amazonaws.com",
    "eventName": "DeleteInstanceProfile",
    "awsRegion": "region-code",
    "sourceIPAddress": "eks-nodegroup.amazonaws.com",
    "userAgent": "eks-nodegroup.amazonaws.com",
    "requestParameters": {
```

```
        "instanceProfileName": "eks-11111111-2222-3333-4444-abcdef123456"
    },
    "responseElements": null,
    "requestID": "11111111-2222-3333-4444-abcdef123456",
    "eventID": "11111111-2222-3333-4444-abcdef123456",
    "eventType": "AwsApiCall",
    "recipientAccountId": "111122223333"
}
```

檢視 Amazon EC2 Auto Scaling 群組的指標

Amazon EKS 受管節點群組預設會啟用 Amazon EC2 Auto Scaling 群組指標，無需額外費用。Auto Scaling 群組每分鐘將取樣資料傳送至 Amazon CloudWatch。這些指標可以依 Auto Scaling 群組的名稱進行精簡。它們可讓您持續查看為受管節點群組提供支援的 Auto Scaling 群組歷史記錄，例如群組大小隨時間的變化。Auto Scaling 群組指標可在 Amazon CloudWatch 或 Auto Scaling 主控台中使用。如需詳細資訊，請參閱監控 Auto Scaling 群組和執行個體的 CloudWatch 指標。

使用 Auto Scaling 群組指標集合，您可以監控受管節點群組的擴展。Auto Scaling 群組指標報告 Auto Scaling 群組最小、最大和所需規模。如果節點群組中的節點數量低於最小規模，這代表節點群組運作狀態不佳，您可以建立警示。追蹤節點群組大小在調整最大計數時也很有用，因此您的資料平面不會耗盡容量。

如果您不想收集這些指標，您可以選擇停用所有或僅停用部分指標。例如，您可以執行此操作，以避免 CloudWatch 儀表板中的雜訊。如需詳細資訊，請參閱 Amazon EC2 Auto Scaling 的 Amazon CloudWatch 指標Amazon EC2 Auto Scaling。

使用 ADOT Operator 傳送指標和追蹤資料

Amazon EKS 支援使用 AWS Management Console、 AWS CLI 和 Amazon EKS API 來安裝和管理 AWS Distro for OpenTelemetry (ADOT) Operator。這可讓您在 Amazon EKS 上執行的應用程式更輕鬆地將指標和追蹤資料傳送至多個監控服務選項，例如 Amazon CloudWatch、Prometheus 和 X-Ray。

如需詳細資訊，請參閱 AWS Distro for OpenTelemetry 文件中的使用 EKS 附加元件的 Distro for OpenTelemetry 入門。 AWS OpenTelemetry

使用 EKS 儀表板檢視叢集資源的彙總資料

image::images/eks-dashboard.png[screenshot of account level cluster metrics]

什麼是 Amazon EKS 儀表板？

Amazon EKS 儀表板提供跨多個 AWS 區域和 AWS 帳戶的 Kubernetes 叢集整合可見性。使用此儀表板，您可以：

- 追蹤排程在未來 90 天內end-of-support自動升級的叢集。
- 延伸支援中叢集的專案 EKS 控制平面成本。
- 使用在升級之前需要注意的洞察來檢閱叢集。
- 識別執行特定 AMI 版本的受管節點群組。
- 監控叢集支援類型分佈 （ 標準與延伸支援相比 ）。

EKS 儀表板會與 EKS Cluster Insights 整合，以解決叢集的問題，例如使用已取代的 Kubernetes APIs。如需詳細資訊，請參閱the section called “叢集洞察”。

> **Note**
>
> EKS 儀表板不是即時的，每 12 小時更新一次。如需即時叢集監控，請參閱 監控叢集

儀表板如何使用 AWS Organizations？

Amazon EKS 儀表板需要 AWS Organizations 整合功能。它利用 AWS Organizations 跨帳戶安全地收集叢集資訊。隨著您的 AWS 基礎設施擴展，此整合提供集中式管理和管控。

如果您的基礎設施未啟用 AWS Organizations，請參閱 AWS Organizations 使用者指南中的設定說明。

跨區域和跨帳戶存取

EKS 儀表板可以在屬於 AWS 組織成員的任何帳戶中查看叢集資源。若要產生組織中 AWS 的帳戶清單，請參閱匯出組織中所有帳戶的詳細資訊。

us-east-1 AWS region 會產生儀表板。您必須登入此區域才能查看儀表板。儀表板會跨 AWS 區域彙總資料，但這不包括 GovCloud或 中國區域。

重要用語

- AWS 組織：多個 AWS 帳戶的統一管理結構。

- 管理帳戶：控制 AWS Organization 的主要帳戶。
- 成員帳戶：組織內管理帳戶以外的任何帳戶。
- 委派管理員：成員帳戶授予特定跨帳戶管理許可。在管理帳戶中，您可以為每個 AWS 服務選擇一個委派管理員帳戶。
- 受信任存取：授權 EKS 儀表板跨組織帳戶存取叢集資訊。
- 服務連結角色 (SLRs)：直接連結至 AWS 服務的唯一 IAM 角色類型。EKS 儀表板使用 SLR 來讀取有關您的帳戶和組織的資訊。
- 如需詳細資訊，請參閱《 AWS Organizations 使用者指南》中的術語和概念。

一般概觀

1. 存取您 AWS Organization 的管理帳戶。
   - 存取管理帳戶的步驟取決於您設定 AWS Organization 的方式。例如，您可以透過 Identity Center 或 Okta AWS 存取管理帳戶。
2. 透過 EKS 主控台啟用受信任存取。
3. 使用其 AWS 帳戶 ID 指派委派管理員。
4. 切換到委派管理員帳戶。
5. 使用整個組織的可見性存取增強型 EKS 主控台。

使用 AWS 主控台啟用 EKS 儀表板

> **Important**
>
> 您必須登入您 AWS 組織的管理帳戶，才能啟用 EKS 儀表板。

存取 EKS 儀表板設定

1. 請確認以下內容：
   a. 您已啟用並設定 AWS Organizations。
   b. 您會登入組織的管理帳戶。
   c. 您正在檢視 us-east-1 區域中的 AWS 管理主控台。
2. 導覽至 EKS 主控台。

3. 在左側邊欄中，開啟儀表板設定。

設定 Amazon EKS 儀表板的存取權

1. 尋找您要允許檢視 EKS 儀表板之 AWS 帳戶 AWS 的帳戶 ID。
   a. 此步驟是選用的，但建議使用。如果沒有，您只能從管理帳戶存取儀表板。最佳實務是，您應該限制對 管理帳戶的存取。
2. 按一下啟用受信任存取。
   a. 您現在可以從管理帳戶檢視儀表板。
3. 按一下註冊委派管理員，然後輸入您將用來檢視儀表板之 AWS 帳戶的帳戶 ID。
   a. 您現在可以從委派管理員帳戶或管理帳戶檢視儀表板。

如需啟用儀表板所需的許可資訊，請參閱所需的最低 IAM 政策。

檢視 EKS 儀表板

1. 登入委派管理員帳戶 （ 建議 ） 或管理帳戶。
2. 登入 us-east-1 區域。
3. 前往 EKS 服務，然後從左側邊欄中選取儀表板。

> **Note**
>
> 檢閱檢視 EKS 儀表板所需的 IAM 許可。

設定儀表板

您可以設定儀表板的檢視，並篩選資源。

可用的資源

- 叢集：檢視 EKS 叢集狀態和位置的彙總資訊。
  - 具有運作狀態問題的叢集。
  - EKS 延伸支援的叢集。
  - 依 Kubernetes 版本分類的叢集。

- 受管節點群組：檢閱受管節點群組和 EC2 執行個體。
  - 依 AMI 類型的節點群組，例如 Amazon Linux 或 Bottlerocket。
  - 節點群組運作狀態問題。
  - 執行個體類型分佈。
- 附加元件：了解您已安裝的 Amazon EKS 附加元件及其狀態。
  - 每個附加元件的安裝數量。
  - 具有運作狀態問題的附加元件。
  - 每個附加元件的版本分佈。

可用的檢視

- 圖形檢視
  - 可自訂的小工具檢視，顯示所選資源的圖形和視覺化效果。
  - EKS 儀表板的所有使用者都可看見圖形檢視的變更，例如移除小工具。
- 資源檢視
  - 所選資源的清單檢視，支援篩選條件。
- 地圖檢視
  - 檢視所選資源的地理分佈。

篩選 EKS 儀表板

您可以透過以下方式篩選 EKS 儀表板：

- AWS 帳戶
- Organizations 定義的 AWS 組織單位
- AWS 區域

使用 AWS 主控台停用 EKS 儀表板

1. 請確認以下內容：
   a. 您已啟用並設定 AWS Organizations。
   b. 您會登入組織的管理帳戶。
   c. 您正在檢視 us-east-1 區域中的 AWS 管理主控台。

2. 導覽至 EKS 主控台。
3. 在左側邊欄中，開啟儀表板設定。
4. 按一下停用受信任存取。

EKS 儀表板故障診斷

啟用 EKS 儀表板的問題

- 您必須登入 AWS 組織的管理帳戶。
  - 如果您沒有 AWS 組織，請建立一個組織。了解如何建立和設定組織。
  - 如果 AWS 您的帳戶已經是 AWS 組織的成員，請識別組織的管理員。
- 您必須以足夠的 IAM 許可登入 AWS 帳戶，才能建立和更新 AWS Organizations 資源。

檢視 EKS 儀表板時發生問題

- 您必須登入下列其中一個 AWS 帳戶：
  - AWS 組織的管理帳戶
  - 委派管理員帳戶，在管理帳戶的 EKS 儀表板設定中識別。
- 如果您最近已啟用 EKS 儀表板，請注意，初始資料人口最多可能需要 12 小時。
- 嘗試使用 CLI 重新啟用儀表板，包括建立服務連結角色。

儀表板小工具意外移動

- EKS 儀表板會在 AWS 帳戶層級儲存可設定的小工具檢視。如果您變更小工具檢視，使用相同 AWS 帳戶的其他人員將看到變更。

設定 EKS 儀表板與 AWS Organizations 的整合

本節提供step-by-step說明。 AWS 您將了解如何啟用和停用服務之間的受信任存取，以及如何註冊和取消註冊委派管理員帳戶。您可以使用 AWS 主控台或 CLI AWS 來執行每個組態任務。

啟用受信任存取

受信任存取會授權 EKS 儀表板安全地存取組織中所有帳戶的叢集資訊。

使用 AWS 主控台

1. 登入您 AWS Organization 的管理帳戶。
2. 導覽至 us-east-1 區域中的 EKS 主控台。
3. 在左側邊欄中，選取儀表板設定。
4. 按一下啟用受信任存取。

> **Note**
>
> 當您透過 EKS 主控台啟用受信任存取時，系統會自動建立`AWSServiceRoleForAmazonEKSDashboard`服務連結角色。如果您使用 CLI 或 AWS Organizations AWS 主控台啟用受信任存取，則不會發生此自動建立。

使用 AWS CLI

1. 登入您 AWS Organization 的管理帳戶。
2. 執行下列命令：

```
aws iam create-service-linked-role --aws-service-name dashboard.eks.amazonaws.com
aws organizations enable-aws-service-access --service-principal eks.amazonaws.com
```

停用受信任存取

停用受信任存取會撤銷 EKS 儀表板在組織帳戶中存取叢集資訊的許可。

使用 AWS 主控台

1. 登入您 AWS Organization 的管理帳戶。
2. 導覽至 us-east-1 區域中的 EKS 主控台。
3. 在左側邊欄中，選取儀表板設定。
4. 按一下停用受信任存取。

使用 AWS CLI

1. 登入您 AWS Organization 的管理帳戶。

2. 執行以下命令：

```
aws organizations disable-aws-service-access --service-principal eks.amazonaws.com
```

啟用委派管理員帳戶

委派管理員是有權存取 EKS 儀表板的成員帳戶。

使用 AWS 主控台

1. 登入您 AWS Organization 的管理帳戶。
2. 導覽至 us-east-1 區域中的 EKS 主控台。
3. 在左側邊欄中，選取儀表板設定。
4. 按一下註冊委派管理員。
5. 輸入您要選擇做為委派管理員之 AWS 帳戶的帳戶 ID。
6. 確認註冊。

使用 AWS CLI

1. 登入您 AWS Organization 的管理帳戶。
2. 執行下列命令，將 取代123456789012為您的帳戶 ID：

```
aws organizations register-delegated-administrator --account-id 123456789012 --
service-principal eks.amazonaws.com
```

停用委派管理員帳戶

停用委派管理員會移除帳戶存取 EKS 儀表板的許可。

使用 AWS 主控台

1. 登入您 AWS Organization 的管理帳戶。
2. 導覽至 us-east-1 區域中的 EKS 主控台。
3. 在左側邊欄中，選取儀表板設定。
4. 在清單中尋找委派管理員。

5. 按一下您要以委派管理員身分移除的帳戶旁邊的取消註冊。

使用 AWS CLI

1. 登入您 AWS Organization 的管理帳戶。
2. 執行下列命令，123456789012以委派管理員的帳戶 ID 取代 ：

```
aws organizations deregister-delegated-administrator --account-id 123456789012 --
service-principal eks.amazonaws.com
```

所需的最低 IAM 政策

本節概述啟用受信任存取所需的最低 IAM 政策，並委派管理員與 AWS Organizations 進行 EKS 儀表板整合。

啟用受信任存取的政策

若要在 EKS 儀表板和 AWS Organizations 之間啟用受信任存取，您需要下列許可：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "organizations:EnableAWSServiceAccess",
                "organizations:DescribeOrganization",
                "organizations:ListAWSServiceAccessForOrganization"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:CreateServiceLinkedRole"
            ],
            "Resource": "arn:aws: iam::*:role/aws-service-role/
dashboard.eks.amazonaws.com/AWSServiceRoleForAmazonEKSDashboard"
        }
    ]
```

```
}
```

委派管理員的政策

若要註冊或取消註冊 EKS 儀表板的委派管理員，您需要下列許可：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "organizations:RegisterDelegatedAdministrator",
                "organizations:DeregisterDelegatedAdministrator",
                "organizations:ListDelegatedAdministrators"
            ],
            "Resource": "*"
        }
    ]
}
```

檢視 EKS 儀表板的政策

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AmazonEKSDashboardReadOnly",
            "Effect": "Allow",
            "Action": [
                "eks:ListDashboardData",
                "eks:ListDashboardResources",
                "eks:DescribeClusterVersions"
            ],
            "Resource": "*"
        },
        {
            "Sid": "AmazonOrganizationsReadOnly",
            "Effect": "Allow",
            "Action": [
                "organizations:DescribeOrganization",
                "organizations:ListAWSServiceAccessForOrganization",
                "organizations:ListRoots",
```

```
                "organizations:ListAccountsForParent",
                "organizations:ListOrganizationalUnitsForParent"
            ],
            "Resource": "*"
        },
        {
            "Sid": "AmazonOrganizationsDelegatedAdmin",
            "Effect": "Allow",
            "Action": [
                "organizations:ListDelegatedAdministrators"
            ],
            "Resource": [
                "*"
            ],
            "Condition": {
                "StringEquals": {
                    "organizations:ServicePrincipal": "eks.amazonaws.com"
                }
            }
        }
    ]
}
```

> **Note**
>
> 這些政策必須連接到 AWS Organization 管理帳戶中的 IAM 主體（使用者或角色）。成員帳戶無法啟用受信任存取或委派管理員。

使用整合 AWS 服務增強 EKS

除了其他章節涵蓋的服務之外，Amazon EKS 還與更多 AWS 服務搭配使用，以提供其他解決方案。本主題識別使用 Amazon EKS 來新增功能的其他服務，或 Amazon EKS 用來執行任務的服務。

主題

- 使用 AWS CloudFormation 建立 Amazon EKS 資源
- 使用 Amazon Detective 分析 EKS 上的安全事件
- 使用 Amazon GuardDuty 偵測威脅
- 使用彈性中樞評估 EKS 叢集 AWS 彈性
- 使用 Security Lake 集中和分析 EKS 安全資料
- 使用 Amazon VPC Lattice 啟用安全的跨叢集連線
- 使用 AWS Local Zones 啟動低延遲 EKS 叢集

使用 AWS CloudFormation 建立 Amazon EKS 資源

Amazon EKS 已與 AWS CloudFormation 整合，這項服務可協助您建立和設定 AWS 資源的模型，讓您可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 EKS 叢集， AWS CloudFormation 會為您佈建和設定這些資源。

當您使用 AWS CloudFormation 時，您可以重複使用範本來持續且重複地設定 Amazon EKS 資源。只需描述您的資源一次，然後在多個 AWS 帳戶和區域中逐一佈建相同的資源。

Amazon EKS 和 AWS CloudFormation 範本

若要佈建和設定 Amazon EKS 和相關服務的資源，您必須了解 AWS CloudFormation 範本。範本是以 JSON 或 YAML 格式化的文本檔案。這些範本說明您要在 AWS CloudFormation 堆疊中佈建的資源。如果您不熟悉 JSON 或 YAML，您可以使用 AWS CloudFormation 設計工具來協助您開始使用 AWS CloudFormation 範本。如需詳細資訊，請參閱《AWS CloudFormation 使用者指南》中的什麼是 CloudFormation 設計工具？。 AWS CloudFormation

Amazon EKS 支援在 AWS CloudFormation 中建立叢集和節點群組。如需詳細資訊，包括 Amazon EKS 資源的 JSON 和 YAML 範本範例，請參閱《 AWS CloudFormation 使用者指南》中的 Amazon EKS 資源類型參考。

進一步了解 AWS CloudFormation

若要進一步了解 AWS CloudFormation，請參閱下列資源：

- AWS CloudFormation
- AWS CloudFormation 使用者指南
- AWS CloudFormation 命令列界面使用者指南

使用 Amazon Detective 分析 EKS 上的安全事件

Amazon Detective 會協助您分析、調查並快速識別安全調查結果或可疑活動的根本原因。Detective 會自動從您的 AWS 資源收集日誌資料。Detective 接著會使用機器學習、統計分析和圖論來產生視覺化內容，協助您更快地進行有效率的安全調查。Detective 提供預先建置的資料彙總、摘要和內容，可協助您快速分析並判斷潛在安全問題的本質和範圍。如需詳細資訊，請參閱 Amazon Detective 使用者指南。

Detective 會將 Kubernetes 和 AWS 資料整理成問題清單，例如：

- Amazon EKS 叢集詳細資訊，包括建立叢集的 IAM 身分和叢集的服務角色。您可以使用 Detective 調查這些 IAM 身分的 AWS 和 Kubernetes API 活動。
- 容器詳細資料，例如映像和安全性內容。您也可以檢閱已終止 Pod 的詳細資訊。
- Kubernetes API 活動，包括 API 活動的整體趨勢和特定 API 呼叫的詳細資訊。例如，您可以顯示所選時間範圍內發出的成功和失敗 Kubernetes API 呼叫數量。此外，最新觀察到的 API 呼叫部分可能有助於識別可疑活動。

Amazon EKS 稽核日誌是選用的資料來源套件，可新增至您的 Detective 行為圖表。您可以在帳戶中檢視可用的選用來源套件及其狀態。如需詳細資訊，請參閱《Amazon Detective 使用者指南》中的 Detective 的 Amazon EKS 稽核日誌。

將 Amazon Detective 與 Amazon EKS 搭配使用

您必須先在叢集所在的相同 AWS 區域中啟用 Detective 至少 48 小時，才能檢閱問題清單。如需詳細資訊，請參閱《Amazon Detective 使用者指南》中的 設定 Amazon Detective。

1. 打開 Detective 主控台，網址為 https://console.aws.amazon.com/detective/。
2. 從左側導覽窗格選取搜尋。
3. 選取選擇類型，然後選取 EKS 叢集。

4. 輸入叢集名稱或 ARN，然後選擇搜尋。
5. 在搜尋結果中，選取要檢視其活動的叢集名稱。如需有關您可以檢視之內容的詳細資訊，請參閱《Amazon Detective 使用者指南》中的涉及 Amazon EKS 叢集的整體 Kubernetes API 活動。

使用 Amazon GuardDuty 偵測威脅

Amazon GuardDuty 是一種威脅偵測服務，可協助保護您的 帳戶、容器、工作負載和資料與您的 AWS 環境。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間活動，以識別環境中的潛在安全風險和惡意活動，並排定其優先順序。

除了其他功能之外，GuardDuty 還提供以下兩種功能，可偵測對 EKS 叢集的潛在威脅：EKS 保護和執行期監控。

> **Note**
>
> 新功能：Amazon EKS Auto Mode 與 GuardDuty 整合。

EKS 保護

此功能提供威脅偵測涵蓋範圍，可協助您透過監控相關聯的 Kubernetes 稽核日誌來保護 Amazon EKS 叢集。Kubernetes 稽核日誌會擷取叢集內的循序動作，包括來自使用者的活動、使用 Kubernetes API 的應用程式，以及控制平面。例如，GuardDuty 可以識別名為 的 APIs 可能竄改 Kubernetes 叢集中的資源，是由未驗證的使用者叫用。

當您啟用 EKS 保護時，GuardDuty 將只能存取您的 Amazon EKS 稽核日誌以進行持續威脅偵測。如果 GuardDuty 識別叢集的潛在威脅，它會產生特定類型的關聯 Kubernetes 稽核日誌問題清單。如需 Kubernetes 稽核日誌中可用問題清單類型的詳細資訊，請參閱《Amazon GuardDuty 使用者指南》中的 Kubernetes 稽核日誌問題清單類型。

如需詳細資訊，請參閱《Amazon GuardDuty 使用者指南》中的 EKS 保護。

執行期監控

此功能會監控和分析作業系統層級、聯網和檔案事件，以協助您偵測環境中特定 AWS 工作負載的潛在威脅。

當您啟用執行期監控並在 Amazon EKS 叢集中安裝 GuardDuty 代理程式時，GuardDuty 會開始監控與此叢集相關聯的執行期事件。請注意，GuardDuty 代理程式和執行期監控不適用於 Amazon EKS 混合節點，因此執行期監控不適用於混合節點上發生的執行期事件。如果 GuardDuty 識別叢

集的潛在威脅，則會產生相關聯的執行期監控調查結果。例如，威脅可能從入侵執行易受攻擊 Web 應用程式的單一容器開始。此 Web 應用程式可能具有基礎容器和工作負載的存取許可。在此案例中，設定不正確的登入資料可能會導致對 帳戶以及其中存放的資料進行更廣泛的存取。

若要設定執行期監控，請將 GuardDuty 代理程式安裝至叢集，做為 Amazon EKS 附加元件。如需附加元件的詳細資訊，請參閱 the section called “ AWS 附加元件”。

如需詳細資訊，請參閱《Amazon GuardDuty 使用者指南》中的執行期監控。

使用彈性中樞評估 EKS 叢集 AWS 彈性

AWS Resilience Hub 透過分析 Amazon EKS 叢集的基礎設施來評估其彈性。 AWS Resilience Hub 使用 Kubernetes 角色型存取控制 (RBAC) 組態來評估部署至叢集的 Kubernetes 工作負載。如需詳細資訊，請參閱《 AWS 復原中心使用者指南》中的啟用對 Amazon EKS 叢集的復原中心存取權。 AWS

使用 Security Lake 集中和分析 EKS 安全資料

Amazon Security Lake 是一項全受管安全資料湖服務，可讓您集中各種來源的安全資料，包括 Amazon EKS。透過將 Amazon EKS 與 Security Lake 整合，您可以深入了解在 Kubernetes 資源上執行的活動，並增強 Amazon EKS 叢集的安全狀態。

> **Note**
>
> 如需搭配 Amazon EKS 使用 Security Lake 和設定資料來源的詳細資訊，請參閱 Amazon Security Lake 文件。

搭配 Amazon EKS 使用 Security Lake 的優勢

集中式安全資料 — Security Lake 會自動收集和集中來自 Amazon EKS 叢集的安全資料，以及其他 AWS 服務、SaaS 供應商、內部部署來源和第三方來源的資料。這可讓您全面檢視整個組織的安全狀態。

標準化資料格式 — Security Lake 將收集的資料轉換為開放網路安全結構描述架構 (OCSF) 格式，這是標準的開放原始碼結構描述。此標準化可讓分析更輕鬆，並與其他安全工具和服務整合。

改善威脅偵測 – 透過分析集中式安全資料，包括 Amazon EKS 控制平面日誌，您可以更有效地偵測 Amazon EKS 叢集內的潛在可疑活動。這有助於快速識別和回應安全事件。

簡化資料管理 — Security Lake 使用可自訂的保留和複寫設定來管理安全資料的生命週期。這可簡化資料管理任務，並確保您保留必要的資料，以用於合規和稽核目的。

啟用 Security Lake for Amazon EKS

1. 為您的 EKS 叢集啟用 Amazon EKS 控制平面記錄。如需詳細說明，請參閱啟用和停用控制平面日誌。
2. 在 Security Lake 中新增 Amazon EKS 稽核日誌作為來源。然後，Security Lake 將開始收集在 EKS 叢集中執行的 Kubernetes 資源上執行之活動的深入資訊。
3. 根據您的需求，在 Security Lake 中設定安全資料的保留和複寫設定。
4. 使用存放在 Security Lake 中的標準化 OCSF 資料，進行事件回應、安全分析，以及與其他 AWS 服務或第三方工具的整合。例如，您可以使用 Amazon OpenSearch Ingestion 從 Amazon Security Lake 資料產生安全洞見。

在 Security Lake 中分析 EKS 日誌

Security Lake 會將 EKS 日誌事件標準化為 OCSF 格式，讓您更輕鬆地分析資料，並將其與其他安全事件建立關聯。您可以使用各種工具和服務，例如 Amazon Athena、Amazon QuickSight 或第三方安全分析工具，來查詢和視覺化標準化資料。

如需 EKS 日誌事件 OCSF 映射的詳細資訊，請參閱 OCSF GitHub 儲存庫中的 https://github.com/ocsf/examples/tree/main/mappings/markdown/ AWS：/v1.1.0/EKS 稽核日誌【映射參考】。

使用 Amazon VPC Lattice 啟用安全的跨叢集連線

Amazon VPC Lattice 是直接內建於 AWS 網路基礎設施的全受管應用程式聯網服務，可用於跨多個帳戶和虛擬私有雲端 (VPCs) 來連接、保護和監控您的服務。透過 Amazon EKS，您可以透過使用 AWS Gateway API 控制器來利用 Amazon VPC Lattice，這是 Kubernetes Gateway API 的實作。您可以使用 Amazon VPC Lattice，以簡單且一致的方式使用標準 Kubernetes 語意設定跨叢集連線。若要開始將 Amazon VPC Lattice 與 Amazon EKS 搭配使用，請參閱 AWS 閘道 API 控制器使用者指南。

使用 AWS Local Zones 啟動低延遲 EKS 叢集

Local AWS 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 AWS Direct Connect。在本機區域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《Amazon EC2 AWS 使用者指南》中的 Local Zones 使用者指南和 Local Zones。 Amazon EC2

Amazon EKS 支援本機區域中的特定資源。這包括受管節點群組、自我管理的 Amazon EC2 節點、Amazon EBS 磁碟區和 Application Load Balancer (ALBs)。在使用本機區域作為 Amazon EKS 叢集的一部分時，建議您考慮以下事項。

- 您無法使用 Amazon EKS 在 Local Zones 中建立 Fargate 節點。
- Amazon EKS 受管 Kubernetes 控制平面一律會在 區域中執行 AWS 。Amazon EKS 受管 Kubernetes 控制平面無法在 Local Zone 中執行。由於 Local Zones 在 VPC 內顯示為子網路，因此 Kubernetes 會將您的本機區域資源作為該子網路的一部分。
- Amazon EKS Kubernetes 叢集會使用 Amazon EKS 受管彈性網路介面，與您在區域或本機區域中執行的 Amazon EC2 執行個體通訊。 AWS https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/using-eni.html若要進一步了解 Amazon EKS 聯網架構，請參閱 設定聯網。
- 與區域子網路不同，Amazon EKS 無法將網路介面放入您的 Local Zone 子網路。這表示建立叢集時，您不得指定本機區域子網路。不過，您可以在不同的多個本機區域中將工作者節點連接到相同的叢集。

使用 Amazon EKS 連接器將 Kubernetes 叢集連接至 Amazon EKS 管理主控台

您可以使用 Amazon EKS 連接器來註冊任何符合標準的 Kubernetes 叢集，並將其連接到 ， AWS 並在 Amazon EKS 主控台中將其視覺化。將叢集連線後，您可以在 Amazon EKS 主控台中查看叢集的狀態、組態和工作負載。您可以使用此功能在 Amazon EKS 主控台中檢視連線的叢集，但無法管理它們。Amazon EKS 連接器需要屬於 Github 上的開放原始碼專案的代理程式。如需其他技術內容，包括常見問答集和故障診斷，請參閱 the section called “對 EKS 連接器進行故障診斷”。

Amazon EKS 連接器會將下列類型的 Kubernetes 叢集連線至 Amazon EKS。

- 內部部署 Kubernetes 叢集
- 在 Amazon EC2 上執行的自我管理叢集
- 其他雲端供應商的受管叢集

Amazon EKS 連接器考量事項

使用 Amazon EKS 連接器之前，請先詳讀下列注意事項：

- 您必須擁有 Kubernetes 叢集的管理權限，才能將其連線至 Amazon EKS。
- 在連線之前，Kubernetes 叢集必須已具備 Linux 64 位元 (x86) 的工作節點。不支援 ARM 工作者節點。
- 您的 Kubernetes 叢集中必須具備工作節點，這些節點可向外存取 `ssm.` 和 `ssmmessages.` Systems Manager 端點。如需詳細資訊，請參閱《 AWS 一般參考》中的 Systems Manager 端點。
- 根據預設，一個區域中最多可連線 10 個叢集。您可以透過 service quota 主控台請求增加配額。如需詳細資訊，請參閱請求增加配額。
- 外部 Kubernetes 叢集僅支援 Amazon EKS `RegisterCluster`、`ListClusters`、`DescribeCluster` 以及 `DeregisterCluster` API。
- 您必須擁有以下許可才能註冊叢集：
  - eks:RegisterCluster
  - ssm:CreateActivation
  - ssm:DeleteActivation
  - iam:PassRole

- 您必須擁有以下許可才能取消註冊叢集：
  - eks:DeregisterCluster
  - ssm:DeleteActivation
  - ssm:DeregisterManagedInstance

Amazon EKS 連接器的必要 IAM 角色

使用 Amazon EKS 連接器需要以下兩種 IAM 角色：

- Amazon EKS 連接器服務連結角色是在第一次註冊叢集時建立。
- 您必須建立 Amazon EKS 連接器代理程式 IAM 角色。如需詳細資訊，請參閱 the section called “連接器 IAM 角色”。

若要為 IAM 主體啟用叢集和工作負載檢視許可，您必須將 `eks-connector` 和 Amazon EKS 連接器叢集角色套用至叢集。請遵循授予存取權中的步驟，在 Amazon EKS 主控台上檢視 Kubernetes 叢集資源。

將外部 Kubernetes 叢集連線至 Amazon EKS 管理主控台

您可以在下列程序中使用多種方法，將外部 Kubernetes 叢集連線至 Amazon EKS。此程序包含兩個步驟：向 Amazon EKS 註冊叢集，以及在叢集中安裝 `eks-connector` 代理程式。

> **Important**
>
> 您必須在完成第一個步驟後的 3 天內完成第二個步驟，以免註冊過期。

考量事項

您可以在安裝代理程式時使用 YAML 資訊清單。或者，如果您向 AWS Management Console 或 AWS 命令列界面註冊叢集，則可以使用 Helm。不過，如果您向 註冊叢集，則無法使用 Helm 安裝代理程式`eksctl`。

先決條件

- 確認已建立 Amazon EKS 連接器代理程式角色。請遵循建立 Amazon EKS 連接器代理程式角色中的步驟。

- 您必須擁有以下許可才能註冊叢集：
  - `eks:RegisterCluster`
  - `ssm:CreateActivation`
  - `ssm:DeleteActivation`
  - `iam:PassRole`

步驟 1：註冊叢集

若要將叢集註冊至 Amazon EKS 連接器，您可以使用下列其中一個工具：

- the section called “AWS CLI”
- the section called “AWS Management Console”
- the section called “`eksctl`”

AWS CLI

1. AWS 必須安裝 CLI。若要安裝或升級，請參閱安裝 AWS CLI。
2. 對於連接器組態，請指定 Amazon EKS 連接器代理程式 IAM 角色。如需詳細資訊，請參閱the section called “Amazon EKS 連接器的必要 IAM 角色”。

```
aws eks register-cluster \
     --name my-first-registered-cluster \
     --connector-config roleArn=arn:aws: iam::111122223333:role/
AmazonEKSConnectorAgentRole,provider="OTHER" \
     --region aws-region
```

範例輸出如下。

```
{
    "cluster": {
        "name": "my-first-registered-cluster",
        "arn": "arn:aws: eks:region:111122223333:cluster/my-first-registered-
cluster",
        "createdAt": 1627669203.531,
        "ConnectorConfig": {
            "activationId": "xxxxxxxxACTIVATION_IDxxxxxxxx",
            "activationCode": "xxxxxxxxACTIVATION_CODExxxxxxxx",
```

```
            "activationExpiry": 1627672543.0,
            "provider": "OTHER",
            "roleArn": "arn:aws: iam::111122223333:role/AmazonEKSConnectorAgentRole"
        },
        "status": "CREATING"
    }
}
```

您會在接下來的步驟中使用 `aws-region`、`activationId` 和 `activationCode` 值。

AWS Management Console

1. 開啟 Amazon EKS 主控台。
2. 選擇 Add cluster (新增叢集)，然後選取 Register (註冊) 以顯示組態頁面。
3. 在 Configure cluster (設定叢集) 區段上，填寫下列欄位：
   - Name (名稱) – 叢集的唯一名稱。
   - Provider (供應商) – 選擇以顯示 Kubernetes 叢集供應商的下拉式清單。如果您不知道特定提供者，請選取其他。
   - EKS Connector role (EKS 連接器角色)：選取用於連接叢集的角色。
4. 選取 Register cluster (註冊叢集)。
5. 系統隨即會顯示叢集概觀頁面。如果您想使用 Helm Chart，請複製 `helm install` 命令並繼續下一個步驟。如果您想要使用 YAML 清單檔案，請選擇下載 YAML 檔案，將清單檔案檔案下載至本機磁碟機。

> **Important**
>
> 這是您複製 `helm install` 命令或下載此檔案的唯一機會。請勿離開此頁面，因為連結將無法存取，您必須取消註冊叢集並從頭開始步驟。

僅能針對已註冊叢集使用命令或清單檔案檔案一次。如果從 Kubernetes 叢集刪除資源，您必須重新註冊叢集並取得新的清單檔案檔案。

繼續執行下一步，將清單檔案檔案套用到 Kubernetes 叢集。

eksctl

1. 必須安裝 eksctl 版本 0.68 或更新版本。若要將其安裝或升級，請參閱 the section called “建立叢集 (eksctl)”。
2. 透過提供名稱、供應商和區域來註冊叢集。

```
eksctl register cluster --name my-cluster --provider my-provider --region region-code
```

輸出範例：

```
2021-08-19 13:47:26 [#]  creating IAM role "eksctl-20210819194112186040"
2021-08-19 13:47:26 [#]  registered cluster "<name>" successfully
2021-08-19 13:47:26 [#]  wrote file eks-connector.yaml to <current directory>
2021-08-19 13:47:26 [#]  wrote file eks-connector-clusterrole.yaml to <current
 directory>
2021-08-19 13:47:26 [#]  wrote file eks-connector-console-dashboard-full-access-
group.yaml to <current directory>
2021-08-19 13:47:26 [!]  note: "eks-connector-clusterrole.yaml" and "eks-connector-
console-dashboard-full-access-group.yaml" give full EKS Console access to IAM
 identity "<aws-arn>", edit if required; read https://eksctl.io/usage/eks-connector
 for more info
2021-08-19 13:47:26 [#]  run `kubectl apply -f eks-connector.yaml,eks-connector-
clusterrole.yaml,eks-connector-console-dashboard-full-access-group.yaml` before
 expiry> to connect the cluster
```

這會在您的本機電腦上建立檔案。這些檔案必須在 3 天內套用至外部叢集，否則註冊將會過期。

3. 在可存取叢集的終端機中，套用 eks-connector-binding.yaml 檔案：

```
kubectl apply -f eks-connector-binding.yaml
```

步驟 2：安裝 eks-connector 代理程式

若要安裝代理eks-connector程式，請使用下列其中一個工具：

- the section called “Helm”
- the section called “yaml”

Helm

> **Note**
>
> 如果您已向 註冊叢集`eksctl`，請使用 YAML 資訊清單方法，而非 Helm Chart 方法。

1. 如果您在上一個步驟中使用 AWS CLI，請將下列命令`ACTIVATION_ID`中的 `ACTIVATION_CODE`和分別取代為 `activationId`和 `activationCode`值。`aws-region` 將 取代為您在上一個步驟中使用的 AWS 區域。接著執行命令，在註冊的叢集上安裝 `eks-connector` 代理程式：

   ```
   $ helm install eks-connector \
     --namespace eks-connector \
     oci://public.ecr.aws/eks-connector/eks-connector-chart \
     --set eks.activationCode=ACTIVATION_CODE \
     --set eks.activationId=ACTIVATION_ID \
     --set eks.agentRegion=aws-region
   ```

   如果您在上一個步驟 AWS Management Console 中使用 ，請使用您從上一個步驟複製的命令，其中已填入這些值。
2. 檢查已安裝 `eks-connector` 部署的健康狀態，並等待 Amazon EKS 中已註冊叢集的狀態成為 `ACTIVE`。

yaml

將 Amazon EKS 連接器清單檔案檔案套用至 Kubernetes 叢集，以便完成連線。若要執行此動作，您必須使用先前描述的方法。如果資訊清單未在三天內套用，Amazon EKS 連接器註冊就會過期。如果叢集連線過期，則必須先取消註冊叢集，才能再次連接叢集。

1. 下載 Amazon EKS 連接器 YAML 檔案。

   ```
   curl -O https://amazon-eks.s3.us-west-2.amazonaws.com/eks-connector/manifests/eks-connector/latest/eks-connector.yaml
   ```
2. 編輯 Amazon EKS 連接器 YAML 檔案，以來自上一個步驟輸出的 `aws-region`、`activationId` 和 `activationCode` 來取代 `%AWS_REGION%`、`%EKS_ACTIVATION_ID%`、`%EKS_ACTIVATION_CODE%` 的所有參考。

   下列範例命令可以取代這些值。

```
sed -i "s~%AWS_REGION%~$aws-region~g; s~%EKS_ACTIVATION_ID%~$EKS_ACTIVATION_ID~g; s~
%EKS_ACTIVATION_CODE%~$(echo -n $EKS_ACTIVATION_CODE | base64)~g" eks-connector.yaml
```

> **Important**
>
> 確認您的啟用代碼格式為 Base64。

3. 在可存取叢集的終端機中，您可以執行以下命令來套用已更新的清單檔案檔案：

```
kubectl apply -f eks-connector.yaml
```

4. 將 Amazon EKS 連接器清單檔案，以及角色繫結 YAML 檔案套用至 Kubernetes 叢集後，請確認叢集已成功連線。

```
aws eks describe-cluster \
     --name "my-first-registered-cluster" \
     --region AWS_REGION
```

   輸出應該包含 `status=ACTIVE`。

5. (選用) 將標籤新增至您的叢集。如需詳細資訊，請參閱the section called “標記您的 資源”。

後續步驟

如果您對這些步驟有任何問題，請參閱 the section called “對 EKS 連接器進行故障診斷”。

若要授予其他 IAM 主體存取 Amazon EKS 主控台以檢視連線叢集中的 Kubernetes 資源，請參閱 the section called “授予對叢集的存取權”。

授予在 Amazon EKS 主控台上檢視 Kubernetes 叢集資源的存取權

授予 IAM 主體對 Amazon EKS 主控台的存取權，以檢視在連線叢集上執行之 Kubernetes 資源的相關資訊。

先決條件

您用來存取 的 IAM 主體 AWS Management Console 必須符合下列要求：

- 它必須具有 eks:AccessKubernetesApi IAM 許可。
- Amazon EKS 連接器服務帳戶可以模擬叢集中的 IAM 主體。這可讓 Amazon EKS 連接器將 IAM 主體映射至 Kubernetes 使用者。

建立和套用 Amazon EKS Connector 叢集角色

1. 下載 eks-connector 叢集角色範本。

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/eks-connector/manifests/eks-connector-console-roles/eks-connector-clusterrole.yaml
```

2. 編輯叢集角色範本 YAML 檔案。將對 %IAM_ARN% 的參考取代為 IAM 主體的 Amazon Resource Name (ARN)。
3. 將 Amazon EKS 連接器叢集角色 YAML 套用至 Kubernetes 叢集。

```
kubectl apply -f eks-connector-clusterrole.yaml
```

若要讓 IAM 主體在 Amazon EKS 主控台中檢視 Kubernetes 資源，該主體必須與 Kubernetes 相關聯，role或clusterrole具有讀取資源的必要許可。如需詳細資訊，請參閱 Kubernetes 文件中的使用 RBAC 授權。

設定 IAM 主體以存取連線的叢集

1. 您可以下載照下範例清單檔案來分別建立 clusterrole 和 clusterrolebinding 或 role 和 rolebinding：

   檢視所有命名空間中的 Kubernetes 資源

   - eks-connector-console-dashboard-full-access-clusterrole 叢集角色可讓您存取所有可在主控台中視覺化的命名空間和資源。您可以變更 role、clusterrole 及其相應繫結的名稱，然後再將其套用至叢集。使用以下命令下載範例檔案。

     ```
     curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/eks-connector/manifests/eks-connector-console-roles/eks-connector-console-dashboard-full-access-group.yaml
     ```

   檢視特定命名空間中的 Kubernetes 資源

   - 此檔案中的命名空間為 default，因此若要指定不同的命名空間，請在將其套用至叢集之前編輯該檔案。使用以下命令下載範例檔案。

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/eks-connector/manifests/
eks-connector-console-roles/eks-connector-console-dashboard-restricted-access-
group.yaml
```

2. 編輯 YAML 檔案的完整存取權或受限存取權，會將 %IAM_ARN% 的參考取代為 IAM 主體的 Amazon Resource Name (ARN)。
3. 將 YAML 檔案的完整存取權或受限存取權套用到 Kubernetes 叢集。以您自己的值取代 YAML 檔案值。

```
kubectl apply -f eks-connector-console-dashboard-full-access-group.yaml
```

若要檢視已連接叢集中的 Kubernetes 資源，請參閱 the section called “存取叢集資源”。資源索引標籤上某些資源類型的資料不適用於連線的叢集。

從 Amazon EKS 主控台取消註冊 Kubernetes 叢集

如果連接叢集已使用完畢，即可將其取消註冊。取消註冊後，叢集就不會再出現在 Amazon EKS 主控台中。

您必須擁有以下許可，才能呼叫 deregisterCluster API：

- `eks:DeregisterCluster`
- `ssm:DeleteActivation`
- `ssm:DeregisterManagedInstance`

此程序包含兩個步驟：向 Amazon EKS 取消註冊叢集，以及在叢集中取消安裝 eks-connector 代理程式。

取消註冊 Kubernetes 叢集

若要從 Amazon EKS 連接器取消註冊叢集，您可以使用下列其中一個工具：

- the section called “AWS CLI”
- the section called “AWS Management Console”
- the section called “`eksctl`”

AWS CLI

1. AWS 必須安裝 CLI。若要安裝或升級，請參閱安裝 AWS CLI。
2. 確認已建立 Amazon EKS 連接器代理程式角色。
3. 取消註冊已連接叢集。

```
aws eks deregister-cluster \
    --name my-cluster \
    --region region-code
```

AWS Management Console

1. 開啟 Amazon EKS 主控台。
2. 選擇 Clusters (叢集)。
3. 在 Clusters (叢集) 頁面上，選取已連接叢集，然後選取 Deregister (取消註冊)。
4. 確認您要取消註冊叢集。

`eksctl`

1. 安裝 `eksctl` 版本 `0.68` 或更新版本。若要將其安裝或升級，請參閱 the section called “建立叢集 (eksctl)”。
2. 確認已建立 Amazon EKS 連接器代理程式角色。
3. 取消註冊連線的叢集：

```
eksctl deregister cluster --name my-cluster
```

清除 Kubernetes 叢集中的資源

若要解除安裝`eks-connector`代理程式，請使用下列其中一個工具：

- the section called “helm”
- the section called “yaml”

helm

執行下列命令以解除安裝代理程式。

```
helm -n eks-connector uninstall eks-connector
```

yaml

1. 從 Kubernetes 叢集中刪除 Amazon EKS 連接器 YAML 檔案。

   ```
   kubectl delete -f eks-connector.yaml
   ```

2. 如果您建立 `clusterrole`或 `clusterrolebindings` ，讓其他 IAM 主體存取叢集，請從 Kubernetes 叢集刪除它們。

對 Amazon EKS 連接器問題進行故障診斷

本主題涵蓋您在使用 Amazon EKS 連接器時可能遇到的一些常見錯誤，包括有關如何解決錯誤和因應措施的說明。

基本疑難排解

本節說明診斷 Amazon EKS 連接器問題的步驟。

檢查 Amazon EKS 連接器狀態

若要檢查 Amazon EKS 連接器狀態，請輸入：

```
kubectl get pods -n eks-connector
```

檢查 Amazon EKS 連接器日誌

Amazon EKS 連接器 Pod 由三個容器組成。若要擷取所有這些容器的完整日誌，以便您可以檢查它們，請執行以下命令：

- `connector-init`

  ```
  kubectl logs eks-connector-0 --container connector-init -n eks-connector
  ```

```
kubectl logs eks-connector-1 --container connector-init -n eks-connector
```

- connector-proxy

```
kubectl logs eks-connector-0 --container connector-proxy -n eks-connector
kubectl logs eks-connector-1 --container connector-proxy -n eks-connector
```

- connector-agent

```
kubectl exec eks-connector-0 --container connector-agent -n eks-connector -- cat /
var/log/amazon/ssm/amazon-ssm-agent.log
kubectl exec eks-connector-1 --container connector-agent -n eks-connector -- cat /
var/log/amazon/ssm/amazon-ssm-agent.log
```

取得有效的叢集名稱

Amazon EKS 叢集由clusterName單一 AWS 帳戶和 AWS 區域內的唯一識別。如果您在 Amazon EKS 中有多個連線叢集，您可以確認目前 Kubernetes 叢集已註冊到哪個 Amazon EKS 叢集。為此，請輸入下列內容來找出目前叢集的 clusterName。

```
kubectl exec eks-connector-0 --container connector-agent -n eks-connector \
  -- cat /var/log/amazon/ssm/amazon-ssm-agent.log | grep -m1 -oE "eks_c:[a-zA-Z0-9_-]+"
 | sed -E "s/^.*eks_c:([a-zA-Z0-9_-]+)_[a-zA-Z0-9]+.*$/\1/"
kubectl exec eks-connector-1 --container connector-agent -n eks-connector \
  -- cat /var/log/amazon/ssm/amazon-ssm-agent.log | grep -m1 -oE "eks_c:[a-zA-Z0-9_-]+"
 | sed -E "s/^.*eks_c:([a-zA-Z0-9_-]+)_[a-zA-Z0-9]+.*$/\1/"
```

其他命令

擷取所需資訊以進行問題疑難排解時，以下命令很有用。

- 使用下列命令來收集 Pod 在 Amazon EKS 連接器中使用的映像。

```
kubectl get pods -n eks-connector -o jsonpath="{.items[*].spec.containers[*].image}"
 | tr -s '[[:space:]]' '\n'
```

- 使用以下命令判斷正在執行 Amazon EKS 連接器的節點名稱。

```
kubectl get pods -n eks-connector -o jsonpath="{.items[*].spec.nodeName}" | tr -s
 '[[:space:]]' '\n'
```

- 執行以下命令，以取得 Kubernetes 用戶端和伺服器版本。

```
kubectl version
```

- 執行以下命令，以取得有關節點的資訊。

```
kubectl get nodes -o wide --show-labels
```

Helm 問題：403 禁止

如果您在執行 helm 安裝命令時收到下列錯誤：

```
Error: INSTALLATION FAILED: unexpected status from HEAD request to https://
public.ecr.aws/v2/eks-connector/eks-connector-chart/manifests/0.0.6: 403 Forbidden
```

您可以執行下列行來修復錯誤：

```
docker logout public.ecr.aws
```

主控台錯誤：叢集停留在待定狀態

如果叢集在您註冊之後卡在 Amazon EKS 主控台`Pending`的狀態，可能是因為 Amazon EKS 連接器尚未成功將叢集連線至 AWS 。對於已註冊的叢集，`Pending`狀態表示連線未成功建立。若要解決此問題，請確定您已將清單檔案應用於目標 Kubernetes 叢集。如果您將其套用於叢集，但叢集仍處於 `Pending` 狀態，則 `eks-connector` statefulset 很可能無法正常運作。若要排除此問題，請參閱本主題中的 the section called “Amazon EKS 連接器 Pod 正在損毀迴圈”。

主控台錯誤：使用者系統：serviceaccount：eks-connector：eks-connector 無法在叢集範圍內模擬 API 群組中的資源使用者

Amazon EKS 連接器使用 Kubernetes 使用者模擬，代表來自 的 IAM 主體。 AWS Management Console每個從 AWS `eks-connector`服務帳戶存取 Kubernetes API 的主體都必須獲得許可，才能以 IAM ARN 做為其 Kubernetes 使用者名稱來模擬對應的 Kubernetes 使用者。在以下範例中，IAM ARN 映射到 Kubernetes 使用者。

- 從 AWS 帳戶 *111122223333 ##*的 IAM 使用者會映射到 Kubernetes 使用者。IAM 最佳實務建議您將許可授予角色而非使用者。

```
arn:aws: iam::111122223333:user/john
```

- AWS 帳戶 *111122223333* 的 IAM 角色*###*會映射至 Kubernetes 使用者：

```
arn:aws: iam::111122223333:role/admin
```

結果是 IAM 角色 ARN，而不是 AWS STS 工作階段 ARN。

如需如何設定 `ClusterRole` 和 `ClusterRoleBinding` 以授予 `eks-connector` 服務帳戶權限來模擬映射使用者的說明，請參閱 the section called “授予對叢集的存取權”。請確定在範本中，`%IAM_ARN%` 已取代為 IAM 主體的 AWS Management Console IAM ARN。

主控台錯誤：【... 】 被禁止：使用者 【... 】 無法在叢集範圍內列出 API 群組中的資源 【... 】

請考量下列問題。Amazon EKS 連接器已成功模擬目標 Kubernetes AWS Management Console 叢集中的請求 IAM 主體。不過，模擬主體沒有 Kubernetes API 操作的 RBAC 許可。

若要解決此問題，有兩種方法可將許可授予其他使用者。如果您先前透過 Helm Chart 安裝了 eks-connector，您可以透過執行下列命令輕鬆授予使用者存取權。將 `userARN1`和 取代`userARN2`為 IAM 角色的 ARNs 清單，以授予檢視 Kubernetes 資源的存取權：

```
helm upgrade eks-connector oci://public.ecr.aws/eks-connector/eks-connector-chart \
    --reuse-values \
    --set 'authentication.allowedUserARNs={userARN1,userARN2}'
```

或者，身為叢集管理員，將適當層級的 RBAC 權限授予個別 Kubernetes 使用者。如需詳細資訊和範例，請參閱 the section called “授予對叢集的存取權”。

主控台錯誤：Amazon EKS 無法與您的 Kubernetes 叢集 API 伺服器通訊。叢集必須處於作用中狀態，才能成功連接。請過幾分鐘後再試。

如果 Amazon EKS 服務無法與目標叢集中的 Amazon EKS 連接器通訊，可能是因為下列其中一個原因：

- 目標叢集中的 Amazon EKS 連接器無法正常運作。
- 目標叢集與 AWS 區域之間的連線能力不佳或連線中斷。

若要解決此問題，請查看 Amazon EKS 連接器日誌。如果您未看到 Amazon EKS 連接器的錯誤，請在幾分鐘後重試連線。如果您經常遇到目標叢集的高延遲或間歇性連線，請考慮將叢集重新註冊到離您較近 AWS 的區域。

Amazon EKS 連接器 Pod 正在損毀迴圈

有許多原因可能導致 Amazon EKS 連接器 Pod 進入 `CrashLoopBackOff` 狀態。此問題可能涉及 `connector-init` 容器。檢查 Amazon EKS 連接器 Pod 的狀態。

```
kubectl get pods -n eks-connector
```

範例輸出如下。

```
NAME               READY   STATUS                  RESTARTS   AGE
eks-connector-0    0/2     Init:CrashLoopBackOff   1          7s
```

如果您的輸出與之前的輸出相似，請參閱 the section called “檢查 Amazon EKS 連接器日誌” 以排除問題。

無法啟動 eks-connector： InvalidActivation

當您第一次啟動 Amazon EKS 連接器時，它會使用 Amazon Web Services 註冊 `activationId` 和 `activationCode`。註冊可能會失敗，這可能會導致 `connector-init` 容器損毀，產生與以下錯誤相似的錯誤。

```
F1116 20:30:47.261469       1 init.go:43] failed to initiate eks-connector:
 InvalidActivation:
```

若要對此問題進行疑難排解，請考慮以下原因和建議的修正：

- 註冊可能失敗，因為 `activationId`和 `activationCode` 不在資訊清單檔案中。如果是這種情況，請確保它們是從 `RegisterCluster` API 操作傳回的正確值，並且 `activationCode` 位於清單檔案檔案中。`activationCode` 會新增至 Kubernetes `base64` 秘密，因此必須對其進行編碼。如需詳細資訊，請參閱the section called “步驟 1：註冊叢集”。
- 註冊可能因啟用過期而失敗。因為出於安全原因，您必須在註冊叢集後的 3 天內啟用 Amazon EKS 連接器。若要解決此問題，請確定 Amazon EKS 連接器資訊清單在到期日期和時間之前套用至目標 Kubernetes 叢集。若要確認您的啟用過期日期，請呼叫 `DescribeCluster` API 操作。

```
aws eks describe-cluster --name my-cluster
```

在以下範例回應中，過期日期和時間記錄為 2021-11-12T22:28:51.101000-08:00。

```
{
    "cluster": {
        "name": "my-cluster",
        "arn": "arn:aws: eks:region:111122223333:cluster/my-cluster",
        "createdAt": "2021-11-09T22:28:51.449000-08:00",
        "status": "FAILED",
        "tags": {
        },
        "connectorConfig": {
            "activationId": "00000000-0000-0000-0000-000000000000",
            "activationExpiry": "2021-11-12T22:28:51.101000-08:00",
            "provider": "OTHER",
            "roleArn": "arn:aws: iam::111122223333:role/my-connector-role"
        }
    }
}
```

如果超過 activationExpiry，請取消註冊叢集，並將其重新註冊。這樣做會產生新的啟用。

叢集節點遺漏對外連線

若要正常運作，Amazon EKS 連接器需要與多個 AWS 端點的傳出連線。您無法在沒有對外連線至目標 AWS 區域的私有叢集。若要解決此問題，您必須新增必要的對外連線。如需連接器要求的相關資訊，請參閱 the section called “Amazon EKS 連接器考量事項”。

Amazon EKS 連接器 Pod 處於 **ImagePullBackOff** 狀態

如果您執行 get pods命令且 Pod 處於 ImagePullBackOff 狀態，則無法正常運作。如果 Amazon EKS 連接器 Pod 處於 ImagePullBackOff 狀態，則無法正常運作。檢查 Amazon EKS 連接器 Pod 的狀態。

```
kubectl get pods -n eks-connector
```

範例輸出如下。

```
NAME              READY   STATUS                  RESTARTS   AGE
eks-connector-0   0/2     Init:ImagePullBackOff   0          4s
```

預設的 Amazon EKS 連接器清單檔案檔案引用來自 Amazon ECR Public Gallery 的映像。目標 Kubernetes 叢集可能無法從 Amazon ECR Public Gallery 提取映像。解決 Amazon ECR Public Gallery 映像提取問題，或者考慮選擇在私有容器登錄檔中鏡像映像。

AWS 連接器常見問答集

問：Amazon EKS 連接器背後的基礎技術如何運作？

答：Amazon EKS 連接器是以 AWS Systems Manager (Systems Manager) 代理程式為基礎。Amazon EKS 連接器會在 Kubernetes 叢集`StatefulSet`上以 的形式執行。它建立叢集的 API 伺服器與 Amazon Web Services 之間的連線並代理通訊。這樣做會在 Amazon EKS 主控台中顯示叢集資料，直到您中斷叢集與 的連線為止 AWS。Systems Manager 代理程式是開放原始碼專案。如需有關此專案的詳細資訊，請參閱 GitHub 專案頁面。

問：我有一個想要連接的內部部署 Kubernetes 叢集。我是否需要開啟防火牆連接埠才能連接此叢集？

答：否，您不需要開啟任何防火牆連接埠。Kubernetes 叢集只需要對 AWS Regions. AWS services 的傳出連線，就永遠無法存取內部部署網路中的資源。Amazon EKS 連接器會在您的叢集上執行，並啟動 的連線 AWS。當叢集註冊完成時， AWS 只有在您從 Amazon EKS 主控台啟動需要叢集上 Kubernetes API 伺服器資訊的動作之後， 才會向 Amazon EKS 連接器發出命令。

問：Amazon AWS EKS 連接器會將哪些資料從我的叢集傳送至 ？

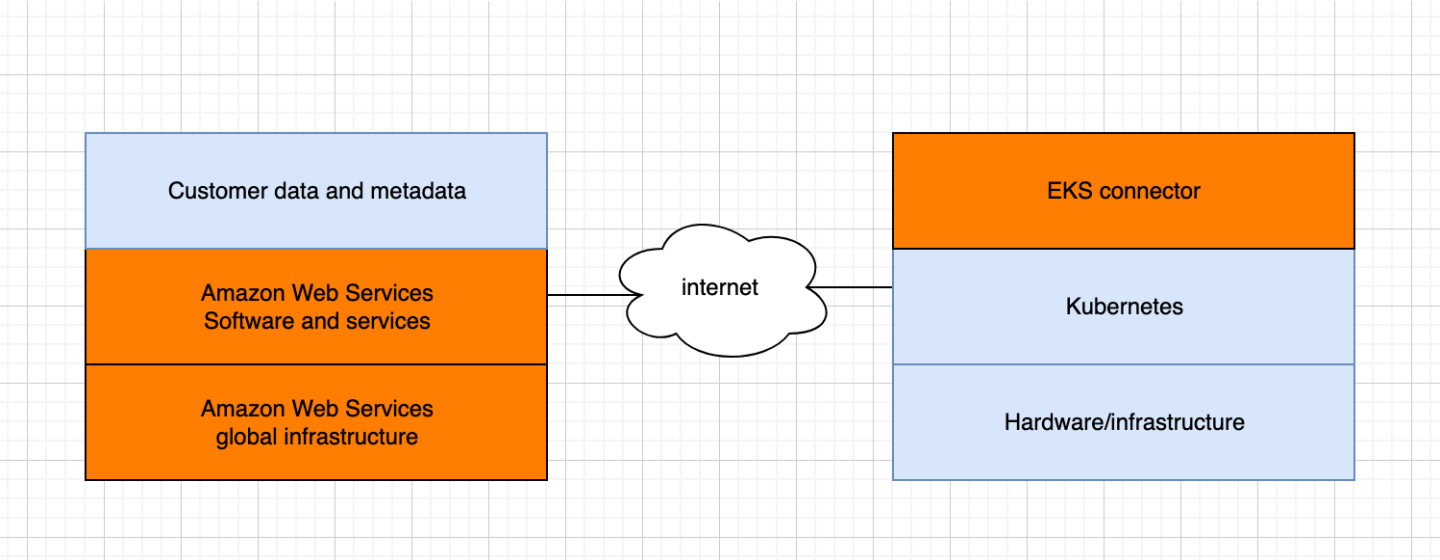
答：Amazon EKS 連接器會傳送註冊叢集所需的技術資訊 AWS。它還會針對客戶請求的 Amazon EKS 主控台功能傳送叢集和工作負載中繼資料。僅在您從必須將資料傳送至 AWS的 Amazon EKS 主控台或 Amazon EKS API 啟動動作時，Amazon EKS 連接器才會收集或傳送此資料。除了 Kubernetes 版本編號之外，預設 AWS 不會儲存任何資料。它只有在您授權時才會存放資料。

問：我可以在 AWS 區域之外連接叢集嗎？

答：您可以將來自任何位置的叢集連接到 Amazon EKS。此外，您的 Amazon EKS 服務可以位於任何 AWS 公有商業 AWS 區域。這適用於從叢集到目標 AWS 區域的有效網路連線。我們建議您挑選最接近叢集位置 AWS 的區域，以進行 UI 效能最佳化。例如，如果您的叢集在東京執行，請將叢集連線到東京的 AWS 區域 （ 也就是 `ap-northeast-1` AWS 區域 ） 以獲得低延遲。您可以將叢集從任何位置連接到任何公有商業區域的 Amazon EKS AWS ，中國或 GovCloud AWS 區域除外。

了解 Amazon EKS 連接器的安全性

Amazon EKS 連接器是一個在您的 Kubernetes 叢集上執行的開放原始碼元件。此叢集可以位於 AWS 環境之外。這對安全責任產生了其他考量。可透過下圖說明此設定。橘色代表 AWS 責任，藍色代表客戶責任：



本主題說明了如果連接的叢集位於 AWS之外的責任模型內的差異。

AWS 責任

- 維護、建置和交付 Amazon EKS 連接器，這是在客戶 Kubernetes 叢集上執行並與之通訊的開放原始碼元件 AWS。
- 維護連線 Kubernetes 叢集和服務之間的傳輸和應用程式層通訊安全性 AWS 。

客戶責任

- Kubernetes 叢集的特定安全，尤其是沿著以下幾行：
  - Kubernetes 秘密必須經過適當加密和保護。
  - 封鎖對 `eks-connector` 命名空間的存取。
- 設定角色型存取控制 (RBAC) 許可，以管理從 AWS的 IAM 主體存取。如需說明，請參閱 the section called “授予對叢集的存取權”。
- 安裝和升級 Amazon EKS 連接器。
- 維護支援連接的 Kubernetes 叢集的硬體、軟體和基礎架構。

- 保護其 AWS 帳戶 （例如，透過保護您的根使用者憑證)。

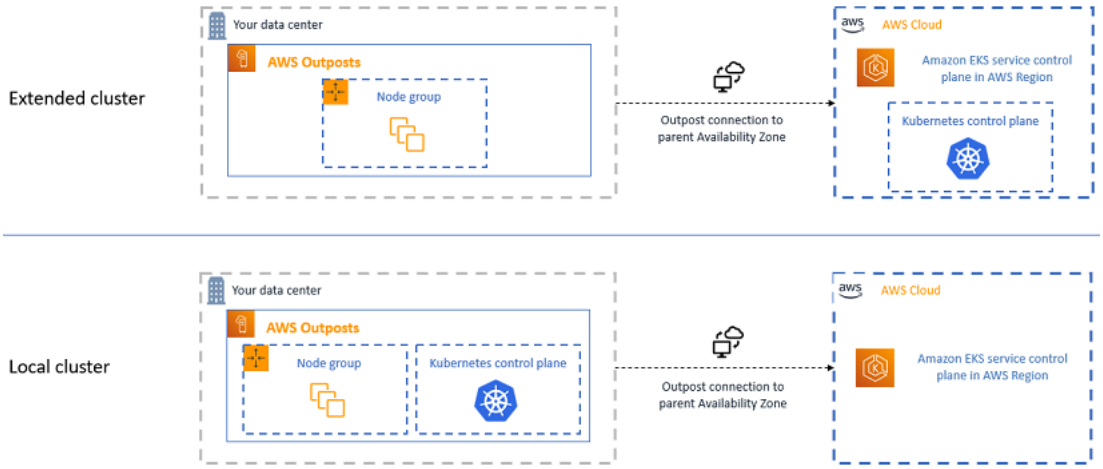
使用 AWS Outposts 部署 Amazon EKS 內部部署

您可以使用 Amazon EKS 在 AWS Outposts 上執行內部部署 Kubernetes 應用程式。您可使用以下方式在 Outpost 上部署 Amazon EKS：

- 擴充叢集 – 在 Outpost 的 AWS 區域和節點中執行 Kubernetes 控制平面。
- 本機叢集 – 在 Outpost 上執行 Kubernetes 控制平面和節點。

對於這兩種部署選項，Kubernetes 控制平面都由 完全管理 AWS。您可以使用在雲端中使用的相同 Amazon EKS API、工具和主控台，在 Outpost 上建立和執行 Amazon EKS。

下圖顯示這些部署選項。



使用每個部署選項的時機

本機叢集和擴充叢集皆為一般用途的部署選項，可用於多種應用程式。

使用本機叢集，您可以在 Outposts 上本機執行整個 Amazon EKS 叢集。此選項可降低因網路暫時與雲端中斷連線而可能導致的應用程式停機風險。這些網路連線中斷可能是因光纖切斷或天氣事件引起的。由於整個 Amazon EKS 叢集會在 Outposts 本機執行，因此仍可使用應用程式。您可在網路與雲端中斷連線期間執行叢集操作。如需詳細資訊，請參閱the section called “準備中斷連線”。如果您擔心從 Outpost 到父 AWS 區域的網路連線品質，並透過網路中斷連線需要高可用性，請使用本機叢集部署選項。

透過擴充叢集，您可以節省 Outpost 的容量，因為 Kubernetes 控制平面在父 AWS 區域中執行。如果您可以投資從 Outpost 到 AWS 區域的可靠備援網路連線，則此選項是合適的。此選項的網路連線品質

十分重要。Kubernetes 處理 Kubernetes 控制平面和節點之間網路中斷連線的方式可能會導致應用程式停機。如需 Kubernetes 行為的詳細資訊，請參閱 Kubernetes 文件中的排程、先佔和移除。

比較部署選項

下表會比較這兩個選項之間的差異。

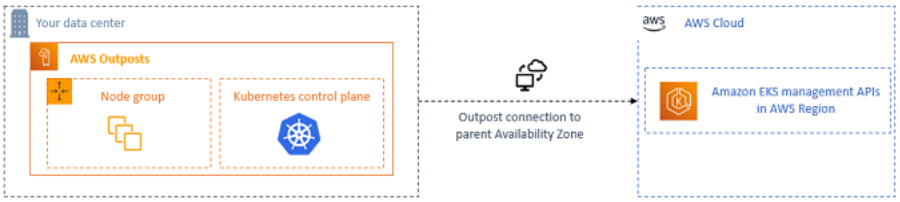
功能	擴充叢集	本機叢集
Kubernetes 控制平面位置	AWS 區域	Outpost
Kubernetes 控制平面帳戶	AWS 帳戶	您的帳戶
區域可用性	請參閱服務端點	美國東部 (俄亥俄)、美國東部 (維吉尼亞北部)、美國西部 (加利佛尼亞北部)、美國西部 (奧勒岡)、亞太區域 (首爾)、亞太區域 (新加坡)、亞太區域 (雪梨)、亞太區域 (東京)、加拿大 (中部)、歐洲 (法蘭克福)、歐洲 (愛爾蘭)、歐洲 (倫敦)、中東 (巴林) 和南美洲 (聖保羅)
Kubernetes 次要版本	eks/latest/userguide/kubernetes-versions.html【支援的 Amazon EKS 版本，type="documentation"】。	eks/latest/userguide/kubernetes-versions.html【支援的 Amazon EKS 版本，type="documentation"】。
平台版本	請參閱 eks/latest/userguide/platform-versions.html【EKS platform-versions，type="documentation"】	請參閱 the section called “EKS 平台版本”
Outpost 外形規格	Outpost 機架	Outpost 機架
使用者介面	AWS Management Console、AWS CLI、Amazon EKS	AWS Management Console、AWS CLI、Amazon EKS

功能	擴充叢集	本機叢集
	API、、`eksctl` AWS CloudFormation 和 Terraform	API、、`eksctl` AWS CloudFormation 和 Terraform
受管政策	AmazonEKSClusterPolicy 和 the section called “AWS 受管政策：AmazonEKSServiceRolePolicy”	AmazonEKSLocalOutpostClusterPolicy 和 the section called “AWS 受管政策：AmazonEKSLocalOutpostServiceRolePolicy”
叢集 VPC 和子網路	請參閱 the section called “VPC 和子網要求”	請參閱 the section called “建立 VPC 和子網路”
叢集端點存取	公有或私有，或兩者兼具	僅限私有
Kubernetes API 伺服器身分驗證	AWS Identity and Access Management (IAM) 和 OIDC	IAM 和 `x.509` 憑證
節點類型	僅限自我管理	僅限自我管理
節點運算類型	Amazon EC2 隨需	Amazon EC2 隨需
節點儲存類型	Amazon EBS gp2 和本機 NVMe SSD	Amazon EBS gp2 和本機 NVMe SSD
Amazon EKS 最佳化 AMI	Amazon Linux、Windows 和 Bottlerocket	僅限 Amazon Linux
IP 版本	僅限 `IPv4`	僅限 `IPv4`
附加元件	Amazon EKS 附加元件，或自我管理附加元件	僅限自我管理附加元件
預設容器網路介面	Kubernetes 專用 Amazon VPC CNI 外掛程式	Kubernetes 專用 Amazon VPC CNI 外掛程式
Kubernetes 控制平面日誌	Amazon CloudWatch Logs	Amazon CloudWatch Logs

功能	擴充叢集	本機叢集
負載平衡	使用AWS Load Balancer控制器僅佈建 Application Load Balancer （無 Network Load Balancer)	使用AWS Load Balancer控制器僅佈建 Application Load Balancer （無 Network Load Balancer)
秘密封套加密	請參閱 the section called “啟用秘密加密”	不支援
服務帳戶的 IAM 角色	請參閱 the section called “具有 IRSA 的登入資料”	不支援
故障診斷	請參閱 故障診斷	請參閱 the section called “對叢集進行疑難排解”

在 AWS Outposts 上建立本機 Amazon EKS 叢集以獲得高可用性

您可以使用本機叢集在 AWS Outposts 本機執行整個 Amazon EKS 叢集。這有助於降低因網路暫時與雲端中斷連線而可能導致的應用程式停機風險。這些連線中斷可能是因光纖切斷或天氣事件引起的。由於整個 Kubernetes 叢集在 Outposts 上於本機執行，因此應用程式仍然可用。您可在網路與雲端中斷連線期間執行叢集操作。如需詳細資訊，請參閱the section called “準備中斷連線”。下圖顯示本機叢集部署。



本機叢集通常可與 Outpost 機架搭配使用。

支援 AWS 的區域

您可以在下列 AWS 區域中建立本機叢集：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）、亞太區域 （首爾）、亞太區域 （新加坡）、亞太區域 （雪梨）、亞太區域 （東京）、加拿大 （中部）、歐洲 （法蘭克福）、歐洲 （愛爾蘭）、歐洲 （倫敦）、中東 （巴林） 和南美洲 （聖保羅）。如需有關支援功能的詳細資訊，請參閱the section called “比較部署選項”。

在 AWS Outposts 上部署 Amazon EKS 叢集

本主題概述了在 Outpost 上執行本機叢集時要考量的事項。本主題也提供如何在 Outpost 上部署本機叢集的指示。

> **Important**
>
> - 這些考量事項不會複寫在相關的 Amazon EKS 文件中。如果其他 Amazon EKS 文件主題與此處的考量衝突，則請遵循此處的考量事項。
> - 這些考量事項隨時會變更，而且可能會經常變更。因此，我們建議您定期檢閱此主題。
> - 許多考量事項與在 AWS 雲端上建立叢集的考量事項不同。

- 本機叢集僅支援 Outpost 機架。單一本機叢集可以在包含單一邏輯 Outpost 的多個實體 Outpost 機架上執行。單一本機叢集無法跨多個邏輯 Outpost 執行。每個邏輯 Outpost 都有一個單一的 Outpost ARN。
- 本機叢集會在 Outpost 上執行和管理您帳戶中的 Kubernetes 控制平面。您無法在 Kubernetes 控制平面執行個體上執行工作負載，或修改 Kubernetes 控制平面元件。這些節點由 Amazon EKS 服務管理。Kubernetes 控制平面的變更不會透過自動 Amazon EKS 管理動作持續進行，例如修補。
- 本機叢集支援自我管理的附加元件和自我管理的 Amazon Linux 節點群組。適用於 Kubernetes、kube-proxy 和 CoreDNS 附加元件的 Amazon VPC CNI 外掛程式會自動安裝在本機叢集上。 ??? CoreDNS
- 本機叢集需要在 Outpost 上使用 Amazon EBS。Outpost 必須有 Amazon EBS 可供 Kubernetes 控制平面儲存使用。Outpost 僅支援 Amazon EBS gp2 磁碟區。
- Amazon EBS 支援的 Kubernetes `PersistentVolumes`支援使用 Amazon EBS CSI 驅動程式。
- 本機叢集的控制平面執行個體是在堆疊的高可用性拓撲中設定。三分之二的控制平面執行個體必須隨時正常運作，才能維持規定人數。如果規定人數遺失，請聯絡 AWS 支援，因為需要一些服務端動作才能啟用新的受管執行個體。

先決條件

- 熟悉 Outposts 部署選項、根據容量考量以及 VPC 需求和考量，為 AWS Outposts 上的 Amazon EKS 叢集選取執行個體類型和置放群組。 ???
- 現有的 Outpost。如需詳細資訊，請參閱什麼是 AWS Outpost。

- kubectl 命令列工具安裝在您的電腦或 AWS CloudShell 上。版本可以與叢集的 Kubernetes 版本相同，也可以比叢集的 Kubernetes 版本更早或更晚一個次要版本。例如，如果您的叢集版本為 1.29，則可以搭配使用 kubectl 1.28、1.29 或 1.30 版。若要安裝或升級 kubectl，請參閱 the section called “設定 kubectl 和 eksctl”。
- 在您的裝置或 AWS CloudShell 上安裝和設定的 AWS 命令列界面 (AWS CLI) 版本 1.27.160 2.12.3或更新版本。若要檢查您目前的版本，請使用 aws --version | cut -d / -f2 | cut -d ' ' -f1。適用於 macOS 的 yum、 apt-get或 Homebrew 等套件管理員通常是最新版本 CLI AWS 後面的幾個版本。若要安裝最新版本，請參閱《 AWS 命令列界面使用者指南》中的使用 aws 設定安裝 和快速組態。 https://docs.aws.amazon.com/cli/latest/userguide/cli-configure-quickstart.html#cli-configure-quickstart-config在 AWS CloudShell 中安裝的 AWS CLI 版本也可能是最新版本後面的幾個版本。若要更新它，請參閱《CloudShell AWS 使用者指南》中的將 CLI 安裝到您的主目錄。 AWS CloudShell
- 具有 create 和 describe Amazon EKS 叢集之許可的 IAM 主體 (使用者或角色)。如需詳細資訊，請參閱the section called “在 Outpost 上建立本機 Kubernetes 叢集”及the section called “所有叢集的清單或描述”。

建立本機 Amazon EKS 叢集後，系統會永久新增建立叢集的 IAM 主體。委託人會以管理員身分特別新增至 Kubernetes RBAC 授權表。此實體具有 system:masters 許可。叢集組態中看不到此實體的身分。因此，請務必注意建立叢集的實體，並確保您永遠不會將其刪除。一開始，只有建立伺服器的委託人可以使用 呼叫 Kubernetes API 伺服器kubectl。如果您使用 主控台建立叢集，請確定當您在叢集上執行kubectl命令時，相同的 IAM 登入資料位於 AWS SDK 登入資料鏈中。建立叢集之後，您可以授予其他 IAM 主體存取您的叢集。

建立 Amazon EKS 本機叢集

您可以使用此頁面所述的下列工具建立本機叢集：

- the section called “eksctl”
- the section called “AWS Management Console”

您也可以使用 AWS CLI、Amazon EKS API、 AWS SDKs、AWS CloudFormation 或 Terraform 在 Outposts 上建立叢集。

eksctl

使用 建立本機叢集 **eksctl**

1. 在您的裝置或 AWS CloudShell 上安裝版本 0.210.0或更新版本的eksctl命令列工具。如需有關安裝或更新 eksctl 的指示，請參閱 eksctl 文件中的安裝一節。
2. 將隨後的內容複製到您的裝置。取代以下值，然後執行修改後的命令來建立 outpost-control-plane.yaml 檔案：
   - 將 *region-code* 取代為您要在其中建立叢集的支援 AWS 區域。
   - 使用叢集的名稱取代 *my-cluster*。此名稱僅能使用英數字元 (區分大小寫) 和連字號。它必須以英數字元開頭，且長度不可超過 100 個字元。在您建立叢集 AWS 的區域和 AWS 帳戶中，名稱必須是唯一的。名稱在您要建立叢集 AWS 的區域和 AWS 帳戶中必須是唯一的。
   - 使用現有 VPC 和子網路的 ID 取代 *vpc-ExampleID1* 和 *subnet-ExampleID1*。VPC 和子網路必須符合在 AWS Outpost 上為 Amazon EKS 叢集建立 VPC 和子網路的要求。
   - 使用您的 Outpost 的 ID 取代 *uniqueid*。
   - 使用您 Outpost 上可用的執行個體類型取代 *m5.large*。選擇執行個體類型之前，請先參閱the section called “容量考量事項”。部署了三個控制平面執行個體。您無法變更此號碼。

```
cat >outpost-control-plane.yaml <<EOF
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: my-cluster
  region: region-code
  version: "1.33"

vpc:
  clusterEndpoints:
    privateAccess: true
  id: "vpc-vpc-ExampleID1"
  subnets:
    private:
      outpost-subnet-1:
        id: "subnet-subnet-ExampleID1"

outpost:
  controlPlaneOutpostARN: arn:aws: outposts:region-code:111122223333:outpost/op-
uniqueid
  controlPlaneInstanceType: m5.large
EOF
```

如需所有可用選項和預設值的完整清單，請參閱 eksctl 文件中的 AWS Outposts Support 和 Config 檔案結構描述。

3. 使用您在上一步驟中建立的組態檔案建立叢集。eksctl 會在 Outpost 上建立 VPC 和一個子網路，以便在其中部署叢集。

```
eksctl create cluster -f outpost-control-plane.yaml
```

叢集佈建需要幾分鐘的時間。建立叢集時，會出現幾行輸出。輸出的最後一行類似於下面的範例行。

```
[#]  EKS cluster "my-cluster" in "region-code" region is ready
```

> **Tip**
>
> 若要查看使用 eksctl 建立叢集時可指定的大部分選項，請使用 `eksctl create cluster --help` 命令。若要查看所有可用的選項，您可使用 config 檔案。如需詳細資訊，請參閱 eksctl 文件中的使用組態檔與組態檔結構描述。您可以在 GitHub 上找到組態檔範例。

eksctl 命令會自動為建立叢集的 IAM 主體（使用者或角色）建立存取項目，並將 IAM 主體管理員許可授予叢集上的 Kubernetes 物件。如果您不希望叢集建立者具有叢集上 Kubernetes 物件的管理員存取權，請將下列文字新增至先前的組態檔案：`bootstrapClusterCreatorAdminPermissions: false`（與 metadata、 vpc和 相同層級outpost)。如果您已新增 選項，則在建立叢集之後，您需要為至少一個 IAM 主體建立存取項目，否則 IAM 主體將無法存取叢集上的 Kubernetes 物件。

AWS Management Console

使用 建立叢集 AWS Management Console

1. 您需要符合 Amazon EKS 要求的現有 VPC 和子網路。如需詳細資訊，請參閱the section called “建立 VPC 和子網路”。
2. 如果您已有本機叢集 IAM 角色，或者您要使用 建立叢集eksctl，則可以略過此步驟。根據預設，eksctl 會為您建立角色。

   a. 執行下列命令以建立 IAM 信任政策 JSON 檔案。

```
cat >eks-local-cluster-role-trust-policy.json <<EOF
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "ec2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF
```

b. 建立 Amazon EKS 叢集 IAM 角色。若要建立 IAM 角色，必須為建立角色的 IAM 主體指派以下 `iam:CreateRole` 動作 (許可)。

```
aws iam create-role --role-name myAmazonEKSLocalClusterRole --assume-role-policy-
document file://"eks-local-cluster-role-trust-policy.json"
```

c. 將名為 AmazonEKSLocalOutpostClusterPolicy 的 Amazon EKS 受管政策連接至角色。若要將 IAM 政策連接至 IAM 主體，必須為連接政策的 IAM 實體指派以下 IAM 動作之一 (許可)：`iam:AttachUserPolicy` 或 `iam:AttachRolePolicy`。

```
aws iam attach-role-policy --policy-arn arn:aws: iam::aws:policy/
AmazonEKSLocalOutpostClusterPolicy --role-name myAmazonEKSLocalClusterRole
```

3. 開啟 Amazon EKS 主控台。
4. 在主控台畫面頂端，請確定您已選取支援 AWS 的區域。
5. 選取 Add cluster (新增叢集)，然後選取 Create (建立)。
6. 在 Configure cluster (設定叢集) 頁面上，輸入或選取下列欄位的值：
   - Kubernetes 控制平面位置 – 選擇 AWS Outposts。
   - Outpost ID：選擇要在其上建立控制平面的 Outpost 的 ID。
   - Instance type (執行個體類型)：選取執行個體類型。只會顯示 Outpost 中可用的執行個體類型。在下拉式清單中，每個執行個體類型都會說明為其建議的節點數量。選擇執行個體類型之前，請先參閱the section called “容量考量事項”。所有複本均使用相同的執行個體類型進行部署。建立叢集後，您無法變更執行個體類型。部署了三個控制平面執行個體。您無法變更此號碼。

- Name (名稱)：叢集的名稱。它在您的帳戶中必須是唯一的 AWS 。此名稱僅能使用英數字元 (區分大小寫) 和連字號。它必須以英數字元開頭，且長度不可超過 100 個字元。名稱在您要建立叢集 AWS 的區域和 AWS 帳戶中必須是唯一的。在您建立叢集 AWS 的區域和 AWS 帳戶中，名稱必須是唯一的。
- Kubernetes 版本 – 選擇您要用於叢集的 Kubernetes 版本。我們建議選取最新版本，除非您需要使用較早版本。
- 叢集服務角色 – 選擇您在上一個步驟中建立的 Amazon EKS 叢集 IAM 角色，以允許 Kubernetes 控制平面管理 AWS 資源。
- Kubernetes 叢集管理員存取權 – 如果您希望建立叢集的 IAM 主體 （角色或使用者） 具有叢集上 Kubernetes 物件的管理員存取權，請接受預設 （允許）。Amazon EKS 會為該 IAM 主體建立存取項目，並向該存取項目授予叢集管理員許可。如需存取項目的詳細資訊，請參閱 the section called “授予許可”。

  如果您希望與建立叢集的主體不同的 IAM 主體具有 Kubernetes 叢集物件的管理員存取權，請選擇不允許選項。建立叢集後，任何具有 IAM 許可來建立存取項目的 IAM 主體，都可以為需要存取 Kubernetes 叢集物件的任何 IAM 主體新增存取項目。如需所需 IAM 許可的詳細資訊，請參閱《服務授權參考》中的 Amazon Elastic Kubernetes Service 定義的動作一節。如果您選擇不允許選項且未建立任何存取項目，則 IAM 主體將無法存取叢集上的 Kubernetes 物件。
- Tags (標籤) – (選用) 將任何標籤新增到您的叢集。如需詳細資訊，請參閱the section called “標記您的 資源”。完成此頁面後，請選擇下一步。

7. 在 Specify networking (指定網路) 頁面上，選取下列欄位的值：
   - VPC：選擇現有的 VPC。針對您要建立的叢集、任何節點和其他 Kubernetes 資源，VPC 必須有足夠數目的 IP 地址。您的 VPC 必須符合 VPC 要求和考量事項中的要求。
   - Subnets (子網路)：根據預設，前一個欄位指定的 VPC 中的所有可用子網路會預先選取。您選擇的子網路必須符合子網路需求和考量中的要求。
   - 安全群組:(選用) 指定一或多個您希望 Amazon EKS 與其建立的網路介面相關聯的安全群組。Amazon EKS 會自動建立一個安全群組，以支援您的叢集和 VPC 之間的通訊。Amazon EKS 將此安全群組及您選擇的任何群組與其建立的網路介面相關聯。如需有關 Amazon EKS 建立的叢集安全群組的詳細資訊，請參閱 the section called “安全群組要求”。您可以修改 Amazon EKS 建立的叢集安全群組中的規則。如果您選擇新增自己的安全群組，則無法在叢集建立後變更您選擇的安全群組。若要使內部部署主機與叢集端點通訊，您必須允許來自叢集安全群組的傳入流量。對於沒有輸入和輸出網際網路連線的叢集 （也稱為私有叢集），您必須執行下列其中一項：
     - 新增與所需 VPC 端點關聯的安全群組。如需必要端點的詳細資訊，請參閱子網路存取 AWS 服務the section called “使用 介面 VPC 端點”中的 。

- 修改 Amazon EKS 建立的安全群組，以允許來自與 VPC 端點關聯之安全群組的流量。完成此頁面後，請選擇下一步。

8. 在設定可觀測性頁面上，您可以選擇性地選擇要開啟的指標和控制平面日誌記錄選項。根據預設，系統會關閉每個日誌類型。
   - 如需 Prometheus 指標選項的詳細資訊，請參閱 the section called “步驟 1：開啟 Prometheus 指標”。
   - 如需控制平面日誌記錄選項的詳細資訊，請參閱 the section called “控制平面日誌”。完成此頁面後，請選擇下一步。
9. 在 Review and create (檢閱並建立) 頁面上，檢閱您在先前頁面上輸入或選取的資訊。如需變更，請選擇 Edit (編輯)。當您滿意時，請選擇建立。在佈建叢集時，Status (狀態) 欄位顯示 CREATING (正在建立)。

   叢集佈建需要幾分鐘的時間。

檢視您的 Amazon EKS 本機叢集

1. 叢集建立之後，您便可以檢視已建立的 Amazon EC2 控制平面執行個體。

```
aws ec2 describe-instances --query 'Reservations[*].Instances[*].{Name:Tags[?
Key==`Name`]|[0].Value}' | grep my-cluster-control-plane
```

   範例輸出如下。

```
"Name": "my-cluster-control-plane-id1"
"Name": "my-cluster-control-plane-id2"
"Name": "my-cluster-control-plane-id3"
```

   每個執行個體均受到 `node-role.eks-local.amazonaws.com/control-plane` 污染，因此沒有在控制平面執行個體上排程工作負載。如需污點的詳細資訊，請參閱 Kubernetes 文件中的污點和容錯。Amazon EKS 會持續監控本機叢集的狀態。我們會執行自動管理動作，如安全性修補程式和修復運作狀態不良的執行個體。如果本機叢集與雲端中斷連線，我們會完成動作，以確保在重新連線時將叢集修復為健全狀態。

2. 如果您使用 `eksctl` 建立叢集，則可以略過此步驟。`eksctl` 會為您完成此步驟。透過向 `kubectl config` 檔案新增內容，使 `kubectl` 能夠與您的叢集通訊。如需如何建立和更新檔案的說明，請參閱 the section called “使用 kubectl 存取叢集”。

```
aws eks update-kubeconfig --region region-code --name my-cluster
```

範例輸出如下。

```
Added new context arn:aws: eks:region-code:111122223333:cluster/my-cluster to /home/
username/.kube/config
```

3. 若要連線至本機叢集的 Kubernetes API 伺服器，請存取子網路的本機閘道，或從 VPC 內連線。如需將 Outpost 機架連線至內部部署網路的詳細資訊，請參閱 AWS Outposts 使用者指南中的機架的本機閘道運作方式。如果您使用直接 VPC 路由，且 Outpost 子網路具有通往本機閘道的路由，則 Kubernetes 控制平面執行個體的私有 IP 地址會自動透過本機網路廣播。本機叢集的 Kubernetes API 伺服器端點託管於 Amazon Route 53 (Route 53)。API 服務端點可透過公有 DNS 伺服器將其解析為 Kubernetes API 伺服器的私有 IP 地址。

   本機叢集的 Kubernetes 控制平面執行個體使用靜態彈性網路介面進行設定，其固定私有 IP 地址不會在整個叢集生命週期中變更。在網路中斷連線期間，與 Kubernetes API 伺服器互動的機器可能無法連線至 Route 53。如果是這種情況，建議使用靜態私有 IP 地址來設定 `/etc/hosts` 以繼續操作。我們還建議您設定本機 DNS 伺服器，並將其連接至 Outpost。如需詳細資訊，請參閱 AWS Outposts 文件。執行下列命令，以確認已與您的叢集建立通訊。

   ```
   kubectl get svc
   ```

   範例輸出如下。

   ```
   NAME         TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
   kubernetes   ClusterIP   10.100.0.1   <none>        443/TCP   28h
   ```

4. （選用）當本機叢集與 AWS 雲端處於中斷連線狀態時，對本機叢集測試身分驗證。如需說明，請參閱the section called “準備中斷連線”。

內部資源

Amazon EKS 會在您的叢集上建立下列資源。這些資源供 Amazon EKS 內部使用。為了讓叢集正常運作，請勿編輯或修改這些資源。

- 下列鏡像 Pod：
  - `aws-iam-authenticator-`*`node-hostname`*

  - eks-certificates-controller-*node-hostname*
  - etcd-*node-hostname*
  - kube-apiserver-*node-hostname*
  - kube-controller-manager-*node-hostname*
  - kube-scheduler-*node-hostname*
- 下列自我管理的附加元件：
  - kube-system/coredns
  - kube-system/ kube-proxy （在您新增第一個節點之前不會建立）
  - kube-system/aws-node (在新增第一個節點之前不會建立)。本機叢集使用 Amazon VPC CNI 外掛程式進行叢集聯網。請勿變更控制平面執行個體 (名為 aws-node-controlplane-* 的 Pod) 的組態。外掛程式建立新網路介面時，您可以使用組態變數來變更預設值。如需詳細資訊，請參閱 GitHub 上的文件。
- 下列服務：
  - default/kubernetes
  - kube-system/kube-dns
- 名為 eks.system 的 PodSecurityPolicy
- 名為 eks:system:podsecuritypolicy 的 ClusterRole
- 名為 eks:system 的 ClusterRoleBinding
- 除了叢集安全群組之外，Amazon EKS 還會在您的 AWS 帳戶中建立名為 的安全群組eks-local-internal-do-not-use-or-edit-*cluster-name*-*uniqueid* 。此安全群組可讓流量在控制平面執行個體上執行的 Kubernetes 元件之間自由流動。

建議的後續步驟：

- 授予建立叢集的 IAM 主體在 中檢視 Kubernetes 資源所需的許可 AWS Management Console
- 授予 IAM 實體對叢集的存取權。如果您希望實體在 Amazon EKS 主控台中檢視 Kubernetes 資源，請將必要許可授予實體。
- 設定叢集的日誌
- 熟悉網路連線中斷期間發生的情況。
- 將節點新增至叢集

- 考慮為您的 etcd 設定備份計劃。Amazon EKS 不支援本機叢集etcd的 自動備份和還原。如需詳細資訊，請參閱 Kubernetes 文件中的備份 等化叢集。兩個主要選項是使用 etcdctl 自動拍攝快照或使用 Amazon EBS 儲存磁碟區備份。

了解 AWS Outposts 的 Kubernetes 和 Amazon EKS 平台版本

本機叢集平台版本代表 AWS Outposts 上 Amazon EKS 叢集的功能。這些版本包括在 Kubernetes 控制平面上執行的元件，已啟用哪些 Kubernetes API 伺服器旗標。它們也包含目前的 Kubernetes 修補程式版本。每個 Kubernetes 次要版本皆有一或多個相關的 平台版本。適用於不同 Kubernetes 次要版本的平台版本都是彼此獨立。雲端中本機叢集和 Amazon EKS 叢集的平台版本皆各自獨立。

當新的 Kubernetes 次要版本可供本機叢集使用時，例如 1.31，該 Kubernetes 次要版本的初始平台版本會從 開始eks-local-outposts.1。但是，Amazon EKS 會定期發佈新平台版本，以啟用新的 Kubernetes 控制平面設定，以及提供安全性問題修正。

當有新的本機叢集平台版本可供次要版本使用時：

- 平台版本編號將會遞增 (eks-local-outposts.n+1)。
- Amazon EKS 會自動將所有現有的本機叢集更新為其對應 Kubernetes 次要版本的最新平台版本。現有平台版本的自動更新將會逐步發佈。推展程序包含替換在 Outpost 上執行的受管 Kubernetes 控制平面執行個體，一次一個，直到全部 3 個執行個體都被新的執行個體取代為止。
- 如果存在服務中斷的風險，Kubernetes 控制平面執行個體替換程序將停止進行。Amazon EKS 只會嘗試在其他 2 個 Kubernetes 控制平面執行個體運作狀態良好，並以叢集節點的形式傳遞所有整備條件時，來取代執行個體。
- 平台版本推展通常需要不到 30 分鐘的時間才能完成。如果叢集長時間處於 UPDATING 狀態，請參閱 the section called “對叢集進行疑難排解”並向 AWS Support 尋求協助。除非 AWS Support 指示，否則請勿手動終止 Kubernetes 控制平面執行個體。
- Amazon EKS 可能會發佈具有對應之修補程式版本的新節點 AMI。所有修補程式版本都與單一 Kubernetes 次要版本的 Kubernetes 控制平面和節點 AMIs 相容。

新的平台版本不會帶來重大變更或導致服務中斷。

本機叢集一律使用指定 Kubernetes 版本的最新可用平台版本 (eks-local-outposts.n) 建立。

的目前及最新平台版本如下表所示。

若要接收此特定文件頁面所有來源檔案變更的通知，您可以使用 RSS 讀取器訂閱下列 URL：

```
https://github.com/awsdocs/amazon-eks-user-guide/commits/mainline/latest/ug/outposts/
eks-outposts-platform-versions.adoc.atom
```

Kubernetes 版本 **1.31**

所有1.31平台版本都已啟用下列許可控制
器：CertificateApproval、CertificateSigning、CertificateSubjectRestriction、Cluste
ValidatingAdmissionPolicy和 ValidatingAdmissionWebhook。

Kubernetes 版本	Amazon EKS 平台版本	版本備註	版本日期
1.31.6	eks-local-outposts.1	Outposts 上v1.31本機 Amazon EKS 叢集的 Kubernetes 版本初始版本。	2025 年 4 月 9 日

Kubernetes 版本 **1.30**

所有1.30平台版本都已啟用下列許可控制
器：CertificateApproval、CertificateSigning、CertificateSubjectRestriction、Cluste
ValidatingAdmissionPolicy和 ValidatingAdmissionWebhook。

Kubernetes 版本	Amazon EKS 平台版本	版本備註	版本日期
1.30.10	eks-local-outposts.3	具有安全性修正和增強功能的新平台版本。kube-proxy 已更新至 v1.30.10. AWS IAM Authenticator，已更新至 v0.6.29。Kubernetes 專用 Amazon VPC CNI 外掛程式已更新至	2025 年 3 月 27 日

Kubernetes 版本	Amazon EKS 平台版本	版本備註	版本日期
		v1.19.2。CoreDNS 已更新至 v1.11.4. AWS Cloud Controller Manager 已更新至 v1.30.8。Bottlerocket 版本已更新為 v1.34.0。	
1.30.7	eks-local-outposts.2	具有安全性修正和增強功能的新平台版本。kube-proxy 已更新至 v1.30.7. AWS IAM Authenticator 已更新至 v0.6.28。Kubernetes 專用 Amazon VPC CNI 外掛程式已更新至 v1.19.0。已更新 Bottlerocket 版本至 v1.29.0。	2025 年 1 月 10 日
1.30.5	eks-local-outposts.1	Outposts 上v1.30本機 Amazon EKS 叢集的 Kubernetes 版本初始版本。	2024 年 11 月 13 日

Kubernetes 版本 **1.29**

所有1.29平台版本都已啟用下列許可控制器：CertificateApproval、CertificateSigning、CertificateSubjectRestriction、ClusterValidatingAdmissionPolicy和 ValidatingAdmissionWebhook。

Kubernetes 版本	Amazon EKS 平台版本	版本備註	版本日期
v1.29.14	eks-local-outposts.6	具有安全性修正和增強功能的新平台版本。kube-proxy 已更新為 v1.29.14。Kubernetes 專用 Amazon VPC CNI 外掛程式已更新至 v1.19.2。CoreDNS 已更新至 v1.11.4. AWS Cloud Controller Manager 已更新至 v1.29.8。Bottlerocket 版本已更新為 v1.34.0。	2025 年 3 月 27 日
v1.29.11	eks-local-outposts.5	具有安全性修正和增強功能的新平台版本。kube-proxy 已更新為 v1.29.11。Kubernetes 專用 Amazon VPC CNI 外掛程式已更新至 v1.19.0。將 CoreDNS 映像更新為 v1.11.3。已更新 Bottlerocket 版本至 v1.29.0。	2025 年 1 月 10 日
1.29.9	eks-local-outposts.4	具有安全性修正和增強功能的新平台版本。kube-proxy 已更新至 v1.29.9. AWS IAM Authenticator 已	2024 年 11 月 8 日

Kubernetes 版本	Amazon EKS 平台版本	版本備註	版本日期
		更新至 v0.6.26。已更新 Bottlerocket 版本至 v1.26.0。	
1.29.6	eks-local-outposts.3	具有安全性修正和增強功能的新平台版本。已更新 Bottlerocket 版本至 v1.22.0。	2024 年 10 月 22 日
1.29.6	eks-local-outposts.2	具有安全性修正和增強功能的新平台版本。已更新 Bottlerocket 版本至 v1.21.0。	2024 年 8 月 27 日
1.29.6	eks-local-outposts.1	Outposts 上v1.29本機 Amazon EKS 叢集的 Kubernetes 版本初始版本。	2024 年 8 月 20 日

Kubernetes 版本 **1.28**

所有1.28平台版本都已啟用下列許可控制器：CertificateApproval、CertificateSigning、CertificateSubjectRestriction、Cluste
ValidatingAdmissionPolicy和 ValidatingAdmissionWebhook。

Kubernetes 版本	Amazon EKS 平台版本	版本備註	版本日期
1.28.15	eks-local-outposts.13	具有安全性修正和增強功能的新平台版本。kube-proxy 組v1.28.15建已更新。適用於 Kubernetes 的 Amazon VPC	2025 年 3 月 27 日

Kubernetes 版本	Amazon EKS 平台版本	版本備註	版本日期
		CNI 外掛程式已更新至 v1.19.0. AWS Cloud Controller Manager 已更新至 v1.28.11。	
1.28.15	eks-local-outposts.12	具有安全性修正和增強功能的新平台版本。kube-proxy 已更新為 v1.28.15。Kubernetes 專用 Amazon VPC CNI 外掛程式已更新至 v1.19.0。已更新 Bottlerocket 版本至 v1.29.0。	2025 年 1 月 10 日
1.28.14	eks-local-outposts.11	具有安全性修正和增強功能的新平台版本。kube-proxy 已更新至 v1.28.14. AWS IAM Authenticator 已更新至 v0.6.26。將 CoreDNS 映像更新為 v1.11.1。已更新 Bottlerocket 版本至 v1.26.0。	2024 年 11 月 8 日
1.28.10	eks-local-outposts.10	具有安全性修正和增強功能的新平台版本。已更新 Bottlerocket 版本至 v1.22.0。	2024 年 10 月 22 日

Kubernetes 版本	Amazon EKS 平台版本	版本備註	版本日期
1.28.10	eks-local-outposts.9	具有安全性修正和增強功能的新平台版本。已更新 Bottlerocket 版本至 v1.21.0。	2024 年 8 月 27 日
1.28.10	eks-local-outposts.8	具有安全性修正和增強功能的新平台版本。	2024 年 7 月 30 日
1.28.10	eks-local-outposts.6	具有安全性修正和增強功能的新平台版本。kube-proxy 已更新至 v1.28.10. AWS IAM Authenticator 已更新至 v0.6.20。已更新 Bottlerocket 版本至 v1.20.2。	2024 年 6 月 19 日
1.28.6	eks-local-outposts.5	將 Bottlerocket 版本更新為 v1.19.3，其中包含最新的錯誤修正，以支援 Outposts 中的本機開機。	2024 年 4 月 18 日
1.28.6	eks-local-outposts.4	具有安全性修正和增強功能的新平台版本。Outposts 中的還原支援或本機開機。降級 Bottlerocket 版本至 ，v1.15.1以實現相容性。	2024 年 4 月 2 日
1.28.6	eks-local-outposts.3	具有安全性修正和增強功能的新平台版本。	2024 年 3 月 22 日

Kubernetes 版本	Amazon EKS 平台版本	版本備註	版本日期
1.28.6	eks-local-outposts.2	具有安全性修正和增強功能的新平台版本。kube-proxy 已更新至 v1.28.6. AWS IAM Authenticator 已更新至 v0.6.17。v1.13.2 基於相容性原因，適用於 Kubernetes 的 Amazon VPC CNI 外掛程式已降級至 。已更新 Bottlerocket 版本至 v1.19.2。	2024 年 3 月 8 日
1.28.1	eks-local-outposts.1	Outposts 上v1.28本機 Amazon EKS 叢集的 Kubernetes 版本初始版本。	2023 年 10 月 4 日

Kubernetes 版本 **1.27**

所有1.27平台版本都已啟用下列許可控制器：CertificateApproval、CertificateSigning、CertificateSubjectRestriction、ClusteValidatingAdmissionPolicy和 ValidatingAdmissionWebhook。

Kubernetes 版本	Amazon EKS 平台版本	版本備註	版本日期
1.27.16	eks-local-outposts.13	具有安全性修正和增強功能的新平台版本。Kubernetes 專用 Amazon VPC CNI 外掛程式已更新至	2025 年 3 月 27 日

Kubernetes 版本	Amazon EKS 平台版本	版本備註	版本日期
		v1.19.2。Bottlerocket 版本已更新為 v1.34.0。	
1.27.16	eks-local-outposts.12	具有安全性修正和增強功能的新平台版本。Kubernetes 專用 Amazon VPC CNI 外掛程式已更新至 v1.19.0。已更新 Bottlerocket 版本至 v1.29.0。	2025 年 1 月 10 日
1.27.16	eks-local-outposts.11	具有安全性修正和增強功能的新平台版本。kube-proxy 已更新至 v1.27.16. AWS IAM Authenticator 已更新至 v0.6.26。將 CoreDNS 映像更新為 v1.11.1。已更新 Bottlerocket 版本至 v1.26.0。	2024 年 11 月 8 日
1.27.14	eks-local-outposts.10	具有安全性修正和增強功能的新平台版本。已更新 Bottlerocket 版本至 v1.22.0。	2024 年 10 月 22 日
1.27.14	eks-local-outposts.9	具有安全性修正和增強功能的新平台版本。已更新 Bottlerocket 版本至 v1.21.0。	2024 年 8 月 27 日

Kubernetes 版本	Amazon EKS 平台版本	版本備註	版本日期
1.27.14	eks-local-outposts.8	具有安全性修正和增強功能的新平台版本。	2024 年 7 月 30 日
1.27.14	eks-local-outposts.6	具有安全性修正和增強功能的新平台版本。kube-proxy 已更新至 v1.27.14. AWS IAM Authenticator 已更新至 v0.6.20。已更新 Bottlerocket 版本至 v1.20.2。	2024 年 6 月 19 日
1.27.10	eks-local-outposts.5	具有安全性修正和增強功能的新平台版本。	2024 年 4 月 2 日
1.27.10	eks-local-outposts.4	具有安全性修正和增強功能的新平台版本。kube-proxy 已更新至 v1.27.10. AWS IAM Authenticator 已更新至 v0.6.17。已更新 Bottlerocket 版本至 v1.19.2。	2024 年 3 月 22 日
1.27.3	eks-local-outposts.3	具有安全性修正和增強功能的新平台版本。kube-proxy 已更新為 v1.27.3。Kubernetes 專用 Amazon VPC CNI 外掛程式已更新至 v1.13.2。	2023 年 7 月 14 日

Kubernetes 版本	Amazon EKS 平台版本	版本備註	版本日期
1.27.1	eks-local-outposts.2	將 CoreDNS 映像更新為 v1.10.1。	2023 年 6 月 22 日
1.27.1	eks-local-outposts.1	Outposts 上1.27本機 Amazon EKS 叢集的 Kubernetes 版本初始版本。	2023 年 5 月 30 日

在 AWS Outpost 上為 Amazon EKS 叢集建立 VPC 和子網路

建立本機叢集時，您需要指定 VPC 與至少一個在 Outpost 上執行的私人子網路。本主題概述了本機叢集的 VPC 和子網路要求和考量事項。

VPC 要求和注意事項

建立本機叢集時，您指定的 VPC 必須符合下列要求和考量事項：

- 請確定 VPC 有足夠的 IP 地址，可用於本機叢集、任何節點，以及您要建立的其他 Kubernetes 資源。如果您想要使用的 VPC 沒有足夠的 IP 地址，請增加可用 IP 地址的數量。您可以透過將其他無類別域間路由 (CIDR) 區塊與 VPC 關聯來完成此操作。您可以在建立叢集的之前或之後，將私有 (RFC 1918) 和公有 (非 RFC 1918) CIDR 區塊與 VPC 關聯。叢集可能最多需要 5 小時才能辨識您與 VPC 關聯的 CIDR 區塊。
- VPC 無法具有指派的 IP 字首或 IPv6 CIDR 區塊。由於這些限制，在將更多 IP 地址指派給字首為 且不適用於 VPC 的 Amazon EKS 節點中涵蓋的資訊。 the section called “IPv6”
- VPC 已啟用 DNS 主機名稱和 DNS 解析。如果沒有這些功能，本機叢集無法建立，您需要啟用這些功能並重新建立叢集。如需詳細資訊，請參閱《Amazon VPC 使用者指南》中的 VPC 的 DNS 屬性。
- 若要透過區域網路存取本機叢集，VPC 必須與 Outpost 的本機閘道路由表關聯。如需詳細資訊，請參閱 AWS Outposts 使用者指南中的 VPC 關聯。

子網需求和注意事項

在建立叢集時，指定至少一個私有子網路。如果您指定多個子網路，Kubernetes 控制平面執行個體會平均分散到子網路。如果指定多個子網路，則子網路必須存在於相同的 Outpost 上。此外，子網路還

必須具有適當的路由和安全群組許可，才能互相通訊。在建立本機叢集時，您指定的子網路必須符合下列要求：

- 子網路都位於相同的邏輯 Outpost 上。
- 子網路總共有至少三個 Kubernetes 控制平面執行個體可用的 IP 地址。如果指定三個子網路，每個子網路必須至少有一個可用的 IP 地址。如果指定兩個子網路，每個子網路必須至少有兩個可用的 IP 地址。如果指定一個子網路，該子網路必須至少有三個可用的 IP 地址。
- 子網路具有通往 Outpost 機架本機閘道的路由，以透過本機網路存取 Kubernetes API 伺服器。如果子網路沒有通往 Outpost 機架本機閘道的路由，您必須從 VPC 內與 Kubernetes API 伺服器通訊。
- 子網必須使用 IP 地址型命名。Amazon EKS 不支援 Amazon EC2 資源型命名。

AWS 服務的子網路存取

Outposts 上的本機叢集私有子網路必須能夠與區域 AWS 服務通訊。為此，您可以使用 NAT 閘道進行網際網路對外存取，或者，如果您想要所有流量在 VPC 內保持私密，可使用介面 VPC 端點。

使用 NAT 閘道

Outposts 上的本機叢集私有子網路必須具有關聯的路由表，其路由至 Outpost 父可用區域中公有子網路中的 NAT 閘道。公有子網路必須擁有到網際網路閘道的路由。此 NAT 閘道會啟用傳出網際網路存取，並防止來自網際網路未經要求的傳入連線連接到 Outpost 上的執行個體。

使用 介面 VPC 端點

如果 Outposts 上的本機叢集私有子網路沒有傳出網際網路連線，或者如果您想要在 VPC 中保持所有流量的私密性，則必須在建立叢集之前，在區域子網路中建立下列界面 VPC 端點和閘道端點。

端點	端點類型
com.amazonaws. *region-code* .ssm	介面
com.amazonaws. *region-code* .ssmmessages	介面
com.amazonaws. *region-code* .ec2messages	介面
com.amazonaws. *region-code* .ec2	介面

端點	端點類型
com.amazonaws. *region-code* .secretsmanager	介面
com.amazonaws. *region-code* .logs	介面
com.amazonaws. *region-code* .sts	介面
com.amazonaws. *region-code* .ecr.api	介面
com.amazonaws. *region-code* .ecr.dkr	介面
com.amazonaws. *region-code* .s3	閘道

端點必須符合下列需求：

- 在位於 Outpost 父可用區域中的私有子網路中建立
- 已啟用私有 DNS 名稱
- 擁有連接的安全群組，該群組允許來自私有 Outpost 子網路之 CIDR 範圍的輸入 HTTPS 流量。

建立端點會產生費用。如需詳細資訊，請參閱 AWS PrivateLink 定價。如果您的 Pod 需要存取其他 AWS 服務，則需要建立其他端點。如需端點的完整清單，請參閱AWS 與 AWS PrivateLink 整合的 服務。

建立 VPC

您可以使用下列其中一個 AWS CloudFormation 範本來建立符合先前需求的 VPC：

- 範本 1 – 此範本會建立 VPC，在 Outpost 上有一個私有子網路，並在 AWS 區域中建立一個公有子網路。私有子網路透過位於 AWS 區域的公有子網路中的 NAT Gateway 路由至網際網路。此範本可用於在具有輸出網際網路存取權的子網路中建立本機叢集。
- 範本 2 – 此範本會在 Outpost 上建立具有一個私有子網路的 VPC，以及在子網路中建立本機叢集所需的最小一組 VPC 端點，該子網路沒有傳入或傳出網際網路存取（也稱為私有子網路）。

在 AWS Outposts 上準備本機 Amazon EKS 叢集，以用於網路連線中斷

如果您的本機網路與 AWS 雲端失去連線，您可以繼續使用 Outpost 上的本機 Amazon EKS 叢集。本主題包含如何準備本機叢集以應對網路連線中斷和相關考量事項。

- 本機叢集可在臨時、計劃外的網路中斷期間實現穩定性和持續操作。 AWS Outposts 仍然是完全連線的方案，可做為資料中心中 AWS 雲端的延伸。如果您的 Outpost 和 AWS Cloud 之間發生網路連線中斷，建議您嘗試還原連線。如需說明，請參閱《AWS Outposts 使用者指南》中的 Outposts 機架網路故障診斷檢查清單。 AWS 如需有關如何針對本機叢集問題進行疑難排解的詳細資訊，請參閱 the section called “對叢集進行疑難排解”。
- Outpost 會發射 `ConnectedStatus` 指標，您可用此指標來監控 Outpost 的連線狀態。如需詳細資訊，請參閱《 Outposts 使用者指南》中的 Outposts 指標。 AWS
- 本機叢集使用 IAM 做為預設身分驗證機制，使用AWS 適用於 Kubernetes 的 Identity and Access Management 驗證器。在網路連線中斷期間，無法使用 IAM。因此，本機叢集會使用 `x.509` 憑證支援替代身分驗證機制，您可使用這些憑證在網路連線中斷期間連線至叢集。如需有關如何取得和使用您叢集 `x.509` 憑證的資訊，請參閱 the section called “在網路連線中斷期間針對本機叢集進行身分驗證”。
- 如果您在網路中斷連線期間無法存取 Route 53，請考慮在內部部署環境中使用本機 DNS 伺服器。Kubernetes 控制平面執行個體使用靜態 IP 地址。您可使用端點主機名稱和 IP 地址來設定用來連線至叢集的主機，以此作為使用本機 DNS 伺服器的替代方法。如需詳細資訊，請參閱 AWS Outposts 使用者指南中的 DNS。
- 如果您預期應用程式流量會在網路連線中斷期間增加，則可在連線至雲端時在叢集中佈建備用運算容量。Amazon EC2 執行個體包含在 AWS Outpost 的價格中。因此，執行備用執行個體不會影響您的 AWS 用量成本。
- 在網路連線中斷期間，若要啟用工作負載的建立、更新和擴展操作，則必須可透過區域網路存取應用程式的容器映像，且叢集須有足夠容量。本機叢集不會為您託管容器登錄檔。如果 Pod 先前已在這些節點上執行，則容器映像會快取在節點上。如果您通常會從雲端中的 Amazon ECR 提取應用程式的容器映像，則請考慮執行本機快取或登錄檔。如果您在網路連線中斷期間需要建立、更新和擴展工作負載資源的操作，則本機快取或登錄會很有幫助。
- 本機叢集會使用 Amazon EBS 作為持久性磁碟區的預設儲存類別，並使用 Amazon EBS CSI 驅動程式管理 Amazon EBS 持久性磁碟區的生命週期。在網路連線中斷期間，Amazon EBS 支援的 Pod 無法建立、更新或擴展。這是因為這些操作需要呼叫雲端中的 Amazon EBS API。如果您要在本機叢集上部署具狀態工作負載，並在網路連線中斷期間需要建立、更新或擴展操作，請考慮使用替代儲存機制。
- 如果 AWS Outposts 無法存取相關的 AWS 區域內 APIs （例如 Amazon EBS 或 Amazon S3 APIs)，則無法建立或刪除 Amazon EBS 快照。

- 將 ALB （輸入）與 AWS Certificate Manager (ACM) 整合時，會將憑證推送並儲存在 AWS Outposts ALB 運算執行個體的記憶體中。如果 AWS 區域中斷連線，目前的 TLS 終止將繼續運作。在此內容中變動操作將會失敗 (例如新的輸入定義、新的 ACM 型憑證 API 操作、ALB 運算規模或憑證輪換)。如需詳細資訊，請參閱 AWS Certificate Manager 使用者指南中的對受管憑證續約進行故障診斷。
- 在網路中斷連線期間，Amazon EKS 控制平面日誌會在 Kubernetes 控制平面執行個體上本機快取。重新連線時，日誌會傳送至父 AWS 區域中的 CloudWatch Logs。您可以使用 Prometheus、Grafana 或 Amazon EKS 合作夥伴解決方案，使用 Kubernetes API 伺服器的指標端點，或使用 Fluent Bit 來監控本機叢集。
- 如果您將 Outposts 上的 AWS Load Balancer控制器用於應用程式流量，則 AWS Load Balancer控制器前端的現有 Pod 會在網路中斷期間繼續接收流量。在網路中斷連線期間建立的新 Pod 不會接收流量，直到 Outpost 重新連線至 AWS 雲端為止。連線至 AWS 雲端時，請考慮為您的應用程式設定複本計數，以因應網路中斷連線期間的擴展需求。
- 適用於 Kubernetes 的 Amazon VPC CNI 外掛程式預設為次要 IP 模式。其設定為 `WARM_ENI_TARGET=1`，允許外掛程式保持可用 IP 地址的「全彈性網路界面」。請依據中斷連線狀態期間的擴展需求來考慮變更 `WARM_ENI_TARGET`、`WARM_IP_TARGET` 和 `MINIMUM_IP_TARGET` 值。如需詳細資訊，請參閱 GitHub 上外掛程式的讀我檔案。如需每個執行個體類型支援的 Pod 數量上限清單，請參閱 GitHub 上的 eni-max-pods.txt 檔案。

在網路連線中斷期間針對本機叢集進行身分驗證

AWS 網路連線中斷期間無法使用 Identity and Access Management (IAM)。中斷連線時，您無法使用 IAM 登入資料向本機叢集進行身分驗證。但是，您可以在中斷連線時使用 x509 憑證，透過區域網路連接至您的叢集。您需要下載並存放客戶端 X509 憑證，以在中斷連線期間使用。在本主題中，您將了解如何建立和使用憑證，在叢集處於中斷連線狀態時進行身分驗證。

1. 建立憑證簽署請求。

   a. 產生憑證簽署請求。

   ```
   openssl req -new -newkey rsa:4096 -nodes -days 365 \
       -keyout admin.key -out admin.csr -subj "/CN=admin"
   ```

   b. 在 Kubernetes 中建立憑證簽署請求。

   ```
   BASE64_CSR=$(cat admin.csr | base64 -w 0)
   cat << EOF > admin-csr.yaml
   apiVersion: certificates.k8s.io/v1
   ```

```
kind: CertificateSigningRequest
metadata:
  name: admin-csr
spec:
  signerName: kubernetes.io/kube-apiserver-client
  request: ${BASE64_CSR}
  usages:
  - client auth
EOF
```

2. 使用 kubectl 建立憑證簽署請求。

```
kubectl create -f admin-csr.yaml
```

3. 檢查憑證簽署請求的狀態。

```
kubectl get csr admin-csr
```

範例輸出如下。

```
NAME       AGE   REQUESTOR                      CONDITION
admin-csr  11m   kubernetes-admin               Pending
```

Kubernetes 已建立憑證簽署請求。

4. 核准憑證簽署請求。

```
kubectl certificate approve admin-csr
```

5. 重新檢查核准的憑證簽署請求狀態。

```
kubectl get csr admin-csr
```

範例輸出如下。

```
NAME       AGE   REQUESTOR                    CONDITION
admin-csr  11m   kubernetes-admin             Approved
```

6. 擷取並驗證憑證。

a. 擷取憑證。

```
kubectl get csr admin-csr -o jsonpath='{.status.certificate}' | base64 --decode >
 admin.crt
```

b. 驗證憑證。

```
cat admin.crt
```

7. 建立與 admin 使用者綁定的叢集角色。

```
kubectl create clusterrolebinding admin --clusterrole=cluster-admin \
    --user=admin --group=system:masters
```

8. 產生中斷連線狀態的使用者範圍 kubeconfig。

您可以使用下載的 admin 憑證來產生 kubeconfig 檔案。取代下列命令中的 *my-cluster* 和 *apiserver-endpoint*。

```
aws eks describe-cluster --name my-cluster \
    --query "cluster.certificateAuthority" \
    --output text | base64 --decode > ca.crt
```

```
kubectl config --kubeconfig admin.kubeconfig set-cluster my-cluster \
    --certificate-authority=ca.crt --server apiserver-endpoint --embed-certs
```

```
kubectl config --kubeconfig admin.kubeconfig set-credentials admin \
    --client-certificate=admin.crt --client-key=admin.key --embed-certs
```

```
kubectl config --kubeconfig admin.kubeconfig set-context admin@my-cluster \
    --cluster my-cluster --user admin
```

```
kubectl config --kubeconfig admin.kubeconfig use-context admin@my-cluster
```

9. 檢視您的 kubeconfig 檔案。

```
kubectl get nodes --kubeconfig admin.kubeconfig
```

10如果您在 Outpost 上已有生產中的服務，請跳過此步驟。如果 Amazon EKS 是在您的 Outpost 上執行的唯一服務，且 Outpost 目前不在生產環境中，您可以模擬網路中斷連線。使用本機叢集進入生產環境之前，請模擬中斷連線，以確保您可以在叢集處於中斷連線狀態時存取叢集。

a. 在將 Outpost 連接到 AWS 區域的聯網裝置上套用防火牆規則。這會中斷 Outpost 連結的服務。您無法建立新的執行個體。目前執行的執行個體失去與 AWS 區域和網際網路的連線。

b. 您可在中斷連線時使用 x509 憑證，測試連結至您本機叢集的連線。請確保將您的 kubeconfig 變更為您在上一步中建立的 admin.kubeconfig。使用您本機叢集的名稱取代 *my-cluster*。

```
kubectl config use-context admin@my-cluster --kubeconfig admin.kubeconfig
```

如果您在本機叢集處於中斷連線狀態時發現任何問題，建議您開立支援票證。

根據容量考量，為 AWS Outposts 上的 Amazon EKS 叢集選取執行個體類型和置放群組

本主題提供使用置放群組選取 Kubernetes 控制平面執行個體類型和 （選用） 的指引，以滿足 Outpost 上本機 Amazon EKS 叢集的高可用性需求。

在您選取要用於 Outposts 上本機叢集 Kubernetes 控制平面的執行個體類型 （例如 m5c5、 或 r5) 之前，請確認 Outpost 組態上可用的執行個體類型。識別可用的執行個體類型之後，根據工作負載所需的節點數量，來選取執行個體大小 (例如 large、xlarge 或 2xlarge)。下表提供有關選擇執行個體大小的建議。

> **Note**
>
> 執行個體大小必須在您的 Outposts 上設定好插槽。確保在本機叢集的生命週期內，您有足夠的容量可容納 Outposts 上可用大小的三個執行個體。如需可用 Amazon EC2 執行個體類型的清單，請參閱 AWS Outposts 機架功能中的運算和儲存區段。

節點數量。	Kubernetes 控制平面執行個體大小
1–20	large
21–100	xlarge
101–250	2xlarge

節點數量。	Kubernetes 控制平面執行個體大小
251–500	`4xlarge`

Kubernetes 控制平面的儲存體需要每個本機叢集 246 GB 的 Amazon EBS 儲存體，才能滿足 所需的 IOPS`etcd`。建立本機叢集時，系統會自動為您佈建 Amazon EBS 磁碟區。

控制平面置放

當您未指定具有 `OutpostConfig.ControlPlanePlacement.GroupName` 屬性的置放群組時，為 Kubernetes 控制平面佈建的 Amazon EC2 執行個體不會在 Outpost 上可用的基礎容量中接收任何特定的硬體置放強制執行。

您可以使用置放群組來滿足 Outpost 上本機 Amazon EKS 叢集的高可用性需求。透過在叢集建立期間指定置放群組，您可以影響 Kubernetes 控制平面執行個體的置放。執行個體分布於獨立的基礎硬體 (機架或主機)，這樣可將相關執行個體對硬體故障事件的影響降至最低。

您能夠設定的分布類型取決於部署中所擁有的 Outpost 機架數量。

- 在單一邏輯 Outpost 中具有一或兩個實體機架的部署 – 您必須至少擁有三個主機，這些主機設定為您為 Kubernetes 控制平面執行個體選擇的執行個體類型。使用主機層級分散的分散置放群組可確保所有 Kubernetes 控制平面執行個體在 Outpost 部署中可用基礎機架內的不同主機上執行。
- 在單一邏輯 Outpost 中具有三個或更多實體機架的部署 – 您必須至少設定三個主機，其為針對 Kubernetes 控制平面執行個體選擇的執行個體類型。使用機架層級分散的分散置放群組可確保 Outpost 部署中所有 Kubernetes 控制平面執行個體在不同的機架上執行。或者，如前一個選項中所述，您也可以使用主機層級分布置放群組。

您負責建立所需的置放群組。您可以在呼叫 `CreateCluster` API 時指定置放群組。如需置放群組及其建立方式的詳細資訊，請參閱《Amazon EC2 使用者指南》中的置放群組。

- 指定放置群組後，Outpost 上必須有可用的開槽容量，才能成功建立本機 Amazon EKS 叢集。容量會根據您使用的是主機還是機架分布類型而有所不同。如果容量不足，叢集會保持 `Creating` 狀態。您可以檢查 DescribeCluster API 回應的運作`Insufficient Capacity Error`狀態欄位。您必須釋放容量才能進行建立程序。
- 在 Amazon EKS 本機叢集平台和版本更新期間，叢集中的 Kubernetes 控制平面執行個體會使用滾動更新策略取代為新執行個體。在此取代過程中，每個控制平面執行個體皆會終止，從而釋放各自的插槽。新更新的執行個體已佈建在其位置中。更新後的執行個體可能會被置放於已釋放的插槽中。若

插槽已被另一個不相關的執行個體使用，且沒有剩餘的容量符合所需的分布拓撲需求，則叢集會維持在 Updating 狀態。您可以在 DescribeCluster API 回應Insufficient Capacity Error的運作狀態欄位中看到各自的 。您必須釋放容量，才能進行更新程序並重新建立先前的高可用性層級。

- 每個 AWS 區域中每個帳戶最多可以建立 500 個置放群組。如需詳細資訊，請參閱《Amazon EC2 使用者指南》中的一般規則和限制。

Outposts 上的本機 Amazon EKS AWS 叢集疑難排解

本主題涵蓋一些使用本機叢集時可能遇到的常見錯誤與排除這些錯誤的方法。本機叢集類似於雲端中的 Amazon EKS 叢集，但 Amazon EKS 管理它們的方式有一些差異。

> **Important**
>
> 除非 AWS Support 明確指示，否則請勿終止在 Outpost 上執行的任何受管 EKS 本機叢集Kubernetes控制平面執行個體。終止這些執行個體會對本機叢集服務可用性造成風險，包括遺失本機叢集，以防同時終止多個執行個體。EC2 執行個體主控台eks-local:controlplane-name上的 標籤可識別 EKS 本機叢集Kubernetes控制平面執行個體。

API 行為

本機叢集透過 Amazon EKS API 建立，但會以非同步方式執行。這表示本機叢集對 Amazon EKS API 的請求會立即傳回。但是，這些請求可能會成功、因輸入驗證錯誤而快速檢錯，或失敗並附上描述性驗證錯誤。此行為類似於 Kubernetes API。

本機叢集不會轉換為 FAILED 狀態。Amazon EKS 會嘗試以連續的方式使叢集狀態與使用者請求的所需狀態一致。因此，本機叢集可能會長期保持在 CREATING 狀態，直到基礎問題解決為止。

描述叢集運作狀態欄位

您可以使用 describe-cluster Amazon EKS AWS CLI 命令來發現本機叢集問題。本機叢集問題會由describe-cluster命令回應的 cluster.health 欄位顯示。此欄位內含的訊息包括錯誤代碼、描述性訊息和相關資源 ID。此資訊僅可透過 Amazon EKS API 和 AWS CLI 取得。在以下範例中，使用您本機叢集的名稱取代 *my-cluster*。

```
aws eks describe-cluster --name my-cluster --query 'cluster.health'
```

範例輸出如下。

```
{
    "issues": [
        {
            "code": "ConfigurationConflict",
            "message": "The instance type 'm5.large' is not supported in Outpost 'my-
outpost-arn'.",
            "resourceIds": [
                "my-cluster-arn"
            ]
        }
    ]
}
```

如果問題無法解決，您可能需要刪除本機叢集並建立新叢集。例如，嘗試使用 Outpost 上無法使用的執行個體類型來佈建叢集。下表包含與運作狀態相關的常見錯誤。

錯誤情況	代碼	訊息	ResourceIds
找不到提供的子網路。	ResourceNotFound	The subnet ID *subnet-id* does not exist	所有提供的子網路 ID
提供的子網路不屬於相同的 VPC。	ConfigurationConflict	Subnets specified must belong to the same VPC	所有提供的子網路 ID
有些提供的子網路不屬於指定的 Outpost。	ConfigurationConflict	Subnet *subnet-id* expected to be in *outpost-arn*, but is in *other-outpost-arn*	有問題的子網路 ID
有些提供的子網路不屬於任何 Outpost。	ConfigurationConflict	Subnet *subnet-id* is not part of any Outpost	有問題的子網路 ID
有些提供的子網路沒有足夠的可用地址，	ResourceLimitExceeded	The specified subnet does	有問題的子網路 ID

錯誤情況	代碼	訊息	ResourceIds
無法為控制平面執行個體建立彈性網路介面。		not have enough free addresses to satisfy the request.	
Outpost 不支援指定的控制平面執行個體類型。	ConfigurationConflict	The instance type *type* is not supported in Outpost *outpost-arn*	叢集 ARN
您已終止控制平面 Amazon EC2 執行個體，或 run-instance 已成功，但觀察到的狀態變更為 Terminated 。您的 Outpost 重新連線後一段時間可能會發生此情況，且 Amazon EBS 內部錯誤會導致 Amazon EC2 內部工作流程失敗。	InternalFailure	EC2 instance state "Terminated" is unexpected	叢集 ARN
您在 Outpost 上的容量不足。如果 Outpost 與 AWS 區域中斷連線，則在建立叢集時也會發生這種情況。	ResourceLimitExceeded	There is not enough capacity on the Outpost to launch or start the instance.	叢集 ARN
您的帳戶已超過您的安全群組配額。	ResourceLimitExceeded	Amazon EC2 API 傳回的錯誤訊息	目標 VPC ID
您的帳戶已超過您的彈性網路介面配額。	ResourceLimitExceeded	Amazon EC2 API 傳回的錯誤訊息	目標子網路 ID

錯誤情況	代碼	訊息	ResourceIds
無法透過 AWS Systems Manager 連線控制平面執行個體。如需解決方案，請參閱the section called “無法透過 AWS Systems Manager 存取控制平面執行個體”。	`ClusterUnreachable`	Amazon EKS 控制平面執行個體無法透過 SSM 連線。請確認您的 SSM 和網路組態，並參考 Outpost 上的 EKS 疑難排解文件。	Amazon EC2 執行個體 ID
取得受管安全群組或彈性網路介面的詳細資訊時發生錯誤。	根據 Amazon EC2 用戶端錯誤代碼。	Amazon EC2 API 傳回的錯誤訊息	所有受管的安全群組 ID
授權或撤銷安全群組輸入規則時發生錯誤。這適用於叢集和控制平面安全群組。	根據 Amazon EC2 用戶端錯誤代碼。	Amazon EC2 API 傳回的錯誤訊息	有問題的安全群組 ID
刪除控制平面執行個體的彈性網路介面時發生錯誤。	根據 Amazon EC2 用戶端錯誤代碼。	Amazon EC2 API 傳回的錯誤訊息	有問題的彈性網路介面 ID

下表列出來自其他 AWS 服務的錯誤，這些錯誤會顯示在`describe-cluster`回應的運作狀態欄位中。

Amazon EC2 錯誤代碼	叢集運作狀態問題代碼	描述
`AuthFailure`	`AccessDenied`	基於各種原因，可能會發生此錯誤。最常見的原因是您意外移除了服務用於從控制平面縮小服務連結角色政策範圍的標籤。如果發生這種情況，

Amazon EC2 錯誤代碼	叢集運作狀態問題代碼	描述
		Amazon EKS 將無法再管理和監控這些 AWS 資源。
UnauthorizedOperation	AccessDenied	基於各種原因，可能會發生此錯誤。最常見的原因是您意外移除了服務用於從控制平面縮小服務連結角色政策範圍的標籤。如果發生這種情況，Amazon EKS 將無法再管理和監控這些 AWS 資源。
InvalidSubnetID.NotFound	ResourceNotFound	當找不到安全群組的傳入規則的子網路 ID 時，就會發生此錯誤。
InvalidPermission.NotFound	ResourceNotFound	當安全群組的輸入規則許可不正確時，就會發生此錯誤。
InvalidGroup.NotFound	ResourceNotFound	當找不到安全群組的輸入規則群組時，就會發生此錯誤。
InvalidNetworkInterfaceID.NotFound	ResourceNotFound	當找不到安全群組的輸入規則的網路介面 ID 時，就會發生此錯誤。
InsufficientFreeAddressesInSubnet	ResourceLimitExceeded	超過子網路資源配額時，會發生此錯誤。
InsufficientCapacityOnOutpost	ResourceLimitExceeded	超過 Outpost 容量配額時，會發生此錯誤。
NetworkInterfaceLimitExceeded	ResourceLimitExceeded	超過彈性網路介面配額時，會發生此錯誤。
SecurityGroupLimitExceeded	ResourceLimitExceeded	超過安全群組配額時，會發生此錯誤。

Amazon EC2 錯誤代碼	叢集運作狀態問題代碼	描述
VcpuLimitExceeded	ResourceLimitExceeded	在新帳戶中建立 Amazon EC2 執行個體時，便會觀察到這種狀況。此錯誤可能類似以下內容："You have requested more vCPU capacity than your current vCPU limit of 32 allows for the instance bucket that the specified instance type belongs to. Please visit http://aws.amazon.com/contact-us/ec2-request to request an adjustment to this limit."
InvalidParameterValue	ConfigurationConflict	如果 Outpost 上不支援指定的執行個體類型，Amazon EC2 會傳回此錯誤代碼。
所有其他失敗	InternalFailure	無

無法建立或修改叢集

本機叢集需要不同於雲端中託管的 Amazon EKS 叢集的許可和政策。當叢集無法建立並產生InvalidPermissions錯誤時，請再次檢查您使用的叢集角色是否已連接 AmazonEKSLocalOutpostClusterPolicy 受管政策。所有其他 API 呼叫需要與雲端中 Amazon EKS 叢集相同的許可集。

叢集卡在 **CREATING** 狀態

建立本機叢集所需的時間量因數個因素而異。這些因素包括您的網路組態、Outpost 組態和叢集的組態。一般而言，本機叢集會在 15-20 分鐘內建立並變更為 ACTIVE 狀態。如果本機叢集仍處於 CREATING 狀態，則您可以在 cluster.health 輸出欄位中呼叫 describe-cluster 以取得有關原因的資訊。

最常見的問題如下：

- 您的叢集無法從 Systems Manager 所在的 AWS 區域連線至控制平面執行個體。您可以從區域內的堡壘主機呼叫 `aws ssm start-session --target instance-id` 進行驗證。如果該命令無法運作，請檢查 Systems Manager 是否正在控制平面執行個體上執行。或者，另一個解決方法是刪除叢集，然後重新建立叢集。
- 由於 EBS 磁碟區的 KMS 金鑰許可，控制平面執行個體無法建立 。使用加密 EBS 磁碟區的使用者受管 KMS 金鑰，如果無法存取金鑰，控制平面執行個體就會終止。如果執行個體終止，請切換到 AWS 受管 KMS 金鑰，或確保您的使用者受管金鑰政策授予叢集角色必要的許可。
- Systems Manager 控制平面執行個體可能無法存取網際網路。檢查您在建立叢集時提供的子網路是否具有 NAT 閘道和具有網際網路閘道的 VPC。使用 VPC Reachability Analyzer 確認控制平面執行個體是否可連線至網際網路閘道。如需詳細資訊，請參閱 Getting started with VPC Reachability Analyzer (VPC Reachability Analyzer 入門)。
- 您提供的角色 ARN 缺少政策。檢查 AWS 受管政策：AmazonEKSLocalOutpostClusterPolicy 是否已從角色中移除。如果 An AWS CloudFormation 堆疊設定錯誤，也可能發生這種情況。
- 所有提供的子網路必須與相同的 Outpost 關聯，並且必須互相連線。如果在建立叢集時指定多個子網路，Amazon EKS 會嘗試將控制平面執行個體散佈到多個子網路。
- Amazon EKS 受管的安全群組將套用至彈性網路介面。但是，其他組態元素 (例如 NACL 防火牆規則) 可能會與彈性網路介面的規則衝突。

VPC 和子網路 DNS 組態設定錯誤或遺失

檢閱在 AWS Outpost 上為 Amazon EKS 叢集建立 VPC 和子網路。

叢集卡在 **`UPDATING`** 狀態

Amazon EKS 會自動將所有現有的本機叢集更新為其對應 Kubernetes 次要版本的最新平台版本。如需平台版本的詳細資訊，請參閱 the section called “EKS 平台版本”。

在自動平台版本推展期間，叢集狀態會變更為 `UPDATING`。更新程序包含將所有 Kubernetes 控制平面執行個體取代為新的執行個體，其中包含針對個別 Kubernetes 次要版本發行的最新安全路徑和錯誤修正。一般而言，本機叢集平台更新程序會在 30 分鐘內完成，且叢集會變更回 `ACTIVE` 狀態。如果本機叢集長時間保持 `UPDATING` 狀態，您可以呼叫 `describe-cluster` 在`cluster.health`輸出欄位中檢查原因的相關資訊。

Amazon EKS 可確保 3 個 Kubernetes 控制平面執行個體中至少有 2 個運作狀態良好且運作中的叢集節點，以維持本機叢集可用性並防止服務中斷。如果本機叢集停滯在 `UPDATING` 狀態，通常是因為有

一些基礎設施或組態問題，導致在程序繼續的情況下，無法保證兩個執行個體的最低可用性。因此，更新程序會停止進行以保護本機叢集服務中斷。

請務必對卡在 UPDATING 狀態的本機叢集進行疑難排解，並解決根本原因，以便更新程序可以使用 3 個 ACTIVE Kubernetes 控制平面執行個體的高可用性完成本機叢集並將其還原回 。

除非 AWS Support 明確指示，否則請勿終止 Outposts 上的任何受管 EKS 本機叢集Kubernetes執行個體。這對卡在 UPDATING 狀態的本機叢集特別重要，因為另一個控制平面節點可能無法完全正常運作，且終止錯誤的執行個體可能會導致服務中斷，並造成本機叢集資料遺失的風險。

最常見的問題如下：

- 一或多個控制平面執行個體由於自本機叢集首次建立以來的網路組態變更而無法連線至 System Manager。您可以從區域內的堡壘主機呼叫 `aws ssm start-session --target instance-id` 進行驗證。如果該命令無法運作，請檢查 Systems Manager 是否正在控制平面執行個體上執行。
- 由於 EBS 磁碟區的 KMS 金鑰許可，無法建立新的控制平面執行個體。使用加密 EBS 磁碟區的使用者受管 KMS 金鑰，如果無法存取金鑰，控制平面執行個體就會終止。如果執行個體終止，請切換到 AWS 受管 KMS 金鑰，或確保您的使用者受管金鑰政策授予叢集角色必要的許可。
- Systems Manager 控制平面執行個體可能遺失網際網路存取。檢查您建立叢集時提供的子網路是否具有 NAT 閘道，以及具有網際網路閘道的 VPC。使用 VPC Reachability Analyzer 確認控制平面執行個體是否可連線至網際網路閘道。如需詳細資訊，請參閱 Getting started with VPC Reachability Analyzer (VPC Reachability Analyzer 入門)。如果您的私有網路沒有傳出網際網路連線，請確保叢集的區域子網路中仍存在所有必要的 VPC 端點和閘道端點 （請參閱 the section called “AWS 服務的子網路存取”)。
- 您提供的角色 ARN 缺少政策。檢查 AWS 受管政策：AmazonEKSLocalOutpostClusterPolicy 是否未從角色中移除。
- 其中一個新的 Kubernetes 控制平面執行個體可能遇到非預期的引導失敗。請在此特殊情況下向AWS支援中心提交票證，以取得疑難排解和收集日誌的進一步指引。

無法將節點加入叢集

- AMI 問題：
  - 您正在使用不支援的 AMI。您必須針對具有最佳化 Amazon Linux AMIs Amazon EKS 最佳化 Amazon Linux 的建立節點使用 v20220620 或更新版本。
  - 如果您使用 an AWS CloudFormation 範本來建立節點，請確定它不是使用不支援的 AMI。

- 遺失 AWS IAM Authenticator `ConfigMap` – 如果遺失，您必須建立它。如需詳細資訊，請參閱 the section called “將 aws-authConfigMap 套用至您的叢集”。
- 使用錯誤的安全群組 – 確保將 `eks-cluster-sg-`*`cluster-name`*`-`*`uniqueid`* 用於您工作節點的安全群組。 AWS CloudFormation 會變更選取的安全群組，以允許每次使用堆疊時有新的安全群組。
- 依照未預期的私人連結 VPC 步驟：傳遞了錯誤的 CA 資料 (`--b64-cluster-ca`) 或 API 端點 (`--apiserver-endpoint`)。

收集日誌

當 Outpost 與其相關聯的 AWS 區域中斷連線時，Kubernetes 叢集可能會繼續正常運作。不過，如果叢集無法正常運作，請遵循在 AWS Outposts 上準備本機 Amazon EKS 叢集以進行網路中斷連線中的疑難排解步驟。如果您遇到其他問題，請聯絡 AWS Support。 AWS Support 可以引導您下載和執行日誌收集工具。如此一來，您可以從 Kubernetes 叢集控制平面執行個體收集日誌，並將其傳送至 AWS Support 以進行進一步調查。

無法透過 AWS Systems Manager 存取控制平面執行個體

當 Amazon EKS 控制平面執行個體無法透過 AWS Systems Manager (Systems Manager) 連線時，Amazon EKS 會顯示叢集的下列錯誤。

```
Amazon EKS control plane instances are not reachable through SSM. Please verify your
 SSM and network configuration, and reference the EKS on Outposts troubleshooting
 documentation.
```

若要解決此問題，請確定您的 VPC 和子網路符合在 AWS Outposts 上為 Amazon EKS 叢集建立 VPC 和子網路的要求，而且您已完成 AWS Systems Manager 使用者指南中設定 Session Manager 的步驟。

在 AWS Outpost 上建立 Amazon Linux 節點

本主題會說明如何啟動 Outpost 上已向 Amazon EKS 叢集註冊的 Amazon Linux 節點的 Auto Scaling 群組。叢集可以在 AWS 雲端或 Outpost 上。

- 現有的 Outpost。如需詳細資訊，請參閱什麼是 AWS Outpost。
- 現有 Amazon EKS 叢集。若要在 AWS 雲端部署叢集，請參閱 the section called “建立叢集”。若要在 Outpost 上部署叢集，請參閱the section called “執行本機叢集”。

- 假設您正在 AWS 雲端的叢集中建立節點，並在已啟用 AWS Outposts、 AWS Wavelength 或 AWS Local Zones 的 AWS 區域中擁有子網路。然後，在您建立叢集時，不得傳入這些子網路。如果您要在 Outpost 上的叢集中建立節點，則必須在建立叢集時在 Outpost 子網路中傳遞 。
- （建議用於 AWS 雲端上的叢集）適用於 Kubernetes 附加元件的 Amazon VPC CNI 外掛程式，設定了其本身具有必要 IAM 政策的 IAM 角色。如需詳細資訊，請參閱the section called “為 IRSA 設定”。本機叢集不支援服務帳戶的 IAM 角色。

您可以使用 `eksctl`或 AWS Management Console（使用 AWS CloudFormation 範本）建立自我管理的 Amazon Linux 節點群組。您也可使用 Terraform。

您可以使用此頁面中所述的下列工具，為本機叢集建立自我管理節點群組：

- the section called “`eksctl`”
- the section called “AWS Management Console”

Important

- 自我管理節點群組包含您帳戶中的 Amazon EC2 執行個體。當您或 Amazon EKS 代表您更新控制平面版本時，這些執行個體不會自動升級。自我管理節點群組在主控台中沒有任何需要更新的指示。您可以檢視安裝在節點上的 `kubelet` 版本，方法是選取您叢集的 Overview (概觀) 標籤上的 Nodes (節點) 清單上的節點，以判斷哪些節點需要更新。您必須手動更新節點。如需詳細資訊，請參閱the section called “更新方法”。
- kubelet 在自我管理節點上使用的憑證會以一年的過期期發出。預設不會啟用憑證輪換（請參閱：https：//https://kubernetes.io/docs/reference/config-api/kubelet-config.v1beta1/#kubelet-config-k8s-io-v1beta1-KubeletConfiguration)，這表示如果您的自我管理節點執行超過一年，就無法再向 Kubernetes API 進行身分驗證。
- 最佳實務是建議客戶定期更新其自我管理節點群組，以從最新的 Amazon EKS 最佳化 AMI 接收 CVEs 和安全性修補程式。更新用於自我管理節點群組的 AMI 也會觸發節點的重新建立，並確保它們不會因為過期的 kubelet 憑證而發生問題。
- 或者，您也可以在建立自我管理節點群組時啟用用戶端憑證輪換（請參閱：https：//https://kubernetes.io/docs/tasks/tls/certificate-rotation/)，以確保 kubelet 憑證隨著目前憑證接近過期而續約。

eksctl

使用 啟動自我管理的 Linux 節點 **eksctl**

1. 在您的裝置0.210.0或 AWS CloudShell 上安裝版本 或更新版本的eksctl命令列工具。如需有關安裝或更新 eksctl 的指示，請參閱 eksctl 文件中的安裝一節。
2. 如果您的叢集位於 AWS 雲端，且 AmazonEKS_CNI_Policy 受管 IAM 政策已連接至您的 Amazon EKS 節點 IAM 角色，建議您改為將其指派給與 Kubernetes aws-node服務帳戶相關聯的 IAM 角色。如需詳細資訊，請參閱the section called “為 IRSA 設定”。若您的叢集位於 Outpost 上，則政策必須連接至您的節點角色。
3. 以下命令會在現有的叢集建立節點群組。叢集必須使用 eksctl 來建立。將 *al-nodes* 取代為您的節點群組名稱。節點群組名稱不能超過 63 個字元。它必須以字母或數字開頭，但剩餘字元也可以包含連字符和底線。使用您叢集的名稱取代 *my-cluster*。此名稱僅能使用英數字元 (區分大小寫) 和連字號。它必須以英數字元開頭，且長度不可超過 100 個字元。名稱在您要建立叢集 AWS 的區域和 AWS 帳戶中必須是唯一的。若您的叢集存在於 Outpost 上，請使用 Outpost 子網路的 ID 來取代 *id*。如果您的叢集存在於 AWS 雲端上，請將 *ID* 取代為您建立叢集時未指定的子網路 ID。使用您 Outpost 支援的執行個體類型來取代 *instance-type*。將剩餘的###取代為您自己的值。依預設，系統會使用與控制平面相同的 Kubernetes 版本來建立節點。

   使用您 Outpost 上可用的執行個體類型來取代 *instance-type* 。

   將 *my-key* 取代為您的 Amazon EC2 金鑰對或公有金鑰的名稱。此金鑰會在節點啟動後用於將 SSH 套用至節點。如果您還沒有 Amazon EC2 金鑰對，您可以在 中建立一個金鑰對 AWS Management Console。如需詳細資訊，請參閱《Amazon EC2 使用者指南》中的 Amazon EC2 金鑰對。

   使用下列命令來建立您的節點群組。

   ```
   eksctl create nodegroup --cluster my-cluster --name al-nodes --node-type instance-
   type \
       --nodes 3 --nodes-min 1 --nodes-max 4 --managed=false --node-volume-type gp2 --
   subnet-ids subnet-id
   ```

   如果您的叢集部署在 AWS 雲端：

   - 您部署的節點群組可以從與執行個體不同的 CIDR 區塊將IPv4地址指派給 Pod。如需詳細資訊，請參閱the section called “自訂聯網”。
   - 您部署的節點群組不需要對外網際網路存取。如需詳細資訊，請參閱the section called “私有叢集”。

如需所有可用選項和預設值的完整清單，請參閱 `eksctl` 文件中的 AWS Outposts Support。

- 如果節點無法加入叢集，請參閱the section called “節點無法加入叢集”疑難排解 Amazon EKS 叢集和節點的問題the section called “無法將節點加入叢集”，以及疑難排解 AWS Outposts 上的本機 Amazon EKS 叢集。
- 範例輸出如下。建立節點時，會有數行輸出。輸出的最後幾行之一類似於以下的範例行。

```
[#]  created 1 nodegroup(s) in cluster "my-cluster"
```

4. (選用) 部署範例應用程式以測試您的叢集和 Linux 節點。

AWS Management Console

步驟 1：使用 啟動自我管理的 Linux 節點 AWS Management Console

1. 下載最新版本的 AWS CloudFormation 範本。

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/cloudformation/2022-12-23/
amazon-eks-nodegroup.yaml
```

2. 開啟 AWS CloudFormation 主控台。
3. 選擇 Create stack (建立堆疊)，然後選取 With new resources (standard) (使用新資源 (標準))。
4. 針對 Specify template (指定範本)，選取 Upload a template file (上傳範本檔案)，然後選取 Choose file (選擇檔案)。選取在上一步驟中下載的 `amazon-eks-nodegroup.yaml` 檔案，然後選取 Next (下一步)。
5. 在 Specify stack details (指定堆疊詳細資訊) 頁面上，據此填寫下列參數，然後選擇 Next (下一步)：
   - 堆疊名稱：為您的 AWS CloudFormation 堆疊選擇堆疊名稱。例如，您可以呼叫 *`al-nodes`*。此名稱僅能使用英數字元 (區分大小寫) 和連字號。它必須以英數字元開頭，且長度不可超過 100 個字元。名稱在您要建立叢集 AWS 的區域和 AWS 帳戶中必須是唯一的。
   - ClusterName：請輸入您叢集的名稱。如果此名稱與您的叢集名稱不符，您的節點將無法加入叢集。
   - ClusterControlPlaneSecurityGroup：從您在建立 VPC 時產生的 AWS CloudFormation 輸出中選擇 SecurityGroups 值。

     下列步驟顯示擷取適用群組的一種操作。

     i. 開啟 Amazon EKS 主控台。

ii. 選擇叢集的名稱。

iii. 選擇 Networking (網路) 索引標籤。

iv. 從 ClusterControlPlaneSecurityGroup 下拉式清單中選取時，請使用 Additional Security Group (其他安全群組) 值作為參考。

- NodeGroupName：輸入節點群組的名稱。此名稱稍後可用來識別為節點建立的 Auto Scaling 節點群組。
- NodeAutoScalingGroupMinSize：輸入節點 Auto Scaling 群組可以縮減的最低節點數。
- NodeAutoScalingGroupDesiredCapacity：堆疊建立時，輸入欲擴展的所需節點數量。
- NodeAutoScalingGroupMaxSize：輸入節點 Auto Scaling 群組可以擴增的最大節點數。
- NodeInstanceType：選擇節點的執行個體類型。如果您的叢集在 AWS 雲端上執行，如需詳細資訊，請參閱 the section called “Amazon EC2 執行個體類型”。若您的叢集正於 Outpost 上執行，則您僅可選取 Outpost 上可用的執行個體類型。
- NodeImageIdSSMParam：預先填入變數 Kubernetes 版本的最新 Amazon EKS 最佳化 AMI 的 Amazon EC2 Systems Manager 參數。若要使用 Amazon EKS 支援的不同 Kubernetes 次要版本，請以不同的 eks/latest/userguide/kubernetes-versions.html【支援的版本，type="documentation"】 取代 *1.XX#*建議指定與叢集相同的 Kubernetes 版本。

  若要使用 Amazon EKS 最佳化加速 AMI，請以 取代 *amazon-linux-2*amazon-linux-2-gpu。若要使用 Amazon EKS 最佳化 Arm AMI，請將 *amazon-linux-2* 取代為 amazon-linux-2-arm64。

  > **Note**
  >
  > Amazon EKS 節點 AMIs 是以 Amazon Linux 為基礎。您可以選擇所需版本的索引標籤，在 Amazon Linux 安全中心追蹤 Amazon Linux 的安全或隱私權事件。您也可以訂閱適用的 RSS 摘要。安全與隱私權事件包含問題的概觀、哪些套件受到影響，以及如何更新您的執行個體以修正問題。

- NodeImageId：（選用）如果您使用自己的自訂 AMI（而非 Amazon EKS 最佳化 AMI)，請輸入您 AWS 區域的節點 AMI ID。如果您在此指定值，則會覆寫 NodeImageIdSSMParam 欄位中的任何值。
- NodeVolumeSize：為您的節點指定根磁碟區大小 (以 GiB 為單位)。
- NodeVolumeType：為您的節點指定根磁碟區類型。
- KeyName：輸入 Amazon EC2 SSH 金鑰對的名稱，您可以在節點啟動後使用該金鑰對來透過 SSH 連接至節點。如果您還沒有 Amazon EC2 金鑰對，您可以在 中建立一個金鑰對 AWS

Management Console。如需詳細資訊，請參閱《Amazon EC2 使用者指南》中的 Amazon EC2 金鑰對。

> **Note**
>
> 如果您未在此處提供金鑰對， AWS CloudFormation 堆疊建立會失敗。

- BootstrapArguments：您可將多個選用引數傳遞至節點。如需詳細資訊，請檢視 GitHub 上的 bootstrap script usage information (啟動程序指令碼使用資訊)。如果您要將節點新增至 AWS Outposts 上的 Amazon EKS 本機叢集 （其中 Kubernetes 控制平面執行個體在 AWS Outpost 上執行），且叢集沒有傳入和傳出網際網路連線 （也稱為私有叢集），則必須提供下列引導引數 （作為單行）。

```
--b64-cluster-ca ${CLUSTER_CA} --apiserver-endpoint https://${APISERVER_ENDPOINT}
 --enable-local-outpost true --cluster-id ${CLUSTER_ID}
```

若要擷取 Amazon CLUSTER_CAEKS 本機叢集CLUSTER_ID的 APISERVER_ENDPOINT、 和值，請執行下列 AWS CLI 命令。將 cluster-name 取代為您的叢集名稱，並將 region （例如 us-east-1) 取代為您的叢集 AWS 區域。

```
echo "CLUSTER_CA=$(aws eks describe-cluster --name cluster-name --region region --
query cluster.certificateAuthority.data --output text)"

echo "APISERVER_ENDPOINT=$(aws eks describe-cluster --name cluster-name --region
 region --query cluster.endpoint --output text)"

echo "CLUSTER_ID=$(aws eks describe-cluster --name cluster-name --region region --
query cluster.id --output text)"
```

- DisableIMDSv1：預設情況下，每個節點都支援執行個體中繼資料服務版本 1 (IMDSv1) 和 IMDSv2。您可以停用 IMDSv1。若要防止節點群組中的未來節點和 Pod 使用 IMDSv1，請將 DisableIMDSv1 設為 true。如需 IMDS 的詳細資訊，請參閱設定執行個體中繼資料服務。如需在節點上限制存取的詳細資訊，請參閱限制存取指派給工作節點的執行個體設定檔。
- VpcId：輸入您建立的 VPC ID。選擇 VPC 之前，請檢閱 VPC 需求和考量事項。
- 子網路：如果叢集位於 Outpost 上，則請至少在 VPC 中選擇一個私有子網路。在選擇子網路之前，請先檢閱子網路需求和注意事項。您可以看到哪些子網是私有子網，方法是從叢集的 Networking (聯網) 標籤打開每一個子網連結。

6. 請在 Configure stack options (設定堆疊選項) 頁面上選取所需的選項，然後選擇 Next (下一頁)。

7. 選取我確認 AWS CloudFormation 可能會建立 IAM 資源的左側核取方塊，然後選擇建立堆疊。
8. 當當堆疊已完成建立時，從主控台將其選取，然後選擇 Outputs (輸出)。
9. 為已建立的節點群組記錄 NodeInstanceRole。當您設定 Amazon EKS 節點時會需要此值。

步驟 2：啟用節點以加入您的叢集

1. 檢查以瞭解是否有 aws-auth ConfigMap。

```
kubectl describe configmap -n kube-system aws-auth
```

2. 如果您看到 aws-auth ConfigMap，請視需要更新之。
   a. 開啟 ConfigMap 進行編輯。

```
kubectl edit -n kube-system configmap/aws-auth
```

   b. 視需要新增 mapRoles 個項目。將 rolearn 值設定為您在先前程序中記錄的 NodeInstanceRole 值。

```
[...]
data:
  mapRoles: |
    - rolearn: <ARN of instance role (not instance profile)>
      username: system:node:{{EC2PrivateDNSName}}
      groups:
        - system:bootstrappers
        - system:nodes
[...]
```

   c. 儲存檔案並結束您的文字編輯器。
3. 如果您收到一個錯誤，說明 "Error from server (NotFound): configmaps "aws-auth" not found，那麼請套用股票 ConfigMap。
   a. 下載組態對應。

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/cloudformation/2020-10-29/
aws-auth-cm.yaml
```

   b. 在 aws-auth-cm.yaml 檔案中，將 rolearn 設定為您在先前程序中紀錄的 NodeInstanceRole 值。您可以使用文字編輯器執行此操作，或取代 *my-node-instance-role* 並執行下列命令：

```
sed -i.bak -e 's|<ARN of instance role (not instance profile)>|my-node-instance-role|' aws-auth-cm.yaml
```

c. 套用組態。此命令可能需要幾分鐘的時間來完成。

```
kubectl apply -f aws-auth-cm.yaml
```

4. 查看節點的狀態，並等待他們到達 Ready 狀態。

```
kubectl get nodes --watch
```

輸入 Ctrl+C 傳回 Shell 提示。

> **Note**
>
> 如果您收到任何授權或資源類型錯誤，請參閱故障診斷主題中的the section called “未經授權或存取遭拒 (kubectl)”。

如果節點無法加入叢集，請參閱the section called “節點無法加入叢集”針對 Amazon EKS 叢集和節點問題進行故障診斷the section called “無法將節點加入叢集”，以及針對 AWS Outposts 上的本機 Amazon EKS 叢集進行故障診斷。

5. 安裝 Amazon EBS CSI 驅動程式。如需詳細資訊，請參閱 GitHub 上的 Installation (安裝)。在 Set up driver permission (設定驅動程式權限) 區段中，請務必依照 Using IAM instance profile (使用 IAM 執行個體設定檔) 選項指示操作。您必須使用 gp2 儲存類別。不支援gp3儲存體方案。

若要在叢集上建立 gp2 儲存類別，請完成以下步驟。

a. 執行下列命令以建立 gp2-storage-class.yaml 檔案。

```
cat >gp2-storage-class.yaml <<EOF
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  annotations:
    storageclass.kubernetes.io/is-default-class: "true"
  name: ebs-sc
provisioner: ebs.csi.aws.com
volumeBindingMode: WaitForFirstConsumer
parameters:
```

```
  type: gp2
  encrypted: "true"
allowVolumeExpansion: true
EOF
```

b. 將清單檔案套用至叢集。

```
kubectl apply -f gp2-storage-class.yaml
```

6. （僅限 GPU 節點）如果您選擇 GPU 執行個體類型和 Amazon EKS 最佳化加速 AMI，則必須將 Kubernetes 的 NVIDIA 裝置外掛程式套用為叢集上的 DaemonSet。在執行下列命令之前，將 *vX.X.X* 取代為所需的 NVIDIA/k8s-device-plugin 版本。

```
kubectl apply -f https://raw.githubusercontent.com/NVIDIA/k8s-device-plugin/vX.X.X/
deployments/static/nvidia-device-plugin.yml
```

Step3：其他動作

1. (選用) 部署範例應用程式以測試您的叢集和 Linux 節點。
2. 若您的叢集部署在 Outpost 上，請跳過此步驟。如果您的叢集部署在 AWS 雲端上，則下列資訊為選用。如果 AmazonEKS_CNI_Policy 受管 IAM 政策連接至您的 Amazon EKS 節點 IAM 角色，建議您改為將其指派給與 Kubernetes aws-node服務帳戶相關聯的 IAM 角色。如需詳細資訊，請參閱the section called “為 IRSA 設定”。

Amazon EKS 上的人工智慧 (AI) 和Machine Learning (ML) 概觀

Amazon Elastic Kubernetes Service (EKS) 是受管的 Kubernetes 平台，可讓組織部署、管理和擴展 AI 和機器學習 (ML) 工作負載，並擁有絕佳的彈性和控制能力。EKS 以開放原始碼 Kubernetes 生態系統為基礎，可讓您利用現有的 Kubernetes 專業知識，同時無縫整合開放原始碼工具和 AWS 服務。

無論您是訓練大規模模型、執行即時線上推論，還是部署生成式 AI 應用程式，EKS 都能提供 AI/ML 專案所需的效能、可擴展性和成本效益。

為什麼選擇適用於 AI/ML 的 EKS？

EKS 是受管 Kubernetes 平台，可協助您部署和管理複雜的 AI/ML 工作負載。以開放原始碼 Kubernetes 生態系統為基礎，與 AWS 服務整合，提供進階專案所需的控制和可擴展性。對於初次使用 AI/ML 部署的團隊，現有的 Kubernetes 技能會直接傳輸，以便有效率地協調多個工作負載。

EKS 支援從作業系統自訂到運算擴展的所有內容，其開放原始碼基礎可提升技術彈性，為未來的基礎設施決策保留選擇。平台提供 AI/ML 工作負載所需的效能和調校選項，支援下列功能：

- 完整叢集控制可微調成本和組態，而不會隱藏抽象概念
- 生產環境中即時推論工作負載的次秒延遲
- 進階自訂，例如多執行個體 GPUs、多雲端策略和作業系統層級調校
- 能夠使用 EKS 作為跨 AI/ML 管道的統一協調器來集中工作負載

金鑰使用案例

Amazon EKS 為各種 AI/ML 工作負載提供強大的平台，支援各種技術和部署模式：

- 即時（線上）推論：EKS 在 Amazon EC2 Inf1 和 Inf2 執行個體上使用 TorchServe、Triton Inference Server 和 KServe 等工具，支援對傳入資料的即時預測，例如詐騙偵測。這些工作負載受益於使用 Karpenter 和 KEDA 的動態擴展，同時利用 Amazon EFS 跨 Pod 分割模型。Amazon ECR Pull Through Cache (PTC) 可加速模型更新，而 Bottlerocket 資料磁碟區搭配 Amazon EBS 最佳化磁碟區可確保快速的資料存取。
- 一般模型訓練：組織使用 Amazon EC2 P4d 和 Amazon EC2 Trn1 執行個體上的 Kubeflow Training Operator (KRO)、Ray Serve 和 Torch Distributed Elastic，利用 EKS 來長期訓練大型資料集上的複

雜模型。 Amazon EC2 P4d Amazon EC2 Trn1 Volcano、Yunikorn 和 Kueue 等工具的批次排程支援這些工作負載。Amazon EFS 可共用模型檢查點，而 Amazon S3 會使用版本管理的生命週期政策來處理模型匯入/匯出。

- 擷取擴增產生 (RAG) 管道：EKS 透過整合擷取和產生程序來管理客戶支援聊天機器人和類似的應用程式。這些工作負載通常會使用 Argo Workflows 和 Kubeflow 等工具進行協同運作、Pinecone、Weaviate 或 Amazon OpenSearch 等向量資料庫，並透過 Application Load Balancer 控制器 (LBC) 向使用者公開應用程式。NVIDIA NIM 最佳化 GPU 使用率，同時 Prometheus 和 Grafana 監控資源使用量。
- 生成式 AI 模型部署：公司在 EKS 上部署即時內容建立服務，例如文字或影像產生，在 Amazon EC2 G5 和 Inferentia 加速器上使用 Ray Serve、vLLM 和 Triton Inference Server。 https://aws.amazon.com/blogs/containers/quora-3x-faster-machine-learning-25-lower-costs-with-nvidia-triton-on-amazon-eks/這些部署可最佳化大規模模型的效能和記憶體使用率。JupyterHub 可進行反覆開發，Gradio 提供簡單的 Web 介面，而 S3 掛載點 CSI 驅動程式允許將 S3 儲存貯體掛載為檔案系統，以存取大型模型檔案。
- 批次（離線）推論： Organizations 透過 AWS Batch 或 Volcano 的排程任務，有效率地處理大型資料集。這些工作負載通常會針對 AWS Inferentia 晶片使用 Inf1 和 Inf2 EC2 執行個體、針對 NVIDIA T4 GPUs 使用 Amazon EC2 G4dn 執行個體，或針對分析任務使用 c5 和 c6i CPU 執行個體，在離峰時間將資源使用率最大化。 G4dn T4 AWS Neuron SDK 和 NVIDIA GPU 驅動程式可最佳化效能，而 MIG/TS 則可啟用 GPU 共用。儲存解決方案包括 Amazon S3 和 Amazon EFS 和 FSx for Lustre，以及各種儲存類別的 CSI 驅動程式。模型管理利用 Kubeflow 管道、Argo 工作流程和 Ray 叢集等工具，而監控則由 Prometheus、Grafana 和自訂模型監控工具處理。

案例研究

客戶因為各種原因而選擇 Amazon EKS，例如最佳化 GPU 用量，或以低於秒的延遲執行即時推論工作負載，如下列案例研究所示。如需 Amazon EKS 的所有案例研究清單，請參閱AWS 客戶成功案例。

- Unitary 每天使用 AI 處理 2，600 萬部影片進行內容管制，需要高輸送量、低延遲推論，並已縮短容器開機時間 80%，確保在流量波動時快速回應擴展事件。
- Miro 是全球支援 7，000 萬名使用者的視覺化協作平台，與先前的自我管理 Kubernetes 叢集相比，運算成本降低了 80%。
- Synthesia 提供生成式 AI 影片建立服務，讓客戶從文字提示建立逼真的影片，在 ML 模型訓練輸送量方面達到 30 倍的改善。
- Harri 為服務業提供人力資源技術，實現了 90% 的擴展速度，以回應需求激增，並透過遷移到 AWS Graviton 處理器將其運算成本降低 30%。

- Ada Support 是採用 AI 技術的客戶服務自動化公司，可降低 15% 的運算成本，同時提高 30% 的運算效率。
- Snorkel AI 可讓企業建置和調整基礎模型和大型語言模型，透過實作 GPU 資源的智慧型擴展機制來節省超過 40% 的成本。

在 EKS 上開始使用Machine Learning

若要在 AWS 雲端的 EKS 上開始規劃和使用Machine Learning平台和工作負載，請繼續 the section called “資源”一節。

在 Amazon EKS 上執行即時線上推論工作負載

本節旨在協助您在 Amazon Elastic Kubernetes Service (EKS) 上部署和操作即時線上推論工作負載。您可以找到使用 GPU 加速節點建置最佳化叢集、整合儲存和自動擴展 AWS 服務、部署驗證範例模型，以及解耦 CPU 和 GPU 任務、選取適當的 AMIs 和執行個體類型，以及確保低延遲暴露推論端點等關鍵架構考量的相關指引。

主題

- Amazon EKS 即時推論最佳實務叢集設定指南

Amazon EKS 即時推論最佳實務叢集設定指南

簡介

本指南提供實作演練，以設定針對即時線上推論工作負載最佳化的 Amazon Elastic Kubernetes Service (EKS) 叢集，並整合 AWS 專家在整個過程中策劃的最佳實務。它使用 EKS Quickstart 架構，這是一組符合模型、加速器和擴展 AWS 最佳實務的精選驅動程式、執行個體類型和組態。此方法可協助您略過選取叢集設定的任務，讓您快速啟動和執行功能正常且預先設定的叢集。在此過程中，我們將部署範例工作負載來驗證您的設定、解釋關鍵架構概念（例如從 GPU 密集型運算解耦 CPU 繫結任務）、解決常見問題（例如，為什麼選擇 Bottlerocket AMI over AL2023？），並概述後續步驟以擴展叢集的功能。

本指南專為Machine Learning (ML) 和人工智慧 (AI) 工程師、平台管理員、運算子和初次使用 AWS 和 EKS 生態系統的資料/AI 專家而設計，假設熟悉 Kubernetes，但先前沒有 EKS 經驗。它旨在協助您了解啟動和執行即時線上推論工作負載所需的步驟和程序。本指南說明建立單一節點推論叢集的基本概念，包括佈建 GPU 資源、整合模型成品的儲存體、啟用安全 AWS 的服務存取，以及公開推論端點。

在整個過程中，它強調了面向使用者的應用程式的低延遲、彈性設計，例如詐騙偵測、即時聊天機器人，以及客戶意見回饋系統中的情緒分析。

在本指南中，我們只專注於使用 G5 EC2 執行個體設定基礎、規範性的起點。如果您要尋找 AWS Inferentia 特定的叢集組態或end-to-end工作流程，請參閱 the section called “使用 Inferentia 設定推論叢集”或 中的研討會the section called “資源”。

開始之前

開始之前，請確定您已執行下列任務：

- 為 Amazon EKS 設定您的環境
- 安裝最新版本的 eksctl
- 安裝 Helm
- （選用）安裝 Docker
- （選用）安裝 NGC CLI

架構

即時線上推論是指使用訓練的機器學習模型，以最低延遲在傳入資料串流上產生預測或輸出的程序。例如，它會啟用即時詐騙偵測、影像分類或產生知識圖表以回應使用者輸入。即時線上推論系統的架構透過從 GPU 密集 AI 運算分離 CPU 繫結的 Web 流量處理，在使用者面向的應用程式中提供低延遲的機器學習預測。此程序通常位於更大的應用程式生態系統中，通常包括後端、前端、向量和模型服務，專注於特殊元件，以啟用獨立擴展、平行開發和應對故障的彈性。隔離專用 GPU 硬體上的推論任務，並利用 APIs 和 WebSockets等界面，確保高度並行、快速處理轉換器等模型，以及透過前端進行使用者互動。請注意，雖然向量資料庫和擷取增強生成 (RAG) 管道通常在即時推論系統中扮演重要角色，但本指南並未涵蓋這些元件。典型架構至少包括：

- 前端服務：作為面向使用者的界面，處理用戶端邏輯，轉譯動態內容，並促進即時互動，它與後端服務通訊以啟動推論請求和顯示結果，通常向後端服務發起請求，該服務使用 WebSockets進行串流更新或 APIs進行結構化資料交換。此服務通常不需要專用負載平衡器，因為它可以託管在內容交付網路 (CDNs) 上，例如靜態資產的 AWS CloudFront 或直接從 Web 伺服器提供，如果動態內容需要，可透過自動擴展群組處理擴展。
- 後端服務：做為應用程式的協調器，管理商業邏輯，例如使用者身分驗證、資料驗證和服務協調（例如，透過 RESTful 端點APIs 或持久性連線WebSockets)。它會與推論服務通訊、在多核心 CPUs 和 RAM 上獨立擴展，在不依賴 GPUs 的情況下處理高 Web 流量，而且通常需要負載平衡器（例如 AWS Application Load Balancer 或 Network Load Balancer) 才能在多個執行個體之間分配傳

入請求，尤其是在高並行情況下。輸入控制器可以進一步管理外部存取和路由規則，以提高安全性和流量形狀。

- 推論服務：作為 AI 運算的核心，在具有足夠 VRAM （例如，8-12 GB 用於 DistilBERT 等模型）的 GPUs 上執行，以使用自訂或開放原始碼模型執行向量內嵌、知識擷取和模型推論 （例如，透過 APIs 用於批次請求或 WebSockets用於即時串流）。此隔離可防止相依性衝突、允許模型更新而不停機，並針對多個並行請求啟用具有負載平衡的水平擴展。為了有效地公開模型服務，它通常位於負載平衡器後方，以在複寫的執行個體之間分配 GPU 繫結工作負載，而傳入資源或控制器 （例如 ALB 傳入控制器 AWS) 則處理外部路由、SSL 終止和路徑型轉送，以確保安全且有效率的存取，而不會讓個別 GPUs 不堪負荷。

解決方案概觀

即時線上推論系統需要高效能的彈性架構，可提供極低的延遲，同時處理無法預測的大量流量暴增。此解決方案概觀說明下列 AWS 元件如何在我們將建立的 Amazon EKS 叢集中一起運作，以確保我們的叢集能夠託管和管理機器學習模型，以盡可能減少最終使用者的即時資料預測。

- Amazon G5 EC2 執行個體 — 對於 GPU 密集型推論任務，我們使用 g5.xlarge 和 g5.2xlarge G5 EC2 執行個體類型，其具有單一 (1) NVIDIA A10G GPU 和 24GB 記憶體 （例如 FP16 的 80 億個參數）。這些 GPUs 以 NVIDIA Ampere 架構為基礎，採用 NVIDIA A10G Tensor 核心 GPUs 和第二代 AMD EPYC 處理器，支援 4-8 個 vCPUs、高達 10 Gbps 的網路頻寬，以及 250-450 GB 的本機 NVMe SSD 儲存體，可確保複雜模型的快速資料移動和運算能力，因此非常適合低延遲、高輸送量的推論任務。選擇 EC2 執行個體類型是應用程式特定的，取決於您的模型 （例如影像、影片、文字模型），以及您的延遲和輸送量需求。例如，如果使用映像和/或影片模型，您可能想要使用 P5 EC2 執行個體來獲得最佳的即時延遲。我們建議您從 G5 EC2 執行個體開始，因為它為快速啟動和執行提供了良好的起點，然後透過效能基準測試評估它是否適合您的工作負載。對於更進階的案例，請考慮 G6 EC2 執行個體。
- Amazon EC2 M7g 執行個體 — 對於資料預先處理、API 請求處理、託管 Karpenter 控制器、附加元件和其他系統元件等 CPU 密集型任務，我們使用 m5.xlarge M7g EC2 執行個體類型。M7g 執行個體是以 ARM 為基礎的執行個體，具有 4 個 vCPUs、16 GB 記憶體、高達 12.5 Gbps 的網路頻寬，並採用 AWS Graviton3 處理器。選擇 EC2 執行個體類型是應用程式特定的，取決於工作負載的運算、記憶體和可擴展性需求。對於運算最佳化工作負載，您可以考慮使用 C7g EC2 執行個體，這些執行個體也使用 Graviton3 處理器，但針對特定使用案例的 M7g 執行個體，已針對更高的運算效能進行最佳化。或者，較新的 C8g EC2 執行個體 （如果有的話） 提供比 C7g 執行個體高 30% 的運算效能。我們建議您從 M7g EC2 執行個體開始，確保其成本效益和與各種工作負載的相容性 （例如，應用程式伺服器、微服務、遊戲伺服器、中型資料存放區），然後透過效能基準測試評估它是否適合您的工作負載。

- Amazon S3 掛載點 CSI 驅動程式 — 對於在單一 GPU 執行個體上的工作負載，其中多個 Pod 共用 GPU（例如，在同一節點上排程多個 Pod 以利用其 GPU 資源），我們使用掛載點 S3 CSI 驅動程式來最佳化記憶體使用量，對於成本敏感、低複雜度設定中的大型模型推論等任務來說至關重要。它將 Amazon S3 儲存貯體公開為 Kubernetes 叢集可用的類似 POSIX 檔案系統，允許推論 Pod 直接將模型成品（例如模型權重）讀取到記憶體中，而無需先下載它們，並使用標準檔案操作輸入資料集。此外，S3 具有幾乎無限制的儲存容量，並加速資料密集型推論工作負載。選擇儲存 CSI 驅動程式是應用程式特定的，取決於工作負載的輸送量和延遲需求。雖然 FSx for OpenZFS CSI 驅動程式跨節點為隨機 I/O 或完全 POSIX 相容共用持久性磁碟區提供低於毫秒的延遲，但我們建議您從掛載點 S3 CSI 驅動程式開始，因為其可擴展性、大型資料集的成本較低，以及內建與 S3-managed物件儲存的整合，用於讀取密集型推論模式（例如，串流模型輸入），然後透過效能基準測試評估它是否適合您的工作負載。
- EKS Pod 身分代理程式 — 為了啟用對 AWS 服務的存取，我們使用 EKS Pod 身分代理程式，該代理程式使用單一服務主體，並促進 Amazon EKS 叢集內的 Pod 層級 IAM 角色關聯。EKS Pod Identity 透過使用單一服務主體 (https：//) 來提供傳統 IAM Roles for Service Accounts (IRSA) 方法的簡化替代方案，而不是依賴每個叢集的個別 OIDC 供應商，這可讓您更輕鬆地指派許可。pods.eks.amazonaws.com 此外，它可讓角色在多個叢集之間重複使用，並支援進階功能，例如 IAM 角色工作階段標籤和目標 IAM 角色。
- EKS 節點監控代理程式 — 為確保推論服務的持續可用性和可靠性，我們使用 EKS 節點監控代理程式搭配自動修復，可自動偵測並取代運作狀態不佳的節點，將停機時間降至最低。它使用增強型運作狀態檢查（例如 KernelReady、NetworkingReady) 持續監控節點是否有硬體、核心、聯網和儲存問題。對於 GPU 節點，它會偵測加速器特定的故障，透過封鎖運作狀態不佳的節點、等待 10 分鐘以解決暫時性 GPU 問題，以及在 30 分鐘後取代節點以持續失敗，來啟動正常修復。
- Bottlerocket AMI — 為了為 EKS 叢集提供安全強化的基礎，我們使用 Bottlerocket AMI，這只包含執行容器所需的基本元件，並提供最短的開機時間以快速擴展。選擇節點 AMI 是應用程式特定的，取決於工作負載的自訂、安全性和可擴展性需求。雖然 AL2023 AMI 為主機層級安裝和自訂提供更大的彈性（例如，在 PV/PVC 中指定專用快取目錄，而不需要任何其他節點組態），但我們建議您從 Bottlerocket AMI 開始，以降低其佔用空間，並針對容器化工作負載（例如微服務、推論伺服器、可擴展APIs) 進行內建最佳化，然後透過效能基準測試來評估其是否適合您的工作負載。
- AWS Load Balancer控制器 (LBC) – 為了公開即時推論端點，我們使用 AWS Load Balancer控制器，其會自動佈建和管理適用於 HTTP/HTTPS 流量的 Application Load Balancer (ALBs)，以及適用於基於 Kubernetes Ingress and Service 資源的 TCP/UDP 流量的 Network Load Balancer (NLBs)，以便將推論模型與外部用戶端整合。此外，它支援路徑型路由等功能，可將推論請求分散到多個 Pod 或節點，確保在流量激增期間可擴展性，並透過連線多工和運作狀態檢查等 AWS原生最佳化將延遲降至最低。

1. 建立您的 EKS 叢集

在此步驟中，我們使用採用 AWS CloudFormation 技術的 eksctl ClusterConfig 範本建立具有 CPU 節點和受管節點群組的叢集。使用僅 CPU 節點初始化叢集，可讓我們專門使用 Karpenter 來管理 CPU 密集型和 GPU 節點，以便使用我們在後續步驟中建立的 Karpenter NodePools 來最佳化資源配置。為了支援即時推論工作負載，我們使用 EKS Bottlerocket AMI、EKS 節點監控代理程式、EKS Pod 身分代理程式、掛載點 S3 CSI 驅動程式、 AWS Load Balancer控制器 (LBC) 和 kube-proxy、vpc-cni 和 coredns 驅動程式佈建叢集。m7g.xlarge 執行個體將用於 CPU 系統任務，包括託管 Karpenter 控制器、附加元件和其他系統元件。

根據預設， `eksctl`會為 CIDR 區塊為 的叢集建立專用 VPC`192.168.0.0/16`。VPC 包含三個公有子網路和三個私有子網路，每個子網路分佈在三個不同的可用區域 （ 或`us-east-1`區域中AZs)，這是部署 Kubernetes 工作負載的理想方法。範本也會部署網際網路閘道，透過路由表中的預設路由和其中一個公有子網路中的單一 NAT 閘道提供公有子網路的網際網路存取，而私有子網路路由表中的預設路由會將傳出流量引導至 NAT 閘道以進行網際網路存取。若要進一步了解此設定，請參閱將節點部署至私有子網路。

檢查您的登入資料

檢查您的 AWS CLI 登入資料是否有效，並且可以使用 AWS 服務進行身分驗證：

```
aws sts get-caller-identity
```

如果成功，CLI 將傳回有關 AWS 您的身分 (UserId、帳戶和 Arn) 的詳細資訊。

檢查執行個體可用性

並非所有區域都提供 G5 執行個體類型。檢查離您最近的區域。例如：

```
aws ec2 describe-instance-types --instance-types g5.xlarge g5.2xlarge --region us-
east-1
```

如果成功，G5 執行個體類型可在您指定的區域中使用。

Bottlerocket AMI 並非所有區域都提供。透過擷取離您最近區域的 Bottlerocket AMI ID 進行檢查。例如：

```
aws ssm get-parameter --name /aws/service/bottlerocket/aws-k8s-1.33/arm64/latest/
image_id \
```

```
    --region us-east-1 --query "Parameter.Value" --output text
```

如果成功，Bottlerocket AMI 可在您指定的區域中使用。

準備您的環境

首先，在新的終端機視窗中設定下列環境變數。注意：請務必以您的唯一值取代範例預留位置，包括叢集名稱、所需區域、Karpenter 發行版本和 Kubernetes 版本。

> **Tip**
>
> 部分變數（例如 ${AWS_REGION}和 ${K8S_VERSION}) 會在區塊的早期定義，然後在稍後的命令中參考，以確保一致性並避免重複。請務必依序執行命令，以便正確匯出這些值，並可用於後續定義。

```
export TEMPOUT="$(mktemp)"
export K8S_VERSION=1.33
export KARPENTER_VERSION="1.5.0"
export AWS_REGION="us-east-1"
export EKS_CLUSTER_NAME="eks-rt-inference-${AWS_REGION}"
export S3_BUCKET_NAME="eks-rt-inference-models-${AWS_REGION}-$(date +%s)"
export NVIDIA_BOTTLEROCKET_AMI="$(aws ssm get-parameter --name /aws/service/
bottlerocket/aws-k8s-${K8S_VERSION}-nvidia/x86_64/latest/image_id --query
 Parameter.Value --output text)"
export STANDARD_BOTTLEROCKET_AMI="$(aws ssm get-parameter --name /aws/service/
bottlerocket/aws-k8s-${K8S_VERSION}/arm64/latest/image_id --query Parameter.Value --
output text)"
export AWS_ACCOUNT_ID="$(aws sts get-caller-identity --query Account --output text)"
export ALIAS_VERSION="$(aws ssm get-parameter --name "/aws/service/eks/optimized-
ami/${K8S_VERSION}/amazon-linux-2023/x86_64/standard/recommended/image_id" --query
 Parameter.Value | xargs aws ec2 describe-images --query 'Images[0].Name' --image-ids |
 sed -r 's/^.*(v[[:digit:]]+).*$/\1/')"
```

建立必要的角色和政策

Karpenter 需要特定的 IAM 角色和政策（例如 Karpenter 控制器 IAM 角色、執行個體描述檔和政策），才能將 EC2 執行個體管理為 Kubernetes 工作者節點。它使用這些角色來執行動作，例如啟動和終止 EC2 執行個體、標記資源，以及與其他 AWS 服務互動。使用 Karpenter 的 cloudformation.yaml 建立 Karpenter 角色和政策：

```
curl -fsSL https://raw.githubusercontent.com/aws/karpenter-provider-aws/v
${KARPENTER_VERSION}/website/content/en/preview/getting-started/getting-started-with-
karpenter/cloudformation.yaml > "${TEMPOUT}" \
&& aws cloudformation deploy \
  --stack-name "Karpenter-${EKS_CLUSTER_NAME}" \
  --template-file "${TEMPOUT}" \
  --capabilities CAPABILITY_NAMED_IAM \
  --parameter-overrides "ClusterName=${EKS_CLUSTER_NAME}"
```

AWS LBC 需要佈建和管理 AWS 負載平衡器的許可，例如為輸入資源建立 ALBs或為 類型的服務建立 NLBsLoadBalancer。我們將在叢集建立期間指定此許可政策。在叢集建立期間，我們會在 ClusterConfig 中使用 eksctl 建立服務帳戶。建立 LBC IAM 政策：

```
aws iam create-policy \
  --policy-name AWSLoadBalancerControllerIAMPolicy \
  --policy-document "$(curl -fsSL https://raw.githubusercontent.com/kubernetes-sigs/
aws-load-balancer-controller/v2.13.0/docs/install/iam_policy.json)"
```

安裝掛載點 S3 CSI 驅動程式時，其 DaemonSet Pod 會設定為使用服務帳戶執行。掛載點 S3 CSI 驅動程式的掛載點需要許可，才能與您稍後在本指南中建立的 Amazon S3 儲存貯體互動。我們將在叢集建立期間指定此許可政策。在叢集建立期間，我們會在 ClusterConfig 中使用 eksctl 建立服務帳戶。建立 S3 IAM 政策：

```
aws iam create-policy \
    --policy-name S3CSIDriverPolicy \
    --policy-document "{\"Version\": \"2012-10-17\", \"Statement\": [{\"Effect\":
 \"Allow\", \"Action\": [\"s3:GetObject\", \"s3:PutObject\", \"s3:AbortMultipartUpload
\", \"s3:DeleteObject\", \"s3:ListBucket\"], \"Resource\": [\"arn:aws:s3:::
${S3_BUCKET_NAME}\", \"arn:aws:s3:::${S3_BUCKET_NAME}/*\"]}]}"
```

注意：如果具有此名稱的角色已存在，請為角色指定不同的名稱。我們在此步驟中建立的角色專屬於您的叢集和 S3 儲存貯體。

建立叢集

在此範本中，eksctl 會自動為 EKS Pod Identity、Node Monitoring Agent、CoreDNS、Kubeproxy、VPC CNI 外掛程式建立 Kubernetes 服務帳戶。截至目前為止，掛載點 S3 CSI 驅動程式不適用於 EKS Pod 身分，因此我們會建立服務帳戶 (IRSA) 的 IAM 角色和 OIDC 端點。此外，我們為 AWS Load Balancer控制器 (LBC) 建立服務帳戶。為了存取 Bottlerocket

節點，eksctl 會自動連接適用於 Bottlerocket 的 AmazonSSMManagedInstanceCore，以允許透過 SSM 的安全 shell 工作階段。

在設定環境變數的相同終端機中，執行下列命令區塊來建立叢集：

```
eksctl create cluster -f - <<EOF
---
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig
metadata:
  name: ${EKS_CLUSTER_NAME}
  region: ${AWS_REGION}
  version: "${K8S_VERSION}"
  tags:
    karpenter.sh/discovery: ${EKS_CLUSTER_NAME}
    # Add more tags if needed for billing
iam:
  # Creates an OIDC endpoint and IRSA service account for the Mountpoint S3 CSI Driver
  # Uses the S3 CSI Driver policy for permissions
  withOIDC: true
  podIdentityAssociations:
  # Creates the pod identity association and service account
  # Uses the Karpenter controller IAM policy for permissions
  - namespace: "kube-system"
    serviceAccountName: karpenter
    roleName: ${EKS_CLUSTER_NAME}-karpenter
    permissionPolicyARNs:
    - arn:aws:iam::${AWS_ACCOUNT_ID}:policy/KarpenterControllerPolicy-
${EKS_CLUSTER_NAME}
  # Creates the pod identity association and service account
  # Uses the AWS LBC policy for permissions
  - namespace: kube-system
    serviceAccountName: aws-load-balancer-controller
    createServiceAccount: true
    roleName: AmazonEKSLoadBalancerControllerRole
    permissionPolicyARNs:
    - arn:aws:iam::${AWS_ACCOUNT_ID}:policy/AWSLoadBalancerControllerIAMPolicy
iamIdentityMappings:
- arn: "arn:aws:iam::${AWS_ACCOUNT_ID}:role/KarpenterNodeRole-${EKS_CLUSTER_NAME}"
  username: system:node:{{EC2PrivateDNSName}}
  groups:
  - system:bootstrappers
  - system:nodes
managedNodeGroups:
```

```
  # Creates 2 CPU nodes for lightweight system tasks
  - name: ${EKS_CLUSTER_NAME}-m7-cpu
    instanceType: m7g.xlarge
    amiFamily: Bottlerocket
    desiredCapacity: 2
    minSize: 1
    maxSize: 10
    labels:
      role: cpu-worker
# Enable automatic Pod Identity associations for VPC CNI Driver, coreDNS, kube-proxy
addonsConfig:
  autoApplyPodIdentityAssociations: true
addons:
  # Installs the S3 CSI Driver addon and creates IAM role
  # Uses the S3 CSI Driver policy for IRSA permissions
  - name: aws-mountpoint-s3-csi-driver
    attachPolicyARNs:
      - "arn:aws:iam::${AWS_ACCOUNT_ID}:policy/S3CSIDriverPolicy"
  - name: eks-pod-identity-agent
  - name: eks-node-monitoring-agent
  - name: coredns
  - name: kube-proxy
  - name: vpc-cni
EOF
```

這個過程需要幾分鐘的時間來完成。如果您想要監控狀態，請參閱 AWS CloudFormation 主控台。

2. 驗證叢集節點和 Pod 運作狀態

讓我們執行幾個運作狀態檢查，以確保叢集準備就緒。當先前的命令完成時，請檢視執行個體類型，並使用下列命令確認您的 CPU 系統節點已達到 Ready 狀態：

```
kubectl get nodes -L node.kubernetes.io/instance-type
```

預期的輸出應如下所示：

```
NAME                              STATUS   ROLES    AGE     VERSION
 INSTANCE-TYPE
ip-192-168-35-103.ec2.internal   Ready    <none>   12m     v1.33.0-eks-802817d
 m7g.xlarge
ip-192-168-7-15.ec2.internal     Ready    <none>   12m     v1.33.0-eks-802817d
 m7g.xlarge
```

使用下列命令，驗證所有 Pod Identity 關聯，以及它們如何將角色映射至叢集中命名空間中的服務帳戶：

```
eksctl get podidentityassociation --cluster ${EKS_CLUSTER_NAME} --region ${AWS_REGION}
```

輸出應會顯示 Karpenter ("karpenter") 和 AWS LBC ("aws-load-balancer-controller") 的 IAM 角色。

驗證 DaemonSets 是否可用：

```
kubectl get daemonsets -n kube-system
```

預期的輸出應如下所示：

```
NAME                           DESIRED CURRENT READY UP-TO-DATE AVAILABLE NODE SELECTOR
          AGE
aws-node                       3       3       3     3          3         <none>
          12m
dcgm-server                    0       0       0     0          0
 kubernetes.io/os=linux 12m
eks-node-monitoring-agent      3       3       3     3          3
 kubernetes.io/os=linux 12m
eks-pod-identity-agent         3       3       3     3          3         <none>
          12m
kube-proxy                     3       3       3     3          3         <none>
          12m
s3-csi-node                    2       2       2     2          2
 kubernetes.io/os=linux 12m
```

確認叢集上安裝了所有附加元件：

```
eksctl get addons --cluster ${EKS_CLUSTER_NAME} --region ${AWS_REGION}
```

預期的輸出應如下所示：

```
NAME                           VERSION              STATUS    ISSUES    IAMROLE
                                      UPDATE AVAILABLE    CONFIGURATION VALUES    POD
 IDENTITY ASSOCIATION ROLES
aws-mountpoint-s3-csi-driver   v1.15.0-eksbuild.1   ACTIVE    0
 arn:aws:iam::143095308808:role/eksctl-eks-rt-inference-us-east-1-addon-aws-m-Role1-
RAUjk4sJnc0L
coredns                        v1.12.1-eksbuild.2   ACTIVE    0
```

```
eks-node-monitoring-agent      v1.3.0-eksbuild.2    ACTIVE    0
eks-pod-identity-agent         v1.3.7-eksbuild.2    ACTIVE    0
kube-proxy                     v1.33.0-eksbuild.2   ACTIVE    0
metrics-server                 v0.7.2-eksbuild.3    ACTIVE    0
vpc-cni                        v1.19.5-eksbuild.1   ACTIVE    0
```

3. 安裝 Karpenter

在 CPU 工作者節點上安裝 Karpenter 控制器 (cpu-worker)，以最佳化成本並節省 GPU 資源。我們將將其安裝在「kube-system」命名空間中，並指定我們在叢集建立期間定義的「karpenter」服務帳戶。此外，此命令會設定 CPU 節點的叢集名稱和 Spot 執行個體中斷佇列。Karpenter 將使用 IRSA 擔任此 IAM 角色。

```
# Logout of helm registry before pulling from public ECR
helm registry logout public.ecr.aws

# Install Karpenter
helm upgrade --install karpenter oci://public.ecr.aws/karpenter/karpenter --version
 "${KARPENTER_VERSION}" --namespace "kube-system" --create-namespace  \
  --set "settings.clusterName=${EKS_CLUSTER_NAME}" \
  --set "settings.interruptionQueue=${EKS_CLUSTER_NAME}" \
  --set controller.resources.requests.cpu=1 \
  --set controller.resources.requests.memory=1Gi \
  --set controller.resources.limits.cpu=1 \
  --set controller.resources.limits.memory=1Gi \
  --set serviceAccount.annotations."eks\.amazonaws\.com/role-arn"="arn:aws:iam::
${AWS_ACCOUNT_ID}:role/${EKS_CLUSTER_NAME}-karpenter" \
  --wait
```

預期的輸出應如下所示：

```
Release "karpenter" does not exist. Installing it now.
Pulled: public.ecr.aws/karpenter/karpenter:1.5.0
Digest: sha256:9a155c7831fbff070669e58500f68d7ccdcf3f7c808dcb4c21d3885aa20c0a1c
NAME: karpenter
LAST DEPLOYED: Thu Jun 19 09:57:06 2025
NAMESPACE: kube-system
STATUS: deployed
REVISION: 1
TEST SUITE: None
```

確認 Karpenter 正在執行：

```
kubectl get pods -n kube-system -l app.kubernetes.io/name=karpenter
```

預期的輸出應如下所示：

```
NAME                         READY   STATUS    RESTARTS   AGE
karpenter-555895dc-865bc   1/1     Running   0          5m58s
karpenter-555895dc-j7tk9   1/1     Running   0          5m58s
```

4. 設定 Karpenter NodePools

在此步驟中，我們會設定互斥 CPU 和 GPU Karpenter NodePools。NodePool 規格中的 `limits` 欄位會限制每個 NodePool 在所有佈建節點中可以使用的資源（例如 CPU、記憶體、GPUs) 總數上限，如果超過這些限制，則可防止其他節點佈建。雖然 NodePools 支援廣泛的執行個體類別（例如 c、g)，但指定特定執行個體類型、容量類型和資源限制可協助您更輕鬆地預估隨需工作負載的成本。在這些 NodePools 中，我們在 G5 執行個體系列中使用一組不同的執行個體類型。這可讓 Karpenter 根據 Pod 資源請求自動選取最適當的執行個體類型，最佳化資源使用率，同時遵守 NodePool 的總限制。若要進一步了解，請參閱建立 NodePools。

設定 GPU NodePool

在此 NodePool 中，我們會設定資源限制來管理具有 GPU 功能的節點佈建。這些限制旨在限制集區中所有節點的總資源，總共允許最多 10 個執行個體。每個執行個體可以是 g5.xlarge (4 個 vCPUs、16 GiB 記憶體、1 個 GPU) 或 g5.2xlarge (8 個 vCPUs、32 GiB 記憶體、1 個 GPU)，只要總 vCPUs 不超過 80 個、總記憶體不超過 320GiB，且總 GPUs 不超過 10 個。例如，集區可以佈建 10 g5.2xlarge 執行個體 (80 個 vCPUs、320 GiB、10 GPUs) 或 10 g5.xlarge 執行個體 (40 個 vCPUs、160 GiB、10 GPUs)，或混合 5 g5.xlarge 和 5 g5.2xlarge (60 個 vCPUs、240 GiB、10 GPUs)，確保根據工作負載需求的彈性，同時遵守資源限制。

此外，我們指定 Bottlerocket AMI 的 Nvidia 變體 ID。最後，我們將中斷政策設定為在 30 分鐘 (`consolidateAfter: 30m`) 後移除空節點，並將節點生命週期上限設定為 30 天 (`expireAfter: 720h`)，以最佳化成本並維護 GPU 密集型任務的節點運作狀態。若要進一步了解，請參閱針對中斷敏感工作負載停用 Karpenter 合併，以及使用 ttlSecondsAfterFinished 自動清除 Kubernetes 任務。

```
cat <<EOF | envsubst | kubectl apply -f -
apiVersion: karpenter.sh/v1
kind: NodePool
metadata:
  name: gpu-a10g-inference-g5
spec:
  template:
```

```
    metadata:
      labels:
        role: gpu-worker
        gpu-type: nvidia-a10g
    spec:
      requirements:
        - key: node.kubernetes.io/instance-type
          operator: In
          values: ["g5.xlarge", "g5.2xlarge"]
        - key: "karpenter.sh/capacity-type"
          operator: In
          values: ["on-demand"]
      taints:
        - key: nvidia.com/gpu
          value: "true"
          effect: NoSchedule
      nodeClassRef:
        name: gpu-a10g-inference-ec2
        group: karpenter.k8s.aws
        kind: EC2NodeClass
      expireAfter: 720h
  limits:
    cpu: "80"
    memory: "320Gi"
    nvidia.com/gpu: "10"
  disruption:
    consolidationPolicy: WhenEmpty
    consolidateAfter: 30m
---
apiVersion: karpenter.k8s.aws/v1
kind: EC2NodeClass
metadata:
  name: gpu-a10g-inference-ec2
spec:
  amiFamily: Bottlerocket
  amiSelectorTerms:
    - id: ${NVIDIA_BOTTLEROCKET_AMI}
  role: "KarpenterNodeRole-${EKS_CLUSTER_NAME}"
  subnetSelectorTerms:
    - tags:
        karpenter.sh/discovery: "${EKS_CLUSTER_NAME}"
  securityGroupSelectorTerms:
    - tags:
        karpenter.sh/discovery: "${EKS_CLUSTER_NAME}"
```

```
  tags:
    nvidia.com/gpu: "true"
EOF
```

預期的輸出應如下所示：

```
nodepool.karpenter.sh/gpu-a10g-inference-g5 created
ec2nodeclass.karpenter.k8s.aws/gpu-a10g-inference-ec2 created
```

確認 NodePool 已建立且運作狀態良好：

```
kubectl get nodepool gpu-a10g-inference-g5 -o yaml
```

尋找`status.conditions`類似 `ValidationSucceeded: True`、 `NodeClassReady: True`和 `Ready: True` ，以確認 NodePool 運作狀態良好。

設定 CPU NodePool

在此 NodePool 中，我們會設定支援約 50 個執行個體的限制，以符合中等 CPU 工作負載 （例如 100-200 個 Pod) 和一般 AWS vCPU 配額 （例如 128-1152)。這些限制的計算假設 NodePool 應擴展到 50 m7.xlarge 執行個體：CPU （每個執行個體 4 個 vCPUs × 50 個執行個體 = 200 個 vCPUs) 和記憶體 （每個執行個體 16 GiB × 50 個執行個體 = 800 GiB)。這些限制旨在限制集區中所有節點的總資源，允許最多 50 m7g.xlarge 執行個體 （每個執行個體都有 4 個 vCPUs和 16 GiB 記憶體），只要總 vCPUs 不超過 200 且總記憶體不超過 800GiB。

此外，我們會指定 Bottlerocket AMI 標準變體的 ID。最後，我們將中斷政策設定為在 60 分鐘後移除空節點 (`consolidateAfter: 60m`)，並將節點生命週期上限設定為 30 天 (`expireAfter: 720h`)，以最佳化成本並維護 GPU 密集型任務的節點運作狀態。若要進一步了解，請參閱針對中斷敏感工作負載停用 Karpenter 合併，以及使用 ttlSecondsAfterFinished 自動清除 Kubernetes 任務。

```
cat <<EOF | envsubst | kubectl apply -f -
apiVersion: karpenter.sh/v1
kind: NodePool
metadata:
  name: cpu-inference-m7gxlarge
spec:
  template:
    metadata:
      labels:
        role: cpu-worker
    spec:
```

```
      requirements:
        - key: node.kubernetes.io/instance-type
          operator: In
          values: ["m7g.xlarge"]
        - key: karpenter.sh/capacity-type
          operator: In
          values: ["on-demand"]
      taints:
        - key: role
          value: cpu-intensive
          effect: NoSchedule
      nodeClassRef:
        name: cpu-inference-m7gxlarge-ec2
        group: karpenter.k8s.aws
        kind: EC2NodeClass
      expireAfter: 720h
  limits:
    cpu: "200"
    memory: "800Gi"
  disruption:
    consolidationPolicy: WhenEmpty
    consolidateAfter: 60m
---
apiVersion: karpenter.k8s.aws/v1
kind: EC2NodeClass
metadata:
  name: cpu-inference-m7gxlarge-ec2
spec:
  amiFamily: Bottlerocket
  amiSelectorTerms:
    - id: ${STANDARD_BOTTLEROCKET_AMI}
  role: "KarpenterNodeRole-${EKS_CLUSTER_NAME}"
  subnetSelectorTerms:
    - tags:
        karpenter.sh/discovery: "${EKS_CLUSTER_NAME}"
  securityGroupSelectorTerms:
    - tags:
        karpenter.sh/discovery: "${EKS_CLUSTER_NAME}"
EOF
```

預期的輸出應如下所示：

```
nodepool.karpenter.sh/cpu-inference-m7gxlarge created
```

```
ec2nodeclass.karpenter.k8s.aws/cpu-inference-m7gxlarge-ec2 created
```

確認 NodePool 已建立且運作狀態良好：

```
kubectl get nodepool cpu-inference-m7gxlarge -o yaml
```

尋找`status.conditions`類似 `ValidationSucceeded: True`、 `NodeClassReady: True`和 `Ready: True` ，以確認 NodePool 運作狀態良好。

5. 部署 GPU Pod 以公開 GPU

您需要 Nvidia 裝置外掛程式，才能讓 Kubernetes 向 Kubernetes 叢集公開 GPU 裝置。一般而言，您需要將外掛程式部署為 DaemonSet；不過， Bottlerocket AMI 會將外掛程式預先安裝為 AMI 的一部分。這表示使用 Bottlerocket AMIs 時，不需要部署 Nvidia 裝置外掛程式 DaemonSet。若要進一步了解，請參閱 Kubernetes 裝置外掛程式以公開 GPUs。

部署範例 Pod

Karpenter 可動態運作：當工作負載 (Pod) 請求 GPU 資源時，它會佈建 GPU 節點。若要驗證 Pod 能夠請求和使用 GPUs，請部署 Pod 以其限制請求`nvidia.com/gpu`資源（例如 `nvidia.com/gpu: 1`)。若要進一步了解這些標籤，請參閱使用 Well-Known 標籤以 GPU 需求排程工作負載。

```
cat <<EOF | envsubst | kubectl apply -f -
apiVersion: v1
kind: Pod
metadata:
  name: gpu-nvidia-smi
spec:
  restartPolicy: OnFailure
  tolerations:
  - key: "nvidia.com/gpu"
    operator: "Exists"
    effect: "NoSchedule"
  nodeSelector:
    role: gpu-worker  # Matches GPU NodePool's label
  containers:
  - name: cuda-container
    image: nvidia/cuda:12.9.1-base-ubuntu20.04
    command: ["nvidia-smi"]
    resources:
      limits:
        nvidia.com/gpu: 1
```

```
      requests:
        nvidia.com/gpu: 1
EOF
```

預期的輸出應如下所示：

```
pod/gpu-ndivia-smi created
```

給它一分鐘，然後檢查 Pod 是否具有「待定」、「ContainerCreating」、「執行中」以及「已完成」狀態：

```
kubectl get pod gpu-nvidia-smi -w
```

驗證 Pod 的節點是否屬於 GPU NodePool：

```
kubectl get node $(kubectl get pod gpu-nvidia-smi -o jsonpath='{.spec.nodeName}') -o
 custom-columns="Name:.metadata.name,Nodepool:.metadata.labels.karpenter\.sh/nodepool"
```

預期的輸出應如下所示：

```
Name                            Nodepool
ip-192-168-83-245.ec2.internal   gpu-a10g-inference-g5
```

檢查 Pod 的日誌：

```
kubectl logs gpu-nvidia-smi
```

預期的輸出應如下所示：

```
Thu Jul 17 04:31:33 2025
+-----------------------------------------------------------------------------------------
+
| NVIDIA-SMI 570.148.08                 Driver Version: 570.148.08         CUDA
 Version: 12.9 |
|-----------------------------------------+----------------------
+----------------------+
| GPU  Name                 Persistence-M | Bus-Id        Disp.A | Volatile Uncorr. ECC
 |
| Fan  Temp   Perf          Pwr:Usage/Cap |         Memory-Usage | GPU-Util  Compute M.
 |
```

```
|                                         |                      |               MIG M.
 |
|=========================================+======================
+======================|
|   0  NVIDIA A10G                    On  | 00000000:00:1E.0 Off |                    0
 |
|  0%   30C    P8               9W / 300W |      0MiB / 23028MiB |      0%      Default
 |
|                                         |                      |                  N/A
 |
+-----------------------------------------------------------------------------------------
+

+-----------------------------------------------------------------------------------------
+
| Processes:
 |
|  GPU        GI     CI        PID   Type   Process name                  GPU Memory
 |
|                     ID        ID                                         Usage
 |
|
=========================================================================================|
|  No running processes found
 |
+-----------------------------------------------------------------------------------------
+
```

6. （選用）準備和上傳模型成品以進行部署

在此步驟中，您將部署模型服務以進行即時影像分類，從將模型權重上傳至 Amazon S3 儲存貯體開始。為了示範，我們使用 NVIDIA GPUNet 的開放原始碼 GPUNet-0 視覺模型部分，該部分支援使用 NVIDIA GPUs 和 TensorRT 對影像進行低延遲推論。 GPUNet 此模型在 ImageNet 上預先訓練，可讓我們即時分類相片或影片串流中的物件，並被視為具有 1，190 萬個參數的小型模型。

設定您的環境

若要下載 GPUNet-0 模型權重 在此步驟中，您需要存取安裝在本機電腦上的 NVIDIA NGC 目錄和 Docker。請依照下列步驟設定免費帳戶並設定 NGC CLI：

- 註冊免費的 NGC 帳戶，並從 NGC 儀表板產生 API 金鑰 （使用者圖示 > 設定 > 產生 API 金鑰 > 產生個人金鑰 > NGC 目錄）。

- 下載並安裝 NGC CLI (Linux/macOS/Windows)，並使用 設定 CLIngc config set。出現提示時輸入您的 API 金鑰；將 org 設定為 nvidia，然後按 Enter 接受其他人的預設值。如果成功，您應該會看到類似：Successfully saved NGC configuration to /Users/your-username/.ngc/config。

驗證服務帳戶許可

開始之前，請檢查 Kubernetes 服務帳戶許可：

```
kubectl get serviceaccount s3-csi-driver-sa -n kube-system -o yaml
```

在叢集建立期間，我們將 S3CSIDriverPolicy 連接至 IAM 角色，並註釋服務帳戶 ("s3-csi-driver-sa")。掛載點 S3 CSI 驅動程式 Pod 會在與 S3 互動時繼承 IAM 角色的許可。預期的輸出應如下所示：

```
apiVersion: v1
kind: ServiceAccount
metadata:
  annotations:
    eks.amazonaws.com/role-arn: arn:aws:iam::143095308808:role/eksctl-eks-rt-inference-
us-east-1-addon-aws-m-Role1-fpXXjRYdKN8r
  creationTimestamp: "2025-07-17T03:55:29Z"
  labels:
    app.kubernetes.io/component: csi-driver
    app.kubernetes.io/instance: aws-mountpoint-s3-csi-driver
    app.kubernetes.io/managed-by: EKS
    app.kubernetes.io/name: aws-mountpoint-s3-csi-driver
  name: s3-csi-driver-sa
  namespace: kube-system
  resourceVersion: "2278"
  uid: 50b36272-6716-4c68-bdc3-c4054df1177c
```

新增容錯

S3 CSI 驅動程式會在所有節點上以 DaemonSet 身分執行。Pod 在這些節點上使用 CSI 驅動程式來掛載 S3 磁碟區。若要允許它在具有污點的 GPU 節點上排程，請將容錯新增至 DaemonSet：

```
kubectl patch daemonset s3-csi-node -n kube-system --type='json' -p='[{"op": "add",
 "path": "/spec/template/spec/tolerations/-", "value": {"key": "nvidia.com/gpu",
 "operator": "Exists", "effect": "NoSchedule"}}]'
```

預期的輸出應如下所示：

```
daemonset.apps/s3-csi-node patched
```

將模型權重上傳至 S3

在此步驟中，您將建立 Amazon S3 儲存貯體、從 NVIDIA GPU 雲端 (NGC) 下載 GPUNet-0 模型權重，並將其上傳至儲存貯體。我們的應用程式將在執行時間存取這些權重以進行推論。

建立 Amazon S3 儲存貯體：

```
aws s3 mb s3://${S3_BUCKET_NAME} --region ${AWS_REGION}
```

啟用儲存貯體的 S3 版本控制，以防止意外刪除和覆寫造成立即和永久的資料遺失：

```
aws s3api put-bucket-versioning --bucket ${S3_BUCKET_NAME} --versioning-configuration
 Status=Enabled
```

將生命週期規則套用至儲存貯體，以在物件版本變成非最新版本的 14 天後移除覆寫或刪除的物件版本、移除過期的刪除標記，並在 7 天後移除不完整的分段上傳。若要進一步了解，請參閱 S3 生命週期組態的範例。

```
aws s3api put-bucket-lifecycle-configuration --bucket $S3_BUCKET_NAME --lifecycle-
configuration '{"Rules":[{"ID":"LifecycleRule","Status":"Enabled","Filter":
{},"Expiration":{"ExpiredObjectDeleteMarker":true},"NoncurrentVersionExpiration":
{"NoncurrentDays":14},"AbortIncompleteMultipartUpload":{"DaysAfterInitiation":7}}]}'
```

從 NGC 下載 GPUNet-0 模型權重。例如，在 macOS 上：

```
ngc registry model download-version nvidia/dle/gpunet_0_pyt_ckpt:21.12.0_amp --dest ~/
downloads
```

> **Note**
>
> 您可能需要為您的作業系統調整此下載命令。若要讓此命令在 Linux 系統上運作，您可能需要建立 目錄做為命令的一部分 （ 例如 `mkdir ~/downloads`)。

預期的輸出應如下所示：

```
{
  "download_end": "2025-07-18 08:22:39",
  "download_start": "2025-07-18 08:22:33",
  "download_time": "6s",
  "files_downloaded": 1,
  "local_path": "/Users/your-username/downloads/gpunet_0_pyt_ckpt_v21.12.0_amp",
  "size_downloaded": "181.85 MB",
  "status": "Completed",
  "transfer_id": "gpunet_0_pyt_ckpt[version=21.12.0_amp]"
}
```

重新命名檢查點檔案以符合稍後步驟中應用程式程式碼中預期的命名（不需要擷取，因為它是包含模型狀態字典的標準 PyTorch *.pth.tar 檢查點）：

```
mv ~/downloads/gpunet_0_pyt_ckpt_v21.12.0_amp/0.65ms.pth.tar gpunet-0.pth
```

在 CLI AWS 中啟用AWS 通用執行期，以最佳化 S3 輸送量：

```
aws configure set s3.preferred_transfer_client crt
```

將模型權重上傳至 S3 儲存貯體：

```
aws s3 cp gpunet-0.pth s3://${S3_BUCKET_NAME}/gpunet-0.pth
```

預期的輸出應如下所示：

```
upload: ./gpunet-0.pth to s3://eks-rt-inference-models-us-east-1-1752722786/
gpunet-0.pth
```

建立模型服務

在此步驟中，您將使用 GPUNet-0 視覺模型為 GPU 加速影像分類設定 FastAPI Web 應用程式。應用程式會在執行時間從 Amazon S3 下載模型權重、從 NVIDIA 的儲存庫擷取模型架構以進行快取，並透過 HTTP 下載 ImageNet 類別標籤。應用程式包含映像預先處理轉換並公開兩個端點：用於狀態檢查的根 GET 和接受映像 URL 的 POST `/predict`端點。

我們使用 FastAPI 搭配 PyTorch 為模型提供服務，在容器化設定中於執行時間從 Amazon S3 載入權重，以進行快速原型設計和 Kubernetes 部署。如需最佳化批次處理或高輸送量引擎等其他方法，請參閱服務 ML 模型。

建立應用程式

為您的應用程式檔案建立目錄，例如 model-testing，然後將目錄變更為其中，並將下列程式碼新增至名為 的新檔案app.py：

```
import os
import torch
import json
import requests
from fastapi import FastAPI, HTTPException
from PIL import Image
from io import BytesIO, StringIO
import torchvision.transforms as transforms
from torch.nn.functional import softmax
import warnings
from contextlib import redirect_stdout, redirect_stderr
import argparse
import boto3
app = FastAPI()

# Suppress specific warnings from the model code (quantization is optional and unused
 here)
warnings.simplefilter("ignore", UserWarning)

device = torch.device("cuda" if torch.cuda.is_available() else "cpu")

# Load model code from cache (if present)
# Use backed cache directory
torch.hub.set_dir('/cache/torch/hub')

# Allowlist for secure deserialization (handles potential issues in older checkpoints)
torch.serialization.add_safe_globals([argparse.Namespace])
# Load the model architecture only on container startup (changed to pretrained=False)
# Precision (FP32 for full accuracy, could be 'fp16' for speed on Ampere+ GPUs)
with redirect_stdout(StringIO()), redirect_stderr(StringIO()):
    gpunet = torch.hub.load('NVIDIA/DeepLearningExamples:torchhub', 'nvidia_gpunet',
 pretrained=False, model_type='GPUNet-0', model_math='fp32')

# Download weights from S3 if not present, then load them
model_path = os.getenv('MODEL_PATH', '/cache/torch/hub/checkpoints/gpunet-0.pth')
os.makedirs(os.path.dirname(model_path), exist_ok=True)  # Ensure checkpoints dir
 exists
if not os.path.exists(model_path):
    s3 = boto3.client('s3')
```

```
    s3.download_file(os.getenv('S3_BUCKET_NAME'), 'gpunet-0.pth', model_path)
checkpoint = torch.load(model_path, map_location=device, weights_only=True)
gpunet.load_state_dict(checkpoint['state_dict'])
# Move to GPU/CPU
gpunet.to(device)
gpunet.eval()

# Preprocessing
preprocess = transforms.Compose([
    transforms.Resize(256),
    transforms.CenterCrop(224),
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225]),
])

# Load ImageNet labels
labels_url = "https://s3.amazonaws.com/deep-learning-models/image-models/
imagenet_class_index.json"
response = requests.get(labels_url)
json_data = json.loads(response.text)
labels = [json_data[str(i)][1].replace('_', ' ') for i in range(1000)]

# Required, FastAPI root
@app.get("/")
async def hello():
    return {"status": "hello"}

# Serve model requests
@app.post("/predict")
async def predict(image_url: str):
    try:
        response = requests.get(image_url)
        response.raise_for_status()
        img = Image.open(BytesIO(response.content)).convert("RGB")
        input_tensor = preprocess(img).unsqueeze(0).to(device)

        with torch.no_grad():
            output = gpunet(input_tensor)

        probs = softmax(output, dim=1)[0]
        top5_idx = probs.topk(5).indices.cpu().numpy()
        top5_probs = probs.topk(5).values.cpu().numpy()
```

```
        results = [{ "label": labels[idx], "probability": float(prob) } for idx, prob
 in zip(top5_idx, top5_probs)]

        return {"predictions": results}
    except Exception as e:
        raise HTTPException(status_code=400, detail=str(e))
```

建立 Dockerfile

下列 Dockerfile 會使用 NVIDIA 深度學習範例的 Tensor 核心 GitHub 儲存庫中的 GPUNet 模型，為應用程式建立容器映像。

我們透過使用僅限執行時間的 PyTorch 基礎、僅安裝具有快取清除的基本套件、預先快取模型程式碼，以及避免容器映像中的「製作」權重，來實現更快的提取和更新。若要進一步了解，請參閱減少容器映像大小。

在與 相同的目錄中app.py，建立 Dockerfile：

```
FROM pytorch/pytorch:2.4.0-cuda12.4-cudnn9-runtime

# Install required system packages required for git cloning
RUN apt-get update && apt-get install -y git && rm -rf /var/lib/apt/lists/*

# Install application dependencies
RUN pip install --no-cache-dir fastapi uvicorn requests pillow boto3 timm==0.5.4

# Pre-cache the GPUNet code from Torch Hub (without weights)
# Clone the repository containing the GPUNet code
RUN mkdir -p /cache/torch/hub && \
    cd /cache/torch/hub && \
    git clone --branch torchhub --depth 1 https://github.com/NVIDIA/
DeepLearningExamples NVIDIA_DeepLearningExamples_torchhub

COPY app.py /app/app.py

WORKDIR /app

CMD ["uvicorn", "app:app", "--host", "0.0.0.0", "--port", "80"]
```

測試應用程式。

從與 app.py和 相同的目錄Dockerfile，為推論應用程式建置容器映像，以 AMD64 架構為目標：

```
docker build --platform linux/amd64 -t gpunet-inference-app .
```

為您的 AWS 登入資料設定環境變數，並選擇性地設定 AWS 工作階段字符。例如：

```
export AWS_REGION="us-east-1"
export AWS_ACCESS_KEY_ID=ABCEXAMPLESCUJFEIELSMUHHAZ
export AWS_SECRET_ACCESS_KEY=123EXAMPLEMZREoQXr8XkiicsOgWDQ5TpUsq0/Z
```

在本機執行容器，注入 AWS 登入資料做為 S3 存取的環境變數。例如：

```
docker run --platform linux/amd64 -p 8080:80 \
  -e S3_BUCKET_NAME=${S3_BUCKET_NAME} \
  -e AWS_ACCESS_KEY_ID=${AWS_ACCESS_KEY_ID} \
  -e AWS_SECRET_ACCESS_KEY=${AWS_SECRET_ACCESS_KEY} \
  -e AWS_DEFAULT_REGION=${AWS_REGION} \
  gpunet-inference-app
```

預期的輸出應如下所示：

```
INFO:     Started server process [1]
INFO:     Waiting for application startup.
INFO:     Application startup complete.
INFO:     Uvicorn running on http://0.0.0.0:80 (Press CTRL+C to quit)
```

在新的終端機視窗中，使用公有映像 URL 做為查詢參數傳送範例 POST 請求，以測試推論端點：

```
curl -X POST "http://localhost:8080/predict?image_url=http://images.cocodataset.org/
test-stuff2017/000000024309.jpg"
```

預期的輸出應該是具有前 5 個預測的 JSON 回應，與此類似（實際標籤和機率可能會根據影像和模型精確度而略有不同）：

```
{"predictions":[{"label":"desk","probability":0.28885871171951294},
{"label":"laptop","probability":0.24679335951805115},
{"label":"notebook","probability":0.08539070934057236},
{"label":"library","probability":0.030645888298749924},
{"label":"monitor","probability":0.02989606373012066}]}
```

使用「Ctrl + C」退出應用程式。

將容器推送至 Amazon ECR

在此步驟中，我們會將 GPUNet-0 模型服務的容器映像上傳至 Amazon Elastic Container Registry (ECR)，以便在 Amazon EKS 上進行部署。此程序涉及建立新的 ECR 儲存庫來存放映像、使用 ECR 驗證，然後標記容器映像並將其推送至我們的登錄檔。

首先，導覽回您在本指南開頭設定環境變數的目錄。例如：

```
cd ..
```

在 Amazon ECR 中建立儲存庫：

```
aws ecr create-repository --repository-name gpunet-inference-app --region ${AWS_REGION}
```

登入 Amazon ECR：

```
aws ecr get-login-password --region ${AWS_REGION} | docker login --username AWS --
password-stdin ${AWS_ACCOUNT_ID}.dkr.ecr.${AWS_REGION}.amazonaws.com
```

預期的輸出應如下所示：

```
Login Succeeded
```

標記映像：

```
docker tag gpunet-inference-app:latest ${AWS_ACCOUNT_ID}.dkr.ecr.
${AWS_REGION}.amazonaws.com/gpunet-inference-app:latest
```

將映像推送到您的 Amazon ECR 儲存庫：

```
docker push ${AWS_ACCOUNT_ID}.dkr.ecr.${AWS_REGION}.amazonaws.com/gpunet-inference-
app:latest
```

這個最後一個步驟需要幾分鐘的時間才能完成。

7. （選用）公開模型服務

在此步驟中，您將使用 AWS Load Balancer控制器 (LBC) 在 Amazon EKS 外部公開即時推論模型服務。這包括設定 LBC、使用掛載點 Amazon S3 S3 的模型權重掛載為持久性磁碟區、部署 GPU 加速應用程式 Pod、建立服務和輸入以佈建 Application Load Balancer (ALB)，以及測試端點。

首先，驗證 AWS LBC 的 Pod Identity 關聯，確認服務帳戶已正確連結至所需的 IAM 角色：

```
eksctl get podidentityassociation --cluster ${EKS_CLUSTER_NAME} --namespace kube-system
 --service-account-name aws-load-balancer-controller
```

預期的輸出應如下所示：

```
ASSOCIATION ARN                                                    NAMESPACE    SERVICE
 ACCOUNT NAME        IAM ROLE ARN    OWNER ARN
arn:aws:eks:us-east-1:143095308808:podidentityassociation/eks-rt-inference-us-
east-1/a-buavluu2wp1jropya    kube-system     aws-load-balancer-controller
 arn:aws:iam::143095308808:role/AmazonEKSLoadBalancerControllerRole
```

標記您的叢集安全群組

AWS Load Balancer控制器僅支援具有標籤索引鍵的單一安全群組，karpenter.sh/discovery: "${EKS_CLUSTER_NAME}"用於 Karpenter 的安全群組選擇。使用 eksctl 建立叢集時，預設叢集安全群組（具有 "kubernetes.io/cluster/<cluster-name>: owned" 標籤）不會自動加上karpenter.sh/discovery標籤。此標籤對於 Karpenter 探索此安全群組並將其連接到其佈建的節點至關重要。連接此安全群組可確保與 AWS Load Balancer控制器 (LBC) 的相容性，使其能夠自動管理透過輸入公開之服務的傳入流量規則，例如這些步驟中的模型服務。

匯出叢集的 VPC ID：

```
CLUSTER_VPC_ID="$(aws eks describe-cluster --name ${EKS_CLUSTER_NAME} --query
 cluster.resourcesVpcConfig.vpcId --output text)"
```

匯出叢集的預設安全群組：

```
CLUSTER_SG_ID="$(aws ec2 describe-security-groups --filters Name=vpc-id,Values=
$CLUSTER_VPC_ID Name=tag-key,Values=kubernetes.io/cluster/${EKS_CLUSTER_NAME} --query
 'SecurityGroups[].[GroupId]' --output text)"
```

將karpenter.sh/discovery標籤新增至預設叢集安全群組。這將允許我們的 CPU 和 GPU EC2NodeClass 選擇器使用它：

```
aws ec2 create-tags --resources ${CLUSTER_SG_ID} --tags Key=karpenter.sh/
discovery,Value=${EKS_CLUSTER_NAME}
```

確認已新增標籤：

```
aws ec2 describe-security-groups --group-ids ${CLUSTER_SG_ID} --query
 "SecurityGroups[].Tags"
```

在結果中，您應該會看到具有 標籤和叢集名稱的以下內容。例如：

```
{
  "Key": "karpenter.sh/discovery",
  "Value": "eks-rt-inference-us-east-1"
}
```

設定 AWS Load Balancer控制器 (LBC)

AWS LBC 對於在 Amazon EKS 上管理 AI/ML 工作負載的輸入流量至關重要，可確保存取推論端點或資料處理管道。透過與 AWS Application Load Balancer (ALB) 和 Network Load Balancer (NLB) 整合，LBC 可將流量動態路由到容器化應用程式，例如執行大型語言模型、電腦視覺模型或即時推論服務的應用程式。由於我們已在叢集建立期間建立服務帳戶和 Pod Identity Association，因此我們`serviceAccount.name`將 設定為符合叢集組態 () 中定義的項目`aws-load-balancer-controller`。

新增 AWS擁有的 eks-charts Helm Chart 儲存庫：

```
helm repo add eks https://aws.github.io/eks-charts
```

使用最新的圖表重新整理本機 Helm 儲存庫：

```
helm repo update eks
```

使用 Helm 部署 AWS LBC，指定 EKS 叢集名稱並參考預先建立的服務帳戶：

```
helm install aws-load-balancer-controller eks/aws-load-balancer-controller \
  -n kube-system \
  --set clusterName=${EKS_CLUSTER_NAME} \
  --set serviceAccount.create=false \
  --set serviceAccount.name=aws-load-balancer-controller
```

預期的輸出應如下所示：

```
NAME: aws-load-balancer-controller
LAST DEPLOYED: Wed Jul 9 15:03:31 2025
NAMESPACE: kube-system
```

```
STATUS: deployed
REVISION: 1
TEST SUITE: None
NOTES:
AWS Load Balancer controller installed!
```

在持久性磁碟區中掛載模型

在此步驟中，您將使用 Amazon S3 CSI 驅動程式掛載點支援的PersistentVolume (PV)，從 Amazon S3 儲存貯體掛載模型權重。這可讓 Kubernetes Pod 以本機檔案的形式存取 S3 物件，消除暫時性 Pod 儲存或初始化容器的資源密集下載，非常適合大型、多 GB 模型權重。

PV 掛載整個儲存貯體根目錄 (中未指定路徑volumeAttributes)，支援多個 Pod 並行唯讀存取，並在容器內公開模型權重 (/models/gpunet-0.pth) 等檔案以進行推論。這可確保應用程式 (app.py) 中的備用「下載」不會觸發，因為檔案是透過掛載存在。透過從容器映像解耦模型，這可啟用共用存取和獨立模型版本更新，而無需重建映像。

建立PersistentVolume(PV)

建立 PersistentVolume (PV) 資源來掛載包含模型權重的 S3 儲存貯體，為多個 Pod 啟用唯讀存取，而無需在執行時間下載檔案：

```
cat <<EOF | envsubst | kubectl apply -f -
apiVersion: v1
kind: PersistentVolume
metadata:
  name: s3-model-pv
spec:
  capacity:
    storage: 5Gi  # Ignored by the driver; can be any value
  accessModes:
    - ReadOnlyMany  # Read only
  persistentVolumeReclaimPolicy: Retain
  storageClassName: ""  # Required for static provisioning
  claimRef:
    namespace: default  # Adjust if you prefer a different namespace
    name: s3-model-pvc
  mountOptions:
    - allow-other  # Enables multi-user access (useful for non-root pods)
    - region ${AWS_REGION} # Optional, include if your bucket is in a different region
 than the cluster
  csi:
    driver: s3.csi.aws.com
```

```
    volumeHandle: gpunet-model-volume  # Must be unique across all PVs
    volumeAttributes:
      bucketName: ${S3_BUCKET_NAME}
EOF
```

建立 PersistentVolumeClaim (PVC)

建立 PersistentVolumeClaim (PVC) 以繫結至 PV，請求對掛載的 S3 模型資料的唯讀存取權：

```
cat <<EOF | envsubst | kubectl apply -f -
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: s3-model-pvc
spec:
  accessModes:
    - ReadOnlyMany
  storageClassName: ""  # Required for static provisioning
  resources:
    requests:
      storage: 5Gi  # Ignored, match PV capacity
  volumeName: s3-model-pv  # Bind to the PV created above
EOF
```

部署應用程式

將推論應用程式部署為 Kubernetes 部署、掛載 S3-backed持久性磁碟區以進行模型存取、套用 GPU 節點選擇器和容錯，以及設定模型路徑的環境變數。此部署會設定模型路徑 (env var of "`/models/gpunet-0.pth`")，因此我們的應用程式（在 中`app.py`) 預設會使用此路徑。在 部署磁碟區掛載於 `/models`（唯讀）時，如果檔案已透過 PVC 存在，則不會觸發模型下載。

```
cat <<EOF | envsubst | kubectl apply -f -
apiVersion: apps/v1
kind: Deployment
metadata:
  name: gpunet-inference-app
spec:
  replicas: 1
  selector:
    matchLabels:
      app: gpunet-inference-app
  template:
    metadata:
```

```
      labels:
        app: gpunet-inference-app
    spec:
      tolerations:
      - key: "nvidia.com/gpu"
        operator: "Exists"
        effect: "NoSchedule"
      nodeSelector:
        role: gpu-worker
      containers:
      - name: inference
        image: ${AWS_ACCOUNT_ID}.dkr.ecr.${AWS_REGION}.amazonaws.com/gpunet-inference-
app:latest
        ports:
        - containerPort: 80
        env:
        - name: MODEL_PATH
          value: "/models/gpunet-0.pth"
        resources:
          limits:
            nvidia.com/gpu: 1
          requests:
            nvidia.com/gpu: 1
        volumeMounts:
        - name: model-volume
          mountPath: /models
          readOnly: true
      volumes:
      - name: model-volume
        persistentVolumeClaim:
          claimName: s3-model-pvc
EOF
```

如果 GPU 節點尚未提供，Karpenter 需要幾分鐘的時間來佈建 GPU 節點。確認推論 Pod 處於「執行中」狀態：

```
kubectl get pods -l app=gpunet-inference-app
```

預期的輸出應如下所示：

```
NAME                                  READY   STATUS    RESTARTS   AGE
gpunet-inference-app-5d4b6c7f8-abcde   1/1     Running   0          2m
```

使用輸入和Load Balancer公開服務

建立 ClusterIP 服務以針對應用程式的連接埠，在 EKS 叢集內部公開推論部署：

```
cat <<EOF | envsubst | kubectl apply -f -
apiVersion: v1
kind: Service
metadata:
  name: gpunet-model-service
spec:
  type: ClusterIP
  ports:
  - port: 80
    targetPort: 80
  selector:
    app: gpunet-inference-app
EOF
```

建立輸入資源，透過 AWS LBC 佈建面向網際網路的 Application Load Balancer (ALB)，將外部流量路由到推論服務：

```
cat <<EOF | envsubst | kubectl apply -f -
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: gpunet-model-ingress
  annotations:
    alb.ingress.kubernetes.io/scheme: internet-facing
    alb.ingress.kubernetes.io/target-type: ip
spec:
  ingressClassName: alb
  rules:
  - http:
      paths:
      - path: /
        pathType: Prefix
        backend:
          service:
            name: gpunet-model-service
            port:
              number: 80
EOF
```

給它幾分鐘的時間，讓 Application Load Balancer (ALB) 完成佈建。監控傳入資源狀態，以確認已佈建 ALB：

```
kubectl get ingress gpunet-model-ingress
```

預期的輸出應如下所示（填入 ADDRESS 欄位）：

```
NAME                   CLASS    HOSTS    ADDRESS
 PORTS   AGE
gpunet-model-ingress   alb      *        k8s-default-gpunetmo-183de3f819-516310036.us-
east-1.elb.amazonaws.com   80      6m58s
```

從輸入狀態擷取和匯出 ALB 主機名稱，以供後續測試使用：

```
export ALB_HOSTNAME=$(kubectl get ingress gpunet-model-ingress -o
 jsonpath='{.status.loadBalancer.ingress[0].hostname}')
```

測試模型服務

透過使用範例映像 URL（例如，從 COCO 資料集）傳送 POST 請求，模擬即時預測來驗證公開的推論端點：

```
curl -X POST "http://${ALB_HOSTNAME}/predict?image_url=http://images.cocodataset.org/
test-stuff2017/000000024309.jpg"
```

預期的輸出應該是具有前 5 個預測的 JSON 回應，與此類似（實際標籤和機率可能會根據影像和模型精確度而略有不同）：

```
{"predictions":[{"label":"desk","probability":0.2888975441455841},
{"label":"laptop","probability":0.2464350312948227},
{"label":"notebook","probability":0.08554483205080032},
{"label":"library","probability":0.030612602829933167},
{"label":"monitor","probability":0.02989672815084457}]}
```

您可以選擇性地在新的 POST 請求中繼續測試其他映像。例如：

```
http://images.cocodataset.org/test-stuff2017/000000024309.jpg
http://images.cocodataset.org/test-stuff2017/000000028117.jpg
http://images.cocodataset.org/test-stuff2017/000000006149.jpg
http://images.cocodataset.org/test-stuff2017/000000004954.jpg
```

結論

在本指南中，您會設定針對 GPU 加速即時推論工作負載最佳化的 Amazon EKS 叢集。您已使用 G5 EC2 執行個體佈建叢集，安裝掛載點 S3 CSI 驅動程式、EKS Pod 身分代理程式、EKS 節點監控代理程式、 Bottlerocket AMI、AWS Load Balancer控制器 (LBC) 和 Karpenter，以管理 CPU 和 GPU NodePools。您使用 NVIDIA 裝置外掛程式來啟用 GPU 排程，並使用 PersistentVolume 和 PersistentVolumeClaim 設定 S3 以進行模型存取。您透過部署範例 GPU Pod、在 Amazon S3 上設定 NVIDIA GPUNet-0 模型的模型存取權、啟用 Pod 初始化，以及透過 Application Load Balancer 公開推論服務來驗證設定。若要充分利用叢集，請使用自動修復設定 EKS 節點監控代理程式。請務必執行基準測試，包括 GPU 效能、延遲和輸送量評估，以最佳化回應時間。若要進一步了解，請參閱針對 AI/ML 工作負載使用監控和可觀測性工具。

清除

若要避免未來產生費用，您需要手動刪除相關聯的 CloudFormation 堆疊，以刪除本指南期間建立的所有資源，包括 VPC 網路。

使用 `--wait`旗標搭配 eksctl 刪除 CloudFormation 堆疊：

```
eksctl delete cluster --region ${AWS_REGION} --name ${EKS_CLUSTER_NAME} --wait
```

完成後，您應該會看到下列回應輸出：

```
2025-07-29 13:03:55 [#]  all cluster resources were deleted
```

使用 Amazon S3 主控台刪除本指南期間建立的 Amazon S3 儲存貯體。

AI/ML 工作負載的 Amazon EKS 叢集組態

本節旨在協助您設定針對 AI/ML 工作負載最佳化的 Amazon EKS 叢集。您可以找到使用 Linux 和 Windows 最佳化 AMIs 執行 GPU 加速容器、使用 Elastic Fabric Adapter (EFA) 設定訓練叢集以實現高效能聯網，以及使用 AWS Inferentia 執行個體建立推論叢集的指引，包括先決條件、step-by-step程序和部署考量。

主題

- 執行 GPU 加速容器 (EC2 上的 Linux)
- 執行 GPU 加速容器 (Windows on EC2 G-Series)
- 使用 Elastic Fabric Adapter 在 Amazon EKS 上執行機器學習訓練

- 將 AWS Inferentia 執行個體與 Amazon EKS 搭配使用以進行Machine Learning

執行 GPU 加速容器 (EC2 上的 Linux)

Amazon EKS 最佳化加速 Amazon Linux AMIs 是以標準 Amazon EKS 最佳化 Amazon Linux AMIs 為基礎建置。如需這些 AMIs的詳細資訊，請參閱 the section called “Amazon EKS 最佳化加速 Amazon Linux AMI”。下列文字說明如何啟用 AWS Neuron 型工作負載。

啟用 AWS Neuron (ML 加速器） 型工作負載

如需在 Amazon EKS 中使用 Neuron 訓練和推論工作負載的詳細資訊，請參閱下列參考：

- Containers - Kubernetes - Neuron 文件入門 AWS
- GitHub 上的 AWS Neuron EKS 範例訓練
- 在 Amazon EKS 上使用 AWSInferentia 部署 ML 推論工作負載

下列程序說明如何使用 Amazon EKS 最佳化加速 AMIs 在 GPU 型執行個體上執行工作負載。

1. 在 GPU 節點加入叢集後，您必須在叢集上將適用於 Kubernetes 的 NVIDIA 裝置外掛程式套用為 DaemonSet。在執行下列命令之前，將 *vX.X.X* 取代為所需的 NVIDIA/k8s-device-plugin 版本。

```
kubectl apply -f https://raw.githubusercontent.com/NVIDIA/k8s-device-plugin/vX.X.X/
deployments/static/nvidia-device-plugin.yml
```

2. 您可使用以下命令驗證您的節點有否配置 GPU。

```
kubectl get nodes "-o=custom-
columns=NAME:.metadata.name,GPU:.status.allocatable.nvidia\.com/gpu"
```

3. 使用下列內容建立名為 `nvidia-smi.yaml` 的檔案。將*##*取代為 nvidia/cuda 的所需標籤。此資訊清單會啟動在節點`nvidia-smi`上執行的 NVIDIA CUDA 容器。

```
apiVersion: v1
kind: Pod
metadata:
  name: nvidia-smi
spec:
  restartPolicy: OnFailure
  containers:
  - name: nvidia-smi
```

```
    image: nvidia/cuda:tag
    args:
    - "nvidia-smi"
    resources:
      limits:
        nvidia.com/gpu: 1
```

4. 執行以下命令，套用此清單檔案。

```
kubectl apply -f nvidia-smi.yaml
```

5. Pod 執行完成後，請使用下列命令檢視其日誌。

```
kubectl logs nvidia-smi
```

範例輸出如下。

```
Mon Aug  6 20:23:31 20XX
+-----------------------------------------------------------------------------+
| NVIDIA-SMI XXX.XX                 Driver Version: XXX.XX                    |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|===============================+======================+======================|
|   0  Tesla V100-SXM2...  On   | 00000000:00:1C.0 Off |                    0 |
| N/A   46C    P0    47W / 300W |      0MiB / 16160MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                       GPU Memory |
|  GPU       PID   Type   Process name                             Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
```

執行 GPU 加速容器 (Windows on EC2 G-Series)

> **Important**
>
> Kubernetes Device Plugin for DirectX by TensorWorks 是一種第三方工具，未經 背書、支援或維護 AWS。 AWS 對此外掛程式的安全性、可靠性或效能不負任何責任。

了解如何使用 NVIDIA GPU 搭配 Kubernetes Device Plugin for DirectX by TensorWorks，在 Amazon EKS (Elastic Kubernetes Service) 上執行 GPUs 加速的 Windows 容器工作負載。如需詳細資訊，請參閱 Kubernetes Device Plugin for DirectX。

設定 Windows 容器的 GPU 加速有兩種主要方法：

- 選項 1：建置預先安裝必要 GPU 驅動程式的自訂 EKS Windows Optimized AMI。
  - 當您需要準備好執行 GPU 加速 Windows 容器的一致預先設定環境，而且可以投入額外的精力來建置和維護自訂 AMI 時，請使用此方法。
- 選項 2：啟動執行個體後，在 EKS 工作者節點上安裝必要的 GPU 驅動程式。
  - 當您想要更簡單的設定程序，但不介意在每個新的工作者節點上安裝 GPU 驅動程式時，請使用此方法。當您評估或設計 GPU 加速工作負載的原型時，更適合開發環境。

您可以使用本指南中詳述的步驟來利用這兩種方法。

考量事項

本指南提供使用 NVIDIA GPU、NVIDIA GRID 驅動程式和 Kubernetes Device Plugin for DirectX by TensorWorks，為您的 Windows 容器安裝和設定 GPUs 加速的步驟。這些步驟已經過測試和驗證，可為 Amazon EKS 上的 Windows 容器工作負載提供 GPU 加速。如需相容驅動程式和裝置外掛程式的詳細資訊the section called “已知限制”，請參閱 。在繼續之前，請注意下列事項：

- 只有具有 NVIDIA GRID 驅動程式的 G 系列執行個體類型已經過測試和驗證，才能使用本指南。雖然其他執行個體類型和驅動程式組合也可能能夠執行 GPU 加速的 Windows 容器，但它們可能需要本指南中未涵蓋的其他組態步驟。
- 只有 DirectX 型工作負載已經過測試和驗證，才能使用本指南。雖然 OpenGL、Vulkan 和 OpenCL 等其他 GPU APIs 可能相容於執行 GPU 加速的 Windows 容器，但可能需要本指南中未涵蓋的其他組態步驟。
- 在執行 GPU 加速 Windows 容器之前，需要注意一些已知的限制。如需詳細資訊，請參閱the section called “已知限制”一節。

先決條件

若要在 Amazon EKS 上啟用 Windows 容器的 GPU 加速，您必須先準備下列需求，才能繼續：

- 使用 Kubernetes v1.27 或更新版本啟動 Amazon EKS 叢集。
- 佈建 Windows Server 2022 或更新版本的 Windows 節點。

- 佈建 G 系列執行個體類型的 Windows 節點，例如 G4 或 G5。
- 使用容器化 `1.7.x`或 佈建容器執行時間的 Windows 節點`2.x.x`。（請參閱 the section called “取得版本資訊” 驗證 Amazon EKS Optimized AMI 中的容器版本。）

在每個 Windows 節點上安裝 GPU 驅動程式

若要在 EKS 工作者節點上安裝 NVIDIA GRID 驅動程式，請遵循 Amazon EC2 執行個體 NVIDIA 驅動程式中概述的步驟。導覽至安裝選項 - 選項 3：GRID 驅動程式，並遵循安裝步驟。

安裝 for Windows Server Core

對於沒有桌面體驗的 Windows Server Core，請使用下列命令以無提示方式安裝 NVIDIA GRID 驅動程式：

```
$nvidiaInstallerFilePath = nvidia-driver-installer.exe # Replace with path to installer
$installerArguments = "-s -clean -noreboot -noeula"
Start-Process -FilePath $nvidiaInstallerFilePath -ArgumentList $installerArguments -
Wait -NoNewWindow -PassThru
```

驗證您的安裝

執行下列 PowerShell 命令，以顯示執行個體上 GPUs 的診斷資訊：

```
nvidia-smi
```

此命令會顯示 NVIDIA 驅動程式版本，以及 GPU 硬體的相關資訊。請確定此命令的輸出符合您預期安裝的 NVIDIA GRID 驅動程式版本。

在每個節點上部署 GPU 裝置外掛程式

若要啟用探索並將 GPU 資源公開至 Windows 節點上的容器，您需要裝置外掛程式。在 EKS 叢集中以 DaemonSet 身分執行，以 Tensorworks 在每個工作者節點上部署 DirectX 裝置外掛程式。請遵循 https：//README.md 中指定的安裝指南，其中將包含下列步驟。建議：

- 在 `kube-system` 命名空間中部署裝置外掛程式。
- 為 DaemonSet 設定適當的資源限制，以確保它不會在您的節點上消耗過多的資源。

> **Note**
>
> 裝置外掛程式 DaemonSet 會在每個節點上執行，做為具有更高權限的主機程序容器。建議實作 RBAC 控制項來限制對此 DaemonSet 的存取，因此只有授權使用者才能執行特權命令。

執行 GPU 加速容器時，裝置外掛程式支援兩種模式：

- 單一租用模式：此模式會將所有 GPU 資源專用於執行個體上的單一容器。使用下列命令搭配單一租用支援安裝裝置外掛程式。如需詳細資訊，請參閱 README.md：//。

```
kubectl apply -f "https://raw.githubusercontent.com/TensorWorks/directx-device-plugins/main/deployments/default-daemonsets.yml"
```

- 多租用模式：此模式允許在執行個體上的多個容器之間共用 GPU 資源。使用下列命令搭配多租用戶支援安裝裝置外掛程式。如需詳細資訊，請參閱 README.md：//。

```
kubectl apply -f "https://raw.githubusercontent.com/TensorWorks/directx-device-plugins/main/deployments/multitenancy-inline.yml"
```

  或者，使用 ConfigMap 指定多租用戶。

```
kubectl apply -f "https://raw.githubusercontent.com/TensorWorks/directx-device-plugins/main/deployments/multitenancy-configmap.yml"
```

驗證裝置外掛程式部署

部署裝置外掛程式之後，請取代 <namespace>並執行下列命令，以驗證 DirectX 裝置外掛程式是否在您的所有 Windows 節點上正確執行。

```
kubectl get ds device-plugin-wddm -n <namespace>
```

驗證容器已準備好進行部署

裝置外掛程式 DaemonSet 在採用 GPU 的 Windows 工作者節點上執行後，請使用下列命令來驗證每個節點是否具有可配置的 GPUs。對應的數字應與每個節點上的 DirectX 裝置數量相符。

```
kubectl get nodes "-o=custom-
columns=NAME:.metadata.name,DirectX:.status.allocatable.directx\.microsoft\.com/
display"
```

使用 GPU 加速執行 Windows 容器

啟動 Pod 之前，請在 `directx.microsoft.com/display`中指定資源名稱`.spec.containers[].resources`。這表示您的容器需要啟用 GPU 的功能，而 `kube-scheduler`會嘗試將 Pod 放置在具有可用 GPU 資源的預先設定 Windows 節點上。

例如，請參閱以下範例命令，該命令會啟動 `Job`以執行 Monte Carlo 模擬來估計 pi 的值。此範例來自 DirectX GitHub 儲存庫的 Kubernetes Device Plugins，該儲存庫有多個範例可供選擇，您可以執行這些範例來測試您的 Windows 節點 GPU 功能。 GitHub

```
cat <<EOF | kubectl apply -f -
apiVersion: batch/v1
kind: Job
metadata:
  name: example-cuda-montecarlo-wddm
spec:
  template:
    spec:
      containers:
      - name: example-cuda-montecarlo-wddm
        image: "index.docker.io/tensorworks/example-cuda-montecarlo:0.0.1"
        resources:
          limits:
            directx.microsoft.com/display: 1
      nodeSelector:
        "kubernetes.io/os": windows
      restartPolicy: Never
  backoffLimit: 0
EOF
```

已知限制

所有 GPUs 皆可使用

執行個體上的所有 GPUs 都可供主機上每個執行中的容器使用，即使您為指定容器請求特定數量的 GPUs。此外，預設行為是在主機上執行的所有容器都會使用索引為 0 的 GPU，即使節點上有多個

GPUs 可用。因此，若要讓多 GPU 任務正常運作，您必須明確指定要在應用程式程式碼中使用的特定 GPU 裝置。

配置裝置以用於應用程式的確切實作，取決於您使用的程式設計語言或架構。例如，如果您使用 CUDA 程式設計，若要選取特定 GPU，您可以使用 函數 cudaSetDevice() 明確指定要在應用程式程式碼中使用的裝置。

明確指定裝置的需要，是因為已知問題會影響 Windows 容器。您可以在 microsoft/Windows-Containers 問題 #333 中追蹤解決此問題的進度。下表代表此 GPU 配置行為的視覺化呈現和實際範例。

假設存在 EC2 執行個體類型 的單一 Windows 節點`g4dn.12xlarge`，並隨附四個 GPUs。考慮在此執行個體上啟動三個 Pod 的案例。此表格顯示，無論每個容器請求的 GPUs 數量為何，這三個 Pod 都可以存取執行個體上的所有四個 GPUs，預設會使用具有裝置索引 0 的 GPU。

Pod	請求的 GPUs	實際 GPU 存取	預設 GPU 用量	可用的 GPU 索引	執行個體 GPUs 總數
Pod 1	1 個 GPU	所有 4 個 GPUs	索引為 0 的 GPU	0、1、2、3	4
Pod 2	2 GPU	所有 4 個 GPUs	索引為 0 的 GPU	0、1、2、3	4
Pod 3	1 個 GPU	所有 4 個 GPUs	索引為 0 的 GPU	0、1、2、3	4

Kubernetes 裝置外掛程式支援

NVIDIA 的 Kubernetes 裝置外掛程式官方實作不支援 Windows。您可以在 NVIDIA/k8s-device-plugin 問題 #419 中追蹤新增官方 Windows 支援的進度。

GPU 運算執行個體限制

視 AWS 您的帳戶組態而定，您可以啟動的 Amazon EC2 GPU 運算執行個體數量和類型可能會有服務限制。如果您需要額外的容量，您可以請求提高配額。

必須建置 Windows GPU Optimized AMI

Amazon EKS 不提供 EKS Windows GPU Optimized AMI 或 EC2 Image Builder 受管元件。您需要遵循本指南中的步驟，使用預先安裝的必要 GPU 驅動程式建置自訂 EKS Windows Optimized AMI，或在啟動執行個體後在 EKS 工作者節點上安裝必要的 GPU 驅動程式。

不支援 Inferentia 和 Trainium

AWS Windows 不支援 Inferentia 和 AWS Trainium 型工作負載。

使用 Elastic Fabric Adapter 在 Amazon EKS 上執行機器學習訓練

本主題說明如何將 Elastic Fabric Adapter (EFA) 與部署在 Amazon EKS 叢集中的 Pod 整合。Elastic Fabric Adapter (EFA) 是 Amazon EC2 執行個體的網路介面，可讓您在 AWS上大規模執行需要高層級節點間通訊的應用程式。其自訂的作業系統略過硬體介面可增強執行個體間通訊的效能，這對於擴展這些應用程式至關重要。藉由 EFA，使用訊息傳遞介面 (MPI) 的高效能運算 (HPC) 應用程式和使用 NVIDIA 集體通訊程式庫 (NCCL) 的機器學習 (ML) 應用程式可擴展到數千個 CPU 或 GPU。因此，您可以使用 AWS 雲端的隨需彈性和彈性，獲得現場部署 HPC 叢集的應用程式效能。將 EFA 與 Amazon EKS 叢集上執行的應用程式整合，可減少完成大規模分散式訓練工作負載的時間，而無需在叢集中新增其他執行個體。如需 EFA 的詳細資訊，請參閱 Elastic Fabric Adapter。

具有 EFA 的執行個體類型

AWS EFA Kubernetes 裝置外掛程式支援所有具有 EFA 的 Amazon EC2 執行個體類型。若要查看具有 EFA 的所有執行個體類型的清單，請參閱《Amazon EC2 使用者指南》中的支援的執行個體類型。不過，為了快速執行 ML 應用程式，除了 EFA 之外，我們建議執行個體具有硬體加速晶片，例如 nVidia GPUs、AWS Inferentia 晶片或 AWS Trainium 晶片。若要查看具有硬體加速晶片和 EFA 的執行個體類型清單，請參閱《Amazon EC2 使用者指南》中的加速運算。

當您比較執行個體類型以在它們之間進行選擇時，請考慮該執行個體類型可用的 EFA 網路卡數量，以及加速器卡數量、CPU 數量和記憶體數量。對於每張網路卡您最多可以指派一個 EFA。EFA 算作網路界面。若要查看每個具有 EFA 的執行個體類型可使用多少 EFA，請參閱《Amazon EC2 使用者指南》中的網路卡清單。

僅 EFA 和 EFA 介面

Elastic Fabric Adapter (EFA) 是一種網路介面，結合了彈性網路轉接器 (ENA) 和 OS-bypass 介面的功能，採用 AWS 可擴展可靠資料包 (SRD) 通訊協定。EFA 功能可讓應用程式直接與硬體通訊，以進行低延遲傳輸。您可以選擇只使用僅限 EFA 的介面存取 EFA 功能，以限制與相同可用區域內介面的通訊。

若要建立可以具有僅限 EFA 介面的節點，您必須使用自訂 EC2 啟動範本，並將 InterfaceType設定為 efa-only。在自訂啟動範本中，您無法將網路卡設定為僅限 0 EFA 界面，因為這是 EC2 執行個體的主要網路卡和網路界面。對於僅限 EFA 的介面，您必須具有 VPC CNI 版本 1.18.5 或更新版本。如果您使用的是 Amazon Linux 2，則僅限 EfA 界面的 ami 版本必須是 v20240928或更新版本。

下列程序會引導您使用eksctl具有 nVidia GPUs 和 EFA 介面的節點來建立 EKS 叢集。您無法使用 eksctl 建立使用僅限 EFA 介面的節點和節點群組。

先決條件

- 現有 Amazon EKS 叢集。如果您沒有現有的叢集，請使用 建立一個叢集開始使用。您的叢集必須部署在具有至少一個私有子網路的 VPC 中，其中具有足夠的可用 IP 地址，以便部署節點。私有子網路必須具有外部裝置 (例如 NAT 閘道) 所提供的傳出網際網路存取權。

  如果計劃使用 eksctl 來建立您的節點群組，eksctl 也會為您建立叢集。
- 在您的裝置或 AWS CloudShell 上安裝和設定的 AWS 命令列界面 (AWS CLI) 版本 1.27.160 2.12.3或更新版本。若要檢查您目前的版本，請使用 aws --version | cut -d / -f2 | cut -d ' ' -f1。適用於 macOS 的 yum、 apt-get或 Homebrew 等套件管理員通常是最新版本 CLI AWS 後面的幾個版本。若要安裝最新版本，請參閱《 AWS 命令列界面使用者指南》中的使用 aws 設定安裝 和 快速組態。 https://docs.aws.amazon.com/cli/latest/userguide/cli-configure-quickstart.html#cli-configure-quickstart-config安裝在 AWS CloudShell 中的 AWS CLI 版本也可能是最新版本後面的幾個版本。若要更新它，請參閱《CloudShell AWS 使用者指南》中的將 CLI 安裝到您的主目錄。 AWS CloudShell
- kubectl 命令列工具安裝在您的裝置或 AWS CloudShell 上。版本可以與叢集的 Kubernetes 版本相同，也可以比叢集的 Kubernetes 版本更早或更晚一個次要版本。例如，如果您的叢集版本為 1.29，則可以搭配使用 kubectl 1.28、1.29 或 1.30 版。若要安裝或升級 kubectl，請參閱 the section called “設定 kubectl 和 eksctl”。
- 在啟動支援多個 Elastic Fabric Adapters 的工作者節點之前，您必須安裝適用於 Kubernetes 版本 1.7.10 或更新版本的 Amazon VPC CNI 外掛程式，例如 p4d或 p5。如需更新 Kubernetes 版本 Amazon VPC CNI 外掛程式的詳細資訊，請參閱 the section called “Amazon VPC CNI”。
- 對於 p6-b200 執行個體，您必須使用 EFA 裝置外掛程式 0.5.6 版或更新版本。

> **Important**
>
> 搭配 Kubernetes 採用 EFA 所需的一項重要考量，就是將大型頁面作為叢集中的資源進行設定和管理。如需詳細資訊，請參閱 Kubernetes 文件中的管理大型頁面。具有 EFA 驅動程式的

Amazon EC2 執行個體已安裝預先配置的 5128 2MiB Huge 頁面，您可以請求 做為資源，以在任務規格中使用。

建立節點群組

下列程序可協助您建立具有 EFA 介面和 GPUDirect RDMA 之 p4d.24xlarge 支援的節點群組，並針對使用 EFA 的多節點 NCCL 效能執行範例 NVIDIA 集體通訊程式庫 (NCCL) 測試。此範例可用於使用 EFA 在 Amazon EKS 上進行分散式深度學習訓練的範本。

1. 決定您要在其中部署節點的 AWS 區域中可使用哪些支援 EFA 的 Amazon EC2 執行個體類型。將 *region-code* 取代為您要部署節點群組 AWS 的區域。

```
aws ec2 describe-instance-types --region region-code \
    --filters Name=network-info.efa-supported,Values=true \
    --query "InstanceTypes[*].[InstanceType]" --output text
```

   當您部署節點時，您要部署的執行個體類型必須在叢集所在的 AWS 區域中可用。

2. 判斷您想要部署的執行個體類型所在的哪一個 Availability Zone (可用區域) 為可用。在本教學課程中，會使用p5.48xlarge執行個體類型，且必須在您在上一個步驟中指定的 AWS 區域的輸出中傳回。當您在生產叢集中部署節點時，請將 *p5.48xlarge ##*為上一個步驟中傳回的任何執行個體類型。

```
aws ec2 describe-instance-type-offerings --region region-code \
    --location-type availability-zone --filters Name=instance-
type,Values=p4d.24xlarge,p5.48xlarge \
    --query 'InstanceTypeOfferings[*].Location' --output text
```

   範例輸出如下。

```
us-west-2a    us-west-2c    us-west-2b
```

   請注意傳回的 Availability Zone (可用區域) 以供稍後步驟使用。將節點部署到叢集時，您的 VPC 必須在輸出中傳回的其中一個 Availability Zone (可用區域) 中具有可用 IP 位址的子網路。

3. 使用 建立節點群組eksctl。您需要在裝置0.210.0或 AWS CloudShell 上安裝版本 或更新版本的eksctl命令列工具。如需有關安裝或更新 eksctl 的指示，請參閱 eksctl 文件中的安裝一節。

a. 將下列內容複製到名為 *efa-cluster.yaml* 的檔案。使用自己的取代*###*。您可以將 *p5.48xlarge* 取代為不同的執行個體，但如果這樣做，請確定 的值availabilityZones是步驟 1 中針對執行個體類型傳回的可用區域。

```
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: my-efa-cluster
  region: region-code
  version: "1.XX"

iam:
  withOIDC: true

availabilityZones: ["us-west-2a", "us-west-2c"]

managedNodeGroups:
  - name: my-efa-ng
    instanceType: p5.48xlarge
    minSize: 1
    desiredCapacity: 2
    maxSize: 3
    availabilityZones: ["us-west-2a"]
    volumeSize: 300
    privateNetworking: true
    efaEnabled: true
```

b. 在現有叢集中建立受管節點群組。

```
eksctl create nodegroup -f efa-cluster.yaml
```

如果您沒有現有的叢集，您可以執行下列命令來建立叢集和節點群組。

```
eksctl create cluster -f efa-cluster.yaml
```

> **Note**
>
> 由於此範例中使用的執行個體類型具有 GPUs，因此在使用 Amazon Linux 2 時，eksctl會自動在每個執行個體上安裝 NVIDIA Kubernetes 裝置外掛程式。Bottlerocket 不需要這麼做，因為 NVIDIA 裝置外掛程式內建於 Bottlerocket 的 EKS NVIDIA 變體中。

在節點群組組態`true`中`efaEnabled`將 設定為 時， `eksctl` 也會自動在節點上部署 EFA 裝置外掛程式。

搭配 EFA 使用 Bottlerocket

Bottlerocket AMI 1.28.0 版及更新版本包含 EFA 的官方支援。若要將 Bottlerocket 用於已啟用 EFA 的節點，請在您的組態`amiFamily: Bottlerocket`中指定 。如果您需要使用自訂 AMI ID，則必須使用標準 `nodeGroups` 而非 `managedNodeGroups`。

以下是範例組態：

```
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: my-efa-bottlerocket-cluster
  region: region-code
  version: "1.XX"

iam:
  withOIDC: true

availabilityZones: ["us-west-2a", "us-west-2c"]

managedNodeGroups:
  - name: my-efa-bottlerocket-ng
    instanceType: p5.48xlarge
    minSize: 1
    desiredCapacity: 2
    maxSize: 3
    availabilityZones: ["us-west-2a"]
    volumeSize: 300
    privateNetworking: true
    efaEnabled: true
    amiFamily: Bottlerocket
    bottlerocket:
      enableAdminContainer: true
      settings:
        kernel:
          sysctl:
            "vm.nr_hugepages": "3000"  # Configures 3000 * 2Mi = 6000Mi hugepages
```

上面的 vm.nr_hugepages sysctl 設定會設定 2Mi 個巨型分頁的數量。在此範例中，3000 表示 3000 * 2Mi = 6000Mi 的巨型分頁。

驗證 EFA 裝置外掛程式安裝

當您使用 建立節點群組時efaEnabled: true， eksctl會自動為您部署 EFA Kubernetes 裝置外掛程式。您可以驗證裝置外掛程式是否已安裝並正常運作：

1. 檢查 DaemonSet 狀態：

```
kubectl get daemonsets -n kube-system
```

輸出範例：

```
NAME                                  DESIRED   CURRENT   READY   UP-TO-DATE
 AVAILABLE   NODE SELECTOR   AGE
aws-efa-k8s-device-plugin-daemonset   2         2         2       2            2
      <none>          6m16s
...
```

在這裡，EFA 裝置外掛程式 DaemonSet 正在兩個節點上執行。兩者皆為就緒且可用。

2. 接著，驗證 DaemonSet 建立的 Pod：

```
kubectl get pods -n kube-system -l name=aws-efa-k8s-device-plugin
```

輸出範例：

```
NAME                                        READY   STATUS    RESTARTS   AGE
aws-efa-k8s-device-plugin-daemonset-d68bs   1/1     Running   0          6m16s
aws-efa-k8s-device-plugin-daemonset-w4l8t   1/1     Running   0          6m16s
```

EFA 裝置外掛程式 Pod 處於執行中狀態，確認外掛程式已成功部署並運作。

3. 驗證資源註冊：

您可以透過描述節點來確認vpc.amazonaws.com/efa資源已向 kubelet 註冊：

```
kubectl describe nodes
```

如果 EFA 資源已正確註冊，您會在節點的容量和可配置資源下看到它。例如：

```
Capacity:
  ...
  vpc.amazonaws.com/efa:  4
Allocatable:
  ...
  vpc.amazonaws.com/efa:  4
```

此輸出會確認節點可辨識 EFA 資源，使其可供請求它的 Pod 使用。

（選用）測試 EFA 的效能

我們建議您測試 EFA 設定。您可以使用 GitHub 上儲存`aws-samples/awsome-distributed-training`庫中的 NCCL 測試。NCCL Tests 使用 Nvidia Collective Communication Library 評估網路的效能。下列步驟會在 Amazon EKS 上提交 NCCL 測試。

1. 部署 Kubeflow MPI 運算子：

   對於 NCCL 測試，您可以套用 Kubeflow MPI 運算子。MPI 運算子可以很容易地在 Kubernetes 上執行 Allreduce 式分散式訓練。如需詳細資訊，請參閱 GitHub 上的 MPI 運算子。

2. 執行多節點 NCCL 效能測試以驗證 GPUDirectRDMA/EFA：

   若要透過 EFA 使用 GPUDirectRDMA 驗證 NCCL 效能，請執行標準 NCCL 效能測試。如需詳細資訊，請參閱 GitHub 上的官方 NCCL 測試儲存庫。

   請完成以下步驟，以執行雙節點 NCCL 效能測試。在 NCCL 測試任務範例中，每個工作者請求八個 GPUs、5210Mi 的 hugepages-2Mi、四個 EFAs 和 8000Mi 的記憶體，這實際上表示每個工作者都會耗用`p5.48xlarge`執行個體的所有資源。

   a. 建立 MPIJob 資訊清單：

      將以下內容複製到名為 的檔案`nccl-tests.yaml`：

```
apiVersion: kubeflow.org/v2beta1
kind: MPIJob
metadata:
  name: nccl-tests
spec:
  runPolicy:
    cleanPodPolicy: Running
    backoffLimit: 20
```

```
  slotsPerWorker: 8
  mpiReplicaSpecs:
    Launcher:
      replicas: 1
      template:
         spec:
          restartPolicy: OnFailure
          containers:
          - image: public.ecr.aws/hpc-cloud/nccl-tests:latest
            imagePullPolicy: IfNotPresent
            name: test-nccl-launcher
            env:
             - name: PATH
               value: $PATH:/opt/amazon/efa/bin:/usr/bin
             - name: LD_LIBRARY_PATH
               value: /opt/amazon/openmpi/lib:/opt/nccl/build/lib:/opt/amazon/efa/
lib:/opt/aws-ofi-nccl/install/lib:/usr/local/nvidia/lib:$LD_LIBRARY_PATH
             - name: NCCL_DEBUG
               value: INFO
             - name: NCCL_BUFFSIZE
               value: '8388608'
             - name: NCCL_P2P_NET_CHUNKSIZE
               value: '524288'
             - name: NCCL_TUNER_PLUGIN
               value: /opt/aws-ofi-nccl/install/lib/libnccl-ofi-tuner.so
            command:
            - /opt/amazon/openmpi/bin/mpirun
            - --allow-run-as-root
            - --tag-output
            - -np
            - "16"
            - -N
            - "8"
            - --bind-to
            - none
            - -x
            - PATH
            - -x
            - LD_LIBRARY_PATH
            - -x
            - NCCL_DEBUG=INFO
            - -x
            - NCCL_BUFFSIZE
            - -x
```

```
            - NCCL_P2P_NET_CHUNKSIZE
            - -x
            - NCCL_TUNER_PLUGIN
            - --mca
            - pml
            - ^cm,ucx
            - --mca
            - btl
            - tcp,self
            - --mca
            - btl_tcp_if_exclude
            - lo,docker0,veth_def_agent
            - /opt/nccl-tests/build/all_reduce_perf
            - -b
            - "8"
            - -e
            - "16G"
            - -f
            - "2"
            - -g
            - "1"
            - -c
            - "1"
            - -n
            - "100"
    Worker:
      replicas: 2
      template:
        spec:
          nodeSelector:
            node.kubernetes.io/instance-type: "p5.48xlarge"
          containers:
          - image: public.ecr.aws/hpc-cloud/nccl-tests:latest
            imagePullPolicy: IfNotPresent
            name: nccl-tests-worker
            volumeMounts:
            - name: shmem
              mountPath: /dev/shm
            resources:
              limits:
                nvidia.com/gpu: 8
                hugepages-2Mi: 5120Mi
                vpc.amazonaws.com/efa: 32
                memory: 32000Mi
```

```
            requests:
              nvidia.com/gpu: 8
              hugepages-2Mi: 5120Mi
              vpc.amazonaws.com/efa: 32
              memory: 32000Mi
        volumes:
        - name: shmem
          hostPath:
            path: /dev/shm
```

b. 套用 NCCL 測試 MPIJob：

套用資訊清單MPIJob以提交 。這將建立兩個 p5.48xlarge Amazon EC2 執行個體。

```
kubectl apply -f nccl-tests.yaml
```

範例輸出如下。

```
mpijob.kubeflow.org/nccl-tests created
```

c. 確認任務已啟動 Pod：

檢視您執行中的 Pod。

```
kubectl get pods
```

範例輸出如下。

```
NAME                              READY   STATUS     RESTARTS   AGE
nccl-tests-launcher-nbql9    0/1     Init:0/1   0          2m49s
nccl-tests-worker-0          1/1     Running    0          2m49s
nccl-tests-worker-1          1/1     Running    0          2m49s
```

MPI Operator 會建立啟動器 Pod 和 2 個工作者 Pod（每個節點一個）。

d. 使用 日誌確認任務已成功執行：

檢視 Pod `nccl-tests-launcher` 的日誌。將 *nbql9* 取代為輸出中的值。

```
kubectl logs -f nccl-tests-launcher-nbql9
```

如果測試成功完成，您可以部署使用 Nvidia Collective Communication Library 的應用程式。

將 AWS Inferentia 執行個體與 Amazon EKS 搭配使用以進行Machine Learning

本主題描述了如何建立 Amazon EKS 叢集，其中具有執行 Amazon EC2 Inf1 執行個體的節點，以及 (選用) 如何部署範例應用程式。Amazon EC2 Inf1 執行個體採用 AWS Inferentia 晶片，由 自訂建置 AWS ，可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 部署到容器，這是一個專門的軟體開發套件 (SDK)，由編譯器、執行時間和分析工具組成，可最佳化 Inferentia 晶片的機器學習推論效能。 AWS Neuron 支援熱門的機器學習架構，例如 TensorFlow、PyTorch 和 MXNet。

> **Note**
>
> Neuron 裝置邏輯 ID 必須是連續的。如果請求多個 Neuron 裝置的 Pod 排程在 `inf1.6xlarge`或 `inf1.24xlarge`執行個體類型 （具有多個 Neuron 裝置） 上，如果 Kubernetes 排程器選取不連續的裝置 IDs，則該 Pod 將無法啟動。如需詳細資訊，請參閱 GitHub 上的裝置邏輯 ID 必須是連續的。

先決條件

- 將 `eksctl` 安裝在您的電腦上。如果您尚未安裝，請參閱 `eksctl` 文件中的安裝。
- 將 `kubectl` 安裝在您的電腦上。如需詳細資訊，請參閱the section called “設定 kubectl 和 eksctl”。
- (選用) 將 `python3` 安裝在您的電腦上。如果您尚未安裝，請參閱 Python 下載以取得安裝指示。

建立叢集

1. 建立具有 Inf1 Amazon EC2 執行個體之節點的叢集。您可以使用任何 Inf1 執行個體類型取代 *`inf1.2xlarge`*。`eksctl` 公用程式偵測到您正在啟動具有 `Inf1` 執行個體類型的節點群組，並將使用 Amazon EKS 最佳化加速 Amazon Linux AMI 之一來啟動您的節點。

   > **Note**
   >
   > 您無法將 IAM 角色用於具有 TensorFlow Serving 的服務帳戶。 TensorFlow

   ```
   eksctl create cluster \
   ```

```
    --name inferentia \
    --region region-code \
    --nodegroup-name ng-inf1 \
    --node-type inf1.2xlarge \
    --nodes 2 \
    --nodes-min 1 \
    --nodes-max 4 \
    --ssh-access \
    --ssh-public-key your-key \
    --with-oidc
```

> **Note**
>
> 請注意下列輸出行的值。它用於稍後（選用）步驟。

```
[9]  adding identity "arn:aws: iam::111122223333:role/eksctl-inferentia-nodegroup-ng-in-NodeInstanceRole-FI7HIYS3BS09" to auth ConfigMap
```

使用Inf1執行個體啟動節點群組時，eksctl會自動安裝 AWS Neuron Kubernetes 裝置外掛程式。這個外掛程式會將 Neuron 裝置做為系統資源公告給 Kubernetes 排程器，如此就能由容器請求。除了預設 Amazon EKS 節點 IAM 政策之外，還會新增 Amazon S3 唯讀存取政策，以便稍後步驟介紹的範例應用程式可以從 Amazon S3 載入訓練過的模型。

2. 確保所有 Pod 都已正確啟動。

```
kubectl get pods -n kube-system
```

縮寫輸出：

```
NAME                                   READY   STATUS    RESTARTS   AGE
[...]
neuron-device-plugin-daemonset-6djhp   1/1     Running   0          5m
neuron-device-plugin-daemonset-hwjsj   1/1     Running   0          5m
```

(選用) 部署 TensorFlow Serving 應用程式映像

經過訓練的模型必須編譯至 Inferentia 目標，才能部署到 Inferentia 執行個體上。若要繼續，您需要儲存在 Amazon S3 中的 Neuron 最佳化 TensorFlow 模型。如果您還沒有 SavedModel，請遵循建

立 Neuron 相容 ResNet50 模型的教學課程，並將產生的 SavedModel 上傳到 S3。ResNet-50 是一種熱門的機器學習模型，用於圖像識別任務。如需編譯 Neuron 模型的詳細資訊，請參閱 AWS Deep Learning AMIs 開發人員指南中的 AWS Inferentia Chip with DLAMI。 AMIs

範例部署資訊清單會管理 AWS 深度學習容器為 TensorFlow 提供的預先建置推論服務容器。容器內部是 AWS Neuron 執行期和 TensorFlow Serving 應用程式。針對 Neuron 最佳化的預建置 Deep Learning Containers 的完整清單會在 GitHub 上進行維護，位於可用圖像下。啟動時，DLC 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。

分配給服務應用程式的 Nesuron 裝置數量可以進行調整，方法是變更部署 yaml 中的 `aws.amazon.com/neuron` 資源。請注意，TensorFlow Serving 和 Neuron 執行時間之間的通訊透過 GRPC 進行，這需要將 `IPC_LOCK` 功能傳遞給容器。

1. 將 `AmazonS3ReadOnlyAccess` IAM 政策新增至在建立叢集的步驟 1 中建立的節點執行個體角色。這是必要步驟，以便範例應用程式可以從 Amazon S3 載入訓練過的模型。

```
aws iam attach-role-policy \
    --policy-arn arn:aws: iam::aws:policy/AmazonS3ReadOnlyAccess \
    --role-name eksctl-inferentia-nodegroup-ng-in-NodeInstanceRole-FI7HIYS3BS09
```

2. 使用下列內容建立名為 `rn50_deployment.yaml` 的檔案。更新區域碼和模型路徑以符合您想要的設定。當用戶端向 TensorFlow 伺服器發出請求時，模型名稱用於識別目的。此範例使用模型名稱來比對範例 ResNet50 用戶端指令碼，該指令碼將在稍後步驟中用於傳送預測請求。

```
aws ecr list-images --repository-name neuron-rtd --registry-id 790709498068 --region
 us-west-2
```

```
kind: Deployment
apiVersion: apps/v1
metadata:
  name: eks-neuron-test
  labels:
    app: eks-neuron-test
    role: master
spec:
  replicas: 2
  selector:
    matchLabels:
      app: eks-neuron-test
      role: master
  template:
```

```
    metadata:
      labels:
        app: eks-neuron-test
        role: master
    spec:
      containers:
        - name: eks-neuron-test
          image: 763104351884.dkr.ecr.us-east-1.amazonaws.com/tensorflow-inference-
neuron:1.15.4-neuron-py37-ubuntu18.04
          command:
            - /usr/local/bin/entrypoint.sh
          args:
            - --port=8500
            - --rest_api_port=9000
            - --model_name=resnet50_neuron
            - --model_base_path=s3://${your-bucket-of-models}/resnet50_neuron/
          ports:
            - containerPort: 8500
            - containerPort: 9000
          imagePullPolicy: IfNotPresent
          env:
            - name: AWS_REGION
              value: "us-east-1"
            - name: S3_USE_HTTPS
              value: "1"
            - name: S3_VERIFY_SSL
              value: "0"
            - name: S3_ENDPOINT
              value: s3.us-east-1.amazonaws.com
            - name: AWS_LOG_LEVEL
              value: "3"
          resources:
            limits:
              cpu: 4
              memory: 4Gi
              aws.amazon.com/neuron: 1
            requests:
              cpu: "1"
              memory: 1Gi
          securityContext:
            capabilities:
              add:
                - IPC_LOCK
```

3. 部署模型。

```
kubectl apply -f rn50_deployment.yaml
```

4. 使用下列內容建立名為 rn50_service.yaml 的檔案。HTTP 和 gRPC 連接埠會開啟以接受預測請求。

```
kind: Service
apiVersion: v1
metadata:
  name: eks-neuron-test
  labels:
    app: eks-neuron-test
spec:
  type: ClusterIP
  ports:
    - name: http-tf-serving
      port: 8500
      targetPort: 8500
    - name: grpc-tf-serving
      port: 9000
      targetPort: 9000
  selector:
    app: eks-neuron-test
    role: master
```

5. 為您的 TensorFlow 模型服務應用程式建立 Kubernetes 服務。

```
kubectl apply -f rn50_service.yaml
```

(選用) 針對您的 TensorFlow Serving 服務進行預測

1. 若要在本機測試，請將 gRPC 連接埠轉送至 eks-neuron-test 服務。

```
kubectl port-forward service/eks-neuron-test 8500:8500 &
```

2. 建立一個叫做 tensorflow-model-server-infer.py 的 Python 指令碼，具有以下內容。此指令碼透過 gRPC (為一服務框架) 執行推斷。

```
import numpy as np
   import grpc
```

```
    import tensorflow as tf
    from tensorflow.keras.preprocessing import image
    from tensorflow.keras.applications.resnet50 import preprocess_input
    from tensorflow_serving.apis import predict_pb2
    from tensorflow_serving.apis import prediction_service_pb2_grpc
    from tensorflow.keras.applications.resnet50 import decode_predictions

    if __name__ == '__main__':
        channel = grpc.insecure_channel('localhost:8500')
        stub = prediction_service_pb2_grpc.PredictionServiceStub(channel)
        img_file = tf.keras.utils.get_file(
            "./kitten_small.jpg",
            "https://raw.githubusercontent.com/awslabs/mxnet-model-server/master/docs/
images/kitten_small.jpg")
        img = image.load_img(img_file, target_size=(224, 224))
        img_array = preprocess_input(image.img_to_array(img)[None, ...])
        request = predict_pb2.PredictRequest()
        request.model_spec.name = 'resnet50_inf1'
        request.inputs['input'].CopyFrom(
            tf.make_tensor_proto(img_array, shape=img_array.shape))
        result = stub.Predict(request)
        prediction = tf.make_ndarray(result.outputs['output'])
        print(decode_predictions(prediction))
```

3. 執行指令碼，將預測提交至您的服務。

```
python3 tensorflow-model-server-infer.py
```

範例輸出如下。

```
[[(u'n02123045', u'tabby', 0.68817204), (u'n02127052', u'lynx', 0.12701613),
 (u'n02123159', u'tiger_cat', 0.08736559), (u'n02124075', u'Egyptian_cat',
 0.063844085), (u'n02128757', u'snow_leopard', 0.009240591)]]
```

在 Amazon EKS 上管理 AI/ML 工作負載的運算資源

本節旨在協助您管理 Amazon Elastic Kubernetes Service (EKS) 中機器學習工作負載的運算資源。您可以找到使用容量區塊為受管節點群組和自我管理節點預留 GPUs 的詳細資訊，包括先決條件、啟動範本設定、擴展組態、工作負載準備，以及處理保留生命週期和正常節點終止的重要考量。

主題

- 使用 ML 的容量區塊建立受管節點群組
- 使用 ML 的容量區塊建立自我管理節點

使用 ML 的容量區塊建立受管節點群組

機器學習 (ML) 的容量區塊可讓您指定未來日期保留 GPU 執行個體，以支援短期 ML 工作負載。如需詳細資訊，請參閱《Amazon EC2 Linux 執行個體使用者指南》中的 ML 容量區塊。

考量事項

> **Important**
>
> - 容量區塊僅適用於特定 Amazon EC2 執行個體類型和 AWS 區域。如需相容性資訊，請參閱《Amazon EC2 Linux 執行個體使用者指南》中的使用容量區塊先決條件。
> - 如需詳細資訊，請參閱「Amazon EC2 Auto Scaling 使用者指南」的使用容量區塊處理機器學習工作負載。
> - 具有容量區塊的受管節點群組只能使用自訂啟動範本建立。
> - 使用容量區塊升級受管節點群組時，請確定所需的節點群組大小設定為 0。

使用 Amazon EC2 容量區塊建立受管節點群組

您可以使用容量區塊搭配 Amazon EKS 受管節點群組來佈建和擴展 GPU 加速的工作者節點。以下 AWS CloudFormation 範本範例並未涵蓋生產叢集中所需的每個層面。一般而言，您也希望引導指令碼將節點加入叢集，並指定 Amazon EKS 加速 AMI。如需詳細資訊，請參閱the section called “建立”。

1. 建立適合您工作負載的啟動範本，並使用 Amazon EKS 受管節點群組。如需詳細資訊，請參閱the section called “啟動範本”。

   除了上述程序中的要求之外，請確定 `LaunchTemplateData`包含下列項目：

   - `InstanceMarketOptions` 並將 `MarketType` 設為 `"capacity-block"`
   - `CapacityReservationSpecification: CapacityReservationTarget` 將 `CapacityReservationId`設定為容量區塊 `cr-`*`02168da1478b509e0`*（例如：）
   - `InstanceType` 設定為支援容量區塊的執行個體類型（例如：*`p5.48xlarge`*）

   以下是 CloudFormation 範本的摘錄，該範本會建立以容量區塊為目標的啟動範本。若要建立自訂 AMI 受管節點群組，您也可以新增 `ImageId`和 `UserData` 參數。

```
NodeLaunchTemplate:
  Type: "AWS::EC2::LaunchTemplate"
  Properties:
    LaunchTemplateData:
      InstanceMarketOptions:
        MarketType: "capacity-block"
      CapacityReservationSpecification:
        CapacityReservationTarget:
          CapacityReservationId: "cr-02168da1478b509e0"
      InstanceType: p5.48xlarge
```

2. 使用啟動範本來建立受管節點群組。

   以下是容量區塊的建立節點群組命令範例。將###取代為您的叢集適用的值。

   建立容量區塊受管節點群組時，請執行下列動作：

   - 將 capacity-type 設定為 "CAPACITY_BLOCK"。如果容量類型未設定為 ，"CAPACITY_BLOCK"或缺少上述任何其他必要的啟動範本值，則建立請求將被拒絕。
   - 在建立請求subnets中指定 時，請務必只在與容量保留相同的可用區域中指定子網路。
   - 如果您在建立請求desiredSize中指定非零，Amazon EKS 會在建立 Auto Scaling 群組 (ASG) 時遵守該規則。不過，如果建立請求是在容量保留作用中之前提出，則 ASG 將無法啟動 Amazon EC2 執行個體，直到它變成作用中為止。因此，ASG 擴展活動會有啟動錯誤。每當保留變成作用中時，執行個體的啟動就會成功，而且 ASG 會在建立時向上擴展至desiredSize上述。

```
aws eks create-nodegroup \
    --cluster-name my-cluster \
    --nodegroup-name my-mng \
    --node-role node-role-arn \
    --region region-code \
    --subnets subnet-id \
    --scaling-config minSize=node-group-min-size,maxSize=node-group-max-
size,desiredSize=node-group-desired-size \
    --ami-type "AL2023_x86_64_NVIDIA" \
    --capacity-type "CAPACITY_BLOCK" \
    --launch-template id="lt-id",version=1
```

3. 請確保節點在擴展後聯結。使用具有容量區塊的受管節點群組的 Amazon EKS 叢集不會執行執行個體啟動時實際加入叢集並註冊的任何驗證。

4. 如果您在0建立時間desiredSize將 設定為 ，則當容量保留變為作用中時，您有不同的選項可以擴展節點群組：
   - 為 ASG 建立符合容量區塊保留開始時間的排程擴展政策。如需詳細資訊，請參閱《Amazon EC2 Auto Scaling 使用者指南》中的為 Amazon EC2 Auto Scaling 排程擴展。
   - 使用 Amazon EKS 主控台或 `eks update-nodegroup-config`更新擴展組態，並設定所需的節點群組大小。
   - 使用 Kubernetes Cluster Autoscaler。如需詳細資訊，請參閱 Cluster Autoscaler on AWS。
5. 節點群組現在已準備好排程工作負載和 Pod。
6. 為了讓您的 Pod 在保留結束之前正常耗盡，Amazon EKS 會使用排程擴展政策，將節點群組大小縮減為 `0` 。此排程擴展將設定為名為 `Amazon EKS Node Group Capacity Scaledown Before Reservation End` 的名稱。建議您不要編輯或刪除此動作。

   Amazon EC2 會在保留結束時間前 30 分鐘開始關閉執行個體。因此，Amazon EKS 將在節點群組的保留結束前 40 分鐘設定排程縮減規模，以安全且優雅地移出 Pod。

使用 ML 的容量區塊建立自我管理節點

機器學習 (ML) 的容量區塊可讓您指定未來日期保留 GPU 執行個體，以支援短期 ML 工作負載。如需詳細資訊，請參閱《Amazon EC2 Linux 執行個體使用者指南》中的 ML 容量區塊。

考量事項

> **Important**
>
> - 容量區塊僅適用於特定 Amazon EC2 執行個體類型和 AWS 區域。如需相容性資訊，請參閱《Amazon EC2 Linux 執行個體使用者指南》中的使用容量區塊先決條件。
> - 如果您在容量保留變成作用中之前建立自我管理節點群組，請將所需的容量設定為 `0`。
> - 為了能有足夠的時間正常耗盡節點，建議您在容量區塊保留結束前預留 30 分鐘以上，排程擴展至零。
> - 為了讓您的 Pod 正常耗盡，建議您如範例步驟所述設定 AWS 節點終止處理常式。

搭配自我管理節點使用容量區塊

您可以將容量區塊與 Amazon EKS 搭配使用，藉此佈建和擴展您的自我管理節點。下列步驟提供一般範例概觀。 AWS CloudFormation 範本範例並未涵蓋生產工作負載中所需的每個層面。一般而言，您

也希望引導指令碼將節點加入叢集、指定 Amazon EKS 加速 AMI，以及適當的執行個體描述檔以加入叢集。如需詳細資訊，請參閱the section called “Amazon Linux”。

1. 建立適用於工作負載的啟動範本。如需詳細資訊，請參閱「Amazon EC2 Auto Scaling 使用者指南」的使用容量區塊處理機器學習工作負載。

   請確定 LaunchTemplateData包含下列項目：

   - InstanceMarketOptions 並將 MarketType 設為 "capacity-block"
   - CapacityReservationSpecification: CapacityReservationTarget 將 CapacityReservationId設定為容量區塊 cr-*02168da1478b509e0* （例如：）
   - IamInstanceProfile 將 Arn設定為適用的 *iam-instance-profile-arn*
   - ImageId 設定為適用的 *image-id*
   - InstanceType 設定為支援容量區塊的執行個體類型（例如：*p5.48xlarge*）
   - SecurityGroupIds 設定為適用的 IDs（例如：*sg-05b1d815d1EXAMPLE*）
   - UserData 設定為自我管理節點群組適用的*#####*

   以下是 CloudFormation 範本的摘錄，該範本會建立以容量區塊為目標的啟動範本。

   ```
   NodeLaunchTemplate:
     Type: "aws::EC2::LaunchTemplate"
     Properties:
       LaunchTemplateData:
         InstanceMarketOptions:
           MarketType: "capacity-block"
         CapacityReservationSpecification:
           CapacityReservationTarget:
             CapacityReservationId: "cr-02168da1478b509e0"
         IamInstanceProfile:
           Arn: iam-instance-profile-arn
         ImageId: image-id
         InstanceType: p5.48xlarge
         KeyName: key-name
         SecurityGroupIds:
         - sg-05b1d815d1EXAMPLE
         UserData: user-data
   ```

   由於容量區塊為區域性，因此您必須在進行保留的可用區域中傳遞子網路。

2. 使用啟動範本來建立自我管理節點群組。如果您在容量保留變成作用中之前執行此操作，請將所需的容量設定為 0。建立節點群組時，請確定您只為保留容量的可用區域指定個別子網路。

   以下是範例 CloudFormation 範本，您可以在建立適用於工作負載的範例 CloudFormation 範本時參考。此範例會取得上一個步驟中顯示之 AWS::Amazon EC2::LaunchTemplate資源Version的 LaunchTemplateId和 。這個範例也會取得在相同範本中其他位置宣告的 DesiredCapacity、MaxSize、MinSize 和 VPCZoneIdentifier 值。

   ```
   NodeGroup:
     Type: "AWS::AutoScaling::AutoScalingGroup"
     Properties:
       DesiredCapacity: !Ref NodeAutoScalingGroupDesiredCapacity
       LaunchTemplate:
         LaunchTemplateId: !Ref NodeLaunchTemplate
         Version: !GetAtt NodeLaunchTemplate.LatestVersionNumber
       MaxSize: !Ref NodeAutoScalingGroupMaxSize
       MinSize: !Ref NodeAutoScalingGroupMinSize
       VPCZoneIdentifier: !Ref Subnets
       Tags:
         - Key: Name
           PropagateAtLaunch: true
           Value: !Sub ${ClusterName}-${NodeGroupName}-Node
         - Key: !Sub kubernetes.io/cluster/${ClusterName}
           PropagateAtLaunch: true
           Value: owned
   ```

3. 成功建立節點群組之後，請務必記錄所建立節點群組的 NodeInstanceRole。您需要此項目，以確保在擴展節點群組時，新節點會加入叢集，且 Kubernetes 能夠辨識節點。如需詳細資訊，請參閱建立自我管理 Amazon Linux 節點中 AWS Management Console 的指示。

4. 建議您根據容量區塊的保留時間，為 Auto Scaling 群組建立排程擴展政策。如需詳細資訊，請參閱《Amazon EC2 Auto Scaling 使用者指南》中的為 Amazon EC2 Auto Scaling 排程擴展。

   在容量區塊結束時間的 30 分鐘前，您保留的所有執行個體都可以使用。屆時仍在執行的執行個體將會開始終止。為了能有足夠的時間正常耗盡節點，建議您在容量區塊保留結束前預留 30 分鐘以上，排程擴展至零。

   如果您想要在容量保留變成 時改為手動擴展Active，則需要在容量區塊保留開始時更新 Auto Scaling 群組所需的容量。然後，您還需要在容量區塊保留結束前預留 30 分鐘以上手動縮減規模。

5. 節點群組現在已準備好排程工作負載和 Pod。

6. 為了讓您的 Pod 正常耗盡，我們建議您設定 AWS Node Termination Handler。此處理常式將能夠使用 EventBridge 從 Amazon EC2 Auto Scaling 監看「ASG 向內擴展」生命週期事件，並允許 Kubernetes 控制平面在執行個體無法使用之前採取必要的動作。否則，您的 Pod 和 Kubernetes 物件將卡在待定狀態。如需詳細資訊，請參閱 GitHub 上的 AWS 節點終止處理常式。

   如果您未設定節點終止處理常式，建議您先開始手動耗盡 Pod，然後再達到 30 分鐘的時段，讓它們有足夠的時間可以正常耗盡。

針對 AI/ML 工作負載最佳化 Amazon EKS 叢集的配方

本節旨在提供位元大小的配方來最佳化 Amazon EKS 叢集，尤其是涉及特殊硬體的 AI/ML 工作負載。您可以透過將污點新增至受管節點群組，找到防止在特定節點上排程 Pod 的指引，包括先決條件、step-by-step程序和部署考量。

主題

- 配方：防止在特定節點上排程 Pod

配方：防止在特定節點上排程 Pod

概觀

與標準機器上的節點相比，具有 GPUs 等專用處理器的節點執行成本更高。若要保護這些節點免於不需要特殊硬體的工作負載，您可以使用 Kubernetes 污點。污點標記節點以拒絕沒有相符公差的 Pod，確保只排程相容的工作負載。如需詳細資訊，請參閱 Kubernetes 文件中的污點和容錯。

Kubernetes 節點污點可以使用 AWS Management Console 或透過 Amazon EKS API 套用至新的和現有的受管節點群組。此配方說明如何使用 CLI 將污點套用至 Amazon EKS AWS 受管節點群組。如需使用 建立具有污點的節點群組的資訊 AWS Management Console，請參閱 the section called “建立”。

先決條件

- 現有的 Amazon EKS 叢集。
- AWS CLI 已安裝並設定適當的許可。

步驟

步驟 1：建立具有污點的節點群組

使用 `aws eks create-nodegroup`命令建立具有污點的新受管節點群組。此範例會套用具有索引鍵 `dedicated`、值 `gpuGroup`和效果 的污點`NO_SCHEDULE`。

```
aws eks create-nodegroup \
 --cli-input-json '
{
  "clusterName": "my-cluster",
  "nodegroupName": "node-taints-example",
  "subnets": [
     "subnet-1234567890abcdef0",
     "subnet-abcdef01234567890",
     "subnet-021345abcdef67890"
   ],
  "nodeRole": "arn:aws: iam::111122223333:role/AmazonEKSNodeRole",
  "taints": [
     {
         "key": "dedicated",
         "value": "gpuGroup",
         "effect": "NO_SCHEDULE"
     }
   ]
}'
```

如需詳細資訊和範例，請參閱 Kubernetes 參考文件中的污點。

步驟 2：更新現有節點群組上的污點

使用 aws eks update-nodegroup-config AWS CLI 命令來新增、移除或取代受管節點群組的污點。

```
aws eks update-nodegroup-config
  --cluster-name my-cluster
  --nodegroup-name node-taints-example
  --taints 'removeTaints=[{key=dedicated,value=gpuGroup,effect=NO_SCHEDULE}]'
```

備註

- 污點可以在使用 `UpdateNodegroupConfig` API 建立節點群組後更新。

- 污點金鑰必須以字母或數字開頭。可包含字母、數字、連字號 (-)、句點 (.) 及底線 (_)。長度上限為 63 個字元。
- 或者，污點金鑰可以使用 DNS 子網域字首和單一 / 開頭。如果其以 DNS 子網域字首開頭，則長度可為 253 個字元。
- 值是選用的，必須以字母或數字開頭。可包含字母、數字、連字號 (-)、句點 (.) 及底線 (_)。長度上限為 63 個字元。
- 直接使用 Kubernetes 或 時 AWS Management Console，污點效果必須是 `NoSchedule`、`PreferNoSchedule`或 `NoExecute`。不過，使用 AWS CLI 或 API 時，污點效果必須是 `NO_SCHEDULE`、 `PREFER_NO_SCHEDULE`或 `NO_EXECUTE`。
- 每個節點群組最多允許 50 個污點。
- 如果使用受管節點群組建立的污點是從節點手動移除，則 Amazon EKS 不會將污點新增至節點。即使在受管節點群組組態中指定了污點，也是如此。

在 Amazon EKS 上開始使用 AI/ML 的資源

若要開始使用 Machine Learning on EKS，請先從這些規範模式中選擇，以快速取得 EKS 叢集和 ML 軟體和硬體，準備好開始執行 ML 工作負載。

研討會

Amazon EKS 上的生成式 AI 研討會

了解如何在 Amazon EKS 上開始使用大型語言模型 (LLM) 應用程式和推論。探索如何部署和管理生產級 LLM 工作負載。透過實作實驗室，您將探索如何利用 Amazon EKS 以及 AWS 服務和開放原始碼工具來建立強大的 LLM 解決方案。研討會環境提供所有必要的基礎設施和工具，可讓您專注於學習和實作。

使用 Neuron 在 Amazon EKS 上生成 AI

了解如何在 Amazon EKS 上開始使用大型語言模型 (LLM) 應用程式和推論。探索如何部署和管理生產級 LLM 工作負載、使用向量資料庫實作進階 RAG 模式，以及使用開放原始碼架構建置資料支援的 LLM 應用程式。透過實作實驗室，您將探索如何利用 Amazon EKS 以及 AWS 服務和開放原始碼工具來建立強大的 LLM 解決方案。研討會環境提供所有必要的基礎設施和工具，可讓您專注於學習和實作。

最佳實務

Amazon EKS 最佳實務指南中的 AI/ML 重點主題提供下列領域的詳細建議，以最佳化 Amazon EKS 上的 AI/ML 工作負載。

AI/ML 運算和自動擴展

本節概述在 Amazon EKS 中最佳化 AI/ML 運算和自動擴展的最佳實務，著重於 GPU 資源管理、節點彈性和應用程式擴展。它提供策略，例如使用已知的標籤和節點親和性來排程工作負載、使用 ML 容量區塊或隨需容量保留，以及使用 EKS 節點監控代理程式等工具實作節點運作狀態檢查。

AI/ML 網路

本節概述在 Amazon EKS 中最佳化 AI/ML 聯網以增強效能和可擴展性的最佳實務，包括選擇具有較高網路頻寬或 Elastic Fabric Adapter (EFA) 的執行個體以進行分散式訓練、安裝 MPI 和 NCCL 等工具，以及啟用字首委派以增加 IP 地址並改善 Pod 啟動時間等策略。

AI/ML 安全性

本節著重於保護資料儲存的安全並確保 Amazon EKS 上 AI/ML 工作負載的合規性，包括使用 Amazon S3 搭配 AWS Key Management Service (KMS) 進行伺服器端加密 (SSE-KMS)、使用區域 KMS 金鑰和 S3 儲存貯體金鑰設定儲存貯體以降低成本、授予 IAM 許可給 EKS Pod 解密等 KMS 動作，以及使用 AWS CloudTrail 日誌稽核。

AI/ML 儲存體

本節提供最佳化 Amazon EKS 上 AI/ML 工作負載中儲存體的最佳實務，包括使用 CSI 驅動程式部署模型以掛載 S3、FSx for Lustre 或 EFS 等服務作為持久性磁碟區、根據工作負載需求選取儲存體（例如 FSx for Lustre 用於使用 Scratch-SSD 或 Persistent-SSD 等選項進行分散式訓練），以及啟用資料壓縮和分割等功能。

AI/ML 可觀測性

本節著重於監控和最佳化 Amazon EKS 上 AI/ML 工作負載的 GPU 使用率，以提高效率和降低成本，包括策略，例如使用 CloudWatch Container Insights 和與 Prometheus 和 Grafana 整合的 NVIDIA DCGM-Exporter 等工具鎖定高 GPU 使用率，以及我們建議您分析 AI/ML 工作負載的指標。

AI/ML 效能

本節著重於透過容器映像管理和啟動最佳化，增強 Amazon EKS 上 AI/ML 工作負載的應用程式擴展和效能，包括使用小型輕量型基礎映像或具有多階段建置的 AWS 深度學習容器、透過 EBS 快照預先載入映像，或使用 DaemonSets 或 部署預先提取至執行階段快取等實務。

參考架構

探索這些 GitHub 儲存庫的參考架構、範例程式碼和公用程式，以在 Amazon EKS 和其他 AWS 服務上實作 AI/ML 工作負載的分散式訓練和推論。

AWSome 分散式訓練

此儲存庫提供用於訓練大型模型的最佳實務、參考架構、模型訓練範例和公用程式的集合 AWS。它支援 Amazon EKS 的分散式訓練，包括適用於 EKS 叢集的 CloudFormation 範本、自訂 AMI 和容器建置、適用於 PyTorch (DDP/FSDP、MegatronLM、NeMo) 和 JAX 等架構的測試案例，以及用於驗證、可觀測性和效能監控的工具，例如 EFA Prometheus 匯出程式和 Nvidia Nsight Systems。

AWSome 推論

此儲存庫提供參考架構和測試案例，以最佳化 上的推論解決方案 AWS，並著重於 Amazon EKS 和加速 EC2 執行個體。它包含 VPC 和 EKS 叢集的基礎設施設定、NVIDIA NIMs、TensorRT-LLM、Triton Inference Server 和 RayService 等架構的專案，以及 Llama3-8B 和 Llama 3.1 405B 等模型的範例。具有使用 K8s LeaderWorkerSet、EKS Autoscaling、多執行個體 GPUs (MIG) 的多節點部署，以及 ASR、推論和 TTS 的音訊機器人等實際使用案例。

教學課程

如果您有興趣在 EKS 中設定Machine Learning平台和架構，請探索本節所述的教學課程。這些教學課程涵蓋從善用 GPU 處理器到選擇建模工具，再到為專業產業建置架構的各種模式。

在 EKS 上建置生成式 AI 平台

- 在 Amazon EKS 上部署生成式 AI 模型
- 在 Amazon EKS 上建置多租戶 JupyterHub 平台

在 EKS 上執行專用生成式 AI 架構

- 使用 Amazon EKS 上的 NVIDIA NeMo Framework 加速生成式 AI 分散式訓練工作負載

- 在 Amazon Elastic Kubernetes Service 上執行 TorchServe

最大化 ML on EKS 的 NVIDIA GPU 效能

- 實作 GPU 共用，以有效率地為 EKS 叢集使用 NVIDIA GPUs：

  使用 NVIDIA 時間分割和加速 EC2 執行個體在 Amazon EKS 上共用 GPU
- 使用多執行個體 GPUs (MIGs) 和 NIM 微服務，在 EKS 叢集上執行每個 GPU 的更多 Pod：

  在 Amazon EKS 上使用 NVIDIA 的多執行個體 GPU (MIG) 將 GPU 使用率最大化：每個 GPU 執行更多 Pod 以提高效能
- 在 Kubernetes 上使用 Kubeflow 建置和部署可擴展的機器學習系統 AWS

在 EKS 上執行影片編碼工作負載

- 在 Amazon EKS 的容器中使用小數 GPUs 交付影片內容

加速推論工作負載的影像載入

- H2O.ai 如何使用 Karpenter 和 Bottlerocket 最佳化和保護其 AI/ML 基礎設施

監控 ML 工作負載

- 使用 AWS 受管開放原始碼服務監控 Amazon EKS 上的 GPU 工作負載
- 在 Amazon CloudWatch 中啟用 Pod 型 GPU 指標

Amazon EKS 上的版本控制

本節旨在協助您了解 Kubernetes 版本控制如何在 Amazon Elastic Kubernetes Service (EKS) 中運作。您可以找到支援版本的詳細資訊、我們的棄用和支援政策、升級程序，以及管理叢集版本的重要考量。

主題

- 了解 EKS 上的 Kubernetes 版本生命週期
- 檢視每個 Kubernetes 版本的 Amazon EKS 平台版本

了解 EKS 上的 Kubernetes 版本生命週期

Kubernetes 會隨著新功能、設計更新和錯誤修正而快速發展。社群平均每四個月發行一次新的 Kubernetes 次要版本 （例如 1.33)。Amazon EKS 遵循次要版本的上游發行和停用週期。由於 Amazon EKS 會提供新的 Kubernetes 版本，所以我們建議您主動更新您的叢集，以便使用最新的可用版本。

次要版本在 Amazon EKS 發行後的前 14 個月內受到標準支援。一旦版本超過標準支援結束日期，就會進入未來 12 個月的延伸支援。延伸支援可讓您在特定 Kubernetes 版本中停留更長的時間，每個叢集小時需支付額外費用。如果您在延長支援期間結束之前尚未更新叢集，您的叢集會自動升級至最舊的目前支援的延伸版本。

預設會啟用延伸支援。若要停用，請參閱停用 EKS 延伸支援。

建議您使用 Amazon EKS 支援的最新可用 Kubernetes 版本來建立叢集。如果您的應用程式需要特定版本的 Kubernetes，您可以選取較舊的版本。您可以在標準或延長支援中提供的任何版本建立新的 Amazon EKS 叢集。

標準支援的可用版本

Amazon EKS 標準支援目前提供下列 Kubernetes 版本：

- 1.33
- 1.32
- 1.31
- 1.30

如需了解標準支援中每個版本的重要變更，請參閱 Kubernetes 版本標準支援。

延長支援的可用版本

Amazon EKS 延伸支援目前提供下列 Kubernetes 版本：

- 1.29
- 1.28
- 1.27

如需了解延伸支援中每個版本的重要變更，請參閱 Kubernetes 版本延伸支援。

Amazon EKS Kubernetes 發佈日曆

下表顯示每個 Kubernetes 版本應考慮的重要版本和支援日期。在 UTC+0 時區中，延伸支援的費用從版本達到標準支援結束的那一天開始。下表中的日期使用 UTC+0 時區。

> **Note**
>
> 只有月份和年份的日期是近似值，並會在已知確切日期時進行更新。

若要接收此特定文件頁面所有來源檔案變更的通知，您可以使用 RSS 讀取器訂閱下列 URL：

```
https://github.com/awsdocs/amazon-eks-user-guide/commits/mainline/latest/ug/clusters/kubernetes-versions.adoc.atom
```

Kubernetes 版本	上游發佈	Amazon EKS 發佈	標準支援結束	延長支援結束
1.33	2025 年 4 月 23 日	2025 年 5 月 29 日	2026 年 7 月 29 日	2027 年 7 月 29 日
1.32	2024 年 12 月 11 日	2025 年 1 月 23 日	2026 年 3 月 23 日	2027 年 3 月 23 日
1.31	2024 年 8 月 13 日	2024 年 9 月 26 日	2025 年 11 月 26 日	2026 年 11 月 26 日

Kubernetes 版本	上游發佈	Amazon EKS 發佈	標準支援結束	延長支援結束
1.30	2024 年 4 月 17 日	2024 年 5 月 23 日	2025 年 7 月 23 日	2026 年 7 月 23 日
1.29	2023 年 12 月 13 日	2024 年 1 月 23 日	2025 年 3 月 23 日	2026 年 3 月 23 日
1.28	2023 年 8 月 15 日	2023 年 9 月 26 日	2024 年 11 月 26 日	2025 年 11 月 26 日
1.27	2023 年 4 月 11 日	2023 年 5 月 24 日	2024 年 7 月 24 日	2025 年 7 月 24 日

使用 CLI AWS 取得版本資訊

您可以使用 AWS CLI 來取得 EKS 上可用 Kubernetes 版本的相關資訊，例如標準支援的結束日期。

使用 CLI 擷取 EKS 上可用 Kubernetes AWS 版本的相關資訊

1. 開啟終端機。
2. 確保您已安裝並設定 AWS CLI。如需詳細資訊，請參閱安裝或更新至最新版本的 CLI。
3. 執行以下命令：

```
aws eks describe-cluster-versions
```

4. 命令會傳回 JSON 輸出，其中包含可用叢集版本的詳細資訊。以下是輸出的範例：

```
{
    "clusterVersions": [
        {
            "clusterVersion": "1.31",
            "clusterType": "eks",
            "defaultPlatformVersion": "eks.21",
            "defaultVersion": true,
            "releaseDate": "2024-09-25T17:00:00-07:00",
            "endOfStandardSupportDate": "2025-11-25T16:00:00-08:00",
            "endOfExtendedSupportDate": "2026-11-25T16:00:00-08:00",
```

```
            "status": "STANDARD_SUPPORT",
            "kubernetesPatchVersion": "1.31.3"
        }
    ]
}
```

輸出會為每個叢集版本提供下列資訊：

- `clusterVersion`：EKS 叢集的 Kubernetes 版本
- `clusterType`：叢集的類型 （ 例如 "eks")
- `defaultPlatformVersion`：預設 EKS 平台版本
- `defaultVersion`：這是否為預設版本
- `releaseDate`：此版本發佈的日期
- `endOfStandardSupportDate`：標準支援結束的日期
- `endOfExtendedSupportDate`：延伸支援結束的日期
- `status`： 版本的目前支援狀態，例如 `STANDARD_SUPPORT`或 `EXTENDED_SUPPORT`
- `kubernetesPatchVersion`：特定 Kubernetes 修補程式版本

Amazon EKS 版常見問題

標準支援提供多少 Kubernetes 版本？

Amazon EKS 與 Kubernetes 版本的 Kubernetes 社群支援一致，致力於在任何指定時間提供至少三個 Kubernetes 版本的支援。我們將至少提前 60 天宣布特定 Kubernetes 次要版本的標準支援結束日期。由於新 Kubernetes 版本的 Amazon EKS 資格和發行程序，Amazon EKS 上 Kubernetes 版本的標準支援結束日期將在 Kubernetes 專案停止支援上游版本的日期之後。

Kubernetes 獲得 Amazon EKS 標準支援的時間有多長？

Kubernetes 版本在首次在 Amazon EKS 上提供後 14 個月內獲得標準支援。即使上游 Kubernetes 不再支援 Amazon EKS 上可用的版本，也是如此。我們向後移植適用於 Amazon EKS 上支援的 Kubernetes 版本的安全修補程式。

當 Amazon EKS 上的 Kubernetes 版本的標準支援結束時，是否會通知我？

是。如果您帳戶中的任何叢集正在執行即將終止支援的版本，Amazon EKS 會在 Amazon EKS 上發行 Kubernetes 版本後約 12 個月，透過 AWS 運作狀態儀表板傳送通知。此通知包括支援終止的日期。此日期距通知發出日期起至少 60 天。

Amazon EKS 支援哪些 Kubernetes 功能？

Amazon EKS 支援 Kubernetes API 的所有一般可用 (GA) 功能。根據預設，叢集中不會啟用新的 Beta APIs。然而，依預設，仍會繼續啟用先前的 Beta API 和新版本的現有 Beta API。不支援 Alpha 功能。

Amazon EKS 受管節點群組是否與叢集控制平面版本一起自動更新？

不受管理節點群組會在您的帳戶建立 Amazon EC2 執行個體。當您或 Amazon EKS 更新控制平面時，這些執行個體不會自動升級。如需詳細資訊，請參閱the section called “更新”。建議您在控制平面和節點上維持相同的 Kubernetes 版本。

自我管理節點群組是否與叢集控制平面版本一起自動更新？

不會。自我管理的節點群組包含您帳戶的 Amazon EC2 執行個體。當您或 Amazon EKS 代表您更新控制平面版本時，這些執行個體不會自動升級。自我管理節點群組在主控台中沒有任何需要更新的指示。您可以檢視安裝在節點上的 `kubelet` 版本，方法是選取您叢集的 Overview (概觀) 標籤上的 Nodes (節點) 清單上的節點，以判斷哪些節點需要更新。您必須手動更新節點。如需詳細資訊，請參閱the section called “更新方法”。

Kubernetes 專案會測試控制平面與節點之間的相容性，最多可達三個次要版本。例如，協調工作是由 `1.33` 控制平面進行時，`1.30` 節點將可繼續操作。不過，不建議在控制平面後方持續執行具有三個次要版本的節點叢集。如需詳細資訊，請參閱 Kubernetes 文件中的 Kubernetes 版本和版本偏移支援政策。建議您在控制平面和節點上維持相同的 Kubernetes 版本。

在 Fargate 上執行的 Pod 是否透過自動叢集控制平面版本升級自動升級？

否。我們強烈建議執行 Fargate Pods 作為複寫控制器的一部分，例如 Kubernetes 部署。然後，對所有 Fargate Pod 執行滾動重新啟動。新版本的 Fargate Pod 部署的`kubelet`版本與您更新的叢集控制平面版本相同。如需詳細資訊，請參閱 Kubernetes 文件中的部署。

> **Important**
>
> 如果您更新控制平面，您仍必須自行更新 Fargate 節點。若要更新 Fargate 節點，請刪除節點代表的 Fargate Pod，並重新部署 Pod。新的 Pod 部署的`kubelet`版本與叢集的版本相同。

混合節點支援哪些 Kubernetes 版本？

Amazon EKS 混合節點支援與其他節點運算類型的 Amazon EKS 叢集相同的 Kubernetes 版本，包括標準和擴充的 Kubernetes 版本支援。當您升級控制平面版本時，混合節點不會自動升級，而且您必須負責升級混合節點。如需詳細資訊，請參閱the section called “升級混合節點”。

Amazon EKS 延伸支援FAQs

標準支援和延伸支援術語對我來說是新的。這些術語是什麼意思？

Amazon EKS 中 Kubernetes 版本的標準支援會在 Amazon EKS 上發行 Kubernetes 版本時開始，並在發行日期後 14 個月結束。Kubernetes 版本的延伸支援將在標準支援結束後立即開始，並在接下來的 12 個月後結束。例如，Amazon EKS 1.23版本的標準支援已於 2023 年 10 月 11 日結束。延長對 版本的支援從 2023 年 10 月 12 1.23日開始，並於 2024 年 10 月 11 日結束。

我需要做什麼才能取得 Amazon EKS 叢集的延伸支援？

您需要將叢集升級政策變更為 EXTENDED，以啟用叢集的延伸支援 （ 請參閱 EKS 延伸支援)。根據預設，除非另有指定，否則對於所有新的和現有的叢集，升級政策都會設為 EXTENDED。請參閱叢集升級政策以檢視叢集的升級政策。標準支援會在 Amazon EKS 上發行 Kubernetes 版本時開始，並在發行日期後 14 個月結束。Kubernetes 版本的延伸支援將在標準支援結束後立即開始，並在接下來的 12 個月後結束。

哪些 Kubernetes 版本可以獲得延伸支援？

在該版本的標準支援結束後，您可以在任何版本執行叢集，長達 12 個月。這代表每個版本將在 Amazon EKS 支援 26 個月（ 14 個月的標準支援加 12 個月的延長支援）。

如果我不想使用延伸支援該怎麼辦？

如果您不想自動註冊延長支援，您可以將叢集升級至標準 Amazon EKS 支援中的 Kubernetes 版本。若要停用延伸支援，請參閱停用 EKS 延伸支援。注意：如果您停用延伸支援，您的叢集將在標準支援結束時自動升級。

延長支援 12 個月結束時會發生什麼情況？

在 Kubernetes 版本上執行且已完成 26 個月生命週期 (14 個月標準支援加上 12 個月延長支援） 的叢集將自動升級至下一個版本。自動升級僅包含 Kubernetes 控制平面。如果您有 EKS Auto Mode 節點，它們可能會自動更新。自我管理節點和 EKS 受管節點群組將保留在先前的版本中。

在延長支援日期結束時，您就無法再使用不支援的版本來建立新的 Amazon EKS 叢集。現有的控制平面會在終止支援日期後由 Amazon EKS 透過逐步部署程序，自動將控制平面更新為最舊的支援版

本。自動更新控制平面後，請務必手動更新叢集附加元件和 Amazon EC2 節點。如需詳細資訊，請參閱the section called “更新 Kubernetes 版本”。

延長支援日期結束後，我的控制平面何時會自動更新？

Amazon EKS 無法提供特定的時間範圍。於延長支援日期結束後，可隨時進行自動更新。在更新之前，您不會收到任何通知。我們建議您主動更新控制平面，而不需依賴 Amazon EKS 自動更新程序。如需詳細資訊，請參閱the section called “更新 Kubernetes 版本”。

我可以無限期地將控制平面保留在 Kubernetes 版本上嗎？

否。 的雲端安全性 AWS 是最高優先順序。過去某個時間點（通常是一年），Kubernetes 社群會停止發佈常見的漏洞和暴露 (CVE) 修補程式，並阻止 CVE 提交不支援的版本。這表示可能無法報告舊版 Kubernetes 特有的漏洞。讓叢集暴露於漏洞之下，不會發出通知，也沒有修復選項。因此，Amazon EKS 不允許控制平面保持在已結束延伸支援的版本上。

取得延長支援是否需要額外費用？

是，在延伸支援中執行的 Amazon EKS 叢集需支付額外費用。如需定價詳細資訊，請參閱 AWS 部落格或我們的定價頁面上 Amazon EKS 擴充支援 Kubernetes 版本定價。

延伸支援包含哪些內容？

延伸支援中的 Amazon EKS 叢集會收到 Kubernetes 控制平面的持續安全性修補程式。此外，Amazon EKS 將發行延伸支援版本的 Amazon VPC CNI、 `kube-proxy`和 CoreDNS 附加元件修補程式。Amazon EKS 也會針對 AWS Amazon Linux、 Bottlerocket 和 Windows 以及這些版本的 Amazon EKS Fargate 節點，發行已發佈 Amazon EKS 最佳化 AMIs 的修補程式。延伸支援中的所有叢集將繼續從 存取技術支援 AWS。

延伸支援中非 Kubernetes 元件的修補程式是否有任何限制？

雖然延伸支援涵蓋來自 的所有 Kubernetes 特定元件 AWS，但它將始終僅支援 AWS針對 Amazon Linux、 Bottlerocket 和 Windows 發佈的 Amazon EKS 最佳化 AMIs。這代表您在使用延長支援時，您可能會在 Amazon EKS 最佳化 AMI 擁有更新的元件 (例如作業系統或核心)。例如，一旦 Amazon Linux 2 在 2025 年達到生命週期結束，Amazon EKS 最佳化的 Amazon Linux AMI 將使用更新的 Amazon Linux 作業系統構建。Amazon EKS 將針對每個 Kubernetes 版本宣布並記錄重要的支援生命週期差異，例如此差異。

我可以在延伸支援上使用 版本建立新的叢集嗎？

是。

在標準支援上檢閱 Kubernetes 版本的版本備註

本主題提供標準支援中每個 Kubernetes 版本需要注意的重要變更。於升級時，請仔細檢閱叢集舊版和新版本之間發生的變更。

Kubernetes 1.33

Kubernetes `1.33` 現在可在 Amazon EKS 中使用。如需 Kubernetes 的詳細資訊`1.33`，請參閱官方發行公告。

> **Important**
>
> - 動態資源配置 Beta Kubernetes API 已啟用。
>   - 此 Beta API 可改善排程和監控需要 GPUs 等資源之工作負載的體驗。
>   - Beta API 由 Kubernetes 社群定義，未來 Kubernetes 版本可能會變更。
>   - 仔細檢閱 Kubernetes 文件中的功能階段，以了解使用 Beta APIs的影響。
> - AWS 不會發佈適用於 Kubernetes 1.33 的 EKS 最佳化 Amazon Linux 2 AMI。
>   - AWS 鼓勵您遷移至 Amazon Linux 2023。了解如何 the section called “升級至 AL2023”。
>   - 如需詳細資訊，請參閱the section called “Amazon Linux 2 AMI 棄用”。

- 就地 Pod 資源調整大小 (Beta)：就地資源調整大小已提升為 Beta 版，允許對現有 Pod 的 CPU 和記憶體資源進行動態更新，而無需重新啟動 - 啟用狀態工作負載的垂直擴展，無需停機，並根據流量模式無縫調整資源。
- Sidecar Containers Now Stable： Sidecar 容器已進入穩定狀態，`restartPolicy: Always`在應用程式容器之前實作附屬做為特殊初始化容器，在整個 Pod 生命週期中執行，並支援用於操作狀態訊號的探查。
  - 如需詳細資訊，請參閱 Kubernetes 文件中的附屬容器。
- 端點 API 棄用：端點 API 現在已正式棄用，並在存取時傳回警告 - 遷移工作負載和指令碼以改用 EndpointSlices API，這支援雙堆疊聯網等現代功能，並處理每個服務的多個 EndpointSlices。
  - 如需詳細資訊，請參閱 Kubernetes 部落格上的 Kubernetes v1.33：繼續從端點轉換至 EndpointSlice。
- Elastic Fabric Adapter Support：Amazon EKS 叢集的預設安全群組現在支援 Elastic Fabric Adapter (EFA) 流量。預設安全群組具有新的傳出規則，允許目的地為相同安全群組的 EFA 流量。這允許叢集內的 EFA 流量。

- 如需詳細資訊，請參閱《Amazon Elastic Compute Cloud 使用者指南》中的適用於 Amazon EC2 上的 AI/ML 和 HPC 工作負載的 Elastic Fabric Adapter。

如需完整的 Kubernetes 1.33變更日誌，請參閱 https：//https://github.com/kubernetes/kubernetes/blob/master/CHANGELOG/CHANGELOG-1.33.md

Kubernetes 1.32

Kubernetes 1.32 現在可在 Amazon EKS 中使用。如需 Kubernetes 的詳細資訊1.32，請參閱官方發行公告。

> **Important**
>
> - FlowSchema 和 PriorityLevelConfiguration 的 `flowcontrol.apiserver.k8s.io/v1beta3` API 版本已在版本 中移除1.32。如果您使用這些 APIs，則必須在升級之前更新您的組態以使用最新的支援版本。
> - ServiceAccount `metadata.annotations[kubernetes.io/enforce-mountable-secrets]` 已在版本 中棄用1.32，並將在未來的 Kubernetes 次要版本版本中移除。建議使用不同的命名空間來隔離對掛載秘密的存取。
> - Kubernetes 版本1.32是 Amazon EKS 將發行 Amazon Linux 2 (AL2) AMIs最後一個版本。從版本1.33開始，Amazon EKS 將繼續發行 Amazon Linux 2023 (AL2023) 和 Bottlerocket 型 AMIs。

- Memory Manager 功能已在 Kubernetes 版本 中進入「一般可用 (GA)」狀態1.32。此增強功能可為容器化應用程式提供更有效率且可預測的記憶體配置，尤其適用於具有特定記憶體需求的工作負載。
- StatefulSets 建立的 PersistentVolumeClaims (PVCs) 現在包含自動清除功能。當不再需要 PVCs 時，系統會自動刪除它們，同時在 StatefulSet 更新和節點維護操作期間維持資料持久性。此功能可簡化儲存管理，並有助於防止叢集中的孤立 PVCs。
- 自訂資源欄位選取器功能已推出，可讓開發人員將欄位選取器新增至自訂資源。此功能為內建 Kubernetes 物件提供與自訂資源相同的篩選功能，可更精確、更有效率地篩選資源，並提升更好的 API 設計實務。

如需完整的 Kubernetes 1.32變更日誌，請參閱 https：//https://github.com/kubernetes/kubernetes/blob/master/CHANGELOG/CHANGELOG-1.32.md

匿名身分驗證變更

從 Amazon EKS 開始1.32，匿名身分驗證僅限於下列 API 伺服器運作狀態檢查端點：

- /healthz
- /livez
- /readyz

使用 system:unauthenticated 使用者對任何其他端點的請求將會收到 401 Unauthorized HTTP 回應。此安全增強功能有助於防止因設定錯誤的 RBAC 政策而可能發生的意外叢集存取。

> **Note**
>
> RBAC public-info-viewer 角色會繼續套用至上述運作狀態檢查端點。

Amazon Linux 2 AMI 棄用

對於 Kubernetes 1.33 版和更新版本，EKS 不會提供預先建置的最佳化 Amazon Linux 2 (AL2) Amazon Machine Image (AMIs)。

AWS 建議採用 EKS Auto 模式，或遷移至較新的作業系統，例如 Amazon Linux 2023 (AL2023) 或 Bottlerocket。

- the section called “從 MNGs遷移”
- the section called “升級至 AL2023”
- the section called “Bottlerocket”

> **Note**
>
> 此更新適用於 EKS 最佳化 AL2 AMIs。如需作業系統本身的詳細資訊，請參閱 Amazon Linux 2 FAQs。

Kubernetes 1.31

Kubernetes 1.31 現在可在 Amazon EKS 中使用。如需 Kubernetes 的詳細資訊1.31，請參閱官方發行公告。

> **Important**
>
> - 自 2017 以來已`--keep-terminated-pod-volumes`棄用的 kubelet 旗標已移除，做為版本`1.31`發行的一部分。此變更會影響 kubelet 如何處理終止的 Pod 磁碟區。如果您在節點組態中使用此標記，則必須更新引導指令碼和啟動範本，以在升級之前將其移除。

- 在 Amazon EKS 版本 中啟用 Beta `VolumeAttributesClass`功能閘道和 API 資源`1.31`。此功能允許叢集運算子修改由相容 CSI 驅動程式管理的持久性磁碟區 (PVs) 的可變屬性，包括 Amazon EBS CSI 驅動程式。若要利用此功能，請確定您的 CSI 驅動程式支援`VolumeAttributesClass`此功能 （對於 Amazon EBS CSI 驅動程式，請升級至 版本`1.35.0`或更新版本以自動啟用此功能）。您將能夠建立`VolumeAttributesClass`物件來定義所需的磁碟區屬性，例如磁碟區類型和輸送量，並將其與您的持久性磁碟區宣告 (PVCs建立關聯。如需詳細資訊，請參閱官方 Kubernetes 文件以及 CSI 驅動程式的文件。
  - 如需 Amazon EBS CSI 驅動程式的詳細資訊，請參閱 the section called “Amazon EBS”。
- AppArmor 的 Kubernetes 支援已進入穩定狀態，現已正式推出供大眾使用。此功能可讓您在容器的中設定 `appArmorProfile.type` 欄位，以使用 AppArmor 保護您的容器`securityContext`。在 Kubernetes 版本 之前`1.30`，AppArmor 是由註釋控制。從版本 開始`1.30`，它使用 欄位進行控制。若要利用此功能，建議您從註釋遷移並使用 `appArmorProfile.type` 欄位來確保您的工作負載相容。
- PersistentVolume 上次階段轉換時間功能已進入穩定狀態，現已正式在 Kubernetes 版本 中公開使用`1.31`。此功能會在 PersistentVolumeStatus `.status.lastTransitionTime`中引入新欄位 ，提供 PersistentVolume 上次轉換至不同階段時的時間戳記。此增強功能可更好地追蹤和管理 PersistentVolumes，尤其是在了解磁碟區生命週期很重要的情況下。

如需完整的 Kubernetes `1.31`變更日誌，請參閱 https：//https://github.com/kubernetes/kubernetes/blob/master/CHANGELOG/CHANGELOG-1.31.md

Kubernetes 1.30

Kubernetes `1.30` 現在可在 Amazon EKS 中使用。如需 Kubernetes 的詳細資訊`1.30`，請參閱官方版本公告。

> **Important**
>
> - 從 Amazon EKS 版本`1.30`或更新版本開始，任何新建立的受管節點群組都會自動預設為使用 Amazon Linux 2023 (AL2023) 做為節點作業系統。先前，新的節點群組預設為 Amazon Linux 2 (AL2)。您可以在建立新節點群組時選擇 AL2 做為 AMI 類型，以繼續使用它。
>   - 如需從 AL2 遷移至 AL2023 的資訊，請參閱 the section called “升級至 AL2023”。
>   - 如需 Amazon Linux 的詳細資訊，請參閱《Amazon Linux 使用者指南》中的比較 AL2 和 AL2023。
>   - 如需為受管節點群組指定作業系統的詳細資訊，請參閱 the section called “建立”。

- 使用 Amazon EKS 時`1.30`，`topology.k8s.aws/zone-id`標籤會新增至工作者節點。您可以使用可用區域 IDs(AZ IDs) 來判斷一個帳戶中資源相對於另一個帳戶中資源的位置。如需詳細資訊，請參閱《 AWS RAM 使用者指南》中的 AWS 資源的可用區域 IDs。
- 從 開始`1.30`，Amazon EKS 不再包含套用至新建立叢集之`gp2 StorageClass`資源的`default`註釋。如果您依名稱參考此儲存類別，這不會影響。如果您依賴叢集`StorageClass`中的預設值，則必須採取動作。您應該`StorageClass`使用名稱 來參考 gp2。或者，您可以在安裝 版本 `1.31.0`或更新版本時，將 `defaultStorageClass.enabled` 參數設定為 true，以部署 Amazon EBS 建議的預設儲存類別`aws-ebs-csi-driver add-on`。
- Amazon EKS 叢集 IAM 角色所需的最低 IAM 政策已變更。`ec2:DescribeAvailabilityZones` 需要 動作。如需詳細資訊，請參閱the section called “叢集 IAM 角色”。

如需完整的 Kubernetes `1.30`變更日誌，請參閱 https：//https://github.com/kubernetes/kubernetes/blob/master/CHANGELOG/CHANGELOG-1.30.md。

在延伸支援上檢閱 Kubernetes 版本的版本備註

Amazon EKS 支援比上游支援的 Kubernetes 版本更長的 Kubernetes 版本，從 Amazon EKS 發行 Kubernetes 次要版本起 14 個月的標準支援，以及額外 12 個月的 Kubernetes 次要版本延伸支援（每個版本總計 26 個月）。

本主題提供延長支援的每個 Kubernetes 版本需要注意的重要變更。於升級時，請仔細檢閱叢集舊版和新版本之間發生的變更。

Kubernetes 1.29

Kubernetes 1.29 現在可在 Amazon EKS 中使用。如需 Kubernetes 的詳細資訊1.29，請參閱官方發行公告。

> **Important**
>
> - 取代的 flowcontrol.apiserver.k8s.io/v1beta2 API 版本 FlowSchema和 PriorityLevelConfiguration 不再在 Kubernetes 版本 中提供1.29。如果您有使用已棄用 Beta API 群組的資訊清單或用戶端軟體，您應該在升級至版本 之前進行變更1.29。

- 節點物件.status.kubeProxyVersion的欄位現在已棄用，Kubernetes 專案提議在未來版本中移除該欄位。已棄用的欄位並不準確，過去是由 kubelet 管理，這實際上並不知道kube-proxy版本，甚至是否kube-proxy正在執行。如果您在用戶端軟體中使用此欄位，請停止 - 資訊不可靠且欄位現在已棄用。
- 在 Kubernetes 中1.29，為了減少潛在的攻擊面，如果長時間未使用 （預設為 1 年），LegacyServiceAccountTokenCleanUp此功能會將舊版自動產生的秘密型權杖標記為無效 （預設為 1 年），並在長時間未嘗試使用時自動將其移除。若要識別這類字符，您可以執行的 ：

```
kubectl get cm kube-apiserver-legacy-service-account-token-tracking -n kube-system
```

如需完整的 Kubernetes 1.29變更日誌，請參閱 https：//https://github.com/kubernetes/kubernetes/blob/master/CHANGELOG/CHANGELOG-1.29.md#changelog-since-v1280。

Kubernetes 1.28

Kubernetes 1.28 現在可在 Amazon EKS 中使用。如需 Kubernetes 的詳細資訊1.28，請參閱官方發行公告。

- Kubernetes 將核心節點和控制平面元件之間的支援偏斜從 v1.28擴展n-2為一個次要版本，n-3因此最舊支援的次要版本的節點元件 (kubelet 和 kube-proxy) 可以使用最新支援的次要版本的控制平面元件 (kube-apiserver、kube-scheduler、kube-controller-manager、cloud-controller-manager)。
- Pod GC Controller 的指標 force_delete_pods_total 和 force_delete_pod_errors_total 已經增強，負責所有強制 Pod 刪除。將原因新增至指標，以指出 Pod 是否因為終止、孤立、終止out-of-service的污點，或是終止和未排程而被強制刪除。

- 已修改 PersistentVolume（PV）控制器，自動將預設 StorageClass 指定給任何未設定 PersistentVolumeClaim 的未受限制 storageClassName。此外，API 伺服器內的PersistentVolumeClaim許可驗證機制已經過調整，以允許將值從未設定狀態變更為實際StorageClass名稱。

如需完整的 Kubernetes 1.28變更日誌，請參閱 https：//https://github.com/kubernetes/kubernetes/blob/master/CHANGELOG/CHANGELOG-1.28.md#changelog-since-v1270。

Kubernetes 1.27

Kubernetes 1.27 現在可在 Amazon EKS 中使用。如需 Kubernetes 的詳細資訊1.27，請參閱官方發行公告。

Important

- 對 Alpha seccomp 註釋 seccomp.security.alpha.kubernetes.io/pod 和 container.seccomp.security.alpha.kubernetes.io 註釋的支援已移除。Alpha seccomp 註釋已在 1.19 中棄用，在 1.27 中刪除後，seccomp 欄位將不再為包含 seccomp 註釋的 Pods 自動填入。請改為使用 Pods 的 securityContext.seccompProfile 欄位或容器來設定 seccomp 設定檔。若要檢查您是否在叢集中使用已棄用的 Alpha seccomp 註釋，請執行下列命令：

  ```
  kubectl get pods --all-namespaces -o json | grep
   -E 'seccomp.security.alpha.kubernetes.io/pod|
  container.seccomp.security.alpha.kubernetes.io'
  ```

- 系統已移除 kubelet 的 --container-runtime 命令行引數。Amazon EKS 的預設容器執行時間自 containerd開始1.24，因此不需要指定容器執行時間。自 1.27 起，Amazon EKS 將忽略傳遞給任何啟動程序指令碼的 --container-runtime 引數。為了防止節點引導程序期間發生錯誤，請勿將此引數傳遞給 --kubelet-extra-args 。您必須移除所有節點建立工作流程和建置指令碼中的 --container-runtime 引數。

- Kubernetes kubelet中的 將預設值1.27提高kubeAPIQPS為 50，並將預設值提高kubeAPIBurst為 100。這些增強功能允許 kubelet 處理更大量的 API 查詢，改善回應時間和效能。當 Pods 的需求增加時，由於擴展要求，修改後的預設值可確保 kubelet 能有效管理增加的工作負載。因此，Pod 啟動速度更快，且叢集作業更有效率。

- 您可以使用更精細的 Pod 拓撲來傳播政策，例如 minDomain。此參數可讓您指定 Pods 應傳播的網域數目下限。nodeAffinityPolicy 和 nodeTaintPolicy 則允許在管理 Pod 分佈中額外的細部程度。這是根據節點親和性、污點和 Pod's 規格的 topologySpreadConstraints 之中的 matchLabelKeys 欄位而定。如此可允許在滾動升級之後選取 Pods 以供分攤計算之用。
- Kubernetes 1.27提升為 Beta 版，這是StatefulSets控制其 PersistentVolumeClaims() 生命週期的新政策機制PVCs。全新的 PVC 保留政策可讓您指定，當 StatefulSet 被刪除或 StatefulSet 中的複本縮減規模時，從 StatefulSet 規格範本產生的 PVCs 是否會自動刪除或保留。
- Kubernetes API 伺服器中的 goaway-chance 選項可透過隨機關閉連線，協助防止HTTP/2用戶端連線卡在單一 API 伺服器執行個體上。如果連線關閉，用戶端會嘗試重新連線，並且可能會因負載平衡而停留在不同的 API 伺服器上。Amazon EKS 版本 1.27 已啟用 goaway-chance 旗標。如果您在 Amazon EKS 叢集上執行的工作負載使用與 HTTP GOAWAY 不相容的用戶端，建議您在連線終止時重新連線GOAWAY，以更新用戶端來處理。

如需完整的 Kubernetes 1.27變更日誌，請參閱 https：//https://github.com/kubernetes/kubernetes/blob/master/CHANGELOG/CHANGELOG-1.27.md#changelog-since-v1260。

檢視目前的叢集支援期間

AWS 主控台的叢集支援期間區段指出您的叢集目前是標準或延伸支援。如果您的叢集支援期間是延長支援，則會向您收取 EKS 延長支援的費用。

如需標準和延伸支援的詳細資訊，請參閱 eks/latest/userguide/kubernetes-versions.html【Understand the Kubernetes version lifecycle on EKS，type="documentation"】。

1. 導覽至 AWS 主控台 EKS 區段中的叢集頁面。確認主控台已設定為與您要檢閱的叢集相同的 AWS 區域。
2. 檢閱支援期間欄。如果 值在...之前為標準支援，則您目前不需要支付延長支援的費用。您處於標準支援期間。如果 值是延伸支援...此叢集目前正在支付延伸支援的費用。

> **Note**
>
> 無法使用 AWS API 或 CLI 擷取支援期間。

檢視目前的叢集升級政策

叢集升級政策會決定叢集離開標準支援期間時會發生的情況。如果您的升級政策是 EXTENDED，叢集將不會自動升級，且會進入延長支援。如果您的升級政策是 STANDARD，則會自動升級。

Kubernetes 版本政策的 Amazon EKS 控制項可讓您為 EKS 叢集選擇標準支援行為結束。透過這些控制項，您可以決定哪些叢集應輸入延伸支援，以及哪些叢集應在 Kubernetes 版本的標準支援結束時自動升級。

次要版本在發行後前 14 個月內，在 Amazon EKS 中受到標準支援。一旦版本超過標準支援結束日期，就會進入未來 12 個月的延長支援。延長支援可讓您維持在特定的 Kubernetes 版本更長的時間，每個叢集小時需支付額外費用。您可以啟用或停用 EKS 叢集的延伸支援。如果您停用延伸支援，AWS 會在標準支援結束時自動將您的叢集升級至下一個版本。如果您啟用延長支援，您可以在有限的時間內維持在目前版本，但需額外付費。即使您使用延伸支援，也請計劃定期升級 Kubernetes 叢集。

您可以使用 `supportType` 屬性，為新的和現有的叢集設定版本政策。有兩種選項可用來設定版本支援政策：

- **`STANDARD`** — 您的 EKS 叢集在標準支援結束時符合自動升級的資格。使用此設定，您不會產生延長支援費用，但 EKS 叢集會自動升級至標準支援的下一個 Kubernetes 版本。
- **`EXTENDED`** — 一旦 Kubernetes 版本達到標準支援結束，EKS 叢集將進入延伸支援。使用此設定，您將產生延長的支援費用。您可以將叢集升級至標準支援的 Kubernetes 版本，以停止產生延長支援費用。在延伸支援上執行的叢集將有資格在延伸支援結束時自動升級。

新叢集和現有叢集預設會啟用延伸支援。您可以檢視 中的叢集是否已啟用延伸支援 AWS Management Console，或使用 AWS CLI。

> **Important**
>
> 如果您希望叢集保持其目前的 Kubernetes 版本，以利用延長的支援期間，您必須在標準支援期間結束前啟用延長的支援升級政策。

您只能在叢集在標準支援中於 Kubernetes 版本上執行時，設定叢集的版本支援政策。一旦版本進入延伸支援，您將無法變更此設定，直到您在標準支援中的版本上執行。

例如，如果您已將版本支援政策設定為 `standard`，則在叢集上執行的 Kubernetes 版本達到標準支援結束之後，您將無法變更此設定。如果您已將版本支援政策設定為 `extended`，則在叢集上執行的

Kubernetes 版本達到標準支援結束之後，您將無法變更此設定。若要變更版本支援政策設定，您的叢集必須在標準支援的 Kubernetes 版本上執行。

檢視叢集升級政策AWS （主控台）

1. 導覽至 AWS 主控台 EKS 區段中的叢集頁面。確認主控台已設定為與您要檢閱的叢集相同的 AWS 區域。
2. 檢閱升級政策欄。如果值為標準支援，您的叢集將不會進入延伸支援。如果該值為延長支援，您的叢集將進入延長支援。

檢視叢集升級政策 (AWS CLI)

1. 確認已安裝 AWS CLI 且您已登入。了解如何更新和安裝 AWS CLI。
2. 判斷 EKS 叢集的名稱。將 CLI 設定為與 EKS 叢集相同的 AWS 區域。
3. 執行以下命令：

```
aws eks describe-cluster \
--name <cluster-name> \
--query "cluster.upgradePolicy.supportType"
```

4. 如果值為 STANDARD，您的叢集將不會進入延伸支援。如果值為 EXTENDED，您的叢集將進入延伸支援。

啟用 EKS 延伸支援，以增加彈性來規劃 Kubernetes 版本升級

本主題說明如何設定 EKS 叢集的升級政策以啟用延伸支援。EKS 叢集的升級政策會決定當叢集達到標準支援期間結束時會發生什麼情況。如果叢集升級政策已啟用延長支援，則會在標準支援期間結束時進入延長支援期間。叢集不會在標準支援期間結束時自動升級。

叢集實際上在延長的支援期間會產生較高的成本。如果叢集僅將升級政策設定為啟用延伸支援，且在其他情況下處於標準支援期間，則會產生標準成本。

如果您在 AWS 主控台中建立叢集，則會將升級政策設定為停用延伸支援。如果您以其他方式建立叢集，則會將升級政策設定為啟用延伸支援。例如，使用 AWS API 建立的叢集已啟用延伸支援。

如需升級政策的詳細資訊，請參閱叢集升級政策。

> **Important**
>
> 如果您希望叢集保持其目前的 Kubernetes 版本，以利用延長的支援期間，您必須在標準支援期間結束前啟用延伸支援升級政策。
> 如果您未啟用延伸支援，您的叢集將自動升級。

啟用 EKS 延伸支援AWS （主控台）

1. 在 AWS 主控台中導覽至您的 EKS 叢集。選取叢集資訊頁面上的概觀索引標籤。
2. 在 Kubernetes 版本設定區段中，選取管理。
3. 選取延伸支援，然後儲存變更。

啟用 EKS 延伸支援 (AWS CLI)

1. 確認已安裝 AWS CLI 且您已登入。了解如何更新和安裝 AWS CLI。
2. 判斷 EKS 叢集的名稱。
3. 執行以下命令：

```
aws eks update-cluster-config \
--name <cluster-name> \
--upgrade-policy supportType=EXTENDED
```

停用 EKS 延伸支援，防止增加叢集成本

本主題說明如何設定 EKS 叢集的升級政策，以停用延伸支援。EKS 叢集的升級政策會決定當叢集達到標準支援期間結束時會發生什麼情況。如果叢集升級政策已停用延伸支援，則會自動升級至下一個 Kubernetes 版本。

如需升級政策的詳細資訊，請參閱叢集升級政策。

> **Important**
>
> 一旦您的叢集已進入延伸支援，您就無法停用延伸支援。您只能在標準支援上停用叢集的延伸支援。
> AWS 建議在標準支援期間將您的叢集升級至版本。

停用 EKS 延伸支援AWS （主控台）

1. 在 AWS 主控台中導覽至您的 EKS 叢集。選取叢集資訊頁面上的概觀索引標籤。
2. 在 Kubernetes 版本設定區段中，選取管理。
3. 選取標準支援，然後儲存變更。

停用 EKS 延伸支援 (AWS CLI)

1. 確認已安裝 AWS CLI 且您已登入。了解如何更新和安裝 AWS CLI。
2. 判斷 EKS 叢集的名稱。
3. 執行以下命令：

```
aws eks update-cluster-config \
--name <cluster-name> \
--upgrade-policy supportType=STANDARD
```

檢視每個 Kubernetes 版本的 Amazon EKS 平台版本

Amazon EKS 平台版本代表 Amazon EKS 叢集控制平面的功能，例如啟用了哪些 Kubernetes API 伺服器旗標以及目前的 Kubernetes 修補程式版本。每個 Kubernetes 次要版本皆有一或多個相關的 Amazon EKS 平台版本。適用於不同 Kubernetes 次要版本的平台版本都是彼此獨立。您可以使用 CLI 或 擷取叢集目前的平台版本 AWS Management Console。 AWS 如果您在 AWS Outposts 上有本機叢集，請參閱 the section called “EKS 平台版本”而非本主題。

當 Amazon EKS 中提供新的 Kubernetes 次要版本時，例如 1.33，該 Kubernetes 次要版本的初始 Amazon EKS 平台版本會從 開始`eks.1`。但是，Amazon EKS 會定期發佈新平台版本，以啟用新的 Kubernetes 控制平面設定，以及提供安全性問題修正。

當有新的 Amazon EKS 平台版本可供次要版本使用時：

- Amazon EKS 平台版本編號將會遞增 (`eks.<n+1>`)。
- Amazon EKS 會自動將所有現有的叢集升級到與其對應之 Kubernetes 次要版本的最新 Amazon EKS 平台版本。現有 Amazon EKS 平台版本的自動升級將會逐步發佈。發行程序可能需要一些時間。如果您立刻需要最新 Amazon EKS 平台版本的功能，您應該建立新的 Amazon EKS 叢集。

如果您的叢集在目前平台版本後面有兩個以上的平台版本，則 Amazon EKS 可能無法自動更新您的叢集。如需可能導致此問題的詳細資訊，請參閱 the section called “Amazon EKS 平台版本比目前平台版本落後兩個版本以上”。

- Amazon EKS 可能會發佈具有對應之修補程式版本的新節點 AMI。但是，對於指定的 Kubernetes 次要版本而言，所有修補程式版本在 EKS 控制平面和節點 AMI 之間都是相容的。

新的 Amazon EKS 平台版本不會帶來重大變更或導致服務中斷。

叢集一律會透過指定 Kubernetes 版本的最新可用 Amazon EKS 平台版本 (eks.<n>) 建立。如果您將叢集更新為新的 Kubernetes 次要版本，您的叢集會收到適用於您要更新之 Kubernetes 次要版本的目前 Amazon EKS 平台版本。

Amazon EKS 的目前及最新平台版本如下表所示。

> **Note**
>
> AWS 最近停用了 2024 年 6 月發佈的一些平台版本。平台版本存在穩定性問題。不需採取任何動作。

若要接收此特定文件頁面所有來源檔案變更的通知，您可以使用 RSS 讀取器訂閱下列 URL：

```
https://github.com/awsdocs/amazon-eks-user-guide/commits/mainline/latest/ug/clusters/platform-versions.adoc.atom
```

Kubernetes 版本 1.33

下列許可控制器已為所有 1.33 平台版本啟用：NodeRestriction、ExtendedResourceToleration、NamespaceLifecycle、LimitRanger、

Kubernetes 版本	EKS 平台版本	版本備註	版本日期
1.33.2	eks.9	具有安全性修正和增強功能的新平台版本。 1.33 eks.7和 1.33 eks.8 已在內部捨棄，且從未發佈。	2025 年 7 月 30 日

Kubernetes 版本	EKS 平台版本	版本備註	版本日期
1.33.1	eks.6	具有安全性修正和增強功能的新平台版本。	2025 年 6 月 26 日
1.33.1	eks.5	具有安全性修正和增強功能的新平台版本。	2025 年 6 月 11 日
1.33.1	eks.4	1.33 適用於 EKS 的 Kubernetes 版本初始版本。如需詳細資訊，請參閱the section called “Kubernetes 1.33”。	2025 年 5 月 28 日

Kubernetes 版本 **1.32**

下列許可控制器已為所有 1.32 平台版本啟用：NodeRestriction、ExtendedResourceToleration、NamespaceLifecycle、LimitRanger、

Kubernetes 版本	EKS 平台版本	版本備註	版本日期
1.32.6	eks.16	具有安全性修正和增強功能的新平台版本。 1.32 eks.14和 1.32 eks.15 已在內部捨棄，且從未發佈。	2025 年 7 月 30 日
1.32.5	eks.13	具有安全性修正和增強功能的新平台版本。	2025 年 6 月 26 日

Kubernetes 版本	EKS 平台版本	版本備註	版本日期
1.32.5	eks.12	具有安全性修正和增強功能的新平台版本。	2025 年 6 月 11 日
1.32.5	eks.11	具有安全性修正和增強功能的新平台版本。	2025 年 5 月 30 日
1.32.3	eks.10	具有安全性修正和增強功能的新平台版本。	2025 年 5 月 16 日
1.32.3	eks.9	具有安全性修正和增強功能的新平台版本。	2025 年 4 月 29 日
1.32.3	eks.8	具有安全性修正和增強功能的新平台版本。	2025 年 4 月 18 日
1.32.3	eks.7	具有安全性修正和增強功能的新平台版本。	2025 年 4 月 18 日
1.32.3	eks.6	具有安全性修正和增強功能的新平台版本。	2025 年 4 月 2 日
1.32.2	eks.5	具有安全性修正和增強功能的新平台版本。	2025 年 3 月 17 日
1.32.2	eks.4	具有安全性修正和增強功能的新平台版本。	2025 年 3 月 4 日

Kubernetes 版本	EKS 平台版本	版本備註	版本日期
1.32.1	eks.3	具有安全性修正和增強功能的新平台版本。	2025 年 2 月 24 日
1.32.0	eks.2	1.32 適用於 EKS 的 Kubernetes 版本初始版本。如需詳細資訊，請參閱the section called “Kubernetes 1.32”。	2025 年 1 月

Kubernetes 版本 **1.31**

下列許可控制器已為所有 1.31 平台版本啟用：NodeRestriction、ExtendedResourceToleration、NamespaceLifecycle、LimitRanger、

Kubernetes 版本	EKS 平台版本	版本備註	版本日期
1.31.10	eks.32	具有安全性修正和增強功能的新平台版本。 1.31 eks.30和 1.31 eks.31 已在內部捨棄，且從未發佈。	2025 年 7 月 30 日
1.31.9	eks.29	具有安全性修正和增強功能的新平台版本。	2025 年 6 月 26 日
1.31.9	eks.28	具有安全性修正和增強功能的新平台版本。	2025 年 6 月 11 日

Kubernetes 版本	EKS 平台版本	版本備註	版本日期
1.31.9	eks.27	具有安全性修正和增強功能的新平台版本。	2025 年 5 月 30 日
1.31.7	eks.26	具有安全性修正和增強功能的新平台版本。	2025 年 5 月 16 日
1.31.7	eks.25	具有安全性修正和增強功能的新平台版本。	2025 年 4 月 29 日
1.31.7	eks.24	具有安全性修正和增強功能的新平台版本。	2025 年 4 月 18 日
1.31.7	eks.23	具有安全性修正和增強功能的新平台版本。	2025 年 4 月 18 日
1.31.7	eks.22	具有安全性修正和增強功能的新平台版本。	2025 年 4 月 2 日
1.31.6	eks.21	具有安全性修正和增強功能的新平台版本。	2025 年 3 月 17 日
1.31.6	eks.20	具有安全性修正和增強功能的新平台版本。	2025 年 3 月 4 日
1.31.5	eks.19	具有安全性修正和增強功能的新平台版本。	2025 年 2 月 24 日

Kubernetes 版本	EKS 平台版本	版本備註	版本日期
1.31.5	eks.18	具有安全性修正和增強功能的新平台版本。	2025 年 2 月 24 日
1.31.4	eks.17	具有安全性修正和增強功能的新平台版本。	2025 年 1 月 17 日
1.31.2	eks.12	具有 Amazon EKS 混合節點的新平台版本支援和增強控制平面可觀測性。請參閱 the section called “混合節點”和 Amazon EKS 分別增強效能可觀測性。	2024 年 11 月 15 日
1.31.1	eks.6	具有安全性修正和增強功能的新平台版本。	2024 年 10 月 21 日
1.31.0	eks.4	1.31 適用於 EKS 的 Kubernetes 版本初始版本。如需詳細資訊，請參閱the section called “Kubernetes 1.31”。	2024 年 9 月 26 日

Kubernetes 版本 **1.30**

下列許可控制器已為所有 1.30 平台版本啟用：NodeRestriction、ExtendedResourceToleration、NamespaceLifecycle、LimitRanger、

Kubernetes 版本	EKS 平台版本	版本備註	版本日期
1.30.14	eks.40	具有安全性修正和增強功能的新平台版本。 1.30 eks.38和 1.30 eks.39 已在內部捨棄，且從未發佈。	2025 年 7 月 30 日
1.30.13	eks.37	具有安全性修正和增強功能的新平台版本。	2025 年 6 月 26 日
1.30.13	eks.36	具有安全性修正和增強功能的新平台版本。	2025 年 6 月 11 日
1.30.13	eks.35	具有安全性修正和增強功能的新平台版本。	2025 年 5 月 30 日
1.30.11	eks.34	具有安全性修正和增強功能的新平台版本。	2025 年 5 月 16 日
1.30.11	eks.33	具有安全性修正和增強功能的新平台版本。	2025 年 4 月 29 日
1.30.11	eks.32	具有安全性修正和增強功能的新平台版本。	2025 年 4 月 18 日
1.30.11	eks.31	具有安全性修正和增強功能的新平台版本。	2025 年 4 月 18 日

Kubernetes 版本	EKS 平台版本	版本備註	版本日期
1.30.11	eks.30	具有安全性修正和增強功能的新平台版本。	2025 年 4 月 2 日
1.30.10	eks.29	具有安全性修正和增強功能的新平台版本。	2025 年 3 月 17 日
1.30.10	eks.28	具有安全性修正和增強功能的新平台版本。	2025 年 3 月 4 日
1.30.9	eks.27	具有安全性修正和增強功能的新平台版本。	2025 年 2 月 24 日
1.30.8	eks.25	具有安全性修正和增強功能的新平台版本。	2025 年 1 月 17 日
1.30.8	eks.24	具有安全性修正和增強功能的新平台版本。	2025 年 1 月 3 日
1.30.7	eks.23	具有安全性修正和增強功能的新平台版本。	2024 年 12 月 13 日
1.30.6	eks.22	具有安全性修正和增強功能的新平台版本。	2024 年 12 月 13 日
1.30.6	eks.21	具有安全性修正和增強功能的新平台版本。	2024 年 12 月 13 日

Kubernetes 版本	EKS 平台版本	版本備註	版本日期
1.30.6	eks.20	具有 Amazon EKS 混合節點的新平台版本支援和增強控制平面可觀測性。請參閱 the section called “混合節點”和 Amazon EKS 分別增強效能可觀測性。	2024 年 11 月 15 日
1.30.5	eks.12	具有安全性修正和增強功能的新平台版本。	2024 年 10 月 21 日
1.30.4	eks.8	具有安全性修正和增強功能的新平台版本。	2024 年 9 月 3 日
1.30.3	eks.7	具有安全性修正和增強功能的新平台版本。	2024 年 8 月 28 日
1.30.3	eks.6	具有安全性修正和增強功能的新平台版本。	2024 年 8 月 9 日
1.30.2	eks.5	具有安全性修正和增強功能的新平台版本。	2024 年 7 月 2 日
1.30.0	eks.2	1.30 適用於 EKS 的 Kubernetes 版本初始版本。如需詳細資訊，請參閱the section called “Kubernetes 1.30”。	2024 年 5 月 23 日

Kubernetes 版本 **1.29**

下列許可控制器已為所有 1.29 平台版本啟
用：NodeRestriction、ExtendedResourceToleration、NamespaceLifecycle、LimitRanger、

Kubernetes 版本	EKS 平台版本	版本備註	版本日期
1.29.15	eks.43	具有安全性修正和增強功能的新平台版本。 1.29 eks.41和 1.29 eks.42 已在內部捨棄，且從未發佈。	2025 年 7 月 30 日
1.29.15	eks.40	具有安全性修正和增強功能的新平台版本。	2025 年 6 月 26 日
1.29.15	eks.39	具有安全性修正和增強功能的新平台版本。	2025 年 6 月 11 日
1.29.15	eks.38	具有安全性修正和增強功能的新平台版本。	2025 年 5 月 30 日
1.29.15	eks.37	具有安全性修正和增強功能的新平台版本。	2025 年 5 月 16 日
1.29.15	eks.36	具有安全性修正和增強功能的新平台版本。	2025 年 4 月 29 日
1.29.15	eks.35	具有安全性修正和增強功能的新平台版本。	2025 年 4 月 18 日

Kubernetes 版本	EKS 平台版本	版本備註	版本日期
1.29.15	eks.34	具有安全性修正和增強功能的新平台版本。	2025 年 4 月 18 日
1.29.15	eks.33	具有安全性修正和增強功能的新平台版本。	2025 年 4 月 2 日
1.29.14	eks.32	具有安全性修正和增強功能的新平台版本。	2025 年 3 月 17 日
1.29.14	eks.31	具有安全性修正和增強功能的新平台版本。	2025 年 3 月 4 日
1.29.13	eks.30	具有安全性修正和增強功能的新平台版本。	2025 年 2 月 24 日
1.29.13	eks.29	具有安全性修正和增強功能的新平台版本。	2025 年 2 月 24 日
1.29.12	eks.28	具有安全性修正和增強功能的新平台版本。	2025 年 1 月 20 日
1.29.12	eks.27	具有安全性修正和增強功能的新平台版本。	2025 年 1 月 3 日
1.29.11	eks.26	具有安全性修正和增強功能的新平台版本。	2024 年 12 月 13 日

Kubernetes 版本	EKS 平台版本	版本備註	版本日期
1.29.10	eks.25	具有安全性修正和增強功能的新平台版本。	2024 年 12 月 13 日
1.29.10	eks.24	具有安全性修正和增強功能的新平台版本。	2024 年 12 月 13 日
1.29.10	eks.23	具有 Amazon EKS 混合節點的新平台版本支援和增強控制平面可觀測性。請參閱 the section called “混合節點”和 Amazon EKS 分別增強效能可觀測性。	2024 年 11 月 15 日
1.29.9	eks.17	具有安全性修正和增強功能的新平台版本。	2024 年 10 月 21 日
1.29.8	eks.13	具有安全性修正和增強功能的新平台版本。	2024 年 9 月 3 日
1.29.7	eks.12	具有安全性修正和增強功能的新平台版本。	2024 年 8 月 28 日
1.29.7	eks.11	具有安全性修正和增強功能的新平台版本。	2024 年 8 月 9 日
1.29.6	eks.10	具有安全性修正和增強功能的新平台版本。	2024 年 7 月 2 日

Kubernetes 版本	EKS 平台版本	版本備註	版本日期
1.29.4	eks.7	具有 CoreDNS Autoscaling、安全性修正和增強功能的新平台版本。如需 CoreDNS Autoscaling 的詳細資訊，請參閱 the section called “高流量的擴展”。	2024 年 5 月 16 日
1.29.3	eks.6	具有安全性修正和增強功能的新平台版本。	2024 年 4 月 18 日
1.29.1	eks.5	具有安全性修正和增強功能的新平台版本。	2024 年 3 月 29 日
1.29.1	eks.4	具有安全性修正和增強功能的新平台版本。	2024 年 3 月 20 日
1.29.1	eks.3	具有安全性修正和增強功能的新平台版本。	2024 年 3 月 12 日
1.29.0	eks.1	1.29 適用於 EKS 的 Kubernetes 版本初始版本。如需詳細資訊，請參閱the section called “Kubernetes 1.29”。	2024 年 1 月 23 日

Kubernetes 版本 **1.28**

下列許可控制器已為所有 1.28 平台版本啟用：NodeRestriction、ExtendedResourceToleration、NamespaceLifecycle、LimitRanger、

Kubernetes 版本	EKS 平台版本	版本備註	版本日期
1.28.15	eks.49	具有安全性修正和增強功能的新平台版本。 1.28 eks.47和 1.28 eks.48 已在內部捨棄，且從未發佈。	2025 年 7 月 30 日
1.28.15	eks.46	具有安全性修正和增強功能的新平台版本。	2025 年 6 月 26 日
1.28.15	eks.45	具有安全性修正和增強功能的新平台版本。	2025 年 6 月 11 日
1.28.15	eks.44	具有安全性修正和增強功能的新平台版本。	5 月 30 日。2025
1.28.15	eks.43	具有安全性修正和增強功能的新平台版本。	2025 年 5 月 16 日
1.28.15	eks.42	具有安全性修正和增強功能的新平台版本。	2025 年 4 月 29 日
1.28.15	eks.41	具有安全性修正和增強功能的新平台版本。	2025 年 4 月 18 日

Kubernetes 版本	EKS 平台版本	版本備註	版本日期
1.28.15	eks.40	具有安全性修正和增強功能的新平台版本。	2025 年 4 月 18 日
1.28.15	eks.39	具有安全性修正和增強功能的新平台版本。	2025 年 4 月 2 日
1.28.15	eks.38	具有安全性修正和增強功能的新平台版本。	2025 年 3 月 17 日
1.28.15	eks.37	具有安全性修正和增強功能的新平台版本。	2025 年 3 月 4 日
1.28.15	eks.36	具有安全性修正和增強功能的新平台版本。	2025 年 2 月 24 日
1.28.15	eks.34	具有安全性修正和增強功能的新平台版本。	2025 年 1 月 17 日
1.28.15	eks.33	具有安全性修正和增強功能的新平台版本。	2025 年 1 月 3 日
1.28.15	eks.32	具有安全性修正和增強功能的新平台版本。	2024 年 12 月 13 日
1.28.15	eks.31	具有安全性修正和增強功能的新平台版本。	2024 年 12 月 13 日

Kubernetes 版本	EKS 平台版本	版本備註	版本日期
1.28.15	eks.30	具有安全性修正和增強功能的新平台版本。	2024 年 12 月 13 日
1.28.15	eks.29	具有 Amazon EKS 混合節點的新平台版本支援和增強控制平面可觀測性。請參閱 the section called “混合節點”和 Amazon EKS 分別增強效能可觀測性。	2024 年 11 月 15 日
1.28.14	eks.23	具有安全性修正和增強功能的新平台版本。	2024 年 10 月 21 日
1.28.13	eks.19	具有安全性修正和增強功能的新平台版本。	2024 年 9 月 3 日
1.28.12	eks.18	具有安全性修正和增強功能的新平台版本。	2024 年 8 月 28 日
1.28.11	eks.17	具有安全性修正和增強功能的新平台版本。	2024 年 8 月 9 日
1.28.11	eks.16	具有安全性修正和增強功能的新平台版本。	2024 年 7 月 2 日

Kubernetes 版本	EKS 平台版本	版本備註	版本日期
1.28.9	eks.13	具有 CoreDNS Autoscaling、安全性修正和增強功能的新平台版本。如需 CoreDNS Autoscaling 的詳細資訊，請參閱 the section called “高流量的擴展”。	2024 年 5 月 16 日
1.28.8	eks.12	具有安全性修正和增強功能的新平台版本。	2024 年 4 月 18 日
1.28.7	eks.11	具有安全性修正和增強功能的新平台版本。	2024 年 3 月 29 日
1.28.7	eks.10	具有安全性修正和增強功能的新平台版本。	2024 年 3 月 20 日
1.28.6	eks.9	具有安全性修正和增強功能的新平台版本。	2024 年 3 月 12 日
1.28.5	eks.7	具有安全性修正和增強功能的新平台版本。	2024 年 1 月 17 日
1.28.4	eks.6	具有存取項目、安全性修正和增強功能的新平台版本。	2023 年 12 月 14 日
1.28.4	eks.5	具有安全性修正和增強功能的新平台版本。	2023 年 12 月 12 日

Kubernetes 版本	EKS 平台版本	版本備註	版本日期
1.28.3	eks.4	具有 的新平台版本 了解 EKS Pod Identity 如何授予 Pod 對 AWS 服務、安全性修正和增強功能的存取權。	2023 年 11 月 10 日
1.28.3	eks.3	具有安全性修正和增強功能的新平台版本。	2023 年 11 月 3 日
1.28.2	eks.2	具有安全性修正和增強功能的新平台版本。	2023 年 10 月 16 日
1.28.1	eks.1	1.28 適用於 EKS 的 Kubernetes 版本初始版本。如需詳細資訊，請參閱the section called “Kubernetes 1.28”。	2023 年 9 月 26 日

Kubernetes 版本 **1.27**

下列許可控制器已為所有 1.27 平台版本啟用：NodeRestriction、ExtendedResourceToleration、NamespaceLifecycle、LimitRanger、

Kubernetes 版本	EKS 平台版本	版本備註	版本日期
1.27.16	eks.53	具有安全性修正和增強功能的新平台版本。 1.27 eks.51和 1.27 eks.52 已在內部捨棄，且從未發佈。	2025 年 7 月 30 日

Kubernetes 版本	EKS 平台版本	版本備註	版本日期
1.27.16	eks.50	具有安全性修正和增強功能的新平台版本。	2025 年 6 月 26 日
1.27.16	eks.49	具有安全性修正和增強功能的新平台版本。	2025 年 6 月 11 日
1.27.16	eks.48	具有安全性修正和增強功能的新平台版本。	2025 年 5 月 30 日
1.27.16	eks.47	具有安全性修正和增強功能的新平台版本。	2025 年 5 月 16 日
1.27.16	eks.46	具有安全性修正和增強功能的新平台版本。	2025 年 4 月 29 日
1.27.16	eks.45	具有安全性修正和增強功能的新平台版本。	2025 年 4 月 18 日
1.27.16	eks.44	具有安全性修正和增強功能的新平台版本。	2025 年 4 月 18 日
1.27.16	eks.43	具有安全性修正和增強功能的新平台版本。	2025 年 4 月 2 日
1.27.16	eks.42	具有安全性修正和增強功能的新平台版本。	2025 年 3 月 17 日

Kubernetes 版本	EKS 平台版本	版本備註	版本日期
1.27.16	eks.41	具有安全性修正和增強功能的新平台版本。	2025 年 3 月 4 日
1.27.16	eks.40	具有安全性修正和增強功能的新平台版本。	2025 年 3 月 4 日
1.27.16	eks.38	具有安全性修正和增強功能的新平台版本。	2025 年 1 月 17 日
1.27.16	eks.37	具有安全性修正和增強功能的新平台版本。	2025 年 1 月 3 日
1.27.16	eks.36	具有安全性修正和增強功能的新平台版本。	2024 年 12 月 13 日
1.27.16	eks.35	具有安全性修正和增強功能的新平台版本。	2024 年 12 月 13 日
1.27.16	eks.34	具有安全性修正和增強功能的新平台版本。	2024 年 12 月 13 日
1.27.16	eks.33	具有 Amazon EKS 混合節點支援、安全性修正和增強功能的新平台版本。如需 Amazon EKS 混合節點的詳細資訊，請參閱 the section called “混合節點”。	2024 年 11 月 15 日

Kubernetes 版本	EKS 平台版本	版本備註	版本日期
1.27.16	eks.27	具有安全性修正和增強功能的新平台版本。	2024 年 10 月 21 日
1.27.16	eks.23	具有安全性修正和增強功能的新平台版本。	2024 年 9 月 3 日
1.27.16	eks.22	具有安全性修正和增強功能的新平台版本。	2024 年 8 月 28 日
1.27.16	eks.21	具有安全性修正和增強功能的新平台版本。	2024 年 8 月 9 日
1.27.15	eks.20	具有安全性修正和增強功能的新平台版本。	2024 年 7 月 2 日
1.27.13	eks.17	具有 CoreDNS Autoscaling、安全性修正和增強功能的新平台版本。如需 CoreDNS Autoscaling 的詳細資訊，請參閱 the section called “高流量的擴展”。	2024 年 5 月 16 日
1.27.12	eks.16	具有安全性修正和增強功能的新平台版本。	2024 年 4 月 18 日
1.27.11	eks.15	具有安全性修正和增強功能的新平台版本。	2024 年 3 月 29 日

Kubernetes 版本	EKS 平台版本	版本備註	版本日期
1.27.11	eks.14	具有安全性修正和增強功能的新平台版本。	2024 年 3 月 20 日
1.27.10	eks.13	具有安全性修正和增強功能的新平台版本。	2024 年 3 月 12 日
1.27.9	eks.11	具有安全性修正和增強功能的新平台版本。	2024 年 1 月 17 日
1.27.8	eks.10	具有存取項目、安全性修正和增強功能的新平台版本。	2023 年 12 月 14 日
1.27.8	eks.9	具有安全性修正和增強功能的新平台版本。	2023 年 12 月 12 日
1.27.7	eks.8	使用 了解 EKS Pod Identity 如何授予 Pod 對 AWS 服務、安全性修正和增強功能的存取權的新平台版本。	2023 年 11 月 10 日
1.27.7	eks.7	具有安全性修正和增強功能的新平台版本。	2023 年 11 月 3 日
1.27.6	eks.6	具有安全性修正和增強功能的新平台版本。	2023 年 10 月 16 日
1.27.4	eks.5	具有安全性修正和增強功能的新平台版本。	2023 年 8 月 30 日

Kubernetes 版本	EKS 平台版本	版本備註	版本日期
1.27.4	eks.4	具有安全性修正和增強功能的新平台版本。	2023 年 7 月 30 日
1.27.3	eks.3	具有安全性修正和增強功能的新平台版本。	2023 年 6 月 30 日
1.27.2	eks.2	具有安全性修正和增強功能的新平台版本。	2023 年 6 月 9 日
1.27.1	eks.1	1.27 適用於 EKS 的 Kubernetes 版本初始版本。如需詳細資訊，請參閱the section called “Kubernetes 1.27”。	2023 年 5 月 24 日

Kubernetes 版本 **1.26**

下列許可控制器已為所有 1.26 平台版本啟用：NodeRestriction、ExtendedResourceToleration、NamespaceLifecycle、LimitRanger、

Kubernetes 版本	EKS 平台版本	版本備註	版本日期
1.26.15	eks.51	具有安全性修正和增強功能的新平台版本。	2025 年 6 月 11 日
1.26.15	eks.50	具有安全性修正和增強功能的新平台版本。	2025 年 5 月 30 日

Kubernetes 版本	EKS 平台版本	版本備註	版本日期
1.26.15	eks.49	具有安全性修正和增強功能的新平台版本。	2025 年 5 月 16 日
1.26.15	eks.48	具有安全性修正和增強功能的新平台版本。	2025 年 4 月 29 日
1.26.15	eks.47	具有安全性修正和增強功能的新平台版本。	2025 年 4 月 18 日
1.26.15	eks.46	具有安全性修正和增強功能的新平台版本。	2025 年 4 月 18 日
1.26.15	eks.45	具有安全性修正和增強功能的新平台版本。	2025 年 4 月 2 日
1.26.15	eks.44	具有安全性修正和增強功能的新平台版本。	2025 年 3 月 17 日
1.26.15	eks.43	具有安全性修正和增強功能的新平台版本。	2025 年 3 月 4 日
1.26.15	eks.42	具有安全性修正和增強功能的新平台版本。	2025 年 2 月 24 日
1.26.15	eks.40	具有安全性修正和增強功能的新平台版本。	2025 年 1 月 17 日

Kubernetes 版本	EKS 平台版本	版本備註	版本日期
1.26.15	eks.39	具有安全性修正和增強功能的新平台版本。	2025 年 1 月 3 日
1.26.15	eks.38	具有安全性修正和增強功能的新平台版本。	2024 年 12 月 13 日
1.26.15	eks.37	具有安全性修正和增強功能的新平台版本。	2024 年 12 月 13 日
1.26.15	eks.36	具有安全性修正和增強功能的新平台版本。	2024 年 12 月 13 日
1.26.15	eks.35	具有 Amazon EKS 混合節點支援、安全性修正和增強功能的新平台版本。如需 Amazon EKS 混合節點的詳細資訊，請參閱 the section called “混合節點”。	2024 年 11 月 15 日
1.26.15	eks.28	具有安全性修正和增強功能的新平台版本。	2024 年 10 月 21 日
1.26.15	eks.24	具有安全性修正和增強功能的新平台版本。	2024 年 9 月 3 日
1.26.15	eks.23	具有安全性修正和增強功能的新平台版本。	2024 年 8 月 28 日

Kubernetes 版本	EKS 平台版本	版本備註	版本日期
1.26.15	eks.22	具有安全性修正和增強功能的新平台版本。	2024 年 8 月 9 日
1.26.15	eks.21	具有安全性修正和增強功能的新平台版本。	2024 年 7 月 2 日
1.26.15	eks.18	具有 CoreDNS Autoscaling、安全性修正和增強功能的新平台版本。如需 CoreDNS Autoscaling 的詳細資訊，請參閱 the section called “高流量的擴展”。	2024 年 5 月 16 日
1.26.15	eks.17	具有安全性修正和增強功能的新平台版本。	2024 年 4 月 18 日
1.26.14	eks.16	具有安全性修正和增強功能的新平台版本。	2024 年 3 月 29 日
1.26.14	eks.15	具有安全性修正和增強功能的新平台版本。	2024 年 3 月 20 日
1.26.13	eks.14	具有安全性修正和增強功能的新平台版本。	2024 年 3 月 12 日
1.26.12	eks.12	具有安全性修正和增強功能的新平台版本。	2024 年 1 月 17 日

Kubernetes 版本	EKS 平台版本	版本備註	版本日期
1.26.11	eks.11	具有存取項目、安全性修正和增強功能的新平台版本。	2023 年 12 月 14 日
1.26.11	eks.10	具有安全性修正和增強功能的新平台版本。	2023 年 12 月 12 日
1.26.10	eks.9	具有 的新平台版本 了解 EKS Pod Identity 如何授予 Pod 對 AWS 服務、安全性修正和增強功能的存取權。	2023 年 11 月 10 日
1.26.10	eks.8	具有安全性修正和增強功能的新平台版本。	2023 年 11 月 3 日
1.26.9	eks.7	具有安全性修正和增強功能的新平台版本。	2023 年 10 月 16 日
1.26.7	eks.6	具有安全性修正和增強功能的新平台版本。	2023 年 8 月 30 日
1.26.7	eks.5	具有安全性修正和增強功能的新平台版本。	2023 年 7 月 30 日
1.26.6	eks.4	具有安全性修正和增強功能的新平台版本。	2023 年 6 月 30 日
1.26.5	eks.3	具有安全性修正和增強功能的新平台版本。	2023 年 6 月 9 日

Kubernetes 版本	EKS 平台版本	版本備註	版本日期
1.26.4	eks.2	具有安全性修正和增強功能的新平台版本。	2023 年 5 月 5 日
1.26.2	eks.1	1.26 適用於 EKS 的 Kubernetes 版本初始版本。	2023 年 4 月 11 日

取得目前的平台版本

1. 開啟 Amazon EKS 主控台。
2. 在導覽窗格中，選擇叢集。
3. 在叢集清單中，選擇要檢查平台版本的叢集名稱。
4. 選擇 Overview (概觀) 索引標籤。
5. 詳細資訊區段中的 提供平台版本。
6. 決定您要檢查平台版本的叢集名稱。
7. 執行以下命令：

```
aws eks describe-cluster --name my-cluster --query cluster.platformVersion
```

範例輸出如下。

```
"eks.10"
```

變更平台版本

您無法變更 EKS 叢集的平台版本。當新的 Amazon EKS 平台版本可供 Kubernetes 版本使用時，EKS 會自動將所有現有叢集升級至其對應 Kubernetes 版本的最新 Amazon EKS 平台版本。現有 Amazon EKS 平台版本的自動升級將會逐步發佈。您無法使用 AWS 主控台或 CLI 變更平台版本。

如果您升級 Kubernetes 版本，您的叢集將移至 Kubernetes 版本的最新平台版本。

對 Amazon EKS 叢集和節點的問題進行故障診斷

此章節涵蓋一些您在使用 Amazon EKS 時可能遇到的常見錯誤，並提供解決方法。如果您需要針對特定 Amazon EKS 區域進行疑難排解，請參閱個別 the section called “故障診斷”、the section called “對 EKS 連接器進行故障診斷” 和針對使用 EKS 附加元件的 ADOT 進行疑難排解主題。

如需其他疑難排解資訊，請參閱 AWS re：Post 上的 Amazon Elastic Kubernetes Service 知識中心內容。

容量不足

如果您在嘗試建立 Amazon EKS 叢集時收到下列錯誤，則您指定的其中一個可用區域沒有足夠的容量來支援叢集。

```
Cannot create cluster 'example-cluster' because region-1d, the targeted
Availability Zone, does not currently have sufficient capacity to support
the cluster. Retry and choose from these Availability Zones: region-1a,
region-1b, region-1c
```

重新嘗試使用叢集 VPC 中的子網路來建立您的叢集，叢集 VPC 託管於此錯誤訊息傳回之可用區域之中。

有叢集無法駐留在其中的可用區域。比較子網路所在的可用區域與子網路需求和考量事項中的可用區域清單。

節點無法加入叢集

有幾個常見的原因會阻擋節點加入叢集：

- 如果節點是受管節點，則 Amazon EKS 會在您建立節點群組時會將項目新增至 `aws-auth ConfigMap`。如果該項目被移除或修改，則您需要重新新增項目。如需詳細資訊，請在您的終端機中輸入 `eksctl create iamidentitymapping --help`。您可使用您的叢集名稱取代以下命令中的 *`my-cluster`* 部分，然後執行修改後的命令來檢視目前的 `aws-auth ConfigMap` 項目：`eksctl get iamidentitymapping --cluster` *`my-cluster`* 。您指定之角色的 ARN 不能包含 以外的路徑/。例如，如果您的角色名稱是 `development/apps/my-role`，則需要在為角色指定 ARN `my-role`時將其變更為 。請確認您指定的是節點 IAM 角色 ARN (而非執行個體設定檔 ARN)。

如果節點是自我管理的，而且您尚未為節點 IAM 角色的 ARN 建立存取項目，則請執行針對受管節點列出的相同命令。如果您已為節點 IAM 角色 ARN 建立存取項目，則可能無法在存取項目中正確進行設定。請確認將節點 IAM 角色 ARN (而非執行個體設定檔 ARN) 指定為 aws-auth ConfigMap 項目或存取項目中的主體 ARN。如需存取項目的詳細資訊，請參閱 the section called “授予許可”。

- node AWS CloudFormation 範本中的 ClusterName 與您希望節點加入的叢集名稱不完全相符。將不正確的值傳遞至此欄位會導致節點/var/lib/kubelet/kubeconfig檔案的組態不正確，而且節點不會加入叢集。
- 節點未標記為叢集所擁有。您的節點必須套用下列標籤，其中 *my-cluster* 會取代為您的叢集名稱。

金鑰	值
kubernetes.io/cluster/*my-cluster*	owned

- 節點可能無法使用公有 IP 地址存取叢集。確定已將公有子網路中部署的節點指派給公有 IP 地址。如果沒有，您可以在啟動彈性 IP 地址後將其與節點建立關聯。如需詳細資訊，請參閱將彈性 IP 地址與執行中的執行個體或網路介面建立關聯。如果公有子網路未設定為自動將公有 IP 地址指派給部署到該子網路的執行個體，則建議您啟用該設定。如需詳細資訊，請參閱修改子網的公有 IPv4 定址屬性。如果節點部署到私有子網路，則子網路必須具有指派給它的公有 IP 地址的 NAT 閘道路由。
- 您的帳戶未啟用您要部署節點之 AWS 區域的 AWS STS 端點。若要啟用區域，請參閱在 區域中啟用和停用 AWS STS AWS。
- 節點沒有私有 DNS 項目，導致kubelet日誌包含node "" not found錯誤。確保建立節點所在的 VPC 具有為 domain-name 設定的值以及在 DHCP options set 中將 domain-name-servers 設定為 Options。預設值為 domain-name:<region>.compute.internal 和 domain-name-servers:AmazonProvidedDNS。如需詳細資訊，請參閱《Amazon VPC 使用者指南》中的 DHCP 選項集。
- 如果受管節點群組中的節點未在 15 分鐘內連線至叢集，則會發出「NodeCreationFailure」的運作狀態問題，且主控台狀態會設為 Create failed。對於啟動時間緩慢AMIs，可以使用快速啟動來解決此問題。

若要識別和疑難排解導致 Worker 節點無法加入叢集的常見原因，您可以使用 AWSSupport-TroubleshootEKSWorkerNode Runbook。如需詳細資訊，請參閱 AWS Systems Manager Automation Runbook 參考 AWSSupport-TroubleshootEKSWorkerNode 中的 。

未經授權或存取遭拒 (kubectl)

如果您在執行kubectl命令時收到下列其中一個錯誤，表示您未正確kubectl設定 Amazon EKS 或您正在使用的 IAM 委託人（角色或使用者）的登入資料，未對應至對 Amazon EKS 叢集上的 Kubernetes 物件具有足夠許可的 Kubernetes 使用者名稱。

- `could not get token: AccessDenied: Access denied`
- `error: You must be logged in to the server (Unauthorized)`
- `error: the server doesn't have a resource type "svc"`

這種情況可能由下列原因之一造成：

- 該叢集使用一個 IAM 主體的憑證建立，而 kubectl 設定為使用另一個 IAM 主體的憑證。若要解決此問題，請更新 kube config 檔案以使用建立叢集時使用的憑證。如需詳細資訊，請參閱the section called "使用 kubectl 存取叢集"。
- 如果您的叢集符合授予 IAM 使用者使用 EKS 存取項目存取 Kubernetes 的先決條件區段中的最低平台需求，則存取項目不會與您的 IAM 主體存在。如果存在，則它沒有為其定義的必要 Kubernetes 群組名稱，或者沒有與其相關聯的適當存取政策。如需詳細資訊，請參閱the section called "授予許可"。
- 如果您的叢集不符合使用 EKS 存取項目授予 IAM 使用者存取 Kubernetes 的最低平台需求，則 中不存在具有 IAM 主體的項目aws-authConfigMap。如果存在，則不會映射到繫結到 Kubernetes Role或ClusterRole具有必要許可的 Kubernetes 群組名稱。如需 Kubernetes 角色型授權 (RBAC) 物件的詳細資訊，請參閱 Kubernetes 文件中的使用 RBAC 授權。您可使用您的叢集名稱取代以下命令中的 *my-cluster* 部分，然後執行修改後的命令來檢視目前的 aws-auth ConfigMap 項目：eksctl get iamidentitymapping --cluster *my-cluster* 。如果具有 IAM 主體 ARN 的項目不在 中ConfigMap，請在終端機eksctl create iamidentitymapping --help中輸入 以了解如何建立項目。

如果您安裝並設定 AWS CLI，則可以設定您使用的 IAM 登入資料。如需詳細資訊，請參閱《 AWS 命令列界面使用者指南》中的設定 AWS CLI。如果您擔任 IAM 角色來存取叢集上的 Kubernetes 物件，您也可以kubectl將 設定為使用 IAM 角色。如需詳細資訊，請參閱the section called "使用 kubectl 存取叢集"。

hostname doesn't match

您系統的 Python 版本必須是 2.7.9或更新版本。否則，您會收到 CLI AWS 呼叫 Amazon EKS 的hostname doesn't match錯誤。如需詳細資訊，請參閱 Python 請求常見問答集中的什麼是「主機名稱不相符」錯誤？。

getsockopt: no route to host

Docker 會在 Amazon EKS 叢集的 172.17.0.0/16 CIDR 範圍中執行。建議您叢集的 VPC 子網路不重疊此範圍。否則，您會收到以下錯誤：

```
Error: : error upgrading connection: error dialing backend: dial tcp
 172.17.<nn>.<nn>:10250: getsockopt: no route to host
```

Instances failed to join the Kubernetes cluster

如果您在 Instances failed to join the Kubernetes cluster中收到錯誤 AWS Management Console，請確定叢集的私有端點存取已啟用，或您已針對公有端點存取正確設定 CIDR 區塊。如需詳細資訊，請參閱the section called "叢集端點存取"。

受管節點群組錯誤代碼

如果受管節點群組遇到硬體運作狀態問題，Amazon EKS 會傳回錯誤代碼，以協助您診斷問題。這些運作狀態檢查不會偵測到軟體問題，因為它們是以 Amazon EC2 運作狀態檢查為基礎。以下清單描述了這些錯誤代碼。

AccessDenied

Amazon EKS 或一或多個受管節點無法向 Kubernetes 叢集 API 伺服器進行身分驗證或授權。如需有關解決常見原因的詳細資訊，請參閱 the section called "修正受管節點群組的 AccessDenied 錯誤的常見原因"。私有 Windows AMIs 也可能導致此錯誤代碼以及Not authorized for images錯誤訊息。如需詳細資訊，請參閱the section called "Not authorized for images"。

AmiIdNotFound

我們找不到與您的啟動範本相關聯的 AMI ID。請確認 AMI 已存在且已與您的帳戶共用。

AutoScalingGroupNotFound

我們找不到與受管節點群組相關聯的 Auto Scaling 群組。您可以使用相同設定來重新建立 Auto Scaling 群組以復原。

ClusterUnreachable

Amazon EKS 或一或多個受管節點無法與您的 Kubernetes 叢集 API 伺服器通訊。如果發生網路中斷或 API 伺服器處理請求逾時，就會發生這種情況。

Ec2SecurityGroupNotFound

我們找不到叢集的叢集安全群組。您必須重新建立叢集。

Ec2SecurityGroupDeletionFailure

無法刪除受管節點群組的遠端存取安全群組。從安全群組移除任何相依項目。

Ec2LaunchTemplateNotFound

我們找不到受管節點群組的 Amazon EC2 啟動範本。您必須重新建立節點群組以復原。

Ec2LaunchTemplateVersionMismatch

受管節點群組的 Amazon EC2 啟動範本版本與 Amazon EKS 建立的版本不相符。您可以還原為 Amazon EKS 所建立的版本以復原。

IamInstanceProfileNotFound

我們找不到受管節點群組的 IAM 執行個體描述檔。您可以使用相同設定來重新建立執行個體描述檔以復原。

IamNodeRoleNotFound

我們找不到受管節點群組的 IAM 角色。您可以使用相同設定來重新建立 IAM 角色以復原。

AsgInstanceLaunchFailures

您的 Auto Scaling 群組嘗試啟動執行個體時失敗。

NodeCreationFailure

您啟動的執行個體無法向 Amazon EKS 叢集註冊。這項失敗的常見原因是節點 IAM 角色許可不足或節點缺少對外網際網路存取。您的節點必須符合下列任何一項要求：

- 能夠使用公有 IP 地址存取網際網路。與節點所在的子網路相關聯的安全群組必須允許通訊。如需詳細資訊，請參閱the section called “子網需求和注意事項”及the section called “安全群組要求”。
- 您的節點和 VPC 必須符合部署具有有限網際網路存取的私有叢集中的要求。

InstanceLimitExceeded

AWS 您的帳戶無法啟動任何其他指定執行個體類型的執行個體。您可以請求提升 Amazon EC2 執行個體限制來復原。

InsufficientFreeAddresses

與您的受管節點群組相關聯的一或多個子網路沒有足夠的可用 IP 地址供新節點使用。

InternalFailure

這些錯誤通常是由 Amazon EKS 伺服器端問題所造成。

修正受管節點群組的 **AccessDenied** 錯誤的常見原因

在受管節點群組上執行操作時發生 AccessDenied 錯誤的最常見原因，是缺少 eks:node-manager ClusterRole 或 ClusterRoleBinding。Amazon EKS 會在您的叢集中設定這些資源，作為與受管節點群組上線的一部分，這些資源是管理節點群組所必需的。

ClusterRole 可能隨著時間而改變，但看起來應與以下範例相似：

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: eks:node-manager
rules:
- apiGroups:
  - ''
  resources:
  - pods
  verbs:
  - get
  - list
  - watch
  - delete
- apiGroups:
  - ''
  resources:
  - nodes
  verbs:
  - get
  - list
  - watch
  - patch
```

```
- apiGroups:
  - ''
  resources:
  - pods/eviction
  verbs:
  - create
```

ClusterRoleBinding 可能隨著時間而改變，但看起來應與以下範例相似：

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: eks:node-manager
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: eks:node-manager
subjects:
- apiGroup: rbac.authorization.k8s.io
  kind: User
  name: eks:node-manager
```

驗證 eks:node-manager ClusterRole 是否存在。

```
kubectl describe clusterrole eks:node-manager
```

如果存在，請將輸出與前一個 ClusterRole 範例進行比較。

驗證 eks:node-manager ClusterRoleBinding 是否存在。

```
kubectl describe clusterrolebinding eks:node-manager
```

如果存在，請將輸出與前一個 ClusterRoleBinding 範例進行比較。

如果您在請求受AcessDenied管節點群組操作時，發現遺失或損壞ClusterRole或ClusterRoleBinding錯誤的原因，您可以還原它們。將以下內容儲存至名為 *eks-node-manager-role.yaml* 的檔案。

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
```

```
  name: eks:node-manager
rules:
- apiGroups:
  - ''
  resources:
  - pods
  verbs:
  - get
  - list
  - watch
  - delete
- apiGroups:
  - ''
  resources:
  - nodes
  verbs:
  - get
  - list
  - watch
  - patch
- apiGroups:
  - ''
  resources:
  - pods/eviction
  verbs:
  - create
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: eks:node-manager
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: eks:node-manager
subjects:
- apiGroup: rbac.authorization.k8s.io
  kind: User
  name: eks:node-manager
```

套用檔案。

```
kubectl apply -f eks-node-manager-role.yaml
```

重試節點群組操作，看看是否可以解決您的問題。

Not authorized for images

Not authorized for images 錯誤訊息的一個潛在原因是使用私有 Amazon EKS Windows AMI 來啟動 Windows 受管節點群組。發佈新的 Windows AMIs 後， 會將超過 4 個月的 AMIs AWS 設為私有，使其無法再存取。如果您的受管節點群組使用私有 Windows AMI，請考慮更新您的 Windows 受管節點群組。雖然我們無法保證可以提供已設為私有AMIs 存取權，但您可以透過向 AWS Support 提交票證來請求存取權。如需詳細資訊，請參閱《Amazon EC2 使用者指南》中的修補程式。

節點處於 NotReady 狀態

如果您的節點進入 NotReady 狀態，這可能表示節點狀況不良且無法排程新的 Pod。這可能由於各種原因而發生，例如節點缺少足夠 CPU、記憶體或可用磁碟空間的資源。

對於 Amazon EKS 最佳化 Windows AMIs，在kubelet組態中預設不會保留運算資源。為了協助防止資源問題，您可以為 提供 kube 預留和/或系統預留的kubelet組態值，為系統程序預留運算資源。您可以使用引導指令碼中的-KubeletExtraArgs命令列參數來執行此操作。如需詳細資訊，請參閱《Kubernetes 文件》和本使用者指南the section called “引導指令碼組態參數”》中的預留系統精靈的運算資源。

EKS 日誌收集器

若要對 Amazon EKS 節點的問題進行疑難排解，位於 的節點上提供了預先建置的指令碼/etc/eks/log-collector-script/eks-log-collector.sh。您可以使用指令碼來收集支援案例和一般故障診斷的診斷日誌。

在您的節點使用以下命令即可執行指令碼：

```
sudo bash /etc/eks/log-collector-script/eks-log-collector.sh
```

> **Note**
>
> 如果指令碼在該位置不存在。您可使用以下命令手動下載並執行指令碼：
>
> ```
> curl -O https://amazon-eks.s3.amazonaws.com/support/log-collector-script/linux/
> eks-log-collector.sh
> ```

```
sudo bash eks-log-collector.sh
```

該指令碼收集的診斷資訊如下。

```
$ sudo bash /etc/eks/log-collector-script/eks-log-collector.sh

      This is version 0.7.8. New versions can be found at https://github.com/awslabs/
amazon-eks-ami/blob/main/log-collector-script/

Trying to collect common operating system logs...
Trying to collect kernel logs...
Trying to collect mount points and volume information...
...
...

 Done... your bundled logs are located in /var/log/eks_i-EXAMPLE_2025-03-25_0000-
UTC_0.7.8.tar.gz
```

診斷資訊收集後將存放於：

```
/var/log/eks_i-EXAMPLE_2025-03-25_0000-UTC_0.7.8.tar.gz
```

若要擷取 Bottlerocket 節點的日誌套件，請參閱 Bottlerocket Log 以取得更多詳細資訊。

容器執行階段網路尚未就緒

您可能會收到類似下列的 `Container runtime network not ready` 錯誤和授權錯誤：

```
4191 kubelet.go:2130] Container runtime network not ready: NetworkReady=false
 reason:NetworkPluginNotReady message:docker: network plugin is not ready: cni config
 uninitialized
4191 reflector.go:205] k8s.io/kubernetes/pkg/kubelet/kubelet.go:452: Failed to list
 *v1.Service: Unauthorized
4191 kubelet_node_status.go:106] Unable to register node
 "ip-10-40-175-122.ec2.internal" with API server: Unauthorized
4191 reflector.go:205] k8s.io/kubernetes/pkg/kubelet/kubelet.go:452: Failed to list
 *v1.Service: Unauthorized
```

這種情況可能由下列原因之一造成：

1. 您的叢集aws-authConfigMap上沒有 ，或者它不包含您設定節點的 IAM 角色的項目。

   若要解決此問題，您可使用您的叢集名稱取代以下命令中的 *my-cluster* 部分，然後執行修改後的命令來檢視目前 ConfigMap 中的項目：eksctl get iamidentitymapping --cluster *my-cluster* 。如果您從 命令收到錯誤訊息，可能是因為您的叢集沒有 aws-auth ConfigMap。以下命令會將項目新增至 ConfigMap。如果 ConfigMap 不存在，命令也會建立它。將 *111122223333* 取代為 IAM 角色 AWS 的帳戶 ID，將 *myAmazonEKSNodeRole* 取代為節點角色的名稱。

   ```
   eksctl create iamidentitymapping --cluster my-cluster \
       --arn arn:aws: iam::111122223333:role/myAmazonEKSNodeRole --group
    system:bootstrappers,system:nodes \
       --username system:node:{{EC2PrivateDNSName}}
   ```

   您指定之角色的 ARN 不能包含 以外的路徑/。例如，如果您的角色名稱是 development/apps/my-role，則需要在指定角色的 ARN my-role時將其變更為 。請確認您指定的是節點 IAM 角色 ARN (而非執行個體設定檔 ARN)。

2. 您的自我管理節點位於叢集中，其平台版本最低版本列在 授予 IAM 使用者使用 EKS 存取項目主題存取 Kubernetes 的先決條件中，但節點 IAM 角色的項目未列在 aws-authConfigMap (請參閱上一個項目) 中，或角色的存取項目不存在。若要解決此問題，您可使用您的叢集名稱取代以下命令中的 *my-cluster* 部分，然後執行修改後的命令來檢視目前的存取項目：aws eks list-access-entries --cluster-name *my-cluster* 。下列命令會為節點的 IAM 角色新增存取項目。將 *111122223333* 取代為 IAM 角色 AWS 的帳戶 ID，將 *myAmazonEKSNodeRole* 取代為節點角色的名稱。如果您有 Windows 節點，請以 取代 *EC2_LINUX*EC2_Windows。請確認您指定的是節點 IAM 角色 ARN (而非執行個體設定檔 ARN)。

   ```
   aws eks create-access-entry --cluster-name my-cluster --principal-arn arn:aws:
    iam::111122223333:role/myAmazonEKSNodeRole --type EC2_LINUX
   ```

TLS 交握逾時

當節點無法建立與公有 API 伺服器端點的連接時，您可能會收到類似下列的錯誤。

```
server.go:233] failed to run Kubelet: could not init cloud provider "aws": error
 finding instance i-1111f2222f333e44c: "error listing AWS instances: \"RequestError:
 send request failed\\ncaused by: Post  net/http: TLS handshake timeout\""
```

kubelet 程序將持續重新產生並測試 API 伺服器端點。在控制平面中執行叢集滾動更新 (例如組態變更或版本更新) 的任何程序期間，也可能會暫時發生錯誤。

若要解決此問題，請檢查路由表和安全群組，以確保來自節點的流量可以到達公有端點。

InvalidClientTokenId

如果您針對部署至中國 AWS 區域中叢集的 Pod 或 DaemonSet 的服務帳戶使用 IAM 角色，且尚未在規格中設定AWS_DEFAULT_REGION環境變數，Pod 或 DaemonSet 可能會收到下列錯誤：

```
An error occurred (InvalidClientTokenId) when calling the GetCallerIdentity operation:
 The security token included in the request is invalid
```

若要解決問題，您需要將AWS_DEFAULT_REGION環境變數新增至您的 Pod 或 DaemonSet 規格，如下列範例 Pod 規格所示。

```
apiVersion: v1
kind: Pod
metadata:
  name: envar-demo
  labels:
    purpose: demonstrate-envars
spec:
  containers:
  - name: envar-demo-container
    image: gcr.io/google-samples/node-hello:1.0
    env:
    - name: AWS_DEFAULT_REGION
      value: "region-code"
```

節點群組必須符合 Kubernetes 版本，才能升級控制平面

將控制平面升級至新的 Kubernetes 版本之前，叢集中受管節點和 Fargate 節點的次要版本必須與控制平面目前版本的版本相同。在將所有 Amazon EKS 受管節點升級為目前的叢集版本前，Amazon EKS update-cluster-version API 都會拒絕請求。Amazon EKS 提供 API 來升級受管節點。如需有關升級受管節點群組 Kubernetes 版本的資訊，請參閱 the section called “更新”。若要升級 Fargate 節點的版本，請刪除節點代表的 Pod，並在升級控制平面後重新部署 Pod。如需詳細資訊，請參閱the section called “更新 Kubernetes 版本”。

啟動許多節點時，會出現 Too Many Requests 錯誤

如果同時啟動許多節點，您可能會 Amazon EC2 使用者資料執行日誌看見錯誤訊息，其顯示 Too Many Requests。這可能是因為控制平面因 describeCluster 呼叫超載。超載會導致調節、節點無法執行引導指令碼，以及節點無法完全加入叢集。

確定 --apiserver-endpoint、 --b64-cluster-ca和 --dns-cluster-ip引數正在傳遞至節點的引導指令碼。包含這些引數時，不需要引導指令碼進行describeCluster呼叫，這有助於防止控制平面超載。如需詳細資訊，請參閱the section called “提供使用者資料，將引數傳遞至 Amazon EKS 最佳化 Linux/Bottlerocket AMI 隨附的bootstrap.sh檔案”。

針對 Kubernetes API 伺服器請求回應的 HTTP 401 未經授權的錯誤

如果叢集上的 Pod 服務帳戶字符已過期，您會看到這些錯誤。

Amazon EKS 叢集的 Kubernetes API 伺服器會使用超過 90 天的字符拒絕請求。在 Kubernetes 舊版本中，字符沒有過期。這意味著倚賴這些字符的客户端必須在一小時內進行重新整理。若要防止 Kubernetes API 伺服器因為無效的權杖而拒絕您的請求，工作負載所使用的 Kubernetes 用戶端 SDK 版本必須相同，或晚於下列版本：

- Go 版本 0.15.7 和更新版本
- Python 版本 12.0.0 和更新版本
- Java 版本 9.0.0 和更新版本
- JavaScript 版本 0.10.3 和更新版本
- Ruby master 分支
- Haskell 版本 0.3.0.0
- C# 版本 7.0.5和更新版本

您可以識別叢集中所有使用過時字符的現有 Pod。如需詳細資訊，請參閱the section called “服務帳戶字符”。

Amazon EKS 平台版本比目前平台版本落後兩個版本以上

當 Amazon EKS 無法自動更新叢集的 eks/latest/userguide/platform-versions.html【platform-version，type="documentation"】 時，可能會發生這種情況。儘管造成這種情況的原因很多，然而一

些常見的原因如下。如果任何這些問題適用於您的叢集，它可能仍然有效，Amazon EKS 就不會更新其平台版本。

問題

該叢集 IAM 角色已刪除，此角色是在建立叢集時指定。您可以使用以下命令，查看指定了哪個角色。使用您叢集的名稱取代 *my-cluster*。

```
aws eks describe-cluster --name my-cluster --query cluster.roleArn --output text | cut
 -d / -f 2
```

範例輸出如下。

```
eksClusterRole
```

解決方案

建立具有相同名稱的新叢集 IAM 角色。

問題

已刪除叢集建立期間指定的子網路 – 即叢集建立期間指定要與叢集搭配使用的子網路。您可以使用以下命令，查看指定了哪些子網路。使用您叢集的名稱取代 *my-cluster*。

```
aws eks describe-cluster --name my-cluster --query cluster.resourcesVpcConfig.subnetIds
```

範例輸出如下。

```
[
"subnet-EXAMPLE1",
"subnet-EXAMPLE2"
]
```

解決方案

確認您的帳戶中是否存在子網路 ID。

```
vpc_id=$(aws eks describe-cluster --name my-cluster --query
 cluster.resourcesVpcConfig.vpcId --output text)
aws ec2 describe-subnets --filters "Name=vpc-id,Values=$vpc_id" --query
 "Subnets[*].SubnetId"
```

範例輸出如下。

```
[
"subnet-EXAMPLE3",
"subnet-EXAMPLE4"
]
```

如果輸出中傳回的子網路 IDs 與建立叢集時指定的子網路 IDs 不相符，則如果您希望 Amazon EKS 更新叢集，則需要變更叢集使用的子網路。這是因為如果您在建立叢集時指定了兩個以上的子網路，Amazon EKS 會隨機選取您指定在其中建立新彈性網路介面的子網路。這些網路介面可讓控制平面與節點進行通訊。如果選取的子網路不存在，Amazon EKS 不會更新叢集。您無法控制 Amazon EKS 會從您於建立叢集時指定的子網路中選擇哪些在其中建立新網路介面。

當您啟動叢集的 Kubernetes 版本更新時，更新可能會因為相同原因而失敗。

問題

叢集建立期間指定的安全群組已刪除 – 如果您在叢集建立期間指定了安全群組，您可以使用下列命令查看其 ID。使用您叢集的名稱取代 *my-cluster*。

```
aws eks describe-cluster --name my-cluster --query
 cluster.resourcesVpcConfig.securityGroupIds
```

範例輸出如下。

```
[
    "sg-EXAMPLE1"
]
```

如果 [] 傳回，則叢集建立時未指定任何安全群組，而缺少安全群組並非問題所在。如果傳回安全群組，則請確認安全群組位於您的帳戶中。

解決方案

請確認這些安全性群組是否位於您的帳戶中。

```
vpc_id=$(aws eks describe-cluster --name my-cluster --query
 cluster.resourcesVpcConfig.vpcId --output text)
aws ec2 describe-security-groups --filters "Name=vpc-id,Values=$vpc_id" --query
 "SecurityGroups[*].GroupId"
```

範例輸出如下。

```
[
"sg-EXAMPLE2"
]
```

如果輸出中傳回的安全群組 IDs 與建立叢集時指定的安全群組 IDs 不相符，則如果您希望 Amazon EKS 更新叢集，則需要變更叢集使用的安全群組。如果叢集建立時指定的安全群組 IDs 不存在，Amazon EKS 不會更新叢集。

當您啟動叢集的 Kubernetes 版本更新時，更新可能會因為相同原因而失敗。

- 您在建立叢集時指定的每個子網路中沒有至少六個 （雖然我們建議 16 個） 可用的 IP 地址。如果您在子網路中沒有足夠的可用 IP 地址，您需要釋放子網路中的 IP 地址，或者您需要變更叢集使用的子網路，以使用具有足夠可用 IP 地址的子網路。
- 您在建立叢集時啟用秘密加密，且您指定的 AWS KMS 金鑰已刪除。假如您希望 Amazon EKS 更新叢集，您必須建立新的叢集

叢集運作狀態FAQs和具有解析路徑的錯誤代碼

Amazon EKS 會偵測 EKS 叢集和叢集基礎設施的問題，並將其存放在 EKS 叢集資源的運作狀態物件中。藉助叢集運作狀態資訊，您可以更快速地偵測、診斷並解決叢集問題，從而建立更安全且及時更新的應用程式環境。此外，由於必要的基礎設施或叢集組態發生問題，您可能無法升級至較新版本的 Kubernetes 或 Amazon EKS 在降級的叢集上安裝安全性更新。Amazon EKS 可能需要 3 小時才能偵測到問題或偵測某個問題是否已解決。

維護 Amazon EKS 叢集的運作狀態是 Amazon EKS 及其使用者共同的責任。使用者負責 IAM 角色和 Amazon VPC 子網路的必備基礎設施，以及其他必須事先提供的必要基礎設施。Amazon EKS 偵測此基礎設施和叢集的組態變更。

若要在 Amazon EKS 主控台中存取叢集的運作狀態，請在從 Amazon EKS 叢集詳細資訊頁面存取的可觀測性儀表板的叢集運作狀態索引標籤中，尋找名為運作狀態問題的資料表。 也可以透過在 EKS API 中呼叫 `DescribeCluster`動作來使用此資料，例如從 AWS 命令列界面內。

為什麼我應該使用此功能？

> 您可以提高對 Amazon EKS 叢集運作狀態的可見性、快速診斷和修正任何問題，而無需花時間偵錯或開啟 AWS 支援案例。例如：您意外刪除了 Amazon EKS 叢集的子網路，Amazon EKS 將無法建立跨帳戶網路介面和 Kubernetes AWS CLI 命令，例如 `kubectl` exec 或 `kubectl`

log。這些操作將會失敗並顯示錯誤：`"Error from server: error dialing backend: remote error: tls: internal error."`。這時會出現內容為此的 Amazon EKS 運作狀態問題：`subnet-da60e280 was deleted: could not create network interface`。

此功能與其他 AWS 服務有何關聯或搭配？

IAM 角色和 Amazon VPC 子網路是叢集運作狀態對其進行問題偵測的其中兩項必備基礎設施。如果這些資源設定不正確，則此功能將傳回詳細錯誤資訊。

具有運作狀態問題的叢集會產生費用嗎？

是的。每個 Amazon EKS 叢集均以標準 Amazon EKS 定價計費。但叢集運作狀態功能是免費提供的。

此功能是否適用於 AWS Outposts 上的 Amazon EKS 叢集？

是，在 AWS 雲端中偵測到 EKS 叢集的叢集問題，包括 AWS Outpost 上的擴充叢集和 AWS Outpost 上的本機叢集。叢集運作狀態不會偵測 Amazon EKS Anywhere 或 Amazon EKS Distro (EKS-D) 的問題。

偵測到新問題時，我可以收到通知嗎？

是。偵測到新的叢集運作狀態問題時， AWS 會傳送電子郵件和個人運作狀態儀表板通知。

主控台是否會為我的運作狀態問題提供警告？

是。任何存在運作狀態問題的叢集都會在主控台頂部包含一則橫幅。

前兩欄是 API 回應值所需的內容。Health ClusterIssue 物件的第三個欄位是 resourceIds，其傳回取決於問題類型。

代碼	訊息	ResourceIds	叢集可復原？
SUBNET_NOT_FOUND	我們找不到目前與您的叢集相關聯的一或多個子網路。呼叫 Amazon EKS update-cluster-config API 以更新子網路。	子網路 ID	是
SECURITY_GROUP_NOT_FOUND	我們找不到目前與您的叢集相關聯的一或多個安全群組。請	安全群組 ID	是

代碼	訊息	ResourceIds	叢集可復原？
	呼叫 Amazon EKS update-cluster-config API 以更新安全群組。		
IP_NOT_AVAILABLE	與叢集關聯的一個或多個子網路沒有足夠的可用 IP 地址供 Amazon EKS 執行叢集管理操作。請使用 Amazon EKS update-cluster-config API 釋放子網路中的地址或將其他子網路關聯到叢集。	子網路 ID	是
VPC_NOT_FOUND	我們找不到與您的叢集相關聯的 VPC。您必須刪除並重新建立叢集。	VPC ID	否
ASSUME_ROLE_ACCESS_DENIED	叢集未使用 Amazon EKS 服務連結角色。我們無法擔任與您叢集相關聯的角色，以執行必要的 Amazon EKS 管理操作。請檢查相應角色是否存在並具有所需的信任政策。	叢集 IAM 角色	是

代碼	訊息	ResourceIds	叢集可復原？
PERMISSIO N_ACCESS_DENIED	叢集未使用 Amazon EKS 服務連結角色。與叢集關聯的角色未授予 Amazon EKS 執行所需管理操作需要的足夠許可。請檢查連接到叢集角色的政策以及是否套用了任何單獨的拒絕政策。	叢集 IAM 角色	是
ASSUME_RO LE_ACCESS _DENIED_U SING_SLR	我們無法假設 Amazon EKS 叢集管理service-linked-role。請檢查相應角色是否存在並具有所需的信任政策。	Amazon EKS 服務連結角色	是
PERMISSIO N_ACCESS_ DENIED_USING_SLR	Amazon EKS 叢集管理服務連結角色未授予 Amazon EKS 執行所需管理操作需要的足夠許可。請檢查連接到叢集角色的政策以及是否套用了任何單獨的拒絕政策。	Amazon EKS 服務連結角色	是
OPT_IN_REQUIRED	您的帳戶沒有 Amazon EC2 服務訂閱。請在帳戶設定頁面中更新帳戶訂閱。	N/A	是
STS_REGIO NAL_ENDPO INT_DISABLED	STS 區域端點已停用。請啟用相應端點，以讓 Amazon EKS 能夠執行所需的叢集管理操作。	N/A	是

代碼	訊息	ResourceIds	叢集可復原？
KMS_KEY_DISABLED	與叢集相關聯的 AWS KMS 金鑰已停用。請重新啟用該金鑰以復原叢集。	KMS 金鑰 Arn	是
KMS_KEY_NOT_FOUND	我們找不到與您的叢集相關聯的 AWS KMS 金鑰。您必須刪除並重新建立叢集。	KMS 金鑰 ARN	否
KMS_GRANT_REVOKED	與您的叢集相關聯的 AWS KMS 金鑰授予會遭到撤銷。您必須刪除並重新建立叢集。	KMS 金鑰 Arn	否

使用開放原始碼專案擴展 Amazon EKS 功能

這些開放原始碼專案擴展了在 或 外部執行的 Kubernetes 叢集的功能 AWS，包括由 Amazon EKS 管理的叢集。

支援部署到 EKS 的軟體

檢閱 Amazon EKS 文件時，您會在整個程序和範例中看到各種開放原始碼工具和軟體的參考。這些工具包括 Kubernetes Metrics Server 和 Cert Manager。

請注意，您選擇部署的任何第三方或開放原始碼軟體不在 AWS 支援協議範圍內。使用 Kubernetes 的好處是主動開放原始碼社群。我們建議您直接與相關的開放原始碼社群和專案維護者合作，為此類元件建立適當的支援管道。如需詳細資訊，請參閱與雲端原生運算基金會 (CNCF) 相關聯的漸進式和孵化專案。

Kubernetes 生態系統包含許多專案和元件，具有不同層級的社群支援、回應時間和預期使用案例。搭配 EKS 實作這些技術時，請確定您了解每個元件的支援矩陣。

AWS 會維護我們整合到 EKS 控制平面的開放原始碼元件。這包括我們全面的安全管道，涵蓋我們分發的所有容器映像和二進位檔的建置驗證、漏洞掃描、驗證測試和修補程式管理。例如， AWS 負責 Kubernetes API 伺服器。Amazon EKS 服務水準協議涵蓋 Kubernetes API 伺服器。您可以使用 Amazon Web Services Support Plan 來解決 Kubernetes API 伺服器的問題，或取得一般指引。

您需要仔細檢閱各種 Amazon EKS 附加元件提供的支援。 AWS 附加元件是 完全支援的唯一 Amazon EKS 附加元件類型 AWS。 AWS Marketplace 附加元件主要由 AWS 合作夥伴支援。社群附加元件從獲得基本生命週期支援 AWS。如需詳細資訊，請參閱附加元件支援。

無論類型為何，每個 EKS 附加元件都會從 EKS 獲得基本生命週期支援，包括 Marketplace 附加元件。基本生命週期支援包括安裝和解除安裝 附加元件。如需可用 Amazon EKS 附加元件類型和相關支援層級的詳細資訊，請參閱 Amazon EKS 附加元件的支援範圍。若要檢視 完全支援的附加元件 AWS，請參閱 Amazon Web Services 附加元件。

- 如需我們安全實務和支援界限的詳細資訊，請參閱 Amazon EKS 中的安全。
- 如需透過 Amazon EKS 附加元件提供的社群和 AWS 市場附加元件的詳細資訊，請參閱 EKS 附加元件支援。

管理工具

Amazon EKS 和 Kubernetes 叢集的相關管理工具。

eksctl

`eksctl` 是一種簡單的 CLI 工具，可在 Amazon EKS 上建立叢集。

- 專案 URL
- 專案文件
- AWS 開放原始碼部落格：eksctl：Amazon EKS 叢集搭配一個命令

AWS 適用於 Kubernetes 的 控制器

使用 Kubernetes 的 AWS 控制器，您可以直接從 Kubernetes 叢集建立和管理 AWS 資源。

- 專案 URL
- AWS 開放原始碼部落格：AWS 適用於 Kubernetes 的 service Operator 現已推出

Flux CD

Flux 是一個工具，您可以用於使用 Git 來管理您的叢集組態。其使用叢集中的運算子來觸發 Kubernetes 內部的部署。如需運算子的詳細資訊，請參閱 GitHub 上的 OperatorHub.io。

- 專案 URL
- 專案文件

Kubernetes 專用 CDK

使用適用於 Kubernetes 的 CDK (cdk8s)，您可以使用熟悉的程式設計語言定義 Kubernetes 應用程式和元件。cdk8s 應用程式會合成為標準的 Kubernetes 資訊清單，可套用至任何 Kubernetes 叢集。

- 專案 URL
- 專案文件
- AWS 容器部落格：推出 cdk8s+：Kubernetes 物件的意圖驅動 APIs

聯網

Amazon EKS 和 Kubernetes 叢集的相關聯網專案。

Kubernetes 專用 Amazon VPC CNI 外掛程式

Amazon EKS 透過 Kubernetes 專用 Amazon VPC CNI 外掛程式支援原生 VPC 聯網。外掛程式會將 VPC 中的 IP 地址指派給每個 Pod。

- 專案 URL
- 專案文件

Kubernetes 的AWS Load Balancer控制器

The AWS Load Balancer 控制器可協助管理 Kubernetes 叢集的 AWS Elastic Load Balancer。它透過佈建 AWS Application Load Balancer 來滿足 Kubernetes Ingress 資源。它透過佈建 AWS Network Load Balancer 來滿足 Kubernetes 服務資源。

- 專案 URL
- 專案文件

ExternalDNS

ExternalDNS 會將公開的 Kubernetes 服務和輸入與 DNS 供應商同步，包括 Amazon Route 53 和服務 AWS 探索。

- 專案 URL
- 專案文件

機器學習

Amazon EKS 和 Kubernetes 叢集的相關機器學習專案。

Kubeflow

適用於 Kubernetes 的機器學習工具組。

- 專案 URL
- 專案文件
- AWS 開放原始碼部落格：Amazon EKS 上的 Kubeflow

Auto Scaling

Amazon EKS 和 Kubernetes 叢集的相關自動調整規模專案。

Cluster Autoscaler

Cluster Autoscaler 是一種工具，可根據 CPU 和記憶體壓力，自動調整 Kubernetes 叢集的大小。

- 專案 URL
- 專案文件
- Amazon EKS 研討會：Cluster Autoscaler

Karpenter

Karpenter 是一種 Kubernetes Node Autoscaler，專為靈活性、效能和簡易性而打造。

- 專案 URL
- 專案文件
- Amazon EKS 研討會：Karpenter

Escalator

Escalator 是適用於 Kubernetes 的批次或任務最佳化水平自動擴展工具。

- 專案 URL
- 專案文件

監控

Amazon EKS 和 Kubernetes 叢集的相關監控專案。

Prometheus

Prometheus 是一種開放原始碼系統監控和警示工具組。

- 專案 URL
- 專案文件
- Amazon EKS 研討會：https://eksworkshop.com/intermediate/240_monitoring/

持續整合 / 持續部署

Amazon EKS 和 Kubernetes 叢集的相關 CI/CD 專案。

Jenkins X

Amazon EKS 和 Kubernetes 叢集上適用於現代雲端應用程式的 CI/CD 解決方案。

- 專案 URL
- 專案文件

了解 Amazon EKS 新功能和藍圖

您可以捲動至 最新消息頁面上的最新消息摘要，以了解 Amazon EKS AWS新功能。您也可以在 GitHub 上檢視藍圖，讓您了解即將推出的功能和優先順序，以便規劃未來使用 Amazon EKS 的方式。您可以向我們提供有關藍圖優先順序的直接意見回饋。

文件歷史紀錄

下表說明 Amazon EKS 使用者指南的一些主要更新和新功能。若要在此表格取得新項目時接收通知，您可以使用 RSS 讀取器訂閱下列 URL：

```
https://docs.aws.amazon.com/eks/latest/userguide/doc-history.rss
```

變更	描述	日期
Amazon EKS 平台版本更新	這是具有安全性修正和增強功能的新平台版本。這包括 Kubernetes 1.33.2、1.32.6、1.31.10 和 1.30.14 的新修補程式版本。	2025 年 7 月 30 日
適用於多主目錄 Pod 的 VPC CNI 多 NIC 功能	Amazon EKS 會將多主目錄 Pod 新增至 VPC CNI。現在，您可以設定工作負載，以及從 EC2 執行個體上的每個 NIC 到每個 Pod 的 VPC CNI 指派 IP 地址。應用程式可以建立並行連線，以使用每個 NIC 的頻寬。每個網路界面都在與節點相同的子網路和安全群組中設定。先前，您需要使用 Multus CNI 執行多個其他 CNIs 來建立多主目錄 Pod。文件和執行此作業的步驟已移至 eks/latest/userguide/pod-multus.html。	2025 年 7 月 15 日
VPC CNI 疑難排解內容更新	已擴展 VPC CNI 中 Kubernetes 網路政策的故障診斷頁面。新增 CRDs和 RBAC 許可，以及 12 個已知問題。	2025 年 6 月 30 日

Amazon EKS 平台版本更新	這是具有安全性修正和增強功能的新平台版本。這些平台版本中沒有任何 Kubernetes 的新修補程式版本。	2025 年 6 月 26 日
AWS 受管政策更新	新增對 的ssmmessages:OpenDataChannel 許可AmazonEKSLocalOutpostServiceRolePolicy 。	2025 年 6 月 26 日
AWS 受管政策更新	新增對 AmazonEKSServiceRolePolicy 和 的許可AmazonEKSComputePolicy ，以允許 Amazon EKS Auto 模式使用您帳戶中的 EC2 隨需容量預留來啟動執行個體。此外，將許可新增至 AmazonEKSComputePolicy ，讓 Amazon EKS 代表您建立 EC2 Spot 服務連結角色。	2025 年 6 月 20 日
受管政策更新	新增AmazonEKSDashboardConsoleReadOnly 政策。	2025 年 6 月 19 日
Amazon EKS Auto Mode 更新至 NodeClass	自動模式節點的NodeClass 範本新增了個別 Pod 子網路的組態。這會新增選用金鑰 podSubnetSelectorTerms 和 podSecurityGroupSelectorTerms ，以設定 Pod 的子網路和安全群組。	2025 年 6 月 13 日

使用 EKS Pod 身分鎖定次要和跨帳戶角色	Amazon EKS 會將目標 IAM 角色新增至 EKS Pod 身分，以進行自動角色鏈結。您可以使用此功能自動擔任另一個帳戶中的角色，EKS Pod Identity 會輪換臨時憑證。每個 Pod Identity 關聯都必須在相同帳戶中具有 IAM 角色，才能先擔任，然後使用該角色來擔任目標角色。	2025 年 6 月 11 日
Amazon EKS 平台版本更新	這是具有安全性修正和增強功能的新平台版本。這些平台版本中沒有任何 Kubernetes 的新修補程式版本。	2025 年 6 月 11 日
Amazon EKS AWS 區域擴展	Amazon EKS 現已在亞太區域（台北）(ap-east-2) AWS 區域提供。	2025 年 6 月 6 日
IPv6 新IPv6叢集雙堆疊公有端點的存取控制		2025 年 6 月 5 日
Amazon EKS 平台版本更新	這是具有安全性修正和增強功能的新平台版本。這包括 Kubernetes 1.32.5、1.31.9和 的新修補程式版本1.30.13。	2025 年 5 月 30 日
EKS 混合節點的新叢集洞察	Amazon EKS 新增了新的叢集洞見，以檢查混合節點的組態。這些洞見檢查會警告您有關現場部署節點和 Pod 以及叢集遠端網路組態的問題。	2025 年 5 月 29 日
Kubernetes 版本 1.33	新增了對新叢集和版本升級的 Kubernetes 版本1.33支援。	2025 年 5 月 29 日

Amazon FSx CSI 驅動程式的附加元件支援	您現在可以使用 AWS 管理主控台、 AWS CLI 和 API 來管理 Amazon FSx CSI 驅動程式。	2025 年 5 月 23 日
在主控台中編輯 Prometheus 抓取器	您現在可以在 Amazon EKS 主控台中編輯 Amazon Managed Service for Prometheus 湊集器。	2025 年 5 月 22 日
受管政策更新	新增AmazonEKSDashboard ServiceRolePolicy 政策。	2025 年 5 月 21 日
Amazon EKS 平台版本更新	這是具有安全性修正和增強功能的新平台版本。這些平台版本中沒有任何 Kubernetes 的新修補程式版本。	2025 年 5 月 16 日
Amazon FSx for Lustre 效能的新頁面	新增了有關最佳化 Amazon FSx for Lustre 效能的詳細資訊的新主題。	2025 年 5 月 2 日
Amazon EKS Auto Mode 更新至 NodeClass	自動模式節點的NodeClass 範本新增了轉送網路代理的組態。這會新增選用金鑰advancedNetworking 來設定 HTTPS 代理。	2025 年 4 月 30 日
混合節點的 Bottlerocket	Bottlerocket 現在可用於 EKS 混合節點。	2025 年 4 月 29 日
Amazon EKS 平台版本更新	這是具有安全性修正和增強功能的新平台版本。這些平台版本中沒有任何 Kubernetes 的新修補程式版本。	2025 年 4 月 29 日

混合聯網的新概念頁面	新增 EKS 混合節點概念的頁面。這些詳細介紹了現場部署和雲端聯網的圖表。	2025 年 4 月 18 日
AWS 受管政策更新	新增許可至 AmazonEKS ClusterPolicy ，以允許 Amazon EKS 彈性網路介面由 VPC CNI 建立。這是必要的，以便 EKS 可以在 VPC CNI 意外退出時清除留下的彈性網路界面。	2025 年 4 月 16 日
AWS 受管政策更新	新增許可至 AmazonEKS ServiceRolePolicy ，以允許 EKS AI/ML 客戶將輸出規則新增至預設 EKS 叢集安全群組。	2025 年 4 月 14 日
EKS 混合節點的節點運作狀態	您可以在混合節點eks-node-monitoring-agent 上使用 ，從 版本開始1.2.0-eksbuild.1 。執行 eks-node-monitoring-agent 做為 Amazon EKS 附加元件，以偵測和顯示運作狀態問題。	2025 年 3 月 31 日

現有叢集的 EKS 混合節點	您現在可以新增、變更或移除現有叢集的混合節點組態。先前，您只能在建立新叢集時，將混合節點組態新增至新叢集。透過 Amazon EKS 混合節點，您可以使用內部部署和邊緣基礎設施作為 Amazon EKS 叢集中的節點。 AWS 管理 Amazon EKS 叢集的 AWS 託管 Kubernetes 控制平面，並管理在內部部署或邊緣環境中執行的混合節點。	2025 年 3 月 31 日
回復：使用叢集洞見防止意外升級	Amazon EKS 已暫時復原一項功能，在發生特定叢集洞見問題時，會要求您使用`--force`旗標來升級叢集。如需詳細資訊，請參閱 GitHub 上對更新叢集版本強制執行升級洞察的暫時復原。	2025 年 3 月 28 日
Bottlerocket FIPS AMIs	Bottlerocket FIPS AMIs 現在可在標準受管節點群組中使用。	2025 年 3 月 27 日
AWS 受管政策更新	新增許可至 `AmazonEKSServiceRolePolicy` ，以允許 Amazon EKS 終止由自動模式建立的 EC2 執行個體。新增許可至 `AmazonEKSServiceRolePolicy` ，以允許 Amazon EKS 終止由自動模式建立的 EC2 執行個體。	2025 年 2 月 28 日

更新受管節點群組的策略	您現在可以使用更新策略來設定受管節點群組的版本更新程序。這引入了在建立新的節點之前終止節點的最小更新策略，這在容量受限的環境中非常有用。預設更新策略會繼續現有的行為。	2025 年 1 月 27 日
Kubernetes 版本 1.32	新增了對新叢集和版本升級的 Kubernetes 版本1.32支援。	2025 年 1 月 23 日
Amazon EKS AWS 區域擴展	Amazon EKS 現已在亞太區域（泰國）區域 (ap-southeast-7) 和墨西哥（中部）(mx-central-1) AWS 區域提供。EKS API 的 EKS Auto 模式和 VPC 端點不適用於任一區域。	2025 年 1 月 14 日
AWS 受管政策更新	新增多個許可至 AmazonEBSCSIDriverPolicy ，以允許 Amazon EBS CSI 驅動程式還原所有快照、在 EBS 磁碟區上啟用快速快照還原 (FSR)，以及修改磁碟區上的標籤。	2025 年 1 月 13 日
AWS 受管政策更新	新增許可至 AmazonEKSLoadBalancingPolicy 。	2024 年 12 月 26 日
已更新叢集洞察	Amazon EKS 升級洞察現在將警告更多叢集運作狀態和版本相容性問題。它可以偵測不同 Kubernetes 和 Amazon EKS 元件之間的問題kubelet，例如 kube-proxy 、 和 Amazon EKS 附加元件。	2024 年 12 月 20 日

節點監控代理程式和自動修復	您可以使用新的 `eks-node-monitoring-agent` 做為 Amazon EKS 附加元件來偵測和顯示運作狀態問題。您也可以啟用節點自動修復，以便在偵測到問題時自動取代節點。	2024 年 12 月 16 日
Amazon EKS 混合節點	您現在可以執行連線至 Amazon EKS 叢集的現場部署節點。透過 Amazon EKS 混合節點，您可以使用內部部署和邊緣基礎設施作為 Amazon EKS 叢集中的節點。 AWS 管理 Amazon EKS 叢集的 AWS 託管 Kubernetes 控制平面，並管理在內部部署或邊緣環境中執行的混合節點。	2024 年 12 月 1 日
Amazon EKS Auto 模式	Amazon EKS Auto Mode 可完全自動化 Kubernetes 叢集基礎設施管理，以進行運算、儲存和聯網 AWS。它透過自動佈建基礎設施、選取最佳運算執行個體、動態擴展資源、持續最佳化成本、修補作業系統，以及與 AWS 安全服務整合，來簡化 Kubernetes 管理。	2024 年 12 月 1 日
Amazon EKS 平台版本更新	這是具有安全性修正和增強功能的新平台版本。這包括 Kubernetes `1.31.2`、`1.30.6`、`1.29.10` 和 `1.28.15` 的新修補程式版本。	2024 年 11 月 22 日

AWS 受管政策更新	更新AWSServiceRoleForAmazonEKSNodegroup 以相容於中國區域。	2024 年 11 月 22 日
Outposts 上的本機叢集現在可使用 Kubernetes AWS 1.30 版	您現在可以使用 Kubernetes 1.30 版在 AWS Outposts 上建立 Amazon EKS 本機叢集。	2024 年 11 月 21 日
AWS 受管政策更新	EKS 已更新 AWS 受管政策 AmazonEKSLocalOutpostClusterPolicy 。新增ec2:DescribeAvailabilityZones 許可，讓叢集控制平面上的 AWS Cloud Controller Manager 可以識別每個節點所在的可用區域。	2024 年 11 月 21 日
使用 FIPS 140-3 的 Bottlerocket AMIs	Bottlerocket AMIs已預先設定為使用 FIPS 140-3 驗證的密碼編譯模組。這包括 Amazon Linux 2023 核心加密 API 密碼編譯模組和 AWS-LC 密碼編譯模組。	2024 年 11 月 20 日
AWS 受管政策更新	已更新AWSServiceRoleForAmazonEKSNodegroup 政策，以允許 Amazon EKS 受ec2:RebootInstances 管節點群組建立的執行個體。限制 Amazon EC2 資源的ec2:CreateTags 許可。	2024 年 11 月 20 日

可觀測性儀表板	可觀測性儀表板可協助您快速偵測、疑難排解和修復問題。AWS/EKS 命名空間中也有新的 CloudWatch vended 指標可用。	2024 年 11 月 18 日
AWS 受管政策更新	EKS 已更新 AWS 受管政策 AmazonEKSServiceRolePolicy 。新增 EKS 存取政策、負載平衡器管理和自動叢集資源清除的許可。	2024 年 11 月 16 日
在主控台中為支援 EKS Pod 身分的附加元件建立新角色	使用主控台建立或更新支援 EKS Pod 身分的附加元件時有新的步驟，您可以在其中自動產生具有適當名稱、角色政策和附加元件信任政策的 IAM 角色。	2024 年 11 月 15 日
AWS Local Zones 中的受管節點群組	受管節點群組現在可以在 AWS Local Zones 中建立。	2024 年 11 月 15 日
有可用的新指標	API 群組 下有可用的新指標metrics.eks.amazonaws.com 。	2024 年 11 月 11 日
AWS 受管政策更新	EKS 已更新 AWS 受管政策 AmazonEKSComputePolicy 。已更新 iam:AddRoleToInstanceProfile 動作的資源許可。	2024 年 11 月 7 日
AWS 受管政策更新	EKS 新增了新的 AWS 受管政策： AmazonEKSComputePolicy	2024 年 11 月 1 日

AWS 受管政策更新	新增許可至 AmazonEKSClusterPolicy 。新增允許 Amazon EKS 將拓撲資訊連接至節點做為標籤的ec2:DescribeInstanceTopology 許可。	2024 年 11 月 1 日
AWS 受管政策更新	EKS 新增了新的 AWS 受管政策： AmazonEKSBlockStoragePolicy	2024 年 10 月 30 日
AWS 受管政策更新	EKS 新增了新的 AWS 受管政策： AmazonEKSLoadBalancingPolicy	2024 年 10 月 30 日
AWS 受管政策更新	新增AmazonEKSServiceRolePolicy 允許 Amazon EKS 將指標發佈至 Amazon CloudWatch 的cloudwatch:PutMetricData 許可。	2024 年 10 月 29 日
AWS 受管政策更新	EKS 新增了新的 AWS 受管政策： AmazonEKSNetworkingPolicy	2024 年 10 月 28 日
新IPv6叢集的雙堆疊端點	使用雙堆疊的eks-cluster. *region*.api.aws端點連接到新IPv6叢集。當您描述這些叢集時，會傳回此端點。 kubectl 和 IPv4、IPv6或雙堆疊環境中的其他 Kubernetes API 用戶端可以解析並連線至公有或私有叢集的這些端點。	2024 年 10 月 21 日

AWS 受管政策更新	已將 autoscaling:ResumeProcesses 、 autoscaling:SuspendProcesses 和相關聯的許可新增至AWSServiceRoleForAmazonEKSNodegroup 中國區域的 ，以與 Amazon Application Recovery Controller for EKS 整合。其他區域沒有變更。	2024 年 10 月 21 日
AL2023 加速 AMIs	您現在可以針對以 AL2023 為基礎的 AMIs 使用加速NVIDIA和 AWS Neuron 執行個體。	2024 年 10 月 11 日
新的來源格式	我們已切換到具有一些配置變更的新來源格式。我們正在解決暫時的次要格式化問題。	2024 年 10 月 10 日
AWS 受管政策更新	新增許可至 AmazonEKSServicePolicy 和 AmazonEKSServiceRolePolicy 。新增ec2:GetSecurityGroupsForVpc 和相關聯的標籤許可，以允許 EKS 讀取安全群組資訊並更新相關標籤。	2024 年 10 月 10 日
AWS 受管政策更新 - 新政策	EKS 新增了新的 AWS 受管政策。	2024 年 10 月 3 日
Kubernetes 版本 1.31	新增了對新叢集和版本升級的 Kubernetes 版本1.31支援。	2024 年 9 月 24 日
AWS 受管政策更新 - 更新現有政策	Amazon EKS 已更新現有的 AWS 受管政策。	2024 年 8 月 21 日

Kubernetes 1.29 版現已可用於 AWS Outposts 上的本機叢集	您現在可以使用 Kubernetes 1.29 版在 AWS Outposts 上建立 Amazon EKS 本機叢集。	2024 年 8 月 20 日
EKS Pod Identity in AWS GovCloud （美國）	Amazon EKS Pod 身分會將 IAM 角色與 Kubernetes 服務帳戶建立關聯。透過此功能，您不再需要提供延伸許可給節點 IAM 角色。如此一來，該節點上的 Pod 就可以呼叫 AWS APIs。與服務帳戶的 IAM 角色不同，EKS Pod 身分完全位於 EKS 內；您不需要 OIDC 身分提供者。	2024 年 8 月 14 日
案例驅動的內容更新	我們重新命名和更新了主題，以在整個指南中更具情境驅動。	2024 年 8 月 9 日
Amazon EKS 的雙堆疊 VPC 介面端點	您現在可以使用 IPv4和 IPv6 IP 地址和 DNS 名稱為 Amazon EKS 建立雙堆疊 VPC 介面端點。	2024 年 8 月 7 日
具有IPv6地址的 Amazon EKS APIs 的新雙堆疊端點	用於建立和管理叢集的 EKS API，以及叢集的 OIDC 發行者 URLs 具有新的雙堆疊端點。Amazon EKS API 的新 DNS 名稱eks.*region*.api.aws會解析為IPv4地址和IPv6地址。新叢集具有新的雙堆疊 OIDC 發行者 URL (oidc-eks. *region*.api.aws)。	2024 年 8 月 1 日
受管節點群組的容量區塊	您現在可以將容量區塊用於受管節點群組。	2024 年 7 月 1 日

Auto Scaling 群組指標集合預設為啟用	Amazon EKS 受管節點群組現在預設會啟用 Amazon EC2 Auto Scaling 群組指標，無需額外費用。先前，您必須執行數個步驟才能啟用此功能。	2024 年 6 月 28 日
AWS 受管政策更新 - 更新現有政策	Amazon EKS 已更新現有的 AWS 受管政策。	2024 年 6 月 27 日
Kubernetes 版本 1.26	Kubernetes 版本1.26現在支援擴充。	2024 年 6 月 12 日
AMI 資訊參考的改進	我們改善了 AMI 資訊參考，特別是 Bottlerocket。	2024 年 6 月 12 日
Kubernetes 版本 1.30	新增了對新叢集和版本升級的 Kubernetes 版本1.30支援。	2024 年 5 月 23 日
CoreDNS Autoscaling	CoreDNS Autoscaler 會根據節點和 CPU 核心的數量，動態調整 EKS 叢集中 CoreDNS 部署的複本數量。此功能適用於 CoreDNS v1.9和 EKS 發行版本 1.25及更新版本的最新平台版本。	2024 年 5 月 14 日
Amazon EKS 平台版本更新	這是具有安全性修正和增強功能的新平台版本。這包括 Kubernetes 1.29.4、1.28.9和 的新修補程式版本1.27.13。	2024 年 5 月 14 日
適用於 Windows 的 CloudWatch Container Insights 支援	Amazon CloudWatch 可觀測性運算子附加元件現在也允許叢集中 Windows 工作者節點上的 Container Insights。	2024 年 4 月 10 日
Kubernetes 概念	新增 Kubernetes 概念主題。	2024 年 4 月 5 日

重組存取和 IAM 內容	將與存取和 IAM 主題相關的現有頁面，例如身分驗證組態映射、存取項目、Pod ID 和 IRSA 移至新區段。修訂概觀內容。	2024 年 4 月 2 日
Bottlerocket 作業系統支援 Amazon S3 CSI 驅動程式	Amazon S3 CSI 驅動程式的掛載點現在與 Bottlerocket 相容。	2024 年 3 月 13 日
AWS 受管政策更新 - 更新現有政策	Amazon EKS 已更新現有的 AWS 受管政策。	2024 年 3 月 4 日
Amazon Linux 2023	Amazon Linux 2023 (AL2023) 是新的 Linux 作業系統，旨在為您的雲端應用程式提供安全、穩定且高效能的環境。	2024 年 2 月 29 日
EKS Pod Identity 和 IRSA 支援 Kubernetes 中的附屬項目 1.29	在 Kubernetes 中1.29，Amazon EKS 叢集提供附屬容器。服務帳戶或 EKS Pod Identity 的 IAM 角色支援附屬容器。如需附屬項目的詳細資訊，請參閱 Kubernetes 文件中的附屬容器。	2024 年 2 月 26 日
Kubernetes 版本 1.29	新增了對新叢集和版本升級的 Kubernetes 版本1.29支援。	2024 年 1 月 23 日
完整版本：Amazon EKS 擴充支援 Kubernetes 版本	擴充 Kubernetes 版本支援可讓您保留在特定 Kubernetes 版本超過 14 個月。	2024 年 1 月 16 日

AWS 雲端中的 Amazon EKS 叢集運作狀態偵測	Amazon EKS 會偵測叢集運作狀態中 Amazon EKS 叢集和叢集先決條件基礎設施的問題。您可以在 中檢視 EKS 叢集的問題， AWS Management Console 並在 EKS API 中檢視叢集`health`的 。這些問題是主控台偵測到且由主控台顯示之問題之外的問題。先前，叢集運作狀態僅適用於 AWS Outposts 上的本機叢集。	2023 年 12 月 28 日
叢集洞察	您現在可以從定期檢查取得有關您的叢集的建議。	2023 年 12 月 20 日
Amazon EKS AWS 區域擴展	Amazon EKS 現已在加拿大西部（卡加利）(`ca-west-1`) AWS 區域提供。	2023 年 12 月 20 日
您現在可以使用存取項目向 IAM 角色和使用者授予對叢集的存取權	在引進存取項目之前，您透過在 `aws-auth ConfigMap` 中新增項目來授予 IAM 角色和使用者對叢集的存取權。現在每個叢集都有一種存取模式，您可以根據排程轉變為使用存取項目。切換模式之後，您可以在 CLI、 AWS CloudFormation 和 AWS SDKs AWS 中新增存取項目來新增使用者。	2023 年 12 月 18 日
Amazon EKS 平台版本更新	這是具有安全性修正和增強功能的新平台版本。這包括 Kubernetes `1.28.4`、`1.27.8`、`1.26.11` 和 `1.25.16` 的新修補程式版本。	2023 年 12 月 12 日

Amazon S3 CSI 驅動程式掛載點	您現在可以在 Amazon EKS 叢集上安裝 Amazon S3 CSI 驅動程式的掛載點。	2023 年 11 月 27 日
建立叢集時開啟 Prometheus 指標	在 中 AWS Management Console，您現在可以在建立叢集時開啟 Prometheus 指標。您也可以在可觀測性索引標籤中檢視 Prometheus 湊集器詳細資訊。	2023 年 11 月 26 日
Amazon EKS Pod 身分識別	Amazon EKS Pod 身分會將 IAM 角色與 Kubernetes 服務帳戶建立關聯。透過此功能，您不再需要提供延伸許可給節點 IAM 角色。如此一來，該節點上的 Pod 就可以呼叫 AWS APIs。與服務帳戶的 IAM 角色不同，EKS Pod 身分完全位於 EKS 內；您不需要 OIDC 身分提供者。	2023 年 11 月 26 日
AWS 受管政策更新 - 更新現有政策	Amazon EKS 已更新現有的 AWS 受管政策。	2023 年 11 月 26 日
CSI 快照控制器	您現在可以安裝 CSI 快照控制器，以便與相容的 CSI 驅動程式 (例如 Amazon EBS CSI 驅動程式) 搭配使用。	2023 年 11 月 17 日
ADOT 運算子主題重寫	Amazon EKS 附加元件對 ADOT Operator 的支援區段與 AWS Distro for OpenTelemetry 文件備援。我們將剩餘的重要資訊遷移到該資源中，以減少過時和不一致的資訊。	2023 年 11 月 14 日

Prometheus 指標的 CoreDNS EKS 附加元件支援	適用於 CoreDNS 的 EKS 附加元件的 v1.10.1-eksbuild.5 、 v1.9.3-eksbuild.9 和 v1.8.7-eksbuild.8 版本會在kube-dns服務中公開 CoreDNS 發佈指標的連接埠。這可讓您更輕鬆地在監控系統中包含 CoreDNS 指標。	2023 年 11 月 10 日
Amazon EKS CloudWatch 可觀測性運算子附加元件	新增的 Amazon EKS CloudWatch 可觀測性運算子頁面。	2023 年 11 月 6 日
美國東部 (俄亥俄) 推出適用於自我管理 P5 執行個體的容量區塊	在美國東部 (俄亥俄)，您現在可以將容量區塊用於自我管理 P5 執行個體。	2023 年 10 月 31 日
叢集支援修改子網路和安全群組	您可以更新叢集，變更叢集使用的子網路和安全群組。您可以從 AWS Management Console、最新版本的 AWS CLI、 AWS CloudFormation 和eksctl版本 v0.164.0-rc.0 或更新版本進行更新。這麼做才能為子網路提供更多可用 IP 地址，以成功升級叢集版本。	2023 年 10 月 24 日

叢集角色和受管節點群組角色支援客戶受管 AWS 身分和存取管理政策	您可以在叢集角色上使用自訂 IAM 政策，而不是 AmazonEKSClusterPolicy AWS 受管政策。此外，您可以在受管節點群組中的節點角色上使用自訂 IAM 政策，而不是 AmazonEKSWorkerNodePolicy AWS 受管政策。為符合嚴格的合規要求，請建立具有最低權限的政策。	2023 年 10 月 23 日
修復 eksctl 的安裝連結	在頁面移動後修復 eksctl 的安裝連結。	2023 年 10 月 6 日
預覽版本：Amazon EKS 擴充支援 Kubernetes 版本	擴充 Kubernetes 版本支援可讓您保留在特定 Kubernetes 版本超過 14 個月。	2023 年 10 月 4 日
移除對 AWS App Mesh 整合的參考	Amazon EKS 與 AWS App Mesh 的整合僅保留給 App Mesh 的現有客戶。	2023 年 9 月 29 日
Kubernetes 版本 1.28	新增了對新叢集和版本升級的 Kubernetes 版本1.28支援。	2023 年 9 月 26 日
現有的叢集支援在適用於 Kubernetes 的 Amazon VPC CNI 外掛程式中強制執行 Kubernetes 網路政策	您可以在現有叢集中使用 Kubernetes 網路政策搭配適用於 Kubernetes 的 Amazon VPC CNI 外掛程式，而不需要第三方解決方案。您可以在現有叢集中使用 Kubernetes 網路政策搭配適用於 Kubernetes 的 Amazon VPC CNI 外掛程式，而不需要第三方解決方案。	2023 年 9 月 15 日

CoreDNS Amazon EKS 附加元件支援修改 PDB	您可以在版本 和更新版本 v1.9.3-eksbuild.7 和更新版本 和更新版本中修改適用於 CoreDNS PodDisruptionBudget 的 EKS 附加元件v1.10.1-eksbuild.4 的 。	2023 年 9 月 15 日
Amazon EKS 支援共用子網路	用來在共用子網路中製作 Amazon EKS 叢集的新共用子網路要求與考量。	2023 年 9 月 7 日
「什麼是Amazon EKS？」的更新內容	加入了新的常見使用案例和架構主題。重新整理其他主題。	2023 年 9 月 6 日
適用於 Kubernetes 的 Amazon VPC CNI 外掛程式中的 Kubernetes 網路政策強制執行	您可以搭配適用於 Kubernetes 的 Amazon VPC CNI 外掛程式使用 Kubernetes 網路政策，而不需要第三方解決方案。您可以搭配適用於 Kubernetes 的 Amazon VPC CNI 外掛程式使用 Kubernetes 網路政策，而不需要第三方解決方案。	2023 年 8 月 29 日
Amazon EKS AWS 區域擴展	Amazon EKS 現已在以色列（特拉維夫）(il-central-1) AWS 區域提供。	2023 年 8 月 1 日
適用於 Fargate 的可設定暫時性儲存	您可以增加在 Amazon EKS Fargate 上執行的每個 Pod 的暫時性儲存總量。	2023 年 7 月 31 日
Amazon EFS CSI 驅動程式的附加元件支援	您現在可以使用 AWS Management Console、 AWS CLI 和 API 來管理 Amazon EFS CSI 驅動程式。	2023 年 7 月 26 日

AWS 受管政策更新 - 新政策	Amazon EKS 新增了新的 AWS 受管政策。	2023 年 7 月 26 日
Outposts 上的本機叢集現在可使用 1.27、1.26、1.25 和 1.24 的 Kubernetes AWS 版本更新	Outposts 上的本機叢集現在可使用 Kubernetes 版本更新至 1.27.3、1.26.6、1.25.11 AWS 和 1.24.15	2023 年 7 月 20 日
Windows 節點的 IP 字首支援	將 IP 字首指派給節點可讓您在節點上託管的 Pod 數量遠高於將個別次要 IP 地址指派給節點時。	2023 年 7 月 6 日
Amazon FSx for OpenZFS CSI 驅動程式	您現在可以在 Amazon EKS 叢集上安裝 Amazon FSx for OpenZFS CSI 驅動程式。	2023 年 6 月 30 日
IPv4叢集中 Linux 節點上的 Pod 現在可以與IPv6端點通訊。	將 IPv6 地址指派給節點後，您的 Pod IPv4地址會轉譯為其執行節點地址的網路IPv6地址。	2023 年 6 月 19 日
AWS GovCloud (US) 區域中的 Windows 受管節點群組	在 AWS GovCloud (US) 區域中，Amazon EKS 受管節點群組現在可以執行 Windows 容器。	2023 年 5 月 30 日
Kubernetes 版本 1.27	新增了對新叢集和版本升級的 Kubernetes 版本1.27支援。	2023 年 5 月 24 日
Kubernetes 版本 1.26	新增了對新叢集和版本升級的 Kubernetes 版本1.26支援。	2023 年 4 月 11 日
無網域 gMSA	您現在可以將無網域 gMSA 與 Windows Pod 搭配使用。	2023 年 3 月 27 日
Amazon EKS AWS 區域擴展	Amazon EKS 現已在亞太區域（墨爾本）(ap-southeast-4) AWS 區域提供。	2023 年 3 月 10 日

Amazon File Cache CSI 驅動程式	您現在可以在 Amazon EKS 叢集上安裝 Amazon File Cache CSI 驅動程式。	2023 年 3 月 3 日
Kubernetes 1.25 版現已適用於 AWS Outposts 上的本機叢集	您現在可以使用 Kubernetes 版本 – 在 Outpost 1.22 上建立 Amazon EKS 本機叢集1.25。	2023 年 3 月 1 日
Kubernetes 版本 1.25	新增了對新叢集和版本升級的 Kubernetes 版本1.25支援。	2023 年 2 月 22 日
AWS 受管政策更新 - 更新現有政策	Amazon EKS 已更新現有的 AWS 受管政策。	2023 年 2 月 7 日
Amazon EKS AWS 區域擴展	Amazon EKS 現已在亞太區域（海德拉巴）()ap-south-2 、歐洲（蘇黎世）(eu-central-2 ）和歐洲（西班牙）(eu-south-2 ）AWS 區域提供。	2023 年 2 月 6 日
Kubernetes 版本 1.21 – 1.24 現在可供 AWS Outposts 上的本機叢集使用。	您現在可以使用 Kubernetes 版本 – 在 Outpost 1.21 上建立 Amazon EKS 本機叢集1.24。以前，只有 1.21 版本可用。	2023 年 1 月 17 日
Amazon EKS 現在支援 AWS PrivateLink	您可以使用 an AWS PrivateLink 在 VPC 和 Amazon EKS 之間建立私有連線。	2022 年 12 月 16 日
受管節點群組 Windows 支援	您現在可以將 Windows 用於 Amazon EKS 受管節點群組。	2022 年 12 月 15 日
來自獨立軟體廠商的 Amazon EKS 附加元件現在可在 AWS Marketplace 中使用	您現在可以透過 AWS Marketplace 從獨立軟體供應商瀏覽和訂閱 Amazon EKS 附加元件。	2022 年 11 月 28 日

AWS 受管政策更新 - 更新現有政策	Amazon EKS 已更新現有的 AWS 受管政策。	2022 年 11 月 17 日
Kubernetes 版本 1.24	新增了對新叢集和版本升級的 Kubernetes 版本1.24支援。	2022 年 11 月 15 日
Amazon EKS AWS 區域擴展	Amazon EKS 現已在中東（阿拉伯聯合大公國）(me-central-1) AWS 區域提供。	2022 年 11 月 3 日
AWS 受管政策更新 - 更新現有政策	Amazon EKS 已更新現有的 AWS 受管政策。	2022 年 10 月 24 日
AWS 受管政策更新 - 更新現有政策	Amazon EKS 已更新現有的 AWS 受管政策。	2022 年 10 月 20 日
AWS Outpost 上的本機叢集現已推出	您現在可以在 Outpost 上建立 Amazon EKS 本機叢集。	2022 年 9 月 19 日
Fargate vCPU 型配額	Fargate 正在從 Pod 型配額轉換為 vCPU 型配額。	2022 年 9 月 8 日
AWS 受管政策更新 - 更新現有政策	Amazon EKS 已更新現有的 AWS 受管政策。	2022 年 8 月 31 日
成本監控	Amazon EKS 現在支援 Kubecost，可讓您監控依 Kubernetes 資源細分的成本，包括 Pod、節點、命名空間和標籤。	2022 年 8 月 24 日
AWS 受管政策更新 - 新政策	Amazon EKS 新增了新的 AWS 受管政策。	2022 年 8 月 24 日
AWS 受管政策更新 - 新政策	Amazon EKS 新增了新的 AWS 受管政策。	2022 年 8 月 23 日

帳單的標籤資源	對所有叢集新增了 aws:eks:cluster-name 產生的成本分配標籤支援。	2022 年 8 月 16 日
Fargate 設定檔萬用字元	在命名空間、標籤索引鍵和標籤值的選擇器條件中，新增了對 Fargate 設定檔萬用字元的支援。	2022 年 8 月 16 日
Kubernetes 版本 1.23	新增了對新叢集和版本升級的 Kubernetes 版本1.23支援。	2022 年 8 月 11 日
在 中檢視 Kubernetes 資源 AWS Management Console	您現在可以使用 檢視部署到叢集的 Kubernetes 資源的相關資訊 AWS Management Console。	2022 年 5 月 3 日
Amazon EKS AWS 區域擴展	Amazon EKS 現已在亞太區域 （ 雅加達 ）(ap-southeast-3) AWS 區域提供。	2022 年 5 月 2 日
可觀測性頁面和 ADOT 附加元件支援	新增可觀測性頁面和 AWS Distro for OpenTelemetry (ADOT)。	2022 年 4 月 21 日
Kubernetes 版本 1.22	新增了對新叢集和版本升級的 Kubernetes 版本1.22支援。	2022 年 4 月 4 日
AWS 受管政策更新 - 新政策	Amazon EKS 新增了新的 AWS 受管政策。	2022 年 4 月 4 日
新增 Fargate Pod 修補詳細資訊	升級 Fargate Pod 時，Amazon EKS 會先根據您的 Pod 中斷預算嘗試移出 Pod。您可以建立事件規則，在刪除 Pod 之前對失敗的移出做出反應。	2022 年 4 月 1 日

完整版本：Amazon EBS CSI 驅動程式的附加元件支援	您現在可以使用 、 AWS Management Console AWS CLI 和 API 來管理 Amazon EBS CSI 驅動程式。	2022 年 3 月 31 日
AWS Outposts 內容更新	在 AWS Outposts 上部署 Amazon EKS 叢集的說明。	2022 年 3 月 22 日
AWS 受管政策更新 - 更新現有政策	Amazon EKS 已更新現有的 AWS 受管政策。	2022 年 3 月 21 日
Windows `containerd` 支援	您現在可以選擇 Windows 節點的 `containerd` 執行時間。	2022 年 3 月 14 日
在安全文件中新增 Amazon EKS 連接器考量事項	描述與連接叢集相關的共同責任模型。	2022 年 2 月 25 日
將IPv6地址指派給您的 Pod 和服務	您現在可以建立 `1.21`或更新版本的叢集，將IPv6地址指派給您的 Pod 和服務。	2022 年 1 月 6 日
AWS 受管政策更新 - 更新現有政策	Amazon EKS 已更新現有的 AWS 受管政策。	2021 年 12 月 13 日
預覽版本：Amazon EBS CSI 驅動程式的附加元件支援	您現在可以使用 AWS Management Console、 AWS CLI 和 API 來預覽，以管理 Amazon EBS CSI 驅動程式。	2021 年 12 月 9 日
Karpenter Autoscaler 支援	您現在可以使用 Karpenter 開放原始碼專案來自動調整節點。	2021 年 11 月 29 日
Fargate 記錄支援 Fluent Bit Kubernetes 篩選條件	您現在可以將 Fargate 記錄與 Fluent Bit Kubernetes 篩選條件搭配使用。	2021 年 11 月 10 日

控制平面中提供的 Windows 支援	您現在可在控制平面中使用 Windows 支援。您不再需要於資料平面中將其啟用。	2021 年 11 月 9 日
新增 Bottlerocket 作為受管節點群組的 AMI 類型	Bottlerocket 以往僅能作為自我管理節點選項使用。現在，它可以設定為受管節點群組，減少滿足節點合規要求所需的工作量。	2021 年 10 月 28 日
DL1 驅動程式支援	自訂 Amazon Linux AMI 現在支援 Amazon Linux 2 深度學習工作負載。此支援操作允許一般內部部署或雲端基準組態設定。	2021 年 10 月 25 日
VT1 視訊支援	自訂 Amazon Linux AMI 現在支援某些發行版本的 VT1。此支援功能會在您的 Amazon EKS 叢集上公告 Xilinx U30 裝置。	2021 年 9 月 13 日
Amazon EKS Connector 現已推出	您可以使用 Amazon EKS 連接器來註冊任何符合標準的 Kubernetes 叢集，並將其連接到 ， AWS 並在 Amazon EKS 主控台中將其視覺化。	2021 年 9 月 8 日
Amazon EKS Anywhere 現已推出	Amazon EKS Anywhere 是 Amazon EKS 的新部署選項，可用於建立和操作內部部署的 Kubernetes 叢集。	2021 年 9 月 8 日
Amazon FSx for NetApp ONTAP CSI 驅動程式	已新增主題，其中摘要說明 Amazon FSx for NetApp ONTAP CSI 驅動程式，並提供其他參考的連結。	2021 年 9 月 2 日

受管節點群組現在會自動計算節點的 Amazon EKS 建議 Pod 上限	受管節點群組現在會自動為您在沒有啟動範本的情況下部署的節點，或使用您尚未指定 AMI ID 的啟動範本來計算 Amazon EKS 最大 Pod。	2021 年 8 月 30 日
移除附加元件設定的 Amazon EKS 管理，而不移除 Amazon EKS 附加元件軟體	您現在可以移除 Amazon EKS 附加元件，而無需從叢集移除附加元件軟體。	2021 年 8 月 20 日
使用 Multus 建立多主目錄 Pod	您現在可以使用 Multus 將多個網路介面新增至 Pod。	2021 年 8 月 2 日
將更多 IP 地址新增至您的 Linux Amazon EC2 節點	您現在可以將更多的 IP 地址新增至您的 Linux Amazon EC2 節點。這表示您可以在每個節點上執行較高密度的 Pod。您現在可以將更多的 IP 地址新增至您的 Linux Amazon EC2 節點。這表示您可以在每個節點上執行較高密度的 Pod。	2021 年 7 月 27 日
`containerd` 執行階段啟動程序	Amazon EKS 最佳化加速 Amazon Linux Amazon Machine Image (AMI) 現在包含引導旗標，可用於在 Amazon EKS 最佳化和 Bottlerocket AMIs 中啟用`containerd` 執行期。此旗標可用於所有支援 Kubernetes 版本的 AMI 中。	2021 年 7 月 19 日
Kubernetes 版本 `1.21`	新增了 Kubernetes 版本`1.21`支援。	2021 年 7 月 19 日
新增受管政策主題	自 2021 年 6 月 17 日起所有 Amazon EKS IAM 受管政策以及已對其進行變更的清單。	2021 年 6 月 17 日

搭配 Fargate 使用 Pod 的安全群組	您現在可以將安全群組用於搭配 Fargate 的 Pod，以及搭配 Amazon EC2 節點使用。	2021 年 6 月 1 日
新增 CoreDNS 和 kube-proxy Amazon EKS 附加元件	Amazon EKS 現在可以幫助您管理叢集的 CoreDNS 和 kube-proxy Amazon EKS 附加元件。	2021 年 5 月 19 日
Kubernetes 版本 1.20	新增了對新叢集和版本升級的 Kubernetes 版本1.20支援。	2021 年 5 月 18 日
AWS2.2.0已發行Load Balancer控制器	您現在可以使用 AWS Load Balancer控制器，使用執行個體或 IP 目標建立 Elastic Load Balancer。	2021 年 5 月 14 日
受管節點群組的節點污點	Amazon EKS 現在支援將備註污點新增至受管節點群組。	2021 年 5 月 11 日
現有叢集的秘密加密	Amazon EKS 現在支援為現有叢集新增秘密加密。	2021 年 2 月 26 日
Kubernetes 版本 1.19	新增了對新叢集和版本升級的 Kubernetes 版本1.19支援。	2021 年 2 月 16 日
Amazon EKS 現在支援 OpenID Connect (OIDC) 身分提供者，作為對 1.16 或更新版本的叢集驗證使用者身分的方法。	OIDC 身分提供者可以與 AWS Identity and Access Management (IAM) 搭配使用或作為替代方案。	2021 年 2 月 12 日
在 中檢視節點和工作負載資源 AWS Management Console	您現在可以在 中檢視受管、自我管理和 Fargate 節點的詳細資訊，以及部署的 Kubernetes 工作負載 AWS Management Console。	2020 年 12 月 1 日

在受管節點群組中部署 Spot 執行個體類型	您現在可以將多個 Spot 或隨需執行個體類型部署到受管節點群組。	2020 年 12 月 1 日
Amazon EKS 現在可以管理叢集的特定附加元件	您可以自行管理附加元件，或允許 Amazon EKS 透過 Amazon EKS API 控制附加元件的啟動和版本。	2020 年 12 月 1 日
在多個輸入之間共用 ALB	您現在可以跨多個 Kubernetes 輸入共用 an AWS Application Load Balancer (ALB)。在過去，您必須為每個輸入部署個別的 ALB。	2020 年 10 月 23 日
NLB IP 目標支援	您現在可以部署具有 IP 目標的 Network Load Balancer。這表示您可以使用 NLB 來負載平衡網路流量到 Fargate Pod，以及直接到在 Amazon EC2 節點上執行的 Pod。	2020 年 10 月 23 日
Kubernetes 版本 1.18	新增了對新叢集和版本升級的 Kubernetes 版本1.18支援。	2020 年 10 月 13 日
為 Kubernetes 服務 IP 地址指派指定自訂 CIDR 區塊。	您現在可以指定 Kubernetes 從中指派服務 IP 地址的自訂 CIDR 區塊。	2020 年 9 月 29 日
將安全群組指派給個別 Pod	您現在可以將不同的安全群組與在許多 Amazon EC2 執行個體類型上執行的一些個別 Pod 建立關聯。	2020 年 9 月 9 日
在節點上部署 Bottlerocket	您現在可以部署執行 Bottlerocket 的節點。	2020 年 8 月 31 日

啟動 Arm 節點的功能已正式可用	您現在可以在受管節點群組和自我管理節點群組中啟動 Arm 節點。	2020 年 8 月 17 日
受管節點群組啟動範本和自訂 AMI	現在可以使用 Amazon EC2 啟動範本部署受管節點群組。您可以自行選擇是否為啟動範本指定自訂 AMI。	2020 年 8 月 17 日
AWS Fargate 的 EFS 支援	您現在可以搭配 AWS Fargate 使用 Amazon EFS。	2020 年 8 月 17 日
Amazon EKS 平台版本更新	這是具有安全性修正和增強功能的新平台版本。這包括將 Network Load Balancer 與 Kubernetes 版本 `1.15`或更新版本搭配使用`LoadBalancer` 時，對 類型的服務的 UDP 支援。如需詳細資訊，請參閱 GitHub 上的允許 UDP for AWS Network Load Balancer 問題。	2020 年 8 月 12 日
Amazon EKS AWS 區域擴展	Amazon EKS 現已在非洲（開普敦）(`af-south-1`）和歐洲（米蘭）(`eu-south-1`）AWS 區域提供。	2020 年 8 月 6 日
Fargate 用量指標	AWS Fargate 提供 CloudWatch 用量指標，可讓您查看帳戶對 Fargate 隨需資源的使用情形。	2020 年 8 月 3 日
Kubernetes 版本 `1.17`	新增了對新叢集和版本升級的 Kubernetes 版本`1.17`支援。	2020 年 7 月 10 日

使用適用於 Kubernetes 的 App Mesh 控制器，從 Kubernetes 內建立和管理 App Mesh 資源	您可以從 Kubernetes 內部建立和管理 App Mesh 資源。控制器也會自動將 Envoy 代理和初始化容器插入您部署的 Pod。	2020 年 6 月 18 日
Amazon EKS 現在支援 Amazon EC2 Inf1 節點	您可以將 Amazon EC2 Inf1 節點新增至叢集。	2020 年 6 月 4 日
Amazon EKS AWS 區域擴展	Amazon EKS 現已在 AWS GovCloud（美國東部）(`us-gov-east-1`) 和 AWS GovCloud（美國西部）(`us-gov-west-1`) AWS 區域提供。	2020 年 5 月 13 日
Amazon EKS 1.12不再支援 Kubernetes	Amazon EKS 1.12不再支援 Kubernetes 版本。請將任何 1.12 版叢集更新至 1.13 或更新版本，避免服務中斷。	2020 年 5 月 12 日
Kubernetes 版本 1.16	新增了對新叢集和版本升級的 Kubernetes 版本1.16支援。	2020 年 4 月 30 日
新增 AWSServiceRoleForAmazonEKS 服務連結角色	新增 AWSServiceRoleForAmazonEKS 服務連結角色。	2020 年 4 月 16 日
Kubernetes 版本 1.15	新增了對新叢集和版本升級的 Kubernetes 版本1.15支援。	2020 年 3 月 10 日
Amazon EKS AWS 區域擴展	Amazon EKS 現已在北京 (`cn-north-1`) 和寧夏 (`cn-northwest-1`) AWS 區域提供。	2020 年 2 月 26 日
FSx for Lustre CSI 驅動程式	新增在 Kubernetes Amazon EKS 叢集上安裝 FSx for Lustre CSI 1.14 驅動程式的主題。	2019 年 12 月 23 日

限制對叢集公有存取端點的網路存取	透過此更新，您可以使用 Amazon EKS 來限制與 Kubernetes API 伺服器之公有存取端點通訊的 CIDR 範圍。	2019 年 12 月 20 日
從 VPC 外部解析叢集的私有存取端點地址	透過此更新，您可以使用 Amazon EKS 從 VPC 外部解析 Kubernetes API 伺服器的私有存取端點。	2019 年 12 月 13 日
(Beta 版) Amazon EC2 A1 Amazon EC2 執行個體節點	啟動向您的 Amazon EKS 叢集註冊的 Amazon EC2 A1 Amazon EC2 執行個體節點。	2019 年 12 月 4 日
在 AWS Outposts 上建立叢集	Amazon EKS 現在支援在 AWS Outposts 上建立叢集。	2019 年 12 月 3 日
AWS Amazon EKS 上的 Fargate	Amazon EKS Kubernetes 叢集現在支援在 Fargate 上執行 Pod。	2019 年 12 月 3 日
Amazon EKS AWS 區域擴展	Amazon EKS 現已在加拿大（中部）(`ca-central-1`) AWS 區域提供。	2019 年 11 月 21 日
受管節點群組	Amazon EKS 受管節點群組會自動化 Amazon EKS Kubernetes 叢集節點 (Amazon EC2 執行個體) 的佈建和生命週期管理。	2019 年 11 月 18 日
Amazon EKS 平台版本更新	可解決 CVE-2019-11253 問題的新平台版本。	2019 年 11 月 6 日
Amazon EKS `1.11`不再支援 Kubernetes	Amazon EKS `1.11`不再支援 Kubernetes 版本。請將任何 `1.11` 版叢集更新至 `1.12` 或更新版本，避免服務中斷。	2019 年 11 月 4 日

Amazon EKS AWS 區域擴展	Amazon EKS 現已在南美洲（聖保羅）(sa-east-1) AWS 區域提供。	2019 年 10 月 16 日
Windows 支援	執行 Kubernetes 版本的 Amazon EKS 叢集1.14現在支援 Windows 工作負載。	2019 年 10 月 7 日
自動擴展	新增章節以涵蓋 Amazon EKS 叢集上支援的一些不同類型的 Kubernetes 自動擴展。	2019 年 9 月 30 日
Kubernetes Dashboard 更新	更新在 Amazon EKS 叢集上安裝 Kubernetes Dashboard 以使用 Beta 2.0版的主題。	2019 年 9 月 28 日
Amazon EFS CSI 驅動程式	新增在 Kubernetes Amazon EKS 叢集上安裝 Amazon EFS CSI 1.14 驅動程式的主題。	2019 年 9 月 19 日
Amazon EKS 最佳化 AMI ID 的 Amazon EC2 Systems Manager 參數	已新增使用 Amazon EC2 Systems Manager 參數擷取 Amazon EKS 最佳化 AMI ID 的相關主題。此參數讓您無需查詢 AMI ID。	2019 年 9 月 18 日
Amazon EKS 資源標記	您可管理 Amazon EKS 叢集的標記。	2019 年 9 月 16 日
Amazon EBS CSI 驅動程式	新增在 Kubernetes Amazon EKS 叢集上安裝 Amazon EBS CSI 1.14 驅動程式的主題。	2019 年 9 月 9 日
全新 Amazon EKS 最佳化 AMI 為 CVE-2019-9512 和 CVE-2019-9514 進行修補	Amazon EKS 已更新 Amazon EKS 最佳化 AMI，可解決 CVE-2019-9512 和 CVE-2019-9514 問題。	2019 年 9 月 6 日

宣佈在 Amazon EKS 1.11中棄用 Kubernetes	Amazon EKS 1.11已於 2019 年 11 月 4 日停止支援 Kubernetes 版本。	2019 年 9 月 4 日
Kubernetes 版本 1.14	新增了對新叢集和版本升級的 Kubernetes 版本1.14支援。	2019 年 9 月 3 日
服務帳戶的 IAM 角色	透過 Amazon EKS 叢集上服務帳戶的 IAM 角色，您可以建立 IAM 角色與 Kubernetes 服務帳戶的關聯。透過此功能，您不再需要提供延伸許可給節點 IAM 角色。如此一來，該節點上的 Pod 就可以呼叫 AWS APIs。	2019 年 9 月 3 日
Amazon EKS AWS 區域擴展	Amazon EKS 現已在中東（巴林）(me-south-1) AWS 區域提供。	2019 年 8 月 29 日
Amazon EKS 平台版本更新	可解決 CVE-2019-9512 和 CVE-2019-9514 問題的新平台版本。	2019 年 8 月 28 日
Amazon EKS 平台版本更新	可解決 CVE-2019-11247 和 CVE-2019-11249 問題的新平台版本。	2019 年 8 月 5 日
Amazon EKS 區域擴展	Amazon EKS 現已在亞太區域（香港）(ap-east-1) AWS 區域提供。	2019 年 7 月 31 日
Amazon EKS 1.10不再支援 Kubernetes	Amazon EKS 1.10不再支援 Kubernetes 版本。請將任何 1.10 版叢集更新至 1.11 或更新版本，避免服務中斷。	2019 年 7 月 30 日

在 ALB 傳入控制器上新增主題	適用於 Kubernetes 的 AWS ALB 傳入控制器是一種控制器，可在建立傳入資源時建立 ALB。	2019 年 7 月 11 日
全新 Amazon EKS 最佳化 AMI	從 AMI 移除不必要的 `kubectl` 二進位檔。	2019 年 7 月 3 日
Kubernetes 版本 `1.13`	新增了對新叢集和版本升級的 Kubernetes 版本`1.13`支援。	2019 年 6 月 18 日
全新 Amazon EKS 最佳化 AMI 為 `AWS-2019-005` 進行修補	Amazon EKS 已更新 Amazon EKS 最佳化 AMI，以解決 link：security/security-bulletins/AWS-2019-005/【AWS-2019-005，type="marketing"】 中所述的漏洞。	2019 年 6 月 17 日
宣布停止支援 Amazon EKS `1.10`中的 Kubernetes	Amazon EKS `1.10`已於 2019 年 7 月 22 日停止支援 Kubernetes 版本。	2019 年 5 月 21 日
Amazon EKS 平台版本更新	適用於 Kubernetes `1.11`和`1.10`叢集的新平台版本，可支援`kubelet`憑證中的自訂 DNS 名稱並改善`etcd`效能。	2019 年 5 月 21 日
`eksctl` 入門	本入門指南說明如何使用 `eksctl` 來安裝 Amazon EKS 入門所有所需的資源。這是簡單的命令列公用程式，可用於在 Amazon EKS 上建立和管理 Kubernetes 叢集。	2019 年 5 月 10 日

AWS CLI `get-token` 命令	`aws eks get-token` 命令已新增至 AWS CLI。您不再需要安裝 AWS IAM Authenticator for Kubernetes，即可為叢集 API 伺服器通訊建立用戶端安全字符。將您的 AWS CLI 安裝升級至最新版本，以使用此新功能。如需詳細資訊，請參閱《AWS 命令列界面使用者指南》中的安裝 AWS 命令列界面。	2019 年 5 月 10 日
Amazon EKS 平台版本更新	適用於 Kubernetes `1.12`叢集的新平台版本，可支援`kubelet`憑證中的自訂 DNS 名稱並改善`etcd`效能。這個版本修正了造成節點 `kubelet` 協助程式每隔幾秒便發出請求新憑證的錯誤。	2019 年 5 月 8 日
Prometheus 教學課程	已新增有關將 Prometheus 部署至 Amazon EKS 叢集的主題。	2019 年 4 月 5 日
Amazon EKS 控制平面記錄	透過此更新，您可以從 Amazon EKS 控制窗格中，直接取得稽核和診斷日誌。您可以在帳戶中使用這些 CloudWatch 日誌作為保護和執行叢集的參考。	2019 年 4 月 4 日
Kubernetes 版本 `1.12`	新增了對新叢集和版本升級的 Kubernetes 版本`1.12`支援。	2019 年 3 月 28 日
新增 App Mesh 入門指南	新增 App Mesh 和 Kubernetes 入門文件。	2019 年 3 月 27 日

Amazon EKS API 伺服器端點私有存取	新增停用 Amazon EKS 叢集 Kubernetes API 伺服器端點公有存取權的文件。	2019 年 3 月 19 日
新增安裝 Kubernetes 指標伺服器的主題	Kubernetes 指標伺服器是叢集中資源使用狀況資料的彙總工具。	2019 年 3 月 18 日
新增相關的開放原始碼專案清單	這些開放原始碼專案會擴展 Kubernetes 叢集在 上執行的功能 AWS，包括由 Amazon EKS 管理的叢集。	2019 年 3 月 15 日
新增有關本機安裝 Helm 的主題	Kubernetes 的 `helm` 套件管理工具可協助您安裝和管理 Kubernetes 叢集上的應用程式。本主題展示如何本機安裝和執行 `helm` 和 `tiller` 二進位檔案。這樣，您可以在本機系統上使用 Helm CLI 安裝和管理圖表。	2019 年 3 月 11 日
Amazon EKS 平台版本更新	將 Amazon EKS Kubernetes `1.11`叢集更新為修補程式層級`1.11.8`以解決 CVE-2019-1002100 的新平台版本。	2019 年 3 月 8 日
已提高叢集限制	Amazon EKS 已將您可以在 AWS 區域中建立的叢集數量從 3 增加到 50。	2019 年 2 月 13 日
Amazon EKS AWS 區域擴展	Amazon EKS 現已在歐洲（倫敦）()`eu-west-2` 、歐洲（巴黎）(`eu-west-3`) 和亞太區域（孟買）(`ap-south-1`) AWS 區域提供。	2019 年 2 月 13 日

全新 Amazon EKS 最佳化 AMI 為 ALAS-2019-1156 進行修補	Amazon EKS 已更新 Amazon EKS 最佳化 AMI，以解決 ALAS-2019-1156 中所述的漏洞。	2019 年 2 月 11 日
全新 Amazon EKS 最佳化 AMI 為 ALAS2-2019-1141 進行修補	Amazon EKS 已更新 Amazon EKS 最佳化 AMI，可解決 ALAS2-2019-1141 所述的 CVE 問題。	2019 年 1 月 9 日
Amazon EKS AWS 區域擴展	Amazon EKS 現已在亞太區域（首爾）(ap-northeast-2) AWS 區域提供。	2019 年 1 月 9 日
Amazon EKS 區域擴展	Amazon EKS 現已在以下其他 AWS 區域提供：歐洲（法蘭克福）(eu-central-1)、亞太區域（東京）(ap-northeast-1)、亞太區域（新加坡）(ap-southeast-1) 和亞太區域（雪梨）()ap-southeast-2 。	2018 年 12 月 19 日
Amazon EKS 叢集更新	已針對 Amazon EKS 叢集 Kubernetes 版本更新和節點取代。	2018 年 12 月 12 日
Amazon EKS AWS 區域擴展	Amazon EKS 現已在歐洲（斯德哥爾摩）(eu-north-1) AWS 區域提供。	2018 年 12 月 11 日
Amazon EKS 平台版本更新	新的平台版本將 Kubernetes 更新至修補程式層級1.10.11以定址連結：security/security-bulletins/AWS-2018-020/【CVE-2018-1002105，type="marketing"】。	2018 年 12 月 4 日

新增支援 ALB 傳入控制器的 1.0.0 版	ALB 輸入控制器會從 發行1.0.0具有正式支援的版本 AWS。	2018 年 11 月 20 日
新增對 CNI 網路組態的支援	適用於 Kubernetes 版本的 Amazon VPC CNI 外掛程式1.2.1現在支援次要 Pod 網路介面的自訂網路組態。	2018 年 10 月 16 日
新增對 MutatingAdmissionWebhook 和 ValidatingAdmissionWebhook 的支援	Amazon EKS 平台版本 1.10-eks.2 現已支援 MutatingAdmissionWebhook 和 ValidatingAdmissionWebhook 許可控制器。	2018 年 10 月 10 日
新增合作夥伴 AMI 資訊	Canonical 已與 Amazon EKS 合作，建立了可供您用於叢集的節點 AMI。	2018 年 10 月 3 日
新增 CLI AWSupdate-kubeconfig 命令的指示	Amazon EKS 已將 update-kubeconfig 新增至 AWS CLI，以簡化建立kubeconfig 檔案以存取叢集的程序。	2018 年 9 月 21 日
全新 Amazon EKS 最佳化 AMI	Amazon EKS 已更新 Amazon EKS 最佳化 AMI (支援及未支援 GPU)，提供了各項安全性修正和 AMI 最佳化。	2018 年 9 月 13 日
Amazon EKS AWS 區域擴展	歐洲 (愛爾蘭) (eu-west-1) 區域現可使用 Amazon EKS。	2018 年 9 月 5 日
Amazon EKS 平台版本更新	新的平台版本支援 Kubernetes 彙整層和 Horizontal Pod Autoscaler (HPA)。	2018 年 8 月 31 日

全新 Amazon EKS 最佳化 AMI 與 GPU 支援	Amazon EKS 已更新 Amazon EKS 最佳化 AMI，以使用新的 AWS CloudFormation 節點範本和引導指令碼。此外，還推出了全新的支援 GPU 的 Amazon EKS 最佳化 AMI。	2018 年 8 月 22 日
全新 Amazon EKS 最佳化 AMI 為 ALAS2-2018-1058 進行修補	Amazon EKS 已更新 Amazon EKS 最佳化 AMI，可解決 ALAS2-2018-1058 所述的 CVE 問題。	2018 年 8 月 14 日
Amazon EKS 最佳化之 AMI 建置指令碼	Amazon EKS 已採用開放原始碼的形式提供建置指令碼，可用於建置 Amazon EKS 最佳化 AMI。這些建置指令碼現可於 GitHub 取得。	2018 年 7 月 10 日
Amazon EKS 初始版本	服務啟動的初始文件	2018 年 6 月 5 日

EKS 使用者指南的貢獻

AWS 已針對 EKS 使用者指南推出改善的貢獻體驗。

您現在可以直接在 GitHub 上編輯 EKS 使用者指南來源。

文件現在使用 AsciiDoc，這是一種類似於 Markdown 的強大撰寫語言。AsciiDoc 結合了簡單的語法與企業文件功能，例如進階格式化、交叉參考和安全控制。

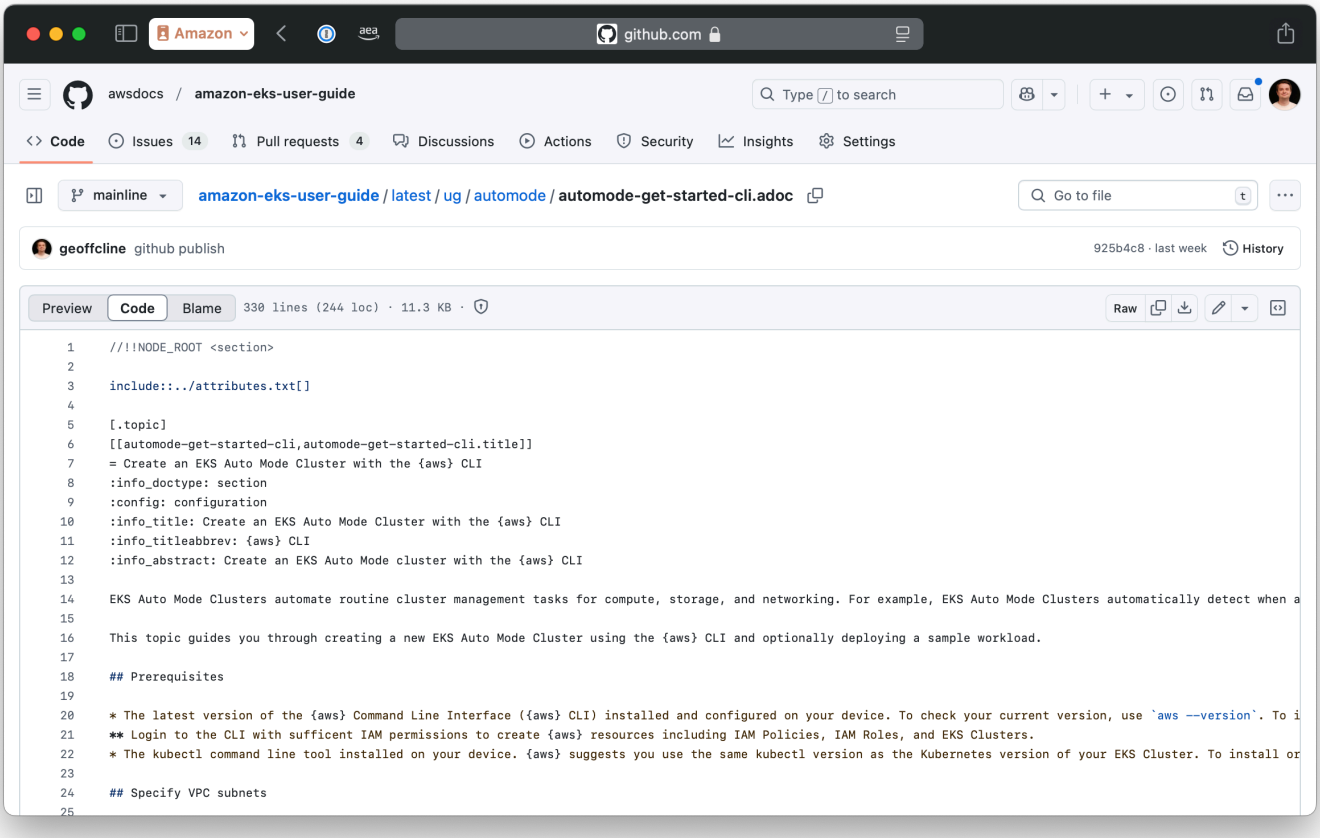
您現在可以直接在 GitHub 上編輯 EKS 文件。我們的簡化程序包括：

- 更快速的提取請求處理
- 減少手動步驟
- 自動化內容品質檢查

我們期待您的貢獻。

從 Web 瀏覽器編輯單一頁面

您可以直接透過 Web 瀏覽器，在 EKS 使用者指南中輕鬆編輯單一頁面。



如果您想要從 Web 瀏覽器編輯多個頁面，請參閱 the section called “使用 GitHub 編輯檔案”。

先決條件

- 要在 Web 瀏覽器中開啟變更的文件頁面。
- 已登入 GitHub。

程序

1. 導覽至《Amazon EKS 使用者指南》中的您要編輯的頁面。
2. 在右窗格中，選擇在 GitHub 上編輯此頁面連結。
3. 在 GitHub 上開啟編輯器，方法為：
   - 按下鍵盤上的 e鍵。
   - 按一下鉛筆圖示，然後從下拉式功能表中選取編輯就地。

- 如果您沒有編輯的選項，則需要登入 GitHub。您的 GitHub 帳戶不需要任何特殊許可即可建議變更。不過，內部 Amazon 貢獻者應該連結其 GitHub 設定檔。

4. 在 GitHub 編輯器中對內容進行必要的變更。
   - 編輯器提供語法反白和預覽功能。
   - 您可以使用 AsciiDoc 標記來格式化您的變更。
   - 您可以使用 `ctrl-f`開啟尋找/取代界面。
5. （選用）預覽您的變更。
   - 使用 `preview`索引標籤，以豐富的格式預覽您的變更。
   - 使用 `show diff`選項反白顯示已變更的區段。移除的區段在左邊界中有一個紅色指標。新區段的左邊界有一個綠色指標。
6. 完成編輯後，按一下編輯器頂端的遞交變更...按鈕
7. 在遞交對話方塊中：
   - 驗證您的電子郵件地址是否正確。
   - 新增簡短但描述性的遞交訊息，說明您的變更。
   - 視需要選擇性地新增較長的描述。
   - 選取 以建立新的分支並提取請求。

您已建立提取請求，包括提議的變更。

提取請求概觀

當您建立 PR 時：

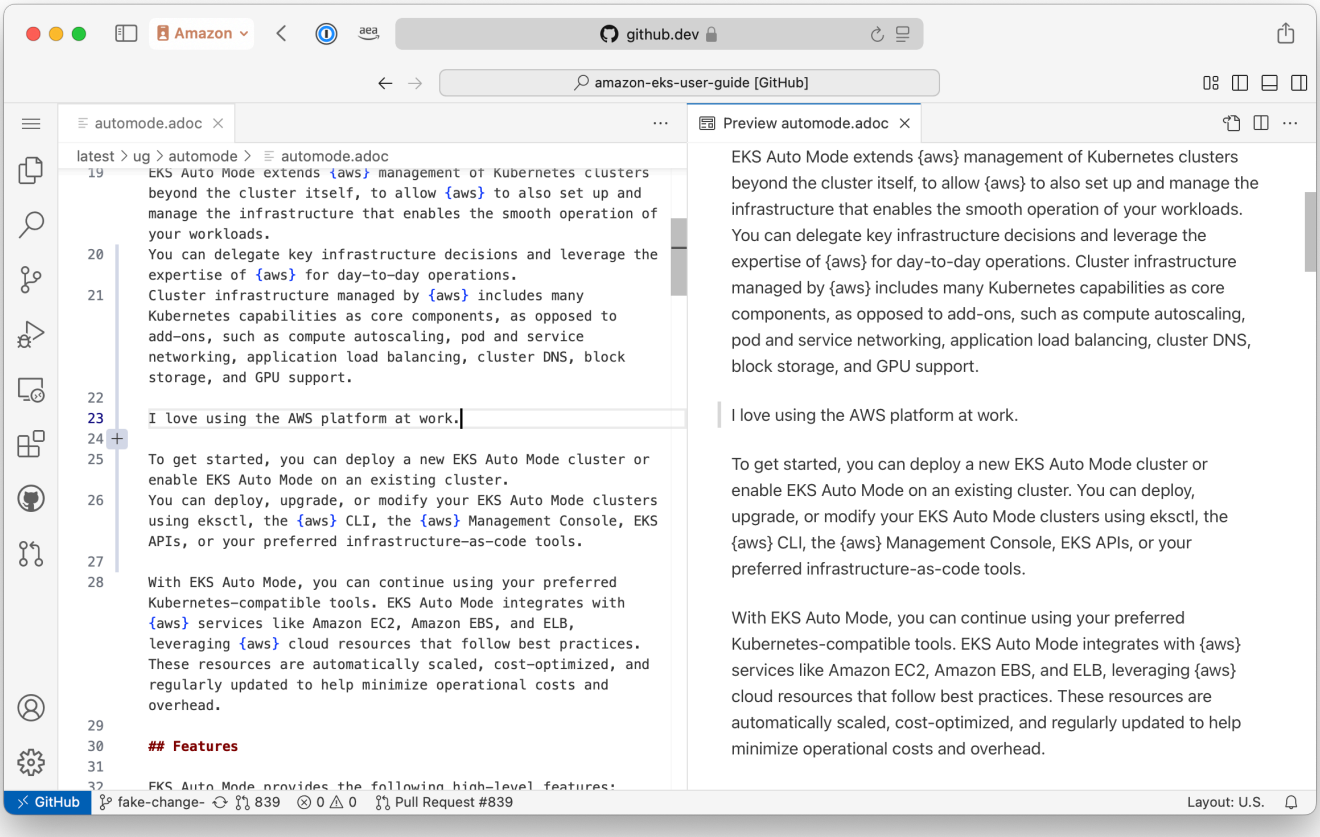
- 儲存庫維護者會提交您的變更以供檢閱。
- 檢閱者可以評論您的變更和請求修改。
- 可能會執行自動化測試來驗證您的變更。
- 核准後，您的變更可以合併到主要儲存庫。

提取請求有助於確保品質，並提供在整合變更之前討論變更的方法。

了解如何在 GitHub 文件中檢閱和核准提取請求。

使用 GitHub Web Editor 從 Web 瀏覽器編輯多個檔案

如果您想要提議變更多個頁面，或建立新的文件頁面，請使用 GitHub.dev Web 編輯器。此 Web 編輯器是以熱門的 Visual Studio Code 文字編輯器為基礎。



先決條件

- 登入 GitHub
- 熟悉 Visual Studio Code 編輯器
- 熟悉 Git

程序

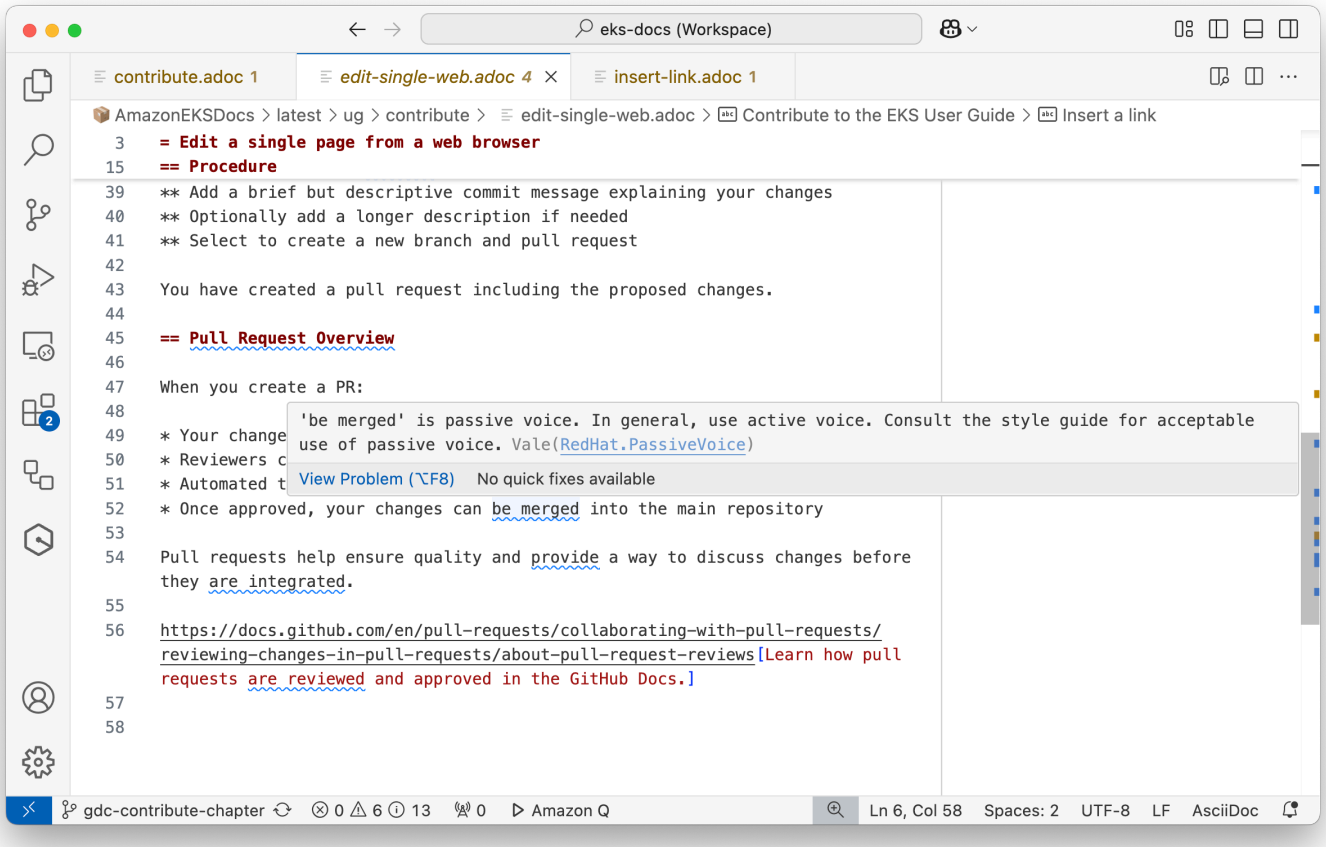
> **Note**
>
> EKS 文件團隊已建立工作區檔案，其中包含編輯器的建議組態，例如文字包裝和 AsciiDoc 語法反白。我們建議您載入此工作區檔案。

1. 在 GitHub.dev 上開啟工作區。 GitHub.dev.
   - 您可以將 URL 加入書籤 `https://github.dev/awsdocs/amazon-eks-user-guide/blob/mainline/eks-docs.code-workspace?workspace=true`
2. （僅限第一次設定）系統可能會提示您在自己的 GitHub 帳戶中建立儲存庫的叉。接受此提示。如需詳細資訊，請參閱 GitHub 文件中的關於叉。
3. （僅限第一次設定）接受右下角的提示來安裝 AsciiDoc 延伸模組。
4. 導覽至 的文件內容`latest/ug`。
   - 文件檔案會依其頂層區段進行組織。例如，「安全性」章節中的頁面在「安全性/」目錄下具有來源檔案。
5. 若要檢視文件頁面的預覽，請使用右上角的開啟預覽到側邊按鈕。圖示包含一個小型放大鏡。
6. 使用左側的來源控制索引標籤遞交變更。如需詳細資訊，請參閱 Visual Studio Code 文件：
   - 遞交變更
   - 建立提取請求

建立提取請求後，文件團隊會檢閱該請求。

在本機安裝 Vale，在輸入時檢視風格意見回饋

您可以在輸入時查看風格意見回饋。這有助於識別尷尬的寫入和錯別字。



概觀：

- Vale CLI 會載入樣式指南，並根據來源檔案執行。
- EKS 文件儲存庫包含一個 vale 組態檔案，可載入樣式指南和本機規則。
- Visual Studio (VS) 的 Vale 擴充功能程式碼會在編輯器內顯示 vale 意見回饋。

安裝 Vale

遵循 Vale CLI 文件中的指示，使用套件管理員安裝 Vale。

安裝 VS Code Vale 延伸模組

1. 開啟 VS 程式碼。
2. 按一下活動列中的延伸項目圖示 （ 或按下 Ctrl+Shift+X)。
3. 搜尋「Vale」。

4. 按一下 Chris Chinchilla 在「Vale VSCode」延伸模組上安裝 。
5. 出現提示時重新載入 VS 程式碼。

同步 Vale

Vale 使用專案根目錄中的`.vale.ini`組態檔案來決定要套用的樣式規則。

1. 開啟 VS 程式碼。
2. 按一下檢視 > 終端機 （或按 Ctrl+`）。
3. 如有需要，導覽至您的專案根目錄。
4. 執行 命令：

```
vale sync
```

5. 等待 Vale 完成下載和同步樣式規則

在 VS 程式碼中檢視樣式意見回饋

1. 在 VS 程式碼中開啟 Markdown 或 AsciiDoc 檔案。
2. Vale 延伸模組會自動根據樣式規則檢查您的文字。
3. 樣式問題會在編輯器中加上底線。
4. 將滑鼠暫留在底線文字上，以查看特定樣式建議。
5. 遵循建議或參閱樣式指南來修正問題。

建立新頁面

了解如何建立新的文件頁面。本主題包含建立初始頁面中繼資料，以及將頁面新增至 目錄的說明。

建立頁面

1. 導覽至章節目錄。例如，如果您想要在「安全性」區段中建立新的頁面，請導覽至 `latest/ug/security`目錄。
2. 判斷頁面 ID。根據慣例，頁面 ID 都是小寫，並以 分隔`-`。此頁面的 ID 為 `create-page`。
3. 使用頁面 ID 和`adoc`副檔名建立新的檔案。例如 `create-page.adoc`。
4. 使用此範本插入頁面中繼資料：

```
include::../attributes.txt[]

[.topic]
[#unique-page-id]
= Page title
:info_titleabbrev: Short title

[abstract]
--
Brief summary of the page's content.
--

Introduction paragraph goes here.
```

將頁面新增至導覽

1. 導覽至父頁面。頂層區段的父頁面是 book.adoc。
2. 在父頁面底部，包含子頁面。

   ```
   include::${filename}[leveloffset=+1]
   ```

   例如：

   ```
   include::create-page.adoc[leveloffset=+1]
   ```

插入連結

AsciiDoc 支援多種類型的連結。使用正確的連結類型非常重要，因此連結可在不同的環境中正常運作。

EKS 使用者指南中的頁面或區段連結

使用交叉參考 (xref) 來連結相同文件網站中的頁面或區段，例如 EKS 使用者指南。如果目標區段移動或重新命名，它們會自動更新。

使用頁面標題做為連結文字

對於本使用者指南中連結至另一個 ID 的大多數情況，請使用下列方法，讓連結文字視需要自動更新至最新的標題。

```
For more information, see <<page-or-section-id>>.
```

定義自訂連結文字

對於您必須有自訂連結文字的情況，請使用下列格式。

```
Here's an example of a <<page-or-section-id,link with custom text>>.
```

連結至 AWS 文件中的其他指南

1. 尋找 AWS 文件頁面的連結。
2. 移除 https://docs.aws.amazon.com/ 字首，只保留路徑。路徑應以字母開頭。
3. 建立連結，如下所示：

```
link:AmazonS3/latest/userguide/create-bucket-overview.html[Create a bucket,
 type="documentation"]
```

外部網頁的連結

此格式會建立非 Amazon 託管頁面的標準連結。例如，將此用於 GitHub 連結。

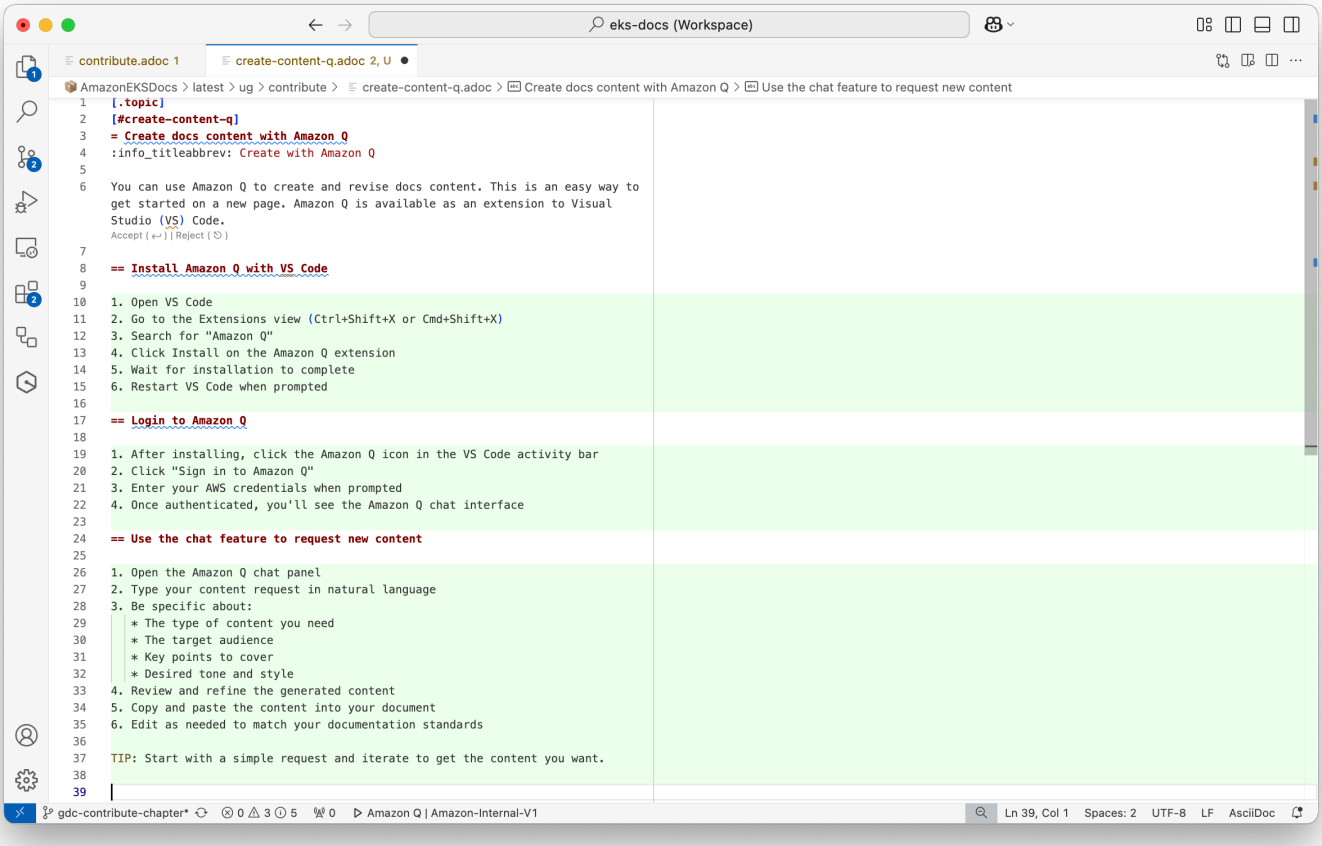
```
https://example.com[Link text]
```

> **Note**
>
> 我們有外部網域的允許清單。允許清單位於 `vale/styles/EksDocs/ExternalDomains.yml`

使用 Amazon Q 建立文件內容

您可以使用 Amazon Q 來建立和修訂文件內容。這是在新頁面上開始使用的簡單方法。Amazon Q 可作為 Visual Studio (VS) 程式碼的延伸。

在下圖中，Amazon Q 產生標示為綠色的行。



使用 VS 程式碼安裝 Amazon Q

1. 開啟 VS 程式碼
2. 前往延伸模組檢視 (Ctrl+Shift+X 或 Cmd+Shift+X)
3. 搜尋「Amazon Q」
4. 按一下在 Amazon Q 延伸模組上安裝
5. 等待安裝完成
6. 出現提示時重新啟動 VS 程式碼

登入 Amazon Q

1. 安裝後，選擇 VS Code 活動列中的 Amazon Q 圖示。
2. 選擇登入 Amazon Q。
3. 出現提示時輸入您的 AWS 登入資料。

4. 驗證後，您會看到 Amazon Q 聊天介面。

使用 Amazon Q 建立內容

1. 在 VS 程式碼中開啟您要編輯的檔案。
2. 選取您要修訂的文字或新內容的位置。
3. 按 Ctrl+I 或 Cmd+I。
4. 在提示中，請具體說明：
   - 您需要的內容類型。
   - 目標對象。
   - 要涵蓋的要點。
   - 所需的色調和風格。
5. 在內嵌預覽中檢閱產生的內容。
6. 使用 Enter 接受變更，或使用 esc 拒絕變更。
7. 視需要進一步編輯。

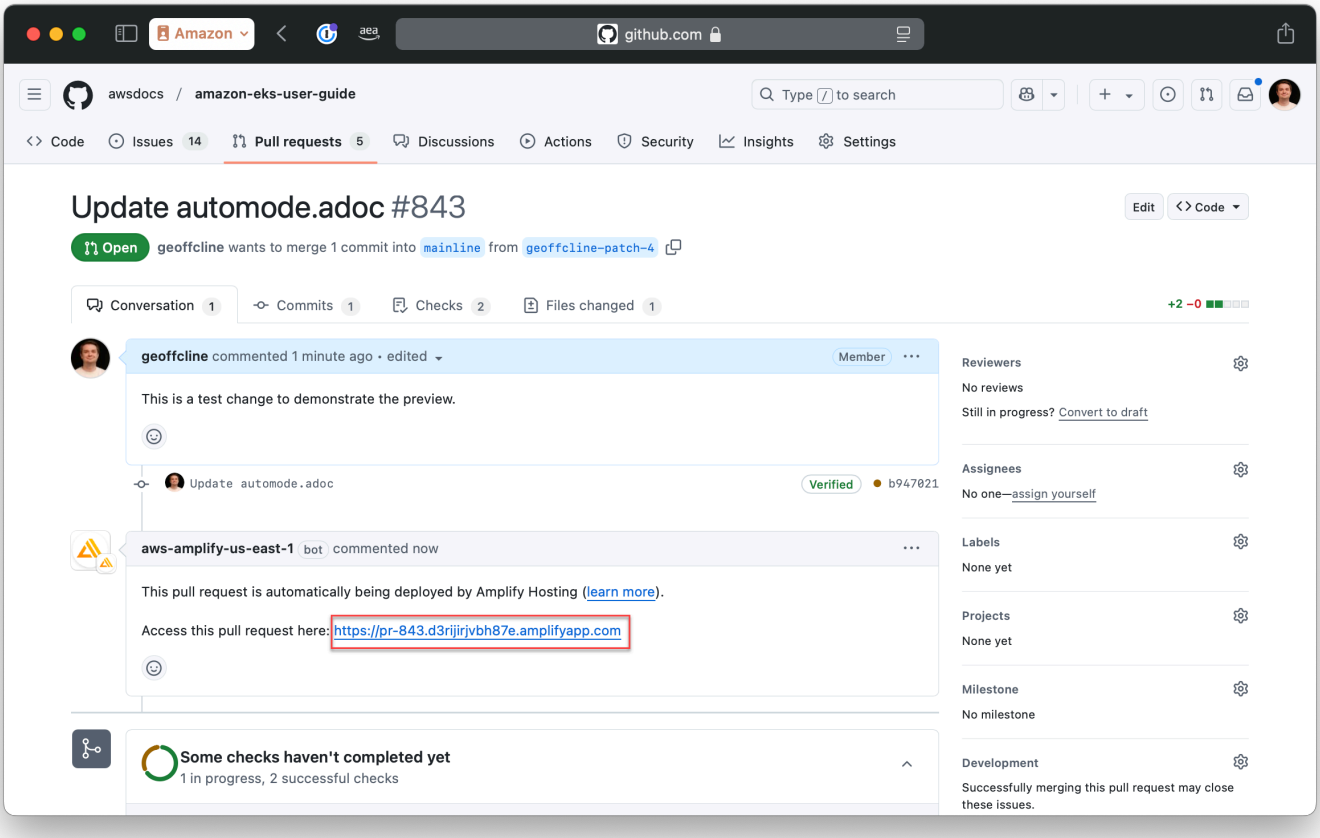
提示

- 從簡單的請求開始，並反覆運算以取得您想要的內容。
- 建立頁面標題的初稿，然後要求 Q 填寫。
- Amazon Q 可能會輸出 Markdown。這沒問題。AsciiDoc 工具可以了解大多數 Markdown 語法。

若要進一步了解 Amazon Q Developer，請參閱 IDE 中的使用 Amazon Q Developer。

檢視提取請求內容的預覽

Amazon EKS 使用者指南 GitHub 設定為建置和產生文件網站的預覽。此預覽沒有完整的 AWS 佈景主題，但確實會檢查內容建置是否正確，連結是否正常運作。



此預覽由 AWS Amplify 託管於暫時 URL。

檢視預覽

當您提交提取請求時， AWS Amplify 會嘗試建置和部署內容預覽。

如果建置成功，aws-amplify-us-east-1 會將註解新增至具有預覽連結的提取請求。選擇「在此處存取此提取請求」右側的連結 （ 如螢幕擷取畫面中以紅色大綱所指出 ）。

如果建置失敗，儲存庫管理員可以查看日誌並提供意見回饋。

> **Note**
>
> 如果您之前沒有做出貢獻，專案維護者可能需要核准執行建置。

預覽限制

預覽會建置為單一大型 HTML 檔案。發佈時，它將顯示為多個頁面。

運作方式：

- 交叉參考 (`xref`)
- 網際網路的連結
- 映像
- 從 託管的內容 `samples/`

什麼沒有作用：

- 使用 連結到其他 AWS 內容`type="documentation"`。這是因為預覽環境中不存在此內容。
- 屬性`{aws}`無法正常顯示。此值會根據環境而變更。

AsciiDoc 語法參考

AsciiDoc 是 AWS 文件的偏好標記語言。雖然工具部分支援 Markdown 語法（例如標題和清單），但我們建議對所有內容使用 AsciiDoc 語法，以確保一致性和適當的轉譯。

本指南涵蓋貢獻 Amazon EKS 文件時所需的常見語法元素。如需更進階的語法和詳細資訊，請參閱 AsciiDoc 文件。

新頁面

每個新的 AsciiDoc 文件都應該以 中定義的結構開頭the section called “建立頁面”。

標題

使用標題以階層方式組織您的內容：

```
= Page/topic title (level 1)
== Section title (level 2)
=== Level 3 heading
==== Level 4 heading
===== Level 5 heading
```

```
====== Level 6 heading
```

注意： AWS 文件標題一律使用句子大小寫。

文字格式

格式化文字以強調重要資訊：

```
Use *bold text* for UI labels.
Use _italic text_ for introducing terms or light emphasis.
Use `monospace text` for code, file names, and commands.
```

清單

未排序清單

為沒有特定序列的項目建立項目符號清單：

```
* First item
* Second item
** Nested item
** Another nested item
* Third item
```

排序清單

為循序步驟或優先項目建立編號清單：

```
. First step
. Second step
.. Substep 1
.. Substep 2
. Third step
```

連結

如需如何正確格式化連結的詳細資訊the section called “插入連結”，請參閱 。不支援 Markdown 樣式連結。

程式碼範例

內嵌程式碼

針對內嵌程式碼使用反引號：

```
Use the `kubectl get pods` command to list all pods.
```

程式碼區塊

建立具有語法反白和屬性支援（類似實體）的程式碼區塊：

```
[source,python,subs="verbatim,attributes"]
----
def hello_world():
    print("Hello, World!")
----
```

映像

使用替代文字插入影像以方便存取：

```
image::images/image-file.png[Alt text description]
```

資料表

建立資料表以組織資訊：

```
[%header,cols="2"]
|===
|Header 1
|Header 2

|Cell 1,1
|Cell 1,2

|Cell 2,1
|Cell 2,2
|===
```

如需更複雜的資料表，請參閱 AsciiDoc 資料表文件。

標註

使用標註反白顯示重要資訊和建議：

```
NOTE: This is a note callout for general information.

TIP: This is a tip callout for helpful advice.

IMPORTANT: This is an important callout for critical information.
```

預覽：

> **Note**
>
> 這是備註標註。

包含其他檔案

包含來自其他檔案的內容：

```
include::filename.adoc[]
```

屬性（類似於實體）

使用預先定義的屬性來維持一致性。特別是，您必須使用 AWS 和 的屬性 `arn:aws:` 。

```
{aws} provides Amazon EKS as a managed Kubernetes service.
```

```
[source,bash,subs="verbatim,attributes"]
----
aws iam attach-role-policy \
    --role-name AmazonEKSAutoClusterRole \
    --policy-arn {arn-aws}iam::aws:policy/AmazonEKSClusterPolicy
----
```

如需屬性清單，請查看 `../attributes.txt` 檔案。

程序

step-by-step程序：

```
To create an Amaozon EKS cluster. do the following steps.

. Sign in to the {aws} Management Console.
. Open the Amazon EKS console.
. Choose *Create cluster*.

...
```