Amazon EMR on EKS 開發指南

# Amazon EMR

Copyright © 2025 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

# Amazon EMR: Amazon EMR on EKS 開發指南

Copyright © 2025 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

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

# Table of Contents

什麼是 Amazon EMR on EKS？ ... 1
適用於 Amazon EMR on EKS 的架構 ... 2
了解 Amazon EMR on EKS 概念和術語 ... 3
Kubernetes 命名空間 ... 3
虛擬叢集 ... 3
作業執行 ... 4
Amazon EMR 容器 ... 4
當您將工作提交至 Amazon EMR on EKS 虛擬叢集時會發生什麼情況 ... 4
Amazon EMR on EKS 入門 ... 6
執行 Spark 應用程式 ... 7
最佳實務 ... 12
安全 ... 12
Pyspark 作業提交 ... 12
儲存 ... 12
中繼存放區整合 ... 12
除錯 ... 13
Amazon EMR on EKS 問題疑難排解 ... 13
節點放置 ... 13
效能 ... 13
成本最佳化 ... 13
使用 AWS Outposts ... 13
自訂 Docker 映像檔 ... 14
如何自訂 Docker 映像檔 ... 14
先決條件 ... 15
步驟 1：從 Amazon Elastic Container Registry (Amazon ECR) 中擷取基礎映像 ... 15
步驟 2：自訂基礎映像 ... 16
步驟 3：(選用但不推薦) 驗證自訂映像 ... 17
步驟 4：發布自訂映像 ... 18
步驟 5：使用自訂映像在 Amazon EMR 中提交 Spark 工作負載 ... 19
為互動端點自訂 Docker 映像檔 ... 21
使用多架構映像 ... 23
選取基礎映像 URI 的詳細資訊 ... 24
Amazon ECR 登錄檔帳戶 ... 25
自訂映像的考量事項 ... 27

執行 Flink 作業 ..... 29
Flink Kubernetes Operator ..... 29
設定 ..... 30
安裝部署 Flink Kubernetes Operator ..... 31
執行 Flink 應用程式 ..... 32
執行 Flink 應用程式的安全角色許可 ..... 36
解除安裝 Operator ..... 38
Flink 原生 Kubernetes ..... 39
設定 ..... 39
開始使用 ..... 40
安全要求 ..... 42
自訂 Flink 和 FluentD 的 Docker 映像 ..... 43
先決條件 ..... 43
從 Amazon Elastic Container Registry 擷取基礎映像 ..... 44
自訂基礎映像 ..... 44
發佈您的自訂映像 ..... 45
提交 Flink 工作負載 ..... 45
監控 ..... 46
使用 Amazon Managed Service for Prometheus ..... 46
使用 Flink UI ..... 48
使用監控組態 ..... 49
Flink 如何支援高可用性和任務彈性 ..... 54
使用高可用性 ..... 54
優化重新啟動時間 ..... 59
正常解除委任 ..... 65
使用 Autoscaler ..... 68
Autoscaler 參數自動調校 ..... 69
Amazon EMR on EKS 上 Flink 任務的維護和故障診斷 ..... 78
維護 Flink 應用程式 ..... 78
疑難排解 ..... 79
支援的版本 ..... 82
執行 Spark 作業 ..... 84
StartJobRun ..... 84
設定 ..... 85
使用 `StartJobRun` 提交作業執行 ..... 109
使用作業提交器分類 ..... 111

使用 Amazon EMR 容器預設分類 ..... 116
Spark Operator ..... 118
設定 ..... 119
開始使用 ..... 119
垂直自動擴展 ..... 123
解除安裝 ..... 128
使用監控組態來監控 Spark ..... 128
安全 ..... 135
spark-submit ..... 144
設定 ..... 144
開始使用 ..... 145
安全 ..... 146
Apache Livy ..... 151
設定 ..... 152
開始使用 ..... 152
執行 Spark 應用程式 ..... 157
解除安裝 ..... 159
安全 ..... 159
安裝屬性 ..... 169
疑難排解常見的環境變數格式錯誤 ..... 173
管理作業執行 ..... 174
使用 CLI 進行管理 ..... 174
執行 Spark SQL 指令碼 ..... 179
作業執行狀態 ..... 182
在主控台中檢視作業 ..... 182
常見作業執行錯誤 ..... 183
使用作業範本 ..... 188
建立並使用任務範本來啟動任務執行 ..... 189
定義作業範本參數 ..... 190
控制對作業範本的存取 ..... 192
使用 Pod 範本 ..... 194
常用案例 ..... 194
使用 Amazon EMR on EKS 啟用 Pod 範本 ..... 196
Pod 範本欄位 ..... 198
附屬容器的考量 ..... 201
使用重試政策 ..... 202

設定重試政策 ..... 202
擷取政策狀態 ..... 204
監控作業 ..... 205
尋找驅動程式日誌 ..... 205
使用 Spark 事件日誌輪換 ..... 206
使用 Spark 容器日誌輪換 ..... 207
使用垂直自動擴展 ..... 209
設定 ..... 209
開始使用 ..... 212
組態 ..... 213
監控建議 ..... 218
解除安裝 ..... 219
執行互動式工作負載 ..... 220
互動端點概觀 ..... 220
互動端點先決條件 ..... 222
AWS CLI ..... 222
eksctl ..... 222
Amazon EKS 叢集 ..... 222
授予叢集存取權 ..... 223
啟用服務帳戶的 IAM 角色 ..... 223
建立 IAM 作業執行角色 ..... 223
授予使用者存取權 ..... 223
向 Amazon EMR 註冊 Amazon EKS 叢集 ..... 224
Load Balancer 控制器 ..... 224
建立互動端點 ..... 224
建立互動端點 ..... 224
指定自訂參數 ..... 225
..... 226
互動端點參數 ..... 227
配置互動端點的設定 ..... 228
監控 Spark 任務 ..... 228
自訂 Pod 範本 ..... 230
將 JEG Pod 部署到節點群組 ..... 230
JEG 組態選項 ..... 234
修改 PySpark 參數 ..... 234
自訂核心映像 ..... 235

監控互動端點 ..... 236
範例 ..... 238
使用自助託管的 Jupyter 筆記本 ..... 239
建立安全群組 ..... 239
建立互動端點 ..... 239
取得閘道伺服器 URL ..... 240
取得驗證字符 ..... 240
部署筆記本 ..... 241
清除 ..... 246
使用 CLI 命令取得互動式端點的相關資訊 ..... 247
..... 247
列出互動端點 ..... 248
刪除互動端點 ..... 250
上傳資料 ..... 251
先決條件 ..... 251
開始使用 ..... 251
監控任務 ..... 253
使用 Amazon CloudWatch Events 監控作業 ..... 253
使用 CloudWatch Events 來自動化 Amazon EMR on EKS ..... 254
範例：設定調用 Lambda 的規則 ..... 255
使用 Amazon CloudWatch Events 透過重試政策來監控作業的驅動程式 Pod ..... 255
管理虛擬叢集 ..... 256
建立虛擬叢集 ..... 256
列出虛擬叢集 ..... 257
描述虛擬叢集 ..... 258
刪除虛擬叢集 ..... 258
虛擬叢集狀態 ..... 258
教學課程 ..... 259
使用 Delta Lake ..... 259
使用 Iceberg ..... 260
目錄整合的 Spark 工作階段組態 ..... 261
使用 PyFlink ..... 262
搭配 Flink 使用 AWS Glue ..... 263
使用 Apache Hudi ..... 266
提交 Apache Hudi 任務 ..... 266
使用 Spark RAPIDS ..... 269

使用 Spark on Redshift ..... 274
啟動 Spark 應用程式 ..... 274
向 Amazon Redshift 進行身分驗證 ..... 275
讀取和寫入 Amazon Redshift ..... 277
考量事項 ..... 279
使用 Volcano ..... 280
概觀 ..... 280
安裝 ..... 280
提交：Spark Operator ..... 281
提交：`spark-submit` ..... 283
使用 YuniKorn ..... 284
概觀 ..... 284
建立 叢集 ..... 284
安裝 YuniKorn ..... 286
提交：Spark Operator ..... 287
提交：`spark-submit` ..... 290
安全 ..... 12
最佳實務 ..... 293
套用最低權限準則 ..... 293
端點的存取控制清單 ..... 293
取得自訂映像的最新安全更新 ..... 293
限制 Pod 憑證存取 ..... 293
隔離不受信任的應用程式碼 ..... 294
角色型存取控制 (RBAC) 許可 ..... 294
限制存取節點群組 IAM 角色或執行個體設定檔憑證 ..... 294
資料保護 ..... 295
靜態加密 ..... 295
傳輸中加密 ..... 297
身分和存取權管理 ..... 298
目標對象 ..... 298
使用身分驗證 ..... 299
使用政策管理存取權 ..... 301
Amazon EMR on EKS 如何搭配 IAM 運作 ..... 303
使用服務連結角色 ..... 309
Amazon EMR on EKS 的受管政策 ..... 312
搭配使用作業執行角色與 Amazon EMR on EKS ..... 313

身分型政策範例 ..... 315
標籤型存取控制政策 ..... 318
故障診斷 ..... 321
搭配 AWS Lake Formation 使用 Amazon EMR on EKS ..... 322
Amazon EMR on EKS 如何與 AWS Lake Formation 搭配使用 ..... 323
使用 Amazon EMR on EKS 啟用 Lake Formation ..... 324
考量與限制 ..... 332
疑難排解 ..... 334
日誌記錄和監控 ..... 335
加密日誌 ..... 336
CloudTrail 日誌 ..... 339
S3 Access Grants ..... 341
概觀 ..... 341
啟動叢集 ..... 342
考量事項 ..... 343
合規驗證 ..... 343
恢復能力 ..... 343
基礎設施安全性 ..... 344
組態與漏洞分析 ..... 344
介面 VPC 端點 ..... 344
為 Amazon EMR on EKS 建立 VPC 端點政策 ..... 345
跨帳戶存取權 ..... 348
先決條件 ..... 348
如何存取跨帳戶 Amazon S3 儲存貯體或 DynamoDB 資料表 ..... 348
標記 資源 ..... 353
標籤基本概念 ..... 353
標記您的 資源 ..... 353
標籤限制 ..... 354
使用 AWS CLI 和 Amazon EMR on EKS API 處理標籤 ..... 355
故障診斷 ..... 13
PVC 作業失敗 ..... 356
驗證 ..... 356
修補程式 ..... 356
手動修補 ..... 360
垂直自動擴展失敗 ..... 362
「403 禁止」錯誤 ..... 363

找不到命名空間 ... 363
Docker 憑證錯誤 ... 363
Spark Operator 失敗 ... 364
Helm Chart 安裝失敗 ... 364
Unsupported filesystem exception ... 364
服務端點和配額 ... 366
服務端點 ... 366
Service Quotas ... 368
檢視配額和請求增加配額 ... 370
發行版本 ... 371
7.9.0 版 ... 372
推出 ... 372
版本備註 ... 374
變更 ... 375
emr-7.9.0-latest ... 375
emr-7.9.0-20250425 ... 376
emr-7.9.0-flink-latest ... 376
emr-7.9.0-flink-20250425 ... 376
7.8.0 版 ... 376
推出 ... 377
版本備註 ... 378
變更 ... 380
emr-7.8.0-latest ... 380
emr-7.8.0-20250228 ... 380
emr-7.8.0-flink-latest ... 380
emr-7.8.0-flink-20250228 ... 381
7.7.0 版 ... 381
推出 ... 381
版本備註 ... 383
變更 ... 384
emr-7.7.0-latest ... 384
emr-7.7.0-20250131 ... 384
emr-7.7.0-flink-latest ... 385
emr-7.7.0-flink-20250131 ... 385
7.6.0 版 ... 385
推出 ... 385

版本備註 ..... 387
功能 ..... 388
變更 ..... 389
emr-7.6.0-latest ..... 389
emr-7.6.0-20241213 ..... 389
emr-7.6.0-flink-latest ..... 389
emr-7.6.0-flink-20241213 ..... 389
7.5.0 版 ..... 390
推出 ..... 390
版本備註 ..... 390
7.4.0 版 ..... 390
推出 ..... 390
版本備註 ..... 390
7.3.0 版 ..... 391
推出 ..... 391
版本備註 ..... 393
功能 ..... 394
變更 ..... 394
emr-7.3.0-latest ..... 394
emr-7.3.0-29240920 ..... 395
emr-7.3.0-flink-latest ..... 395
emr-7.3.0-flink-29240920 ..... 395
7.2.0 版 ..... 395
推出 ..... 396
版本備註 ..... 397
功能 ..... 399
emr-7.2.0-latest ..... 399
emr-7.2.0-20240610 ..... 399
emr-7.2.0-flink-latest ..... 400
emr-7.2.0-flink-20240610 ..... 400
7.1.0 版 ..... 400
推出 ..... 400
版本備註 ..... 402
功能 ..... 403
emr-7.1.0-latest ..... 404
emr-7.1.0-20240321 ..... 404

emr-7.1.0-flink-latest ..... 404
emr-7.1.0-flink-20240321 ..... 404
7.0.0 版 ..... 405
推出 ..... 405
版本備註 ..... 406
功能 ..... 408
變更 ..... 408
emr-7.0.0-latest ..... 408
emr-7.0.0-2024321 ..... 408
emr-7.0.0-20231211 ..... 409
emr-7.0.0-flink-latest ..... 409
emr-7.0.0-flink-2024321 ..... 409
emr-7.0.0-flink-20231211 ..... 409
6.15.0 版 ..... 410
推出 ..... 410
版本備註 ..... 411
功能 ..... 413
emr-6.15.0-latest ..... 413
emr-6.15.0-20240105 ..... 413
emr-6.15.0-20231109 ..... 413
emr-6.15.0-flink-latest ..... 414
emr-6.15.0-flink-20240105 ..... 414
emr-6.15.0-flink-20231109 ..... 414
6.14.0 版 ..... 414
推出 ..... 414
版本備註 ..... 416
功能 ..... 417
emr-6.14.0-latest ..... 417
emr-6.14.0-20231005 ..... 417
6.13.0 版 ..... 418
推出 ..... 418
版本備註 ..... 419
功能 ..... 420
emr-6.13.0-latest ..... 421
emr-6.13.0-20230814 ..... 421
6.12.0 版 ..... 421

推出 ..... 421
版本備註 ..... 422
功能 ..... 424
emr-6.12.0-latest ..... 424
emr-6.12.0-20240321 ..... 424
emr-6.12.0-20230701 ..... 424
6.11.0 版 ..... 425
推出 ..... 425
版本備註 ..... 425
功能 ..... 427
emr-6.11.0-latest ..... 427
emr-6.11.0-20230905 ..... 427
emr-6.11.0-20230509 ..... 428
6.10.0 版 ..... 428
emr-6.10.0-latest ..... 430
emr-6.10.0-20230905 ..... 431
emr-6.10.0-20230624 ..... 431
emr-6.10.0-20230421 ..... 431
emr-6.10.0-20230403 ..... 431
emr-6.10.0-20230220 ..... 432
6.9.0 版 ..... 432
emr-6.9.0-latest ..... 434
emr-6.9.0-20230905 ..... 435
emr-6.9.0-20230624 ..... 435
emr-6.9.0-20221108 ..... 435
6.8.0 版 ..... 435
emr-6.8.0-latest ..... 438
emr-6.8.0-20230905 ..... 439
emr-6.8.0-20230624 ..... 439
emr-6.8.0-20221219 ..... 439
emr-6.8.0-20220802 ..... 439
6.7.0 版 ..... 440
emr-6.7.0-latest ..... 441
emr-6.7.0-20240321 ..... 442
emr-6.7.0-20230624 ..... 442
emr-6.7.0-20221219 ..... 442

emr-6.7.0-20220630 ..... 442
6.6.0 版 ..... 443
emr-6.6.0-latest ..... 444
emr-6.6.0-20240321 ..... 444
emr-6.6.0-20230624 ..... 444
emr-6.6.0-20221219 ..... 445
emr-6.6.0-20220411 ..... 445
6.5.0 版 ..... 445
emr-6.5.0-latest ..... 446
emr-6.5.0-20240321 ..... 446
emr-6.5.0-20221219 ..... 447
emr-6.5.0-20220802 ..... 447
emr-6.5.0-20211119 ..... 447
6.4.0 版 ..... 447
emr-6.4.0-latest ..... 448
emr-6.4.0-20240321 ..... 449
emr-6.4.0-20221219 ..... 449
emr-6.4.0-20210830 ..... 449
6.3.0 版 ..... 449
emr-6.3.0-latest ..... 451
emr-6.3.0-20240321 ..... 451
emr-6.3.0-20220802 ..... 451
emr-6.3.0-20211008 ..... 451
emr-6.3.0-20210802 ..... 451
emr-6.3.0-20210429 ..... 452
6.2.0 版 ..... 452
emr-6.2.0-latest ..... 453
emr-6.2.0-20240321 ..... 453
emr-6.2.0-20220802 ..... 454
emr-6.2.0-20211008 ..... 454
emr-6.2.0-20210802 ..... 454
emr-6.2.0-20210615 ..... 454
emr-6.2.0-20210129 ..... 455
emr-6.2.0-20201218 ..... 455
emr-6.2.0-20201201 ..... 455
5.36.0 版 ..... 455

emr-5.36.0-latest ... 456
emr-5.36.0-20240321 ... 457
emr-5.36.0-20221219 ... 457
emr-5.36.0-20220620 ... 457
emr-5.36.0-20220525 ... 457
5.35.0 版 ... 458
emr-5.35.0-latest ... 459
emr-5.35.0-20240321 ... 459
emr-5.35.0-20221219 ... 459
emr-5.35.0-20220802 ... 459
emr-5.35.0-20220307 ... 460
5.34 版 ... 460
emr-5.34.0-latest ... 461
emr-5.34.0-20240321 ... 461
emr-5.34.0-20220802 ... 461
emr-5.34.0-20211208 ... 462
5.33.0 版 ... 462
emr-5.33.0-latest ... 463
emr-5.33.0-20240321 ... 463
emr-5.33.0-20221219 ... 464
emr-5.33.0-20220802 ... 464
emr-5.33.0-20211008 ... 464
emr-5.33.0-20210802 ... 464
emr-5.33.0-20210615 ... 465
emr-5.33.0-20210323 ... 465
5.32.0 版 ... 465
emr-5.32.0-latest ... 466
emr-5.32.0-20240321 ... 467
emr-5.32.0-20220802 ... 467
emr-5.32.0-20211008 ... 467
emr-5.32.0-20210802 ... 467
emr-5.32.0-20210615 ... 468
emr-5.32.0-20210129 ... 468
emr-5.32.0-20201218 ... 468
emr-5.32.0-20201201 ... 468
文件歷史紀錄 ... 469

..... cdlxxi

# 什麼是 Amazon EMR on EKS？

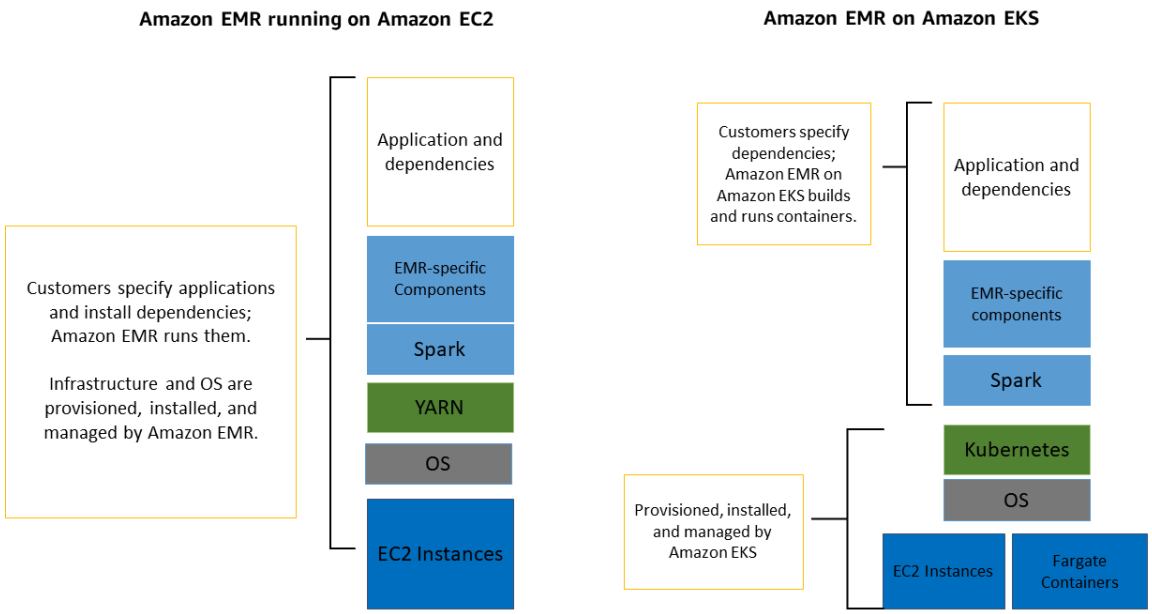
Amazon EMR on EKS 為 Amazon EMR 提供了一個部署選項，可讓您在 Amazon Elastic Kubernetes Service (Amazon EKS) 上執行開放原始碼大數據框架。使用此部署選項，您可以專注於執行分析工作負載，同時 Amazon EMR on EKS 可為開放原始碼應用程式建置、設定和管理容器。

如果您已經使用 Amazon EMR，現在可以在相同的 Amazon EKS 叢集上執行 Amazon EMR 型應用程式和其他類型的應用程式。此部署選項還可改善資源使用率，並簡化多個可用區域的基礎設施管理。如果已經在 Amazon EKS 上執行大數據框架，現在就可以使用 Amazon EMR 來自動化佈建和管理，並更快速地執行 Apache Spark。

Amazon EMR on EKS 可讓您的團隊更有效地協作，以更輕鬆且符合成本效益的方式來處理相當大量的資料：

- 可以在通用資源集區上執行應用程式，而不必佈建基礎設施。您可以使用 Amazon EMR Studio 和 AWS SDK 或 AWS CLI 來開發、提交和診斷在 EKS 叢集上執行的分析應用程式。可以使用自我管理的 Apache Airflow 或 Amazon Managed Workflows for Apache Airflow (MWAA)，在 Amazon EMR 上執行排程作業。
- 基礎設施團隊可以集中管理通用運算平台，將 Amazon EMR 工作負載與其他容器型應用程式合併。可以使用常用的 Amazon EKS 工具簡化基礎設施管理，並利用共用叢集來處理需要不同版本開放原始碼框架的工作負載。也可以透過自動化 Kubernetes 叢集管理和作業系統修補來減少營運成本。透過 Amazon EC2 和 AWS Fargate，您可以啟用多個運算資源，以滿足效能、操作或財務需求。

下圖表示 Amazon EMR 的兩種不同部署模型。




主題

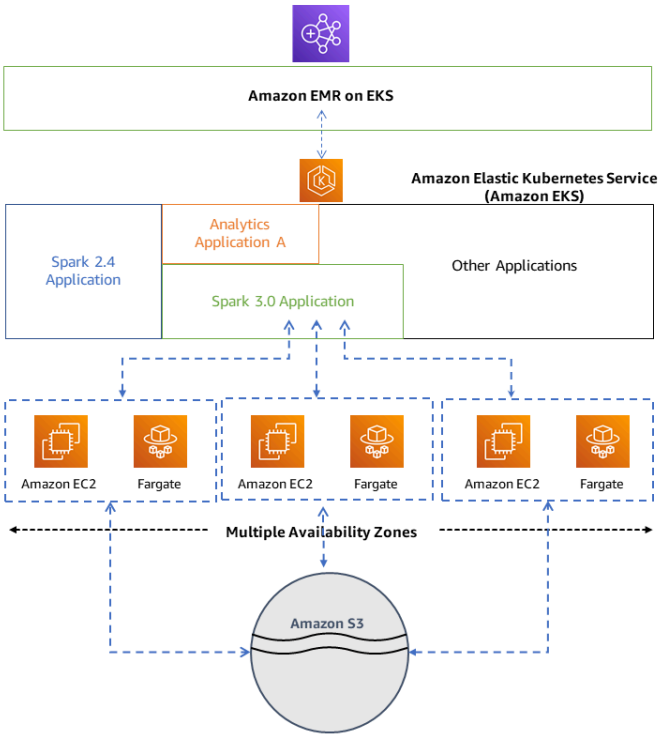
- 適用於 Amazon EMR on EKS 的架構
- 了解 Amazon EMR on EKS 概念和術語
- 當您將工作提交至 Amazon EMR on EKS 虛擬叢集時會發生什麼情況

# 適用於 Amazon EMR on EKS 的架構

Amazon EMR on EKS 會將應用程式鬆散地耦合到它們執行所在的基礎設施。每個基礎設施層級都為後續層級提供協同運作。當您向 Amazon EMR 提交作業時，您的作業定義會包含其所有應用程式特定參數。Amazon EMR 使用這些參數來指示 Amazon EKS 要部署哪些 Pod 和容器。然後，Amazon EKS 將來自 Amazon EC2 的運算資源帶入線上，並且 AWS Fargate 需要此資源才能執行任務。

透過這種鬆散的服務耦合，可以同時執行多個安全隔離的作業。也可以使用不同的運算後端對相同作業進行基準測試，或將作業分散到多個可用區域以提高可用性。

下圖說明 Amazon EMR on EKS 如何與其他 AWS 服務搭配使用。




# 了解 Amazon EMR on EKS 概念和術語

Amazon EMR on EKS 為 Amazon EMR 提供了一個部署選項，可讓您在 Amazon Elastic Kubernetes Service (Amazon EKS) 上執行開放原始碼大數據框架。本主題為您提供一些常見術語的背景，包括命名空間、虛擬叢集和任務執行，這些都是您提交進行處理的工作單位。

## Kubernetes 命名空間

Amazon EKS 使用 Kubernetes 命名空間，在多個使用者和應用程式之間劃分叢集資源。這些命名空間是多租用戶環境的基礎。Kubernetes 命名空間可以具有 Amazon EC2 或 AWS Fargate 作為運算提供者。這種靈活性為您提供了不同的效能和成本選項，以便您的作業繼續執行。

## 虛擬叢集

虛擬叢集是 Amazon EMR 註冊的 Kubernetes 命名空間。Amazon EMR 使用虛擬叢集來執行作業和託管端點。相同實體叢集可支援多個虛擬叢集。不過，每個虛擬叢集都會映射 EKS 叢集上的一個命名空間。虛擬叢集不會建立任何增加帳單或需要在服務之外進行生命週期管理的作用中資源。

## 作業執行

作業執行是您提交至 Amazon EMR on EKS 的作業單位，例如 Spark jar、PySpark 指令碼或 SparkSQL 查詢。一個作業可以有多個作業執行。當您提交作業執行時，會包含下列資訊：

- 應在其中執行作業的虛擬叢集。
- 用於識別作業的作業名稱。
- 執行角色 - 限定範圍的 IAM 角色，它可執行作業並允許您指定作業可存取的資源。
- Amazon EMR 版本標籤，它指定要使用的開放原始碼應用程式的版本。
- 提交作業時要使用的成品，例如 spark-submit 參數。

根據預設，日誌會上傳至 Spark 歷史記錄伺服器，並可從 AWS Management Console中存取。也可以將事件日誌、執行日誌和指標推送到 Amazon S3 和 Amazon CloudWatch。

## Amazon EMR 容器

Amazon EMR 容器是 Amazon EMR on EKS 的 API 名稱。`emr-containers` 字首可用於下列情況：

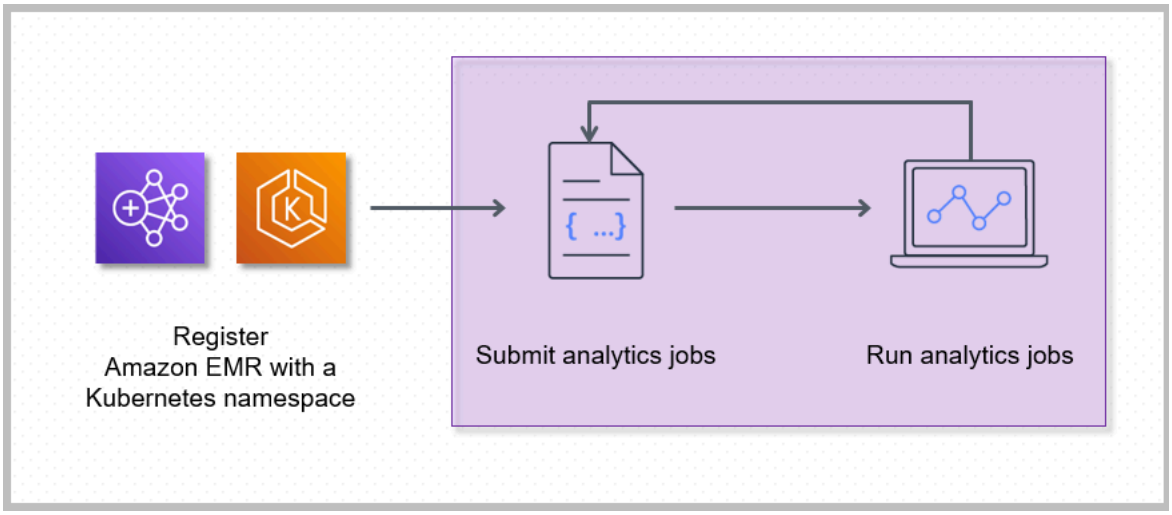
- 它是針對 Amazon EMR on EKS 的 CLI 命令中的字首。例如：`aws emr-containers start-job-run`。
- 它是針對 Amazon EMR on EKS 的 IAM 政策操作之前的字首。例如：`"Action": [ "emr-containers:StartJobRun"]`。如需詳細資訊，請參閱 Amazon EMR on EKS 的政策動作。
- 它是 Amazon EMR on EKS 服務端點中使用的字首。例如：`emr-containers.us-east-1.amazonaws.com`。如需詳細資訊，請參閱 Amazon EMR on EKS 服務端點。

# 當您將工作提交至 Amazon EMR on EKS 虛擬叢集時會發生什麼情況

在 Amazon EKS 叢集上使用 Kubernetes 命名空間註冊 Amazon EMR 可建立虛擬叢集。然後，Amazon EMR 可以在該命名空間上執行分析工作負載。當您使用 Amazon EMR on EKS 將 Spark 作業提交至虛擬叢集時，Amazon EMR on EKS 會請求 Amazon EKS 上的 Kubernetes 排程器來排程 Pod。

以下步驟和圖表說明了 Amazon EMR on EKS 工作流程：

- 使用現有的 Amazon EKS 叢集，或使用 eksctl 命令列公用程式或 Amazon EKS 主控台來建立叢集。
- 透過使用 EKS 叢集上的命名空間註冊 Amazon EMR 來建立虛擬叢集。
- 使用 AWS CLI 或 SDK 將您的任務提交至虛擬叢集。




對於您執行的每個作業，Amazon EMR on EKS 都會建立一個包含 Amazon Linux 2 基礎映像、Apache Spark 和相關相依性的容器。每個作業都會在可下載容器並開始執行該容器的 Pod 中執行。Pod 會在作業終止後終止。如果容器的映像先前已部署至節點，則會使用快取映像並略過下載。附屬容器 (例如日誌或指標轉寄站) 可部署至 Pod。作業終止後，您仍然可以使用 Amazon EMR 主控台中的 Spark 應用程式 UI 對其進行偵錯。

# Amazon EMR on EKS 入門

本主題透過在虛擬叢集上部署 Spark 應用程式，協助您開始使用 Amazon EMR on EKS。它包含設定正確許可和啟動任務的步驟。開始之前，請確定您已完成設定 Amazon EMR on EKS 所述的步驟。這可協助您在建立虛擬叢集之前取得 AWS CLI 等工具。如需可協助您開始的其他範本，請參閱 GitHub 上的 EMR 容器最佳實務指南。

您需要設定步驟中的下列資訊：

- 已向 Amazon EMR 註冊的 Amazon EKS 叢集和 Kubernetes 命名空間的虛擬叢集 ID

> **Important**
>
> 建立 EKS 叢集時，確保使用 m5.xlarge 作為執行個體類型，或使用具有更高 CPU 和記憶體的任何其他執行個體類型。與 m5.xlarge 相比，使用具有較低 CPU 或記憶體的執行個體類型可能會因為叢集中的資源不足而導致作業失敗。

- 用於作業執行的 IAM 角色名稱
- Amazon EMR 版本的發行標籤 (例如，`emr-6.4.0-latest`)
- 用於記錄和監控的目的地目標：
  - Amazon CloudWatch 日誌群組名稱和日誌串流字首
  - 用於儲存事件和容器日誌的 Amazon S3 位置

> **Important**
>
> Amazon EMR on EKS 作業使用 Amazon CloudWatch 和 Amazon S3 作為監控和記錄的目的地目標。透過檢視傳送至這些目的地的作業日誌，可監控作業進度並對故障進行疑難排解。若要啟用日誌，與作業執行的 IAM 角色相關聯的 IAM 政策必須具有存取目標資源所需的許可。如果 IAM 政策沒有所需許可，則在執行此範例作業之前，必須遵循 更新作業執行角色的信任政策、設定作業執行以使用 Amazon S3 日誌，以及設定作業執行以使用 CloudWatch 日誌中列出的步驟。

# 執行 Spark 應用程式

採取以下步驟，在 Amazon EMR on EKS 上執行簡單的 Spark 應用程式。Spark Python 應用程式的應用程式 `entryPoint` 檔案位於 `s3://REGION.elasticmapreduce/emr-containers/samples/wordcount/scripts/wordcount.py`。*REGION* 是 Amazon EMR on EKS 虛擬叢集所在的區域，例如 *us-east-1*。

1. 如下列政策陳述式所示，使用所需許可更新作業執行角色的 IAM 政策。

   JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ReadFromLoggingAndInputScriptBuckets",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:ListBucket"
            ],
            "Resource": [
                "arn:aws:s3:::*.elasticmapreduce",
                "arn:aws:s3:::*.elasticmapreduce/*",
                "arn:aws:s3:::amzn-s3-demo-destination-bucket",
                "arn:aws:s3:::amzn-s3-demo-destination-bucket/*",
                "arn:aws:s3:::amzn-s3-demo-logging-bucket",
                "arn:aws:s3:::amzn-s3-demo-logging-bucket/*"
            ]
        },
        {
            "Sid": "WriteToLoggingAndOutputDataBuckets",
            "Effect": "Allow",
            "Action": [
                "s3:PutObject",
                "s3:DeleteObject"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-destination-bucket/*",
                "arn:aws:s3:::amzn-s3-demo-logging-bucket/*"
            ]
        },
```

```
        {
            "Sid": "DescribeAndCreateCloudwatchLogStream",
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogStream",
                "logs:DescribeLogGroups",
                "logs:DescribeLogStreams"
            ],
            "Resource": [
                "arn:aws:logs:*:*:*"
            ]
        },
        {
            "Sid": "WriteToCloudwatchLogs",
            "Effect": "Allow",
            "Action": [
                "logs:PutLogEvents"
            ],
            "Resource": [
                "arn:aws:logs:*:*:log-group:my_log_group_name:log-
stream:my_log_stream_prefix/*"
            ]
        }
    ]
}
```

- 本政策中的第一個聲明 `ReadFromLoggingAndInputScriptBuckets` 授予 `ListBucket` 和 `GetObjects` 對以下 Amazon S3 儲存貯體的存取權：
  - *`REGION`*`.elasticmapreduce`-應用程式 `entryPoint` 檔案所在的儲存貯體。
  - *`amzn-s3-demo-destination-bucket`* - 您為輸出資料定義的儲存貯體。
  - *`amzn-s3-demo-logging-bucket`* - 您為記錄資料定義的儲存貯體。
- 此政策中的第二個聲明 `WriteToLoggingAndOutputDataBuckets` 會分別授予將資料寫入到輸出和日誌儲存貯體的作業許可。
- 第三個聲明 `DescribeAndCreateCloudwatchLogStream` 為作業授予描述和建立 Amazon CloudWatch Logs 的許可。
- 第四個聲明 `WriteToCloudwatchLogs` 授予的許可能夠將日誌寫入名為 *`my_log_stream_prefix`* 的日誌串流下的名為 *`my_log_group_name`* 的 Amazon CloudWatch 日誌群組。

2. 要執行 Spark Python 應用程式，請使用以下命令。將所有可取代的*####*值取代為適當的值。*REGION* 是 Amazon EMR on EKS 虛擬叢集所在的區域，例如 *us-east-1*。

```
aws emr-containers start-job-run \
--virtual-cluster-id cluster_id \
--name sample-job-name \
--execution-role-arn execution-role-arn \
--release-label emr-6.4.0-latest \
--job-driver '{
  "sparkSubmitJobDriver": {
    "entryPoint": "s3://REGION.elasticmapreduce/emr-containers/samples/wordcount/
scripts/wordcount.py",
    "entryPointArguments": ["s3://amzn-s3-demo-destination-bucket/
wordcount_output"],
    "sparkSubmitParameters": "--conf spark.executor.instances=2 --
conf spark.executor.memory=2G --conf spark.executor.cores=2 --conf
 spark.driver.cores=1"
  }
}' \
--configuration-overrides '{
  "monitoringConfiguration": {
    "cloudWatchMonitoringConfiguration": {
      "logGroupName": "my_log_group_name",
      "logStreamNamePrefix": "my_log_stream_prefix"
    },
    "s3MonitoringConfiguration": {
       "logUri": "s3://amzn-s3-demo-logging-bucket"
    }
  }
}'
```

將在 `s3://`*amzn-s3-demo-destination-bucket*`/wordcount_output` 中提供此作業的輸出資料。

也可以使用指定參數為作業執行建立 JSON 檔案。然後使用 JSON 檔案的路徑執行 `start-job-run` 命令。如需詳細資訊，請參閱使用 `StartJobRun` 提交作業執行。如需有關設定作業執行參數的詳細資訊，請參閱 用於設定作業執行的選項。

3. 要執行 Spark SQL 應用程式，請使用以下命令。將所有*####*值取代為適當的值。*REGION* 是 Amazon EMR on EKS 虛擬叢集所在的區域，例如 *us-east-1*。

```
aws emr-containers start-job-run \
```

```
--virtual-cluster-id cluster_id \
--name sample-job-name \
--execution-role-arn execution-role-arn \
--release-label emr-6.7.0-latest \
--job-driver '{
  "sparkSqlJobDriver": {
    "entryPoint": "s3://query-file.sql",
    "sparkSqlParameters": "--conf spark.executor.instances=2 --
conf spark.executor.memory=2G --conf spark.executor.cores=2 --conf
 spark.driver.cores=1"
  }
}' \
--configuration-overrides '{
  "monitoringConfiguration": {
    "cloudWatchMonitoringConfiguration": {
      "logGroupName": "my_log_group_name",
      "logStreamNamePrefix": "my_log_stream_prefix"
    },
    "s3MonitoringConfiguration": {
       "logUri": "s3://amzn-s3-demo-logging-bucket"
    }
  }
}'
```

範例 SQL 查詢檔案如下所示：您必須擁有外部檔案存放區，例如 S3，用於儲存資料表的資料。

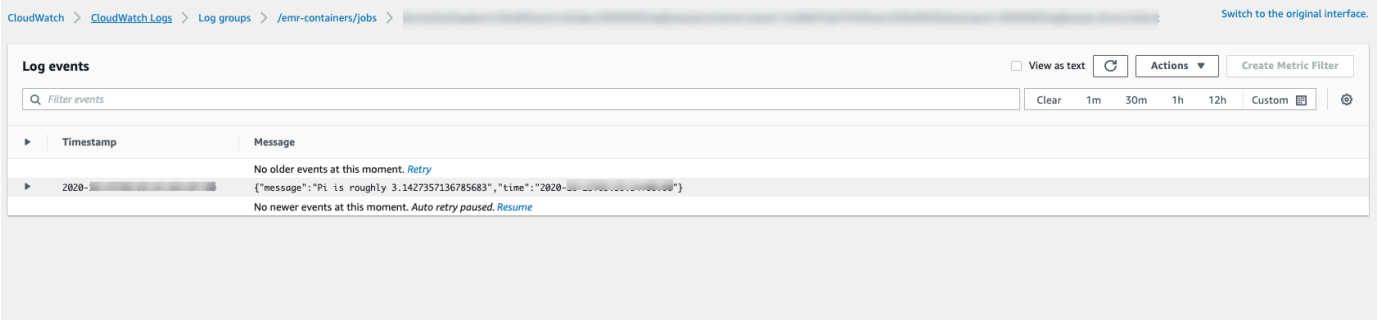
```
CREATE DATABASE demo;
CREATE EXTERNAL TABLE IF NOT EXISTS demo.amazonreview( marketplace string,
 customer_id string, review_id  string, product_id  string, product_parent  string,
 product_title  string, star_rating  integer, helpful_votes  integer, total_votes
  integer, vine  string, verified_purchase  string, review_headline  string,
 review_body  string, review_date  date, year  integer) STORED AS PARQUET LOCATION
 's3://URI to parquet files';
SELECT count(*) FROM demo.amazonreview;
SELECT count(*) FROM demo.amazonreview WHERE star_rating = 3;
```

將在 S3 或 CloudWatch 中的驅動程式的 stdout 日誌中提供此作業的輸出，具體取決於所設定的 `monitoringConfiguration`。

4. 也可以使用指定參數為作業執行建立 JSON 檔案。然後使用 JSON 檔案的路徑執行 start-job-run 命令。如需詳細資訊，請參閱「提交作業執行」。如需有關設定作業執行參數的詳細資訊，請參閱「用於設定作業執行的選項」。

若要監控作業進度或對故障進行偵錯，可以檢查上傳到 Amazon S3、CloudWatch Logs 或兩者的日誌。請參閱 Amazon S3 中設定作業執行以使用 S3 日誌的日誌路徑和設定作業執行以使用 CloudWatch Logs 的 Cloudwatch 日誌。若要查看 CloudWatch Logs 中的日誌，請依以下指示操作。

- 在 https://console.aws.amazon.com/cloudwatch/ 開啟 CloudWatch 主控台。
- 在導覽窗格中，選擇日誌。然後選擇日誌群組。
- 選擇 Amazon EMR on EKS 的日誌群組，然後檢視上傳的日誌事件。




> **Important**
>
> 作業具有預設設定的重試政策。如需有關如何修改或停用組態的資訊，請參閱使用作業重試政策。

# GitHub 上的 Amazon EMR on EKS 最佳實務指南連結

我們已使用開放原始碼社群協作建置 Amazon EMR on EKS 最佳實務指南，以便快速迭代並提供建立和執行虛擬叢集方面的建議。建議您針對這些章節使用 Amazon EMR on EKS 最佳實務指南。選擇各章節中的連結以前往 GitHub 網站。

## 安全

> **Note**
>
> 如果有關 Amazon EMR on EKS 安全性的詳細資訊，請參閱 Amazon EMR on EKS 安全最佳實務。

加密最佳實務：如何對靜態和傳輸中的資料使用加密。

管理網路安全性描述了在連線到託管於 Amazon RDS 和 Amazon Redshift 等 AWS 服務 中的資料來源時，如何為 Amazon EMR on EKS 設定 Pod 的安全群組。

使用 AWS 秘密管理員來存放秘密。

## Pyspark 作業提交

Pyspark 作業提交：使用 zip、egg、wheel 和 pex 等封裝格式，為 PySpark 應用程式指定不同類型的封裝。

## 儲存

使用 EBS 磁碟區：如何針對需要 EBS 磁碟區的作業使用靜態和動態佈建。

使用 Amazon FSx for Lustre 磁碟區：如何針對需要 Amazon FSx for Lustre 磁碟區的作業使用靜態和動態佈建。

使用執行個體儲存體磁碟區：如何使用執行個體儲存體磁碟區來處理作業。

## 中繼存放區整合

使用 Hive 中繼存放區：提供不同方法來使用 Hive 中繼存放區。

使用 AWS Glue： 提供設定 Glue AWS 目錄的不同方法。

## 除錯

使用 Spark 偵錯：如何變更日誌級別。

連線至驅動程式 Pod 上的 Spark 使用者介面。

如何搭配使用自託管的 Spark 歷史記錄伺服器與 Amazon EMR on EKS。

## Amazon EMR on EKS 問題疑難排解

疑難排解。

## 節點放置

將 Kubernetes 節點選取器用於 `single-az` 和其他使用案例。

使用 Fargate 節點放置。

## 效能

使用動態資源配置 (DRA)。

適用於 Amazon VPC Container Network Interface (CNI) 外掛程式、Cluster Autoscaler 和 Core DNS 的 EKS 最佳實務。

## 成本最佳化

使用 Spot 執行個體：Amazon EC2 Spot 執行個體最佳實務，以及如何使用 Spark 節點停用功能。

## 使用 AWS Outposts

使用 在 EKS 上執行 Amazon EMR AWS Outposts

# 自訂 Amazon EMR on EKS 的 Docker 映像檔

可搭配使用自訂 Docker 映像檔與 Amazon EMR on EKS。自訂 Amazon EMR on EKS 執行期映像具有下列優點：

- 將應用程式相依性和執行期環境封裝到單一不可變容器中，以提升可攜性並簡化每個工作負載的相依性管理。
- 安裝和設定針對您的工作負載進行優化的套件。這些套件可能無法在 Amazon EMR 執行期的公開發佈中廣泛使用。
- 將 Amazon EMR on EKS 與組織內目前建立的建置、測試和部署程序整合，包括本機開發和測試。
- 套用已建立的安全程序，例如影像掃描，以符合組織內的合規和監管要求。

主題


- 如何自訂 Docker 映像檔
- 選取基礎映像 URI 的詳細資訊
- 自訂映像的考量事項


## 如何自訂 Docker 映像檔

請依照下列步驟來自訂 Amazon EMR on EKS 的 Docker 映像。這些步驟說明如何取得基礎映像、自訂和發佈映像，以及使用映像提交工作負載。


- 先決條件
- 步驟 1：從 Amazon Elastic Container Registry (Amazon ECR) 中擷取基礎映像
- 步驟 2：自訂基礎映像
- 步驟 3：(選用但不推薦) 驗證自訂映像
- 步驟 4：發布自訂映像
- 步驟 5：使用自訂映像在 Amazon EMR 中提交 Spark 工作負載


> **Note**
>
> 自訂 Docker 映像時，您可能想要考慮的其他選項是針對互動式端點進行自訂，這是為了確保您擁有所需的相依性，或使用多架構容器映像：

- 為互動端點自訂 Docker 映像檔
- 使用多架構映像

## 先決條件

- 完成 Amazon EMR on EKS 的 設定 Amazon EMR on EKS 步驟。
- 在您的環境中安裝 Docker。如需詳細資訊，請參閱獲取 Docker。

## 步驟 1：從 Amazon Elastic Container Registry (Amazon ECR) 中擷取基礎映像

基礎映像包含 Amazon EMR 執行期和用於存取其他 AWS 服務的連接器。對於 Amazon EMR 6.9.0 及更高版本，您可從 Amazon ECR 公共映像庫中獲取基礎映像。瀏覽圖庫以尋找映像連結，然後將映像拉到本地工作區。例如，對於 Amazon EMR 7.8.0 版本，以下`docker pull`命令會取得最新的標準基礎映像。您可以將 `emr-7.8.0:latest` 取代為 `emr-7.8.0-spark-rapids:latest`，以擷取具有 Nvidia RAPIDS Accelerator 的映像。也可以將 `emr-7.8.0:latest` 取代為 `emr-7.8.0-java11:latest`，以使用 Java 11 執行期來擷取映像。

```
docker pull public.ecr.aws/emr-on-eks/spark/emr-7.8.0:latest
```

如果想要擷取 Amazon EMR 6.9.0 或更早版本的基礎映像，或者想要從每個區域的 Amazon ECR 登錄檔帳戶中擷取，請使用下列步驟：

1. 選擇基礎映像 URI。映像 URI 遵循此格式，*ECR-registry-account*.dkr.ecr.*Region*.amazonaws.com/spark/*container-image-tag*，如下列範例所示。

   ```
   895885662937.dkr.ecr.us-west-2.amazonaws.com/spark/emr-6.6.0:latest
   ```

   若要在您的區域中選擇基礎映像，請參閱 選取基礎映像 URI 的詳細資訊。

2. 登入儲存基礎映像的 Amazon ECR 儲存庫。將 *895885662937* 和 *us-west-2* 取代為您選取的 Amazon ECR 登錄帳戶和 AWS 區域。

```
aws ecr get-login-password --region us-west-2 | docker login --username AWS --
password-stdin 895885662937.dkr.ecr.us-west-2.amazonaws.com
```

3. 將基礎映像拉入本機工作區。將 *emr-6.6.0:latest* 取代為您選取的容器映像標籤。

```
docker pull 895885662937.dkr.ecr.us-west-2.amazonaws.com/spark/emr-6.6.0:latest
```

## 步驟 2：自訂基礎映像

請依照下列步驟來自訂您從 Amazon ECR 提取的基礎映像。

1. 在您的本機工作區建立新的 `Dockerfile`。
2. 編輯您剛建立的 `Dockerfile`，並新增下列內容。此 `Dockerfile` 使用您從 `895885662937.dkr.ecr.us-west-2.amazonaws.com/spark/emr-6.6.0:latest` 中提取的容器映像。

```
FROM 895885662937.dkr.ecr.us-west-2.amazonaws.com/spark/emr-6.6.0:latest
USER root
### Add customization commands here ####
USER hadoop:hadoop
```

3. 在 `Dockerfile` 中新增命令以自訂基礎映像。例如，新增一個命令來安裝 Python 程式庫，如以下 `Dockerfile` 所示。

```
FROM 895885662937.dkr.ecr.us-west-2.amazonaws.com/spark/emr-6.6.0:latest
USER root
RUN pip3 install --upgrade boto3 pandas numpy // For python 3
USER hadoop:hadoop
```

4. 從建立 `Dockerfile` 所在的相同目錄中，執行下列命令以建置 Docker 映像檔。提供 Docker 映像檔的名稱，例如 *emr6.6_custom*。

```
docker build -t emr6.6_custom .
```

## 步驟 3：(選用但不推薦) 驗證自訂映像

建議您在發布自訂映像之前先測試它的相容性。您可以使用 Amazon EMR on EKS 自訂映像檔 CLI 來檢查映像是否具有必要的檔案結構和正確的組態，以便在 Amazon EMR on EKS 上執行。

> **Note**
>
> Amazon EMR on EKS 自訂映像 CLI 無法確認您的映像是否沒有錯誤。從基礎映像中移除相依性時，請小心謹慎。

採取下列步驟來驗證自訂映像。

1. 下載並安裝 Amazon EMR on EKS 自訂映像 CLI。如需詳細資訊，請參閱 Amazon EMR on EKS 自訂映像 CLI 安裝指南。
2. 執行下列命令以測試安裝。

```
emr-on-eks-custom-image --version
```

以下顯示輸出範例。

```
Amazon EMR on EKS Custom Image CLI
Version: x.xx
```

3. 執行以下命令來驗證自訂映像。

```
emr-on-eks-custom-image validate-image -i image_name -r release_version [-
t image_type]
```

- `-i` 指定需要驗證的本機映像 URI。這可以是映像 URI、您為映像定義的任何名稱或標記。
- `-r` 指定基礎映像的確切發行版本，例如，`emr-6.6.0-latest`。
- `-t` 指定映像類型。如果為 Spark 映像，請輸入 `spark`。預設值為 `spark`。目前的 Amazon EMR on EKS 自訂映像 CLI 版本僅支援 Spark 執行期映像。

如果成功執行命令，且自訂映像符合所有必要的組態和檔案結構，則傳回的輸出會顯示所有測試結果，如下列範例所示。

```
Amazon EMR on EKS Custom Image Test
```

```
Version: x.xx
... Checking if docker cli is installed
... Checking Image Manifest
[INFO] Image ID: xxx
[INFO] Created On: 2021-05-17T20:50:07.986662904Z
[INFO] Default User Set to hadoop:hadoop : PASS
[INFO] Working Directory Set to /home/hadoop : PASS
[INFO] Entrypoint Set to /usr/bin/entrypoint.sh : PASS
[INFO] SPARK_HOME is set with value: /usr/lib/spark : PASS
[INFO] JAVA_HOME is set with value: /etc/alternatives/jre : PASS
[INFO] File Structure Test for spark-jars in /usr/lib/spark/jars: PASS
[INFO] File Structure Test for hadoop-files in /usr/lib/hadoop: PASS
[INFO] File Structure Test for hadoop-jars in /usr/lib/hadoop/lib: PASS
[INFO] File Structure Test for bin-files in /usr/bin: PASS
... Start Running Sample Spark Job
[INFO] Sample Spark Job Test with local:///usr/lib/spark/examples/jars/spark-
examples.jar : PASS
----------------------------------------------------------------
Overall Custom Image Validation Succeeded.
----------------------------------------------------------------
```

如果自訂映像不符合所需的組態或檔案結構，就會出現錯誤訊息。傳回的輸出會提供有關不正確組態或檔案結構的相關資訊。

## 步驟 4：發布自訂映像

將新的 Docker 映像檔發布至 Amazon ECR 登錄檔。

1. 執行下列命令以建立用於存放 Docker 映像檔的 Amazon ECR 儲存庫。*############ emr6.6_custom_repo*。將 *us-west-2* 取代為您的區域。

```
aws ecr create-repository \
    --repository-name emr6.6_custom_repo \
    --image-scanning-configuration scanOnPush=true \
    --region us-west-2
```

   如需詳細資訊，請參閱《Amazon ECR 使用者指南》中的建立儲存庫。

2. 執行下列命令以驗證預設登錄檔。

```
aws ecr get-login-password --region us-west-2 | docker login --username AWS --
password-stdin aws_account_id.dkr.ecr.us-west-2.amazonaws.com
```

如需詳細資訊，請參閱《Amazon ECR 使用者指南》中的驗證預設登錄檔。

3. 標記映像並將其發布至您建立的 Amazon ECR 儲存庫。

   標記映像。

```
docker tag emr6.6_custom aws_account_id.dkr.ecr.us-
west-2.amazonaws.com/emr6.6_custom_repo
```

   推送映像。

```
docker push aws_account_id.dkr.ecr.us-west-2.amazonaws.com/emr6.6_custom_repo
```

   如需詳細資訊，請參閱《Amazon ECR 使用者指南》中的將映像推送至 Amazon ECR。

## 步驟 5：使用自訂映像在 Amazon EMR 中提交 Spark 工作負載

建立並發布自訂映像後，可以使用自訂映像來提交 Amazon EMR on EKS 作業。

首先，建立 start-job-run-request.json 檔案，並指定 `spark.kubernetes.container.image` 參數來參考自訂映像，如下面的 JSON 檔案範例所示。

> **Note**
>
> 可以使用 `local://` 結構描述來參考自訂映像中的可用檔案，如下面的 JSON 程式碼片段中的 `entryPoint` 引數所示。也可以使用 `local://` 結構描述來參考應用程式相依性。使用 `local://` 結構描述所參考的所有檔案和相依性必須已存在於自訂映像的指定路徑中。

```
{
    "name": "spark-custom-image",
    "virtualClusterId": "virtual-cluster-id",
    "executionRoleArn": "execution-role-arn",
    "releaseLabel": "emr-6.6.0-latest",
```

```
    "jobDriver": {
      "sparkSubmitJobDriver": {
        "entryPoint": "local:///usr/lib/spark/examples/jars/spark-examples.jar",
        "entryPointArguments": [
                  "10"
              ],
         "sparkSubmitParameters": "--class org.apache.spark.examples.SparkPi --conf
 spark.kubernetes.container.image=123456789012.dkr.ecr.us-west-2.amazonaws.com/
emr6.6_custom_repo"
       }
    }
}
```

也可以參考具有 applicationConfiguration 屬性的自訂映像，如下列範例所示。

```
{
    "name": "spark-custom-image",
    "virtualClusterId": "virtual-cluster-id",
    "executionRoleArn": "execution-role-arn",
    "releaseLabel": "emr-6.6.0-latest",
    "jobDriver": {
      "sparkSubmitJobDriver": {
        "entryPoint": "local:///usr/lib/spark/examples/jars/spark-examples.jar",
        "entryPointArguments": [
                  "10"
              ],
         "sparkSubmitParameters": "--class org.apache.spark.examples.SparkPi"
       }
    },
    "configurationOverrides": {
        "applicationConfiguration": [
            {
                "classification": "spark-defaults",
                "properties": {
                    "spark.kubernetes.container.image": "123456789012.dkr.ecr.us-
west-2.amazonaws.com/emr6.6_custom_repo"
                }
            }
        ]
    }
}
```

然後執行 `start-job-run` 命令以提交作業。

```
aws emr-containers start-job-run --cli-input-json file:///./start-job-run-request.json
```

在上面的 JSON 範例中，請將 *emr-6.6.0-latest* 取代為您的 Amazon EMR 發行版本。強烈建議您使用 `-latest` 發行版本，以確保選取的版本包含最新安全更新。如需有關 Amazon EMR 發行版本和其映像標籤的詳細資訊，請參閱 選取基礎映像 URI 的詳細資訊。

> **Note**
>
> 可以使用 `spark.kubernetes.driver.container.image` 和 `spark.kubernetes.executor.container.image` 為驅動程式和執行程式 Pod 指定不同的映像。

## 為互動端點自訂 Docker 映像檔

也可以為互動端點自訂 Docker 映像檔，以便執行自訂的基礎核心映像。這有助於確保從 EMR Studio 執行互動式工作負載時擁有所需的相依性。

1. 請依照上述步驟 1-4 自訂 Docker 映像檔。對於 Amazon EMR 6.9.0 及更高版本，您可從 Amazon ECR 公共映像庫中獲取基礎映像 URI。對於 Amazon EMR 6.9.0 之前的版本，可以在每個 AWS 區域的 Amazon EMR Registry 帳戶中獲取映像，唯一的區別是 Dockerfile 中的基礎映像 URI。基礎映像 URI 的格式如下：

   ```
   ECR-registry-account.dkr.ecr.Region.amazonaws.com/notebook-spark/container-image-tag
   ```

   需要在基礎映像 URI 中使用 `notebook-spark` 而非 `spark`。基礎映像包含 Spark 執行期和隨之一起執行的筆記本核心。如需有關選取區域和容器映像標籤的詳細資訊，請參閱 選取基礎映像 URI 的詳細資訊。

   > **Note**
   >
   > 目前僅支援覆寫基礎映像，且不支援引入非基礎映像 AWS 提供之其他類型的全新核心。

2. 建立可與自訂映像搭配使用的互動端點。

首先，建立稱為 `custom-image-managed-endpoint.json` 的 JSON 檔案，其中具有以下內容。

```
{
    "name": "endpoint-name",
    "virtualClusterId": "virtual-cluster-id",
    "type": "JUPYTER_ENTERPRISE_GATEWAY",
    "releaseLabel": "emr-6.6.0-latest",
    "executionRoleArn": "execution-role-arn",
    "certificateArn": "certificate-arn",
    "configurationOverrides": {
        "applicationConfiguration": [
            {
                "classification": "jupyter-kernel-overrides",
                "configurations": [
                    {
                        "classification": "python3",
                        "properties": {
                            "container-image": "123456789012.dkr.ecr.us-
west-2.amazonaws.com/custom-notebook-python:latest"
                        }
                    },
                    {
                        "classification": "spark-python-kubernetes",
                        "properties": {
                            "container-image": "123456789012.dkr.ecr.us-
west-2.amazonaws.com/custom-notebook-spark:latest"
                        }
                    }
                ]
            }
        ]
    }
}
```

接下來，使用 JSON 檔案中指定的組態建立互動端點，如下列範例所示。

```
aws emr-containers create-managed-endpoint --cli-input-json custom-image-managed-
endpoint.json
```

如需詳細資訊，請參閱為虛擬叢集建立互動端點。

3. 透過 EMR Studio 連接至互動端點。如需詳細資訊，請參閱從 Studio 連接。

## 使用多架構映像

Amazon EMR on EKS 支援 Amazon Elastic Container Registry (Amazon ECR) 的多架構容器映像。如需詳細資訊，請參閱為 Amazon ECR 引入多架構容器映像。

Amazon EMR on EKS 自訂映像支援 AWS Graviton 型 EC2 執行個體non-Graviton-based EC2 執行個體。Graviton 型映像會與非 Graviton 型映像儲存在 Amazon ECR 的相同映像儲存庫中。

例如，若要檢查 Docker 清單檔案是否有 6.6.0 映像，請執行下列命令。

```
docker manifest inspect 895885662937.dkr.ecr.us-west-2.amazonaws.com/spark/
emr-6.6.0:latest
```

此處為輸出。`arm64` 架構適用於 Graviton 執行個體。`amd64` 適用於非 Graviton 執行個體。

```
{
   "schemaVersion": 2,
   "mediaType": "application/vnd.docker.distribution.manifest.list.v2+json",
   "manifests": [
      {
         "mediaType": "application/vnd.docker.distribution.manifest.v2+json",
         "size": 1805,
         "digest":
 "xxx123:6b971cb47d11011ab3d45fff925e9442914b4977ae0f9fbcdcf5cfa99a7593f0",
         "platform": {
            "architecture": "arm64",
            "os": "linux"
         }
      },
      {
         "mediaType": "application/vnd.docker.distribution.manifest.v2+json",
         "size": 1805,
         "digest":
 "xxx123:6f2375582c9c57fa9838c1d3a626f1b4fc281e287d2963a72dfe0bd81117e52f",
         "platform": {
            "architecture": "amd64",
            "os": "linux"
         }
      }
```

```
    ]
}
```

遵循下列步驟來建立多架構映像：

1. 使用以下內容建立 `Dockerfile`，以便可以提取 `arm64` 映像。

```
FROM --platform=arm64 895885662937.dkr.ecr.us-west-2.amazonaws.com/spark/
emr-6.6.0:latest
USER root

RUN pip3 install boto3 // install customizations here
USER hadoop:hadoop
```

2. 遵循為 Amazon ECR 引入多架構容器映像中的說明，以建置多架構映像。

> **Note**
>
> 必須在 `arm64` 執行個體上建立 `arm64` 映像。同樣，必須在 `amd64` 執行個體上建置 `amd64` 映像。

   也可以建置多架構映像，而不必使用 Docker `buildx` 命令在每個特定的執行個體類型上建置。如需詳細資訊，請參閱利用多 CPU 架構支援。

3. 建立多架構映像後，可以提交具有相同 `spark.kubernetes.container.image` 參數的作業，並將其指向映像。在同時具有 AWS Graviton 型non-Graviton-based EC2 執行個體的異質叢集中，執行個體會根據提取映像的執行個體架構來決定正確的架構映像。

# 選取基礎映像 URI 的詳細資訊

> **Note**
>
> 對於 Amazon EMR 6.9.0 版和更高版本，可以從 Amazon ECR 公共映像庫中擷取基礎映像，因此您不需要按照本頁面中的說明直接建構基礎映像 URI。若要尋找基礎映像的容器映像標籤，請參閱版本說明頁面，了解 Amazon EMR on EKS 的對應版本。

您選取的基礎 Docker 映像檔儲存在 Amazon Elastic Container Registry (Amazon ECR) 中。映像 URI 遵循此格式：*ECR-registry-account*.dkr.ecr.*Region*.amazonaws.com/spark/*container-image-tag*，如下列範例所示。

```
895885662937.dkr.ecr.us-west-2.amazonaws.com/spark/emr-7.8.0:latest
```

互動端點的映像 URI 遵循此格式：*ECR-registry-account*.dkr.ecr.*Region*.amazonaws.com/notebook-spark/*container-image-tag*，如下列範例所示。需要在基礎映像 URI 中使用 notebook-spark 而非 spark。

```
895885662937.dkr.ecr.us-west-2.amazonaws.com/notebook-spark/emr-7.8.0:latest
```

同樣，對於互動端點的非 Spark python3 映像，映像 URI 為 *ECR-registry-account*.dkr.ecr.*Region*.amazonaws.com/notebook-python/*container-image-tag*。下列範例 URI 已正確格式化：

```
895885662937.dkr.ecr.us-west-2.amazonaws.com/notebook-python/emr-7.8.0:latest
```

若要尋找基礎映像的容器映像標籤，請參閱版本說明頁面，了解 Amazon EMR on EKS 的對應版本。

## 按區域劃分的 Amazon ECR 登錄檔帳戶

若要避免高網路延遲，請從最近的位置提取基礎映像 AWS 區域。根據下表，選取與您從中提取映像的區域所對應的 Amazon ECR 登錄檔帳戶。

| 區域 | Amazon ECR 登錄檔帳戶 |
| --- | --- |
| ap-east-1 | 736135916053 |
| ap-northeast-1 | 059004520145 |
| ap-northeast-2 | 996579266876 |
| ap-northeast-3 | 705689932349 |
| ap-southeast-3 | 946962994502 |
| ap-south-1 | 235914868574 |

| 區域 | Amazon ECR 登錄檔帳戶 |
| --- | --- |
| ap-south-2 | 691480105545 |
| ap-southeast-1 | 671219180197 |
| ap-southeast-2 | 038297999601 |
| ca-central-1 | 351826393999 |
| eu-central-1 | 107292555468 |
| eu-central-2 | 314408114945 |
| eu-north-1 | 830386416364 |
| eu-west-1 | 483788554619 |
| eu-west-2 | 118780647275 |
| eu-west-3 | 307523725174 |
| eu-south-1 | 238014973495 |
| eu-south-2 | 350796622945 |
| il-central-1 | 395734710648 |
| me-south-1 | 008085056818 |
| me-central-1 | 818935616732 |
| sa-east-1 | 052806832358 |
| us-gov-west-1 | 299385240661 |
| us-gov-east-1 | 299393998622 |
| us-east-1 | 755674844232 |
| us-east-2 | 711395599931 |

| 區域 | Amazon ECR 登錄檔帳戶 |
| --- | --- |
| us-west-1 | 608033475327 |
| us-west-2 | 895885662937 |
| af-south-1 | 358491847878 |
| cn-north-1 | 068337069695 |
| cn-northwest-1 | 068420816659 |

# 自訂映像的考量事項

當您自訂 Docker 映像檔時，可以為您的作業選擇精確的執行期。使用此功能時，請考慮這些最佳實務。其中包括安全性、組態和掛載映像的考量：

- 安全性是 AWS 與您之間的共同責任。您負責對新增至映像的二進位檔案進行安全修補。遵循 Amazon EMR on EKS 安全最佳實務，尤其是 取得自訂映像的最新安全更新 和 套用最低權限準則。
- 當您自訂基礎映像時，必須將 Docker 使用者變更為 `hadoop:hadoop`，以便作業不會與根使用者一起執行。
- Amazon EMR on EKS 會在執行階段將檔案掛載在映像的組態之上，例如 `spark-defaults.conf`。若要覆寫這些組態檔案，建議您在作業提交期間使用 `applicationOverrides` 參數，而不要直接修改自訂映像中的檔案。
- Amazon EMR on EK 會在執行階段掛載特定資料夾。您對這些資料夾所做的任何修改都無法在容器中使用。如果您要為自訂映像新增應用程式或其相依性，建議您選擇不屬於下列預先定義路徑的目錄：
  - `/var/log/fluentd`
  - `/var/log/spark/user`
  - `/var/log/spark/apps`
  - `/mnt`
  - `/tmp`
  - `/home/hadoop`

- 您可以將自訂映像上傳到任何與 Docker 相容的儲存庫，例如 Amazon ECR、Docker Hub 或私有企業儲存庫。如需有關如何使用選取的 Docker 儲存庫設定 Amazon EKS 叢集身分驗證的詳細資訊，請參閱從私有登錄檔提取映像。

# 使用 Amazon EMR on EKS 執行 Flink 作業

Amazon EMR 6.13.0 版及更高版本支援具有 Apache Flink 的 Amazon EMR on EKS 或 Flink Kubernetes Operator，作為 Amazon EMR on EKS 的作業提交模型。使用具有 Apache Flink 的 Amazon EMR on EKS，您可以在自己的 Amazon EKS 叢集上使用 Amazon EMR 發行執行期來部署和管理 Flink 應用程式。在 Amazon EKS 叢集中部署 Flink Kubernetes Operator 之後，即可直接向 Operator 提交 Flink 應用程式。Operator 可管理 Flink 應用程式的生命週期。

主題


- 設定和使用 Flink Kubernetes Operator
- 使用 Flink 原生 Kubernetes
- 自訂 Flink 和 FluentD 的 Docker 映像
- 監控 Flink Kubernetes Operator 和 Flink 作業
- Flink 如何支援高可用性和任務彈性
- 針對 Flink 應用程式使用 Autoscaler
- Amazon EMR on EKS 上 Flink 任務的維護和故障診斷
- 具有 Apache Flink 的 Amazon EMR on EKS 的支援版本


## 設定和使用 Flink Kubernetes Operator

以下頁面描述了如何設定和使用 Flink Kubernetes Operator，以使用 Amazon EMR on EKS 執行 Flink 作業。可用的主題包括必要的先決條件、如何設定您的環境，以及在 Amazon EMR on EKS 上執行 Flink 應用程式。

主題


- 針對 Amazon EMR on EKS 設定 Flink Kubernetes Operator
- 安裝適用於 Amazon EMR on EKS 的 Flink Kubernetes Operator
- 執行 Flink 應用程式
- 執行 Flink 應用程式的安全角色許可
- 針對 Amazon EMR on EKS 解除安裝 Flink Kubernetes Operator

# 針對 Amazon EMR on EKS 設定 Flink Kubernetes Operator

先完成下列任務，然後在 Amazon EKS 上安裝 Flink Kubernetes Operator。如果已經註冊 Amazon Web Services (AWS) 且已在使用 Amazon EKS，則您幾乎可使用 Amazon EMR on EKS。完成下列任務，即可在 Amazon EKS 上設定 Flink Operator。如果已經完成任何先決條件，則可以跳過這些先決條件，然後繼續進行下一個。

- 安裝或更新至最新版本的 AWS CLI- 如果您已安裝 AWS CLI，請確認您擁有最新版本。
- 設定 kubectl 和 eksctl – eksctl 是您用來與 Amazon EKS 通訊的命令列工具。
- 安裝 Helm – 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。
- 開始使用 Amazon EKS – eksctl – 請依照步驟在 Amazon EKS 中建立具有節點的新 Kubernetes 叢集。
- 選擇 Amazon EMR 發行標籤 (6.13.0 版或更新版本） – Amazon EMR 6.13.0 版及更高版本支援 Flink Kubernetes Operator。
- 在 Amazon EKS 叢集上啟用服務帳戶的 IAM 角色 (IRSA)。
- 建立作業執行角色。
- 更新作業執行角色的信任政策 。
- 建立操作員執行角色。此為選擇性步驟。可以對 Flink 作業和操作員使用相同的角色。如果想要為操作員使用不同的 IAM 角色，可以建立單獨的角色。
- 更新操作員執行角色的信任政策。必須針對想要用於 Amazon EMR Flink Kubernetes 操作員服務帳戶的角色明確新增一個信任政策項目。可以遵循以下範例格式：

  JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Federated": "arn:aws:iam::111122223333:oidc-
provider/OIDC_PROVIDER"
            },
            "Action": "sts:AssumeRoleWithWebIdentity",
            "Condition": {
                "StringLike": {
```

```
                    "OIDC_PROVIDER:sub": "system:serviceaccount:NAMESPACE:emr-
containers-sa-flink-operator"
                }
            }
        }
    ]
}
```

# 安裝適用於 Amazon EMR on EKS 的 Flink Kubernetes Operator

本主題透過準備 Flink 部署，協助您開始在 Amazon EKS 上使用 Flink Kubernetes Operator。

## 安裝 Kubernetes Operator

請使用下列步驟來安裝 Kubernetes Operator for Apache Flink。

1. 如果您尚未這麼做，請完成 the section called “設定” 中的步驟。
2. 安裝 *cert-manager* (每個 Amazon EKS 叢集一次) 以允許新增 Webhook 元件。

```
kubectl apply -f https://github.com/cert-manager/cert-manager/releases/download/
v1.12.0/cert-manager.yaml
```

3. 安裝 Helm Chart。

```
export VERSION=7.8.0 # The Amazon EMR release version
export NAMESPACE=The Kubernetes namespace to deploy the operator

helm install flink-kubernetes-operator \
oci://public.ecr.aws/emr-on-eks/flink-kubernetes-operator \
--version $VERSION \
--namespace $NAMESPACE
```

輸出範例：

```
NAME: flink-kubernetes-operator
LAST DEPLOYED: Tue May 31 17:38:56 2022
NAMESPACE: $NAMESPACE
STATUS: deployed
REVISION: 1
TEST SUITE: None
```

4. 等待部署完成，然後驗證 Chart 安裝。

```
kubectl wait deployment flink-kubernetes-operator --namespace $NAMESPACE --for
 condition=Available=True --timeout=30s
```

5. 部署完成時，您應該會看到下列訊息。

```
deployment.apps/flink-kubernetes-operator condition met
```

6. 使用以下命令查看已部署的 Operator。

```
helm list --namespace $NAMESPACE
```

以下顯示範例輸出，其中應用程式版本 `x.y.z-amzn-n` 與 Amazon EMR on EKS 版本的 Flink Operator 版本相對應。如需詳細資訊，請參閱具有 Apache Flink 的 Amazon EMR on EKS 的支援版本。

```
NAME                            NAMESPACE    REVISION    UPDATED
               STATUS      CHART                                  APP VERSION

flink-kubernetes-operator    $NAMESPACE   1           2023-02-22 16:43:45.24148
 -0500 EST    deployed    flink-kubernetes-operator-emr-7.8.0    x.y.z-amzn-n
```

## 執行 Flink 應用程式

使用 Amazon EMR 6.13.0 及更高版本時，您可以在 Amazon EMR on EKS 上使用 Flink Kubernetes Operator 在應用程式模式下執行 Flink 應用程式。使用 Amazon EMR 6.15.0 及更高版本時，您也可以在工作階段模式下執行 Flink 應用程式。本頁面介紹了可用於透過 Amazon EMR on EKS 執行 Flink 應用程式的兩種方法。

> **Note**
>
> 提交 Flink 作業時，必須建立 Amazon S3 儲存貯體來儲存高可用性中繼資料。如果不想使用此功能，可以停用它。依預設會啟用此功能。

必要條件：在使用 Flink Kubernetes Operator 執行 Flink 應用程式之前，請完成 the section called “設定” 和 the section called “安裝 Kubernetes Operator” 中的步驟。

## Application mode

使用 Amazon EMR 6.13.0 及更高版本時，您可以在 Amazon EMR on EKS 上使用 Flink Kubernetes Operator 在應用程式模式下執行 Flink 應用程式。

1. 建立`FlinkDeployment`定義檔案，`basic-example-app-cluster.yaml`如下列範例所示。如果您啟用並使用其中一個選擇加入 AWS 區域，請務必取消註解並設定組態 `fs.s3a.endpoint.region`。

```
apiVersion: flink.apache.org/v1beta1
kind: FlinkDeployment
metadata:
  name: basic-example-app-cluster
spec:
  flinkConfiguration:
    taskmanager.numberOfTaskSlots: "2"
    #fs.s3a.endpoint.region: OPT_IN_AWS_REGION_NAME
    state.checkpoints.dir: CHECKPOINT_S3_STORAGE_PATH
    state.savepoints.dir: SAVEPOINT_S3_STORAGE_PATH
  flinkVersion: v1_17
  executionRoleArn: JOB_EXECUTION_ROLE_ARN
  emrReleaseLabel: "emr-6.13.0-flink-latest" # 6.13 or higher
  jobManager:
    storageDir: HIGH_AVAILABILITY_STORAGE_PATH
    resource:
      memory: "2048m"
      cpu: 1
  taskManager:
    resource:
      memory: "2048m"
      cpu: 1
  job:
    # if you have your job jar in S3 bucket you can use that path as well
    jarURI: local:///opt/flink/examples/streaming/StateMachineExample.jar
    parallelism: 2
    upgradeMode: savepoint
    savepointTriggerNonce: 0
  monitoringConfiguration:
    cloudWatchMonitoringConfiguration:
       logGroupName: LOG_GROUP_NAME
```

2. 使用下列命令提交 Flink 部署。這也將建立名為 `basic-example-app-cluster` 的 `FlinkDeployment` 物件。

```
kubectl create -f basic-example-app-cluster.yaml -n <NAMESPACE>
```

3. 存取 Flink UI。

```
kubectl port-forward deployments/basic-example-app-cluster 8081 -n NAMESPACE
```

4. 開啟 `localhost:8081` 以在本機檢視 Flink 作業。
5. 清除作業。請記得清除為此作業建立的 S3 成品，例如檢查點、高可用性、儲存點中繼資料和 CloudWatch 日誌。

如需有關透過 Flink Kubernetes Operator 提交應用程式至 Flink 的詳細資訊，請參閱 GitHub 上 `apache/flink-kubernetes-operator` 資料夾中的 Flink Kubernetes Operator 範例。

Session mode

使用 Amazon EMR 6.15.0 及更高版本時，您可以在 Amazon EMR on EKS 上使用 Flink Kubernetes Operator 在工作階段模式下執行 Flink 應用程式。

1. 在下列範例中建立名為 `basic-example-app-cluster.yaml` `FlinkDeployment`的定義檔案。如果您啟用並使用其中一個選擇加入 AWS 區域，請務必取消註解並設定組態 `fs.s3a.endpoint.region`。

```
apiVersion: flink.apache.org/v1beta1
kind: FlinkDeployment
metadata:
  name: basic-example-session-cluster
spec:
  flinkConfiguration:
    taskmanager.numberOfTaskSlots: "2"
    #fs.s3a.endpoint.region: OPT_IN_AWS_REGION_NAME
    state.checkpoints.dir: CHECKPOINT_S3_STORAGE_PATH
    state.savepoints.dir: SAVEPOINT_S3_STORAGE_PATH
  flinkVersion: v1_17
  executionRoleArn: JOB_EXECUTION_ROLE_ARN
  emrReleaseLabel: "emr-6.15.0-flink-latest"
  jobManager:
    storageDir: HIGH_AVAILABILITY_S3_STORAGE_PATH
    resource:
```

```
      memory: "2048m"
      cpu: 1
  taskManager:
    resource:
      memory: "2048m"
      cpu: 1
  monitoringConfiguration:
    s3MonitoringConfiguration:
       logUri:
    cloudWatchMonitoringConfiguration:
       logGroupName: LOG_GROUP_NAME
```

2. 使用下列命令提交 Flink 部署。這也將建立名為 `basic-example-session-cluster` 的 `FlinkDeployment` 物件。

```
kubectl create -f basic-example-app-cluster.yaml -n NAMESPACE
```

3. 使用下列命令確認工作階段叢集 `LIFECYCLE` 為 `STABLE`：

```
kubectl get flinkdeployments.flink.apache.org basic-example-session-cluster -
n NAMESPACE
```

輸出應類似以下範例：

```
NAME                                JOB STATUS   LIFECYCLE STATE
basic-example-session-cluster                          STABLE
```

4. 使用以下範例內容，建立 `FlinkSessionJob` 自訂定義資源檔案 `basic-session-job.yaml`：

```
apiVersion: flink.apache.org/v1beta1
kind: FlinkSessionJob
metadata:
  name: basic-session-job
spec:
  deploymentName: basic-session-deployment
  job:
    # If you have your job jar in an S3 bucket you can use that path.
    # To use jar in S3 bucket, set
    # OPERATOR_EXECUTION_ROLE_ARN (--set emrContainers.operatorExecutionRoleArn=
$OPERATOR_EXECUTION_ROLE_ARN)
    # when you install Spark operator
```

```
    jarURI: https://repo1.maven.org/maven2/org/apache/flink/flink-examples-
streaming_2.12/1.16.1/flink-examples-streaming_2.12-1.16.1-TopSpeedWindowing.jar
    parallelism: 2
    upgradeMode: stateless
```

5. 使用下列命令提交 Flink 工作階段作業。這將建立 `FlinkSessionJob` 物件 `basic-session-job`。

```
kubectl apply -f basic-session-job.yaml -n $NAMESPACE
```

6. 使用下列命令確認工作階段叢集 `LIFECYCLE` 為 `STABLE`，且 `JOB STATUS` 為 `RUNNING`：

```
kubectl get flinkdeployments.flink.apache.org basic-example-session-cluster -
n NAMESPACE
```

輸出應類似以下範例：

```
NAME                              JOB STATUS   LIFECYCLE STATE
basic-example-session-cluster     RUNNING      STABLE
```

7. 存取 Flink UI。

```
kubectl port-forward deployments/basic-example-session-cluster 8081 -n NAMESPACE
```

8. 開啟 `localhost:8081` 以在本機檢視 Flink 作業。
9. 清除作業。請記得清除為此作業建立的 S3 成品，例如檢查點、高可用性、儲存點中繼資料和 CloudWatch 日誌。

## 執行 Flink 應用程式的安全角色許可

本主題說明部署和執行 Flink 應用程式的安全角色。管理部署和建立和管理任務、操作員角色和任務角色需要兩個角色。本主題會介紹它們並列出其許可。

### 角色型存取控制

若要部署 Operator 並執行 Flink 作業，必須建立兩個 Kubernetes 角色：一個 Operator 和一個作業角色。Amazon EMR 會在您安裝 Operator 時預設建立兩個角色。

## Operator 角色

我們使用 Operator 角色來管理 `flinkdeployments`，以便為每個 Flink 作業和其他資源 (如服務) 建立和管理 JobManager。

Operator 角色的預設名稱為 `emr-containers-sa-flink-operator` 且需要下列許可。

```
rules:
- apiGroups:
  - ""
  resources:
  - pods
  - services
  - events
  - configmaps
  - secrets
  - serviceaccounts
  verbs:
  - '*'
- apiGroups:
  - rbac.authorization.k8s.io
  resources:
  - roles
  - rolebindings
  verbs:
  - '*'
- apiGroups:
  - apps
  resources:
  - deployments
  - deployments/finalizers
  - replicasets
  verbs:
  - '*'
- apiGroups:
  - extensions
  resources:
  - deployments
  - ingresses
  verbs:
  - '*'
- apiGroups:
  - flink.apache.org
```

```
  resources:
  - flinkdeployments
  - flinkdeployments/status
  - flinksessionjobs
  - flinksessionjobs/status
  verbs:
  - '*'
- apiGroups:
  - networking.k8s.io
  resources:
  - ingresses
  verbs:
  - '*'
- apiGroups:
  - coordination.k8s.io
  resources:
  - leases
  verbs:
  - '*'
```

### 作業角色

JobManager 使用作業角色來建立和管理每個作業的 TaskManagers 和 ConfigMaps。

```
rules:
- apiGroups:
  - ""
  resources:
  - pods
  - configmaps
  verbs:
  - '*'
- apiGroups:
  - apps
  resources:
  - deployments
  - deployments/finalizers
  verbs:
  - '*'
```

## 針對 Amazon EMR on EKS 解除安裝 Flink Kubernetes Operator

請依照下列步驟解除安裝 Flink Kubernetes Operator。

1. 刪除 Operator。

```
helm uninstall flink-kubernetes-operator -n <NAMESPACE>
```

2. 刪除 Helm 不會解除安裝的 Kubernetes 資源。

```
kubectl delete serviceaccounts, roles, rolebindings -l emr-
containers.amazonaws.com/component=flink.operator --namespace <namespace>
kubectl delete crd flinkdeployments.flink.apache.org
 flinksessionjobs.flink.apache.org
```

3. (選用) 刪除 cert-manager。

```
kubectl delete -f https://github.com/jetstack/cert-manager/releases/download/
v1.12.0/cert-manager.yaml
```

# 使用 Flink 原生 Kubernetes

Amazon EMR 6.13.0 版及更高版本支援 Flink Native Kubernetes 作為命令列工具，可以使用它向 Amazon EMR on EKS 叢集提交 Flink 應用程式並執行。

主題


- 針對 Amazon EMR on EKS 設定 Flink Native Kubernetes
- 開始針對 Amazon EMR on EKS 使用 Flink Native Kubernetes
- Native Kubernetes 的 Flink JobManager 服務帳戶安全要求


## 針對 Amazon EMR on EKS 設定 Flink Native Kubernetes

完成下列任務進行設定，然後才能在 Amazon EMR on EKS 上使用 Flink CLI 執行應用程式。如果已經註冊 Amazon Web Services (AWS) 且已在使用 Amazon EKS，則您幾乎可使用 Amazon EMR on EKS。如果已經完成任何先決條件，則可以跳過這些先決條件，然後繼續進行下一個。

- 安裝或更新至最新版本的 AWS CLI- 如果您已安裝 AWS CLI，請確認您擁有最新版本。
- 開始使用 Amazon EKS – eksctl – 請依照步驟，在 Amazon EKS 中建立具有節點的新 Kubernetes 叢集。
- 選擇 Amazon EMR 基礎映像 URI (6.13.0 版或更高版本) - Amazon EMR 6.13.0 版及更高版本支援 Flink Kubernetes 命令。

- 確認 JobManager 服務帳戶具有建立和監控 TaskManager Pod 的適當許可。如需詳細資訊，請參閱 Flink JobManager 服務帳戶對原生 Kubernetes 的安全要求。
- 設定本機 AWS 憑證設定檔。
- 對於您要在其中執行 Flink 應用程式的 Amazon EKS 叢集建立或更新 Kubeconfig 檔案。

## 開始針對 Amazon EMR on EKS 使用 Flink Native Kubernetes

這些步驟說明如何設定、設定服務帳戶，以及執行 Flink 應用程式。Flink Native Kubernetes 用於在執行中的 Kubernetes 叢集上部署 Flink。

### 設定和執行 Flink 應用程式

Amazon EMR 6.13.0 及更高版本支援 Flink Native Kubernetes，以便在 Amazon EKS 叢集上執行 Flink 應用程式。完成以下步驟，以執行 Flink 應用程式：

1. 在使用 Flink Native Kubernetes 命令執行 Flink 應用程式之前，請先完成 the section called “設定” 中的步驟。
2. 下載並安裝 Flink。
3. 設定以下環境變數的值。

```
#Export the FLINK_HOME environment variable to your local installation of Flink
export FLINK_HOME=/usr/local/bin/flink #Will vary depending on your installation
export NAMESPACE=flink
export CLUSTER_ID=flink-application-cluster
export IMAGE=<123456789012.dkr.ecr.sample-AWS ##-.amazonaws.com/flink/emr-6.13.0-
flink:latest>
export FLINK_SERVICE_ACCOUNT=emr-containers-sa-flink
export FLINK_CLUSTER_ROLE_BINDING=emr-containers-crb-flink
```

4. 建立服務帳戶，以管理 Kubernetes 資源。

```
kubectl create serviceaccount $FLINK_SERVICE_ACCOUNT -n $NAMESPACE
kubectl create clusterrolebinding $FLINK_CLUSTER_ROLE_BINDING --clusterrole=edit --
serviceaccount=$NAMESPACE:$FLINK_SERVICE_ACCOUNT
```

5. 執行 `run-application` CLI 命令 。

```
$FLINK_HOME/bin/flink run-application \
    --target kubernetes-application \
    -Dkubernetes.namespace=$NAMESPACE \
```

```
    -Dkubernetes.cluster-id=$CLUSTER_ID \
    -Dkubernetes.container.image.ref=$IMAGE \
    -Dkubernetes.service-account=$FLINK_SERVICE_ACCOUNT \
    local:///opt/flink/examples/streaming/Iteration.jar
2022-12-29 21:13:06,947 INFO  org.apache.flink.kubernetes.utils.KubernetesUtils
            [] - Kubernetes deployment requires a fixed port. Configuration
 blob.server.port will be set to 6124
2022-12-29 21:13:06,948 INFO  org.apache.flink.kubernetes.utils.KubernetesUtils
            [] - Kubernetes deployment requires a fixed port. Configuration
 taskmanager.rpc.port will be set to 6122
2022-12-29 21:13:07,861 WARN
 org.apache.flink.kubernetes.KubernetesClusterDescriptor      [] - Please note that
 Flink client operations(e.g. cancel, list, stop, savepoint, etc.) won't work from
 outside the Kubernetes cluster since 'kubernetes.rest-service.exposed.type' has
 been set to ClusterIP.
2022-12-29 21:13:07,868 INFO
 org.apache.flink.kubernetes.KubernetesClusterDescriptor      [] - Create flink
 application cluster flink-application-cluster successfully, JobManager Web
 Interface: http://flink-application-cluster-rest.flink:8081
```

6. 檢查已建立的 Kubernetes 資源。

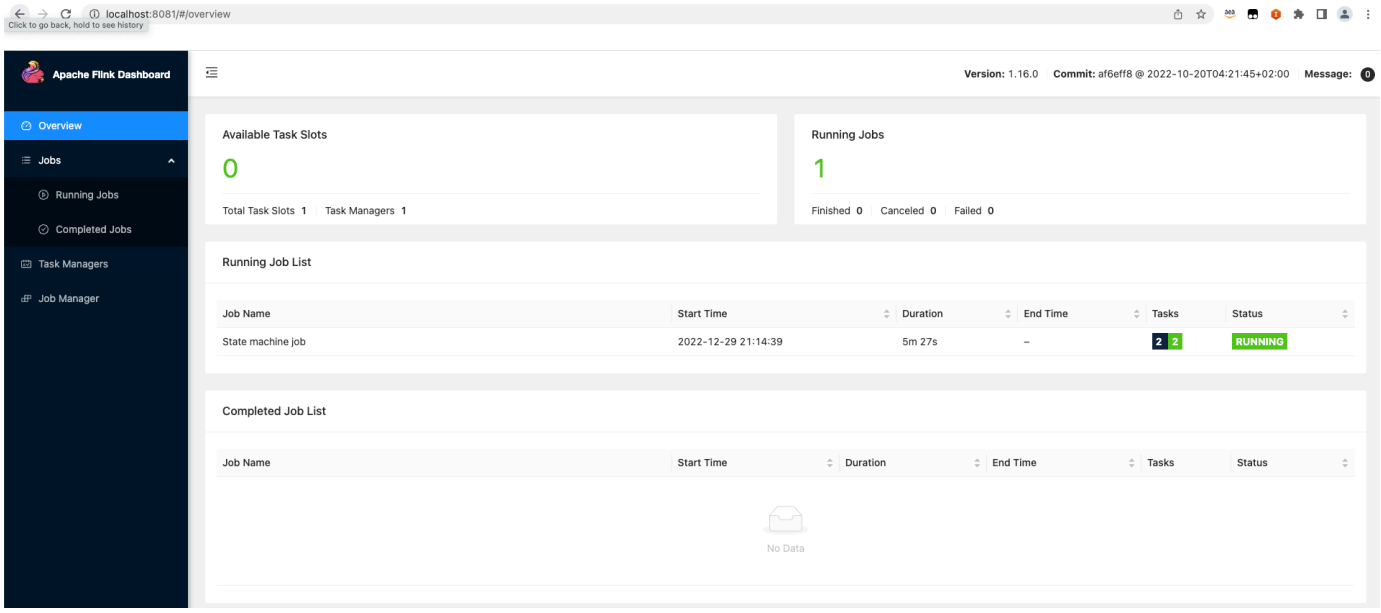
```
kubectl get all -n <namespace>
NAME READY STATUS RESTARTS AGE
pod/flink-application-cluster-546687cb47-w2p2z 1/1 Running 0 3m37s
pod/flink-application-cluster-taskmanager-1-1 1/1 Running 0 3m24s

NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE
service/flink-application-cluster ClusterIP None <none> 6123/TCP,6124/TCP 3m38s
service/flink-application-cluster-rest ClusterIP 10.100.132.158 <none> 8081/TCP
 3m38s

NAME READY UP-TO-DATE AVAILABLE AGE
deployment.apps/flink-application-cluster 1/1 1 1 3m38s

NAME DESIRED CURRENT READY AGE
replicaset.apps/flink-application-cluster-546687cb47 1 1 1 3m38s
```

7. 連接埠轉送到 8081。

```
kubectl port-forward service/flink-application-cluster-rest 8081 -n <namespace>
```

```
Forwarding from 127.0.0.1:8081 -> 8081
```

8. 在本機存取 Flink 使用者介面。




9. 刪除 Flink 應用程式。

```
kubectl delete deployment.apps/flink-application-cluster -n <namespace>
deployment.apps "flink-application-cluster" deleted
```

如需有關提交應用程式至 Flink 的詳細資訊，請參閱 Apache Flink 文件中的 Native Kubernetes。

## Native Kubernetes 的 Flink JobManager 服務帳戶安全要求

Flink JobManager Pod 使用 Kubernetes 服務帳戶來存取 Kubernetes API 伺服器，以建立和監控 TaskManager Pod。JobManager 服務帳戶必須具有適當的許可，才能建立/刪除 TaskManager Pod，並允許 TaskManager 監看領導者 ConfigMaps，以擷取叢集中 JobManager 和 ResourceManager 的地址。

下列規則適用於此服務帳戶。

```
rules:
- apiGroups:
  - ""
  resources:
  - pods
  verbs:
```

```
  - "*"
- apiGroups:
  - ""
  resources:
  - services
  verbs:
  - "*"
- apiGroups:
  - ""
  resources:
  - configmaps
  verbs:
  - "*"
- apiGroups:
  - "apps"
  resources:
  - deployments
  verbs:
  - "*"
```

# 自訂 Flink 和 FluentD 的 Docker 映像

執行下列步驟，使用 Apache Flink 或 FluentD 映像自訂 Amazon EMR on EKS 的 Docker 映像。這包括取得基礎映像、自訂映像、發佈映像以及提交工作負載的技術指導。

主題


- 先決條件
- 步驟 1：從 Amazon Elastic Container Registry 擷取基礎映像
- 步驟 2：自訂基礎映像
- 步驟 3：發佈您的自訂映像
- 步驟 4：使用自訂映像在 Amazon EMR 中提交 Flink 工作負載


## 先決條件

在自訂 Docker 映像之前，請確定您已完成下列先決條件：

- 完成設定 Amazon EMR on EKS 的 Flink Kubernetes Operator 步驟。
- 已在您的環境中安裝 Docker。如需詳細資訊，請參閱獲取 Docker。

## 步驟 1：從 Amazon Elastic Container Registry 擷取基礎映像

基礎映像包含您存取其他 所需的 Amazon EMR 執行期和連接器 AWS 服務。如果您使用 Amazon EMR on EKS 搭配 Flink 6.14.0 版或更新版本，您可以從 Amazon ECR Public Gallery 取得基礎映像。瀏覽圖庫以尋找映像連結，然後將映像拉到本地工作區。例如，對於 Amazon EMR 6.14.0 版本，下列`docker pull`命令會傳回最新的標準基礎映像。將 取代`emr-6.14.0:latest`為您想要的發行版本。

```
docker pull public.ecr.aws/emr-on-eks/flink/emr-6.14.0-flink:latest
```

以下是 Flink 圖庫影像和 Fluentd 圖庫影像的連結：

- emr-on-eks/flink/emr-6.14.0-flink
- emr-on-eks/fluentd/emr-6.14.0(

## 步驟 2：自訂基礎映像

下列步驟說明如何自訂您從 Amazon ECR 提取的基礎映像。

1. 在您的本機工作區建立新的 `Dockerfile`。
2. 編輯 `Dockerfile`並新增下列內容。這會`Dockerfile`使用您從 提取的容器映像`public.ecr.aws/emr-on-eks/flink/emr-7.9.0-flink:latest`。

   ```
   FROM public.ecr.aws/emr-on-eks/flink/emr-7.9.0-flink:latest
   USER root
   ### Add customization commands here ####
   USER hadoop:hadoop
   ```

   如果您使用的是 ，請使用下列組態`Fluentd`。

   ```
   FROM public.ecr.aws/emr-on-eks/fluentd/emr-7.9.0:latest
   USER root
   ### Add customization commands here ####
   USER hadoop:hadoop
   ```

3. 在 `Dockerfile` 中新增命令以自訂基礎映像。下列命令示範如何安裝 Python 程式庫。

   ```
   FROM public.ecr.aws/emr-on-eks/flink/emr-7.9.0-flink:latest
   ```

```
USER root
RUN pip3 install --upgrade boto3 pandas numpy // For python 3
USER hadoop:hadoop
```

4. 在您建立 的相同目錄中`DockerFile`，執行下列命令來建置 Docker 映像。您在`-t`旗標後提供的欄位是影像的自訂名稱。

```
docker build -t <YOUR_ACCOUNT_ID>.dkr.ecr.<YOUR_ECR_REGION>.amazonaws.com/
<ECR_REPO>:<ECR_TAG>
```

## 步驟 3：發佈您的自訂映像

您現在可以將新的 Docker 映像發佈到您的 Amazon ECR 登錄檔。

1. 執行下列命令來建立 Amazon ECR 儲存庫以存放 Docker 映像。為您的儲存庫提供名稱，例如`emr_custom_repo`. 如需詳細資訊，請參閱《Amazon Elastic Container Registry 使用者指南》中的建立儲存庫。

```
aws ecr create-repository \
       --repository-name emr_custom_repo \
       --image-scanning-configuration scanOnPush=true \
       --region <AWS_REGION>
```

2. 執行下列命令以驗證預設登錄檔。如需詳細資訊，請參閱《Amazon Elastic Container Registry 使用者指南》中的驗證您的預設登錄檔。

```
aws ecr get-login-password --region <AWS_REGION> | docker login --username AWS --
password-stdin <AWS_ACCOUNT_ID>.dkr.ecr.<YOUR_ECR_REGION>.amazonaws.com
```

3. 推送映像。如需詳細資訊，請參閱《Amazon Elastic Container Registry 使用者指南》中的將映像推送至 Amazon ECR。

```
docker push <YOUR_ACCOUNT_ID>.dkr.ecr.<YOUR_ECR_REGION>.amazonaws.com/
<ECR_REPO>:<ECR_TAG>
```

## 步驟 4：使用自訂映像在 Amazon EMR 中提交 Flink 工作負載

對`FlinkDeployment`規格進行下列變更，以使用自訂映像。若要這樣做，請在部署規格的`spec.image`行中輸入您自己的映像。

```
apiVersion: flink.apache.org/v1beta1
   kind: FlinkDeployment
   metadata:
     name: basic-example
   spec:
     flinkVersion: v1_18
     image: <YOUR_ACCOUNT_ID>.dkr.ecr.<YOUR_ECR_REGION>.amazonaws.com/
<ECR_REPO>:<ECR_TAG>
     imagePullPolicy: Always
     flinkConfiguration:
           taskmanager.numberOfTaskSlots: "1"
```

若要為您的 Fluentd 任務使用自訂映像，請在部署規格的`monitoringConfiguration.image`行中輸入您自己的映像。

```
  monitoringConfiguration:
       image: <YOUR_ACCOUNT_ID>.dkr.ecr.<YOUR_ECR_REGION>.amazonaws.com/
<ECR_REPO>:<ECR_TAG>
       cloudWatchMonitoringConfiguration:
         logGroupName: flink-log-group
         logStreamNamePrefix: custom-fluentd
```

# 監控 Flink Kubernetes Operator 和 Flink 作業

本章節描述了您可以透過 Amazon EMR on EKS 來監控 Flink 作業的幾種方式。這些包括使用 Flink Web Dashboard 整合 Flink 與 Amazon Managed Service for Prometheus，提供任務狀態和指標，或使用監控組態將日誌資料傳送至 Amazon S3 和 Amazon CloudWatch。

主題


- 使用 Amazon Managed Service for Prometheus 監控 Flink 任務
- 使用 Flink UI 監控 Flink 任務
- 使用監控組態來監控 Flink Kubernetes 運算子和 Flink 任務


## 使用 Amazon Managed Service for Prometheus 監控 Flink 任務

可以整合 Apache Flink 與 Amazon Managed Service for Prometheus (管理入口網站)。Amazon Managed Service for Prometheus 支援從在 Amazon EKS 上執行的叢集的 Amazon Managed Service 中擷取指標。Amazon Managed Service for Prometheus 與 Amazon EKS 叢集上已經執行

的 Prometheus 伺服器搭配使用。執行 Amazon Managed Service for Prometheus 與 Amazon EMR Flink Operator 的整合將自動部署和設定 Prometheus 伺服器，以便與 Amazon Managed Service for Prometheus 整合。

1. 建立 Amazon Managed Service for Prometheus 工作區。此工作區用作擷取端點。您稍後將需要遠端寫入 URL。
2. 設定服務帳戶的 IAM 角色。

   對於這種加入方法，請針對執行 Prometheus 伺服器的 Amazon EKS 叢集中的服務帳戶使用 IAM 角色。這些角色也稱為服務角色。

   如果您還沒有這些角色，請設定服務角色，以便從 Amazon EKS 叢集中擷取指標。

   在繼續之前，請先建立名為 `amp-iamproxy-ingest-role` 的 IAM 角色。
3. 安裝 Amazon EMR Flink Operator 與 Amazon Managed Service for Prometheus。

現在您擁有 Amazon Managed Service for Prometheus 工作區、Amazon Managed Service for Prometheus 的專用 IAM 角色以及必要的許可，可以安裝 Amazon EMR Flink Operator。

建立 `enable-amp.yaml` 檔案。此檔案可讓您使用自訂組態來覆寫 Amazon Managed Service for Prometheus 設定。請務必使用您自己的角色。

```
kube-prometheus-stack:
    prometheus:
    serviceAccount:
        create: true
        name: "amp-iamproxy-ingest-service-account"
        annotations:
            eks.amazonaws.com/role-arn: "arn:aws:iam::<AWS_ACCOUNT_ID>:role/amp-
iamproxy-ingest-role"
    remoteWrite:
        - url: <AMAZON_MANAGED_PROMETHEUS_REMOTE_WRITE_URL>
        sigv4:
            region: <AWS_REGION>
        queueConfig:
            maxSamplesPerSend: 1000
            maxShards: 200
            capacity: 2500
```

使用 `Helm Install --set` 命令將覆寫傳遞至 `flink-kubernetes-operator` 圖表。

```
helm upgrade -n <namespace> flink-kubernetes-operator \
   oci://public.ecr.aws/emr-on-eks/flink-kubernetes-operator \
   --set prometheus.enabled=true
   -f enable-amp.yaml
```

此命令會自動在連接埠 9999 的運算子中安裝 Prometheus 報告程式。任何未來的 `FlinkDeployment` 也會在 9249 上公開 `metrics` 連接埠。

- Flink Operator 指標會顯示在標籤 `flink_k8soperator_` 下的 Prometheus 中。
- Flink Task Manager 指標會顯示在標籤 `flink_taskmanager_` 下的 Prometheus 中。
- Flink Job Manager 指標會顯示在標籤 `flink_jobmanager_` 下的 Prometheus 中。

## 使用 Flink UI 監控 Flink 任務

若要監控執行中 Flink 應用程式的運作狀態和效能，請使用 Flink Web Dashboard。此儀表板提供有關作業狀態、TaskManager 數目以及作業指標和日誌的資訊。它也可讓您檢視和修改 Flink 作業的組態，並與 Flink 叢集互動，以提交或取消作業。

若要存取正在 Kubernetes 上執行的 Flink 應用程式的 Flink Web Dashboard，請執行下列動作：

1. 使用 `kubectl port-forward` 命令將本機連接埠轉送至 Flink 應用程式的 TaskManager Pod 中執行 Flink Web Dashboard 的連接埠。此連接埠預設為 8081。將 *`deployment-name`* 取代為上述 Flink 應用程式部署的名稱。

   ```
   kubectl get deployments -n namespace
   ```

   輸出範例：

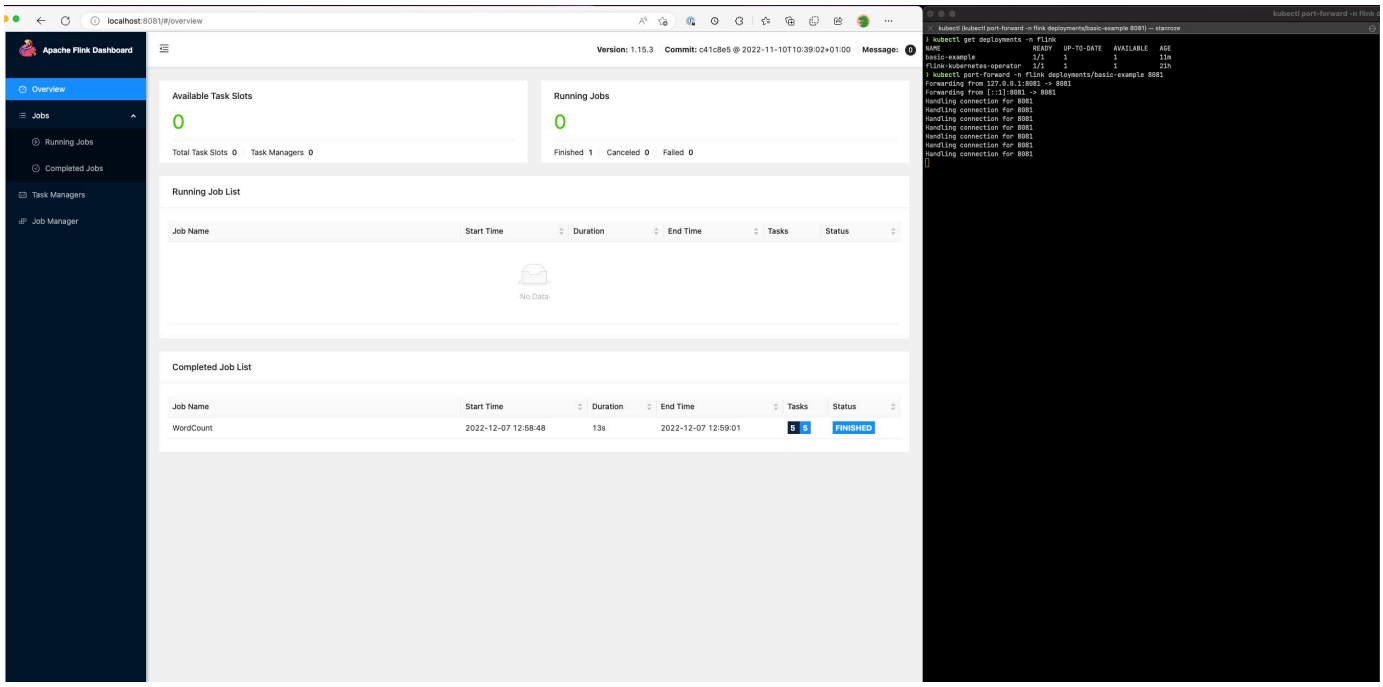
   ```
   kubectl get deployments -n flink-namespace
   NAME                        READY   UP-TO-DATE   AVAILABLE   AGE
   basic-example               1/1       1            1           11m
   flink-kubernetes-operator   1/1       1            1           21h
   ```

   ```
   kubectl port-forward deployments/deployment-name 8081 -n namespace
   ```

2. 如果您想要在本機使用不同的連接埠，請使用 *`local-port`*:8081 參數。

```
kubectl port-forward -n flink deployments/basic-example 8080:8081
```

3. 在網頁瀏覽器中，導覽至 `http://localhost:8081` (或 `http://localhost:local-port`，如果您使用自訂本機連接埠) 以存取 Flink Web Dashboard。此儀表板會顯示有關執行中 Flink 應用程式的資訊，例如作業狀態、TaskManager 數目以及作業指標和日誌。




## 使用監控組態來監控 Flink Kubernetes 運算子和 Flink 任務

監控組態可讓您輕鬆地將 Flink 應用程式和 Operator 日誌的日誌封存設定為 S3 和/或 CloudWatch (您可以選擇其中一個或兩者)。這樣做可將 FluentD 附屬項新增到 JobManager 和 TaskManager Pod，隨後將這些元件的日誌轉發到您設定的接收器。

> **Note**
>
> 必須為 Flink Oerator 和 Flink 作業 (服務帳戶) 的服務帳戶設定 IAM 角色，才能使用此功能，因為它需要與其他 AWS 服務互動。必須在 針對 Amazon EMR on EKS 設定 Flink Kubernetes Operator 中使用 IRSA 進行設定。

### Flink 應用程式日誌

可採用以下方法定義此設定。

```
apiVersion: flink.apache.org/v1beta1
kind: FlinkDeployment
metadata:
  name: basic-example
spec:
  image: FLINK IMAGE TAG
  imagePullPolicy: Always
  flinkVersion: v1_17
  flinkConfiguration:
    taskmanager.numberOfTaskSlots: "2"
  executionRoleArn: JOB EXECUTION ROLE
  jobManager:
    resource:
      memory: "2048m"
      cpu: 1
  taskManager:
    resource:
      memory: "2048m"
      cpu: 1
  job:
    jarURI: local:///opt/flink/examples/streaming/StateMachineExample.jar
  monitoringConfiguration:
    s3MonitoringConfiguration:
      logUri: S3 BUCKET
    cloudWatchMonitoringConfiguration:
      logGroupName: LOG GROUP NAME
      logStreamNamePrefix: LOG GROUP STREAM PREFIX
    sideCarResources:
      limits:
        cpuLimit: 500m
        memoryLimit: 250Mi
    containerLogRotationConfiguration:
        rotationSize: 2GB
        maxFilesToKeep: 10
```

以下是組態選項。

- `s3MonitoringConfiguration` - 用於設定轉送至 S3 的組態金鑰
  - `logUri` (必要) - 想要在其中儲存日誌的 S3 儲存貯體路徑。
  - 上傳日誌後 S3 上的路徑將如下所示。
    - 未啟用日誌輪換：

```
s3://${logUri}/${POD NAME}/STDOUT or STDERR.gz
```

- 日誌輪換已啟用。可以同時使用輪換的檔案和目前檔案 (沒有日期戳記的檔案)。

```
s3://${logUri}/${POD NAME}/STDOUT or STDERR.gz
```

下面格式是遞增的數字。

```
s3://${logUri}/${POD NAME}/stdout_YYYYMMDD_index.gz
```

- 使用此轉寄站需要以下 IAM 許可。

```
{
    "Effect": "Allow",
    "Action": [
        "s3:PutObject"
    ],
    "Resource": [
       "S3_BUCKET_URI/*",
       "S3_BUCKET_URI"
    ]
}
```

- `cloudWatchMonitoringConfiguration` - 用於設定轉送至 CloudWatch 的組態金鑰。
  - `logGroupName` (必要) - 您要向其傳送日誌的 CloudWatch 日誌群組名稱 (如果群組不存在，則自動建立群組)。
  - `logStreamNamePrefix` (選用) - 您想要向其傳送日誌的日誌串流名稱。預設值為空字串。格式如下所示：

```
${logStreamNamePrefix}/${POD NAME}/STDOUT or STDERR
```

  - 使用此轉寄站需要以下 IAM 許可。

```
{
    "Effect": "Allow",
    "Action": [
        "logs:CreateLogStream",
        "logs:CreateLogGroup",
        "logs:PutLogEvents"
    ],
```

```
    "Resource": [
        "arn:aws:logs:REGION:ACCOUNT-ID:log-group:{YOUR_LOG_GROUP_NAME}:*",
        "arn:aws:logs:REGION:ACCOUNT-ID:log-group:{YOUR_LOG_GROUP_NAME}"
    ]
}
```

- `sideCarResources` (選用) - 用於在已啟動的 Fluentbit 附屬容器上設定資源限制的組態金鑰。
  - `memoryLimit` (選用) - 預設值為 512Mi。根據需要進行調整。
  - `cpuLimit` (選用) - 此選項沒有預設值。根據需要進行調整。
- `containerLogRotationConfiguration` (選用) - 控制容器日誌輪換行為。依預設會啟用此功能。
  - `rotationSize` (必要) - 指定日誌輪換的檔案大小。可能的值範圍為 2KB 至 2GB。rotationSize 參數的數值單位部分會以整數形式傳遞。由於不支援小數值，因此可以使用值 1500MB 來指定 1.5GB 的輪換大小。預設值為 2GB。
  - `maxFilesToKeep` (必要) — 指定輪換發生後，要在容器中保留的檔案數上限。下限值是 1，上限值是 50。預設為 10。

## Flink Operator 日誌

也可以使用 Helm Chart 安裝中 `values.yaml` 檔案的下列選項，為 Operator 啟用日誌封存。可以啟用 S3、CloudWatch 或兩者。

```
monitoringConfiguration:
  s3MonitoringConfiguration:
    logUri: "S3-BUCKET"
    totalFileSize: "1G"
    uploadTimeout: "1m"
  cloudWatchMonitoringConfiguration:
    logGroupName: "flink-log-group"
    logStreamNamePrefix: "example-job-prefix-test-2"
  sideCarResources:
    limits:
      cpuLimit: 1
      memoryLimit: 800Mi
  memoryBufferLimit: 700M
```

以下是 `monitoringConfiguration` 下的可用組態選項。

- `s3MonitoringConfiguration` - 將此選項設定為存檔至 S3。

- `logUri` (必要) - 想要在其中儲存日誌的 S3 儲存貯體路徑。
- 以下是上傳日誌後 S3 儲存貯體路徑的可能格式。
  - 未啟用日誌輪換。

    ```
    s3://${logUri}/${POD NAME}/OPERATOR or WEBHOOK/STDOUT or STDERR.gz
    ```

  - 日誌輪換已啟用。可以同時使用輪換的檔案和目前檔案 (沒有日期戳記的檔案)。

    ```
    s3://${logUri}/${POD NAME}/OPERATOR or WEBHOOK/STDOUT or STDERR.gz
    ```

    下面的格式索引是遞增的數字。

    ```
    s3://${logUri}/${POD NAME}/OPERATOR or WEBHOOK/stdout_YYYYMMDD_index.gz
    ```

- `cloudWatchMonitoringConfiguration` - 用於設定轉送至 CloudWatch 的組態金鑰。
  - `logGroupName` (必要) - 您想要向其傳送日誌的 CloudWatch 日誌群組名稱。如果群組不存在，則會自動建立群組。
  - `logStreamNamePrefix` (選用) — 您想要向其傳送日誌的日誌串流名稱。預設值為空字串。CloudWatch 的格式如下所示：

    ```
    ${logStreamNamePrefix}/${POD NAME}/STDOUT or STDERR
    ```

- `sideCarResources` (選用) — 用於在已啟動的 Fluentbit 附屬容器上設定資源限制的組態金鑰。
  - `memoryLimit` (選用) - 記憶體限制。根據需要進行調整。預設值為 512Mi。
  - `cpuLimit` - CPU 限制。根據需要進行調整。無預設值。
- `containerLogRotationConfiguration` (選用) - 控制容器日誌輪換行為。依預設會啟用此功能。
  - `rotationSize` (必要) - 指定日誌輪換的檔案大小。可能的值範圍為 2KB 至 2GB。rotationSize 參數的數值單位部分會以整數形式傳遞。由於不支援小數值，因此可以使用值 1500MB 來指定 1.5GB 的輪換大小。預設值為 2GB。
  - `maxFilesToKeep` (必要) — 指定輪換發生後，要在容器中保留的檔案數上限。下限值是 1，上限值是 50。預設為 10。

# Flink 如何支援高可用性和任務彈性

下列各節概述 Flink 如何讓任務更可靠且高可用性。它透過 Flink 高可用性等內建功能，以及在發生故障時各種復原功能來執行此操作。

主題


- 使用 Flink Operator 和 Flink 應用程式的高可用性 (HA)
- 使用 Amazon EMR on EKS 優化 Flink 作業重新啟動時間，以進行任務復原和擴展操作
- 使用 Amazon EMR on EKS 上的 Flink 正常解除委任 Spot 執行個體


## 使用 Flink Operator 和 Flink 應用程式的高可用性 (HA)

本主題說明如何設定高可用性，並描述它如何適用於幾個不同的使用案例。這包括當您使用 任務管理員時，以及當您使用 Flink 原生 kubernet 時。

### Flink Operator 高可用性

我們啟用 Flink Operator 的高可用性，以便可以容錯移轉至待命 Flink Operator，在發生故障時將 Operator 控制迴圈中的停機時間降至最低。依預設會啟用「高可用性」，且起始 Operator 複本的預設數目為 2。可以在 `values.yaml` 檔案中設定 Helm Chart 的複本欄位。

下列欄位可自訂：

- `replicas` (選用，預設值為 2)：將此數字設定為大於 1 可建立其他待命 Operator，並允許更快速地復原作業。
- `highAvailabilityEnabled` (選用，預設值為 true)：控制是否要啟用 HA。將此參數指定為 true 可啟用多可用區部署支援，並設定正確的 `flink-conf.yaml` 參數。

透過在 `values.yaml` 檔案中設定下列組態，停用 operator 的高可用性。

```
...
imagePullSecrets: []

replicas: 1

# set this to false if you don't want HA
highAvailabilityEnabled: false
...
```

多可用區部署

我們會在多個可用區域建立 operator Pod。這是一個軟約束，如果您在不同的可用區域中沒有足夠的資源，將在相同的可用區域中排程您的 operator Pod 。

確定領導者複本

如果啟用 HA，則複本使用 Lease 來確定哪些 JM 是領導者，並使用 K8s Lease 進行領導者選舉。您可以描述 Lease 並查看 .Spec.Holder Identity 欄位，以確定目前的領導者

```
kubectl describe lease <Helm Install Release Name>-<NAMESPACE>-lease -n <NAMESPACE> |
 grep "Holder Identity"
```

Flink-S3 互動

設定存取憑證

請確定您已設定 IRSA，具有可存取 S3 儲存貯體的適當 IAM 許可。

從 S3 應用程式模式中擷取作業 jar

Flink Operator 也支援從 S3 中擷取應用程式 jar。只需在 FlinkDeployment 規格中提供 jarURI 的 S3 位置即可。

也可以使用此功能來下載其他成品，例如 PyFLink 指令碼。生成的 Python 指令碼放在路徑 `/opt/flink/usrlib/` 下。

以下範例示範如何將此功能用於 PyFLink 作業。請注意 jarURI 和引數欄位。

```
apiVersion: flink.apache.org/v1beta1
kind: FlinkDeployment
metadata:
  name: python-example
spec:
  image: <YOUR CUSTOM PYFLINK IMAGE>
  emrReleaseLabel: "emr-6.12.0-flink-latest"
  flinkVersion: v1_16
  flinkConfiguration:
    taskmanager.numberOfTaskSlots: "1"
  serviceAccount: flink
  jobManager:
    highAvailabilityEnabled: false
```

```
    replicas: 1
    resource:
      memory: "2048m"
      cpu: 1
  taskManager:
    resource:
      memory: "2048m"
      cpu: 1
  job:
    jarURI: "s3:///<S3-BUCKET>/scripts/pyflink.py" # Note, this will trigger the
 artifact download process
    entryClass: "org.apache.flink.client.python.PythonDriver"
    args: ["-pyclientexec", "/usr/local/bin/python3", "-py", "/opt/flink/usrlib/
pyflink.py"]
    parallelism: 1
    upgradeMode: stateless
```

Flink S3 連接器

Flink 隨附有兩個 S3 連接器 (如下所列)。以下各章節討論何時使用哪個連接器。

檢查點：Presto S3 連接器

- 將 S3 結構描述設為 s3p://
- 用於檢查點到 s3 的建議連接器。如需詳細資訊，請參閱 Apache Flink 文件中的 S3-specific。

FlinkDeployment 規格範例：

```
apiVersion: flink.apache.org/v1beta1
kind: FlinkDeployment
metadata:
  name: basic-example
spec:
  flinkConfiguration:
    taskmanager.numberOfTaskSlots: "2"
    state.checkpoints.dir: s3p://<BUCKET-NAME>/flink-checkpoint/
```

讀取和寫入 S3：Hadoop S3 連接器

- 將 S3 結構描述設定為 `s3://` 或 (`s3a://`)
- 用於從 S3 讀取和寫入檔案的建議連接器 (只有 S3 連接器實作 Flinks Filesystem 介面)。

- 根據預設，我們在 `flink-conf.yaml` 檔案`fs.s3a.aws.credentials.provider`中設定 ，也就是 `com.amazonaws.auth.WebIdentityTokenCredentialsProvider`。如果完全覆寫預設 `flink-conf` 並且正在與 S3 進行互動，請確保使用此提供程式。

FlinkDeployment 規格範例

```
apiVersion: flink.apache.org/v1beta1
kind: FlinkDeployment
metadata:
  name: basic-example
spec:
  job:
    jarURI: local:///opt/flink/examples/streaming/WordCount.jar
    args: [ "--input", "s3a://<INPUT BUCKET>/PATH", "--output", "s3a://<OUTPUT BUCKET>/
PATH" ]
    parallelism: 2
    upgradeMode: stateless
```

## Flink Job Manager

Flink 部署的高可用性 (HA) 可讓作業繼續進行，即使遇到暫時性錯誤以及 JobManager 當機。作業將會重新啟動，但會從上次啟用 HA 的成功檢查點開始。如果沒有啟用 HA，Kubernetes 將重新啟動 JobManager，但您的作業將作為新作業開始，並將失去其進度。設定 HA 之後，我們可以告訴 Kubernetes 將 HA 中繼資料儲存在永久性儲存體中，以便在 JobManager 中發生暫時性失敗時進行參考，然後從上次成功的檢查點恢復我們的作業。

Flink 作業預設為啟用 HA (複本計數設為 2，這將要求您提供 S3 儲存位置，以便 HA 中繼資料持續存在)。

HA 組態

```
apiVersion: flink.apache.org/v1beta1
kind: FlinkDeployment
metadata:
  name: basic-example
spec:
  flinkConfiguration:
    taskmanager.numberOfTaskSlots: "2"
  executionRoleArn: "<JOB EXECUTION ROLE ARN>"
  emrReleaseLabel: "emr-6.13.0-flink-latest"
  jobManager:
```

```
    resource:
      memory: "2048m"
      cpu: 1
    replicas: 2
    highAvailabilityEnabled: true
    storageDir: "s3://<S3 PERSISTENT STORAGE DIR>"
  taskManager:
    resource:
      memory: "2048m"
      cpu: 1
```

以下是 Job Manager 中上述 HA 組態的描述 (在 .spec.jobManager 下定義)：

- `highAvailabilityEnabled` (選用，預設值為 true)：如果您不想啟用 HA 且不想使用提供的 HA 組態，請將此設定為 `false` 。您仍然可以操作「複本」欄位以手動設定 HA。
- `replicas` (選用，預設值為 2)：將此數字設定為大於 1 可建立其他待命 JobManager，並允許更快速地復原作業。如果停用 HA，則必須將複本計數設定為 1，否則您將繼續收到驗證錯誤 (如果未啟用 HA，則僅支援 1 個複本)。
- `storageDir` (必填)：因為我們預設使用的複本計數為 2，所以我們必須提供一個持久的 StorageDir。目前，此欄位僅接受 S3 路徑作為儲存位置。

Pod 位置

如果您啟用 HA，我們也會嘗試在同一個可用區域中共置 Pod，進而提升效能 (藉由將 Pod 放在相同可用區域來減少網路延遲)。這是一個竭盡全力的過程，意味著如果您在對大部分 Pod 進行排程的可用區域中沒有足夠的資源，剩餘的 Pod 仍會排程，但最終可能會在此可用區域以外的節點上結束。

確定領導者複本

如果啟用 HA，複本會使用租用來判斷哪個 JM 是領導者，並使用 K8s Configmap 作為儲存此中繼資料的資料儲存。如果您想確定領導者，可以查看 Configmap 的內容，然後查看資料下的關鍵字 `org.apache.flink.k8s.leader.restserver`，以使用該 IP 地址尋找 K8s Pod。也可使用下列 bash 命令。

```
ip=$(kubectl get configmap -n <NAMESPACE> <JOB-NAME>-cluster-config-map -o json | jq -
r ".data[\"org.apache.flink.k8s.leader.restserver\"]" | awk -F: '{print $2}' | awk -F
 '/' '{print $3}')
kubectl get pods -n NAMESPACE  -o json | jq -r ".items[] | select(.status.podIP ==
 \"$ip\") | .metadata.name"
```

## Flink 作業：原生 Kubernetes

Amazon EMR 6.13.0 及更高版本支援 Flink Native Kubernetes，以便在 Amazon EKS 叢集上以高可用性模式執行 Flink 應用程式。

> **Note**
>
> 提交 Flink 作業時，必須建立 Amazon S3 儲存貯體來儲存高可用性中繼資料。如果不想使用此功能，可以停用它。依預設會啟用此功能。

若要開啟 Flink 高可用性功能，請在執行 `run-application` CLI 命令時提供下列 Flink 參數。參數定義於範例下方。

```
-Dhigh-availability.type=kubernetes \
-Dhigh-availability.storageDir=S3://DOC-EXAMPLE-STORAGE-BUCKET \
-
Dfs.s3a.aws.credentials.provider="com.amazonaws.auth.WebIdentityTokenCredentialsProvider"
 \
-Dkubernetes.jobmanager.replicas=3 \
-Dkubernetes.cluster-id=example-cluster
```

- **`Dhigh-availability.storageDir`**：您要在其中存放作業的高可用性中繼資料的 Amazon S3 儲存貯體。

  **`Dkubernetes.jobmanager.replicas`**：要建立的 Job Manager Pod 數量，為大於 1 的整數。

  **`Dkubernetes.cluster-id`**：識別 Flink 叢集的唯一識別碼。

# 使用 Amazon EMR on EKS 優化 Flink 作業重新啟動時間，以進行任務復原和擴展操作

當任務失敗或發生擴展操作時，Flink 會嘗試從最後一個完成的檢查點重新執行任務。根據檢查點狀態的大小和平行任務數量，重新啟動程序可能需要一分鐘或更長的時間才能執行。在重新啟動期間，作業的積壓任務可能會累積。不過，Flink 有一些方法可以優化執行圖表的復原和重新啟動速度，以提高作業穩定性。

此頁面說明 Amazon EMR Flink 在現場執行個體的任務復原或擴展操作期間，可以改善任務重新啟動時間的一些方式。Spot 執行個體是未使用的運算容量，以折扣價提供。它具有獨特的行為，包括偶爾

中斷，因此了解 Amazon EMR on EKS 如何處理這些行為很重要，包括 Amazon EMR on EKS 如何執行除役和任務重新啟動。

主題


- 任務本機復原
- 透過 Amazon EBS 磁碟區掛載進行任務本機復原
- 一般日誌型增量檢查點
- 精細復原
- 調整式排程器中的合併重新啟動機制


## 任務本機復原

> **Note**
>
> Amazon EMR on EKS 6.14.0 及更高版本上的 Flink 支援任務本機復原。

透過 Flink 檢查點，每個任務皆會產生其狀態的快照，Flink 會將其寫入分散式儲存體 (如 Amazon S3)。在復原的情況下，任務會從分散式儲存體還原其狀態。分散式儲存提供容錯能力，並且由於所有節點皆可存取分散式儲存體，因此可以在重新擴展期間重新分配狀態。

但是，遠端分散式存放區也有一個缺點：所有任務均須透過網路從遠端位置讀取其狀態。這可能會導致任務復原或擴展操作期間大型狀態的復原時間較長。

您可透過任務本機復原來解決復原時間較長的問題。任務會將檢查點上的狀態寫入任務本機的次要儲存體，例如本機磁碟上。同時還會將其狀態存放在主要儲存體中 (在此例中為 Amazon S3)。在復原期間，排程器會在較早執行任務的相同 Task Manager 上排定任務，以便其可以從本機狀態存放區復原，而不是從遠端狀態存放區讀取。如需詳細資訊，請參閱《Apache Flink 文件》中的任務本機復原。

我們對範例作業的基準測試指出，啟用任務本機復原後，復原時間已從幾分鐘縮短至幾秒鐘。

若要啟用任務本機復原，請在 `flink-conf.yaml` 檔案中設定下列組態。指定檢查點間隔值 (以毫秒為單位)。

```
state.backend.local-recovery: true
state.backend: hasmap or rocksdb
state.checkpoints.dir: s3://STORAGE-BUCKET-PATH/checkpoint
execution.checkpointing.interval: 15000
```

## 透過 Amazon EBS 磁碟區掛載進行任務本機復原

> **Note**
>
> Amazon EMR on EKS 6.15.0 及更高版本上的 Flink 支援 Amazon EBS 的任務本機復原。

透過 Amazon EMR on EKS 上的 Flink，您可以自動向 TaskManager Pod 佈建 Amazon EBS 磁碟區以進行任務本機復原。預設的覆蓋掛載隨附 10 GB 磁碟區，對於狀態較低的作業而言即足夠。具有大型狀態的作業可以啟用自動 EBS 磁碟區掛載選項。TaskManager Pod 會在 Pod 建立期間自動建立和掛載，並在 Pod 刪除期間移除。

使用下列步驟為 Amazon EMR on EKS 中的 Flink 啟用自動 EBS 磁碟區掛載：

1. 匯出您將在後續步驟中使用的下列變數值。

```
export AWS_REGION=aa-example-1
export FLINK_EKS_CLUSTER_NAME=my-cluster
export AWS_ACCOUNT_ID=111122223333
```

2. 為您的叢集建立或更新 `kubeconfig` YAML 檔案。

```
aws eks update-kubeconfig --name $FLINK_EKS_CLUSTER_NAME --region $AWS_REGION
```

3. 為 Amazon EKS 叢集上的 Amazon EBS 容器儲存介面 (CSI) 驅動程式建立 IAM 服務帳戶。

```
eksctl create iamserviceaccount \
   --name ebs-csi-controller-sa \
   --namespace kube-system \
   --region $AWS_REGION \
   --cluster $FLINK_EKS_CLUSTER_NAME\
   --role-name TLR_${AWS_REGION}_${FLINK_EKS_CLUSTER_NAME} \
   --role-only \
   --attach-policy-arn arn:aws:iam::aws:policy/service-role/
AmazonEBSCSIDriverPolicy \
   --approve
```

4. 使用下列命令建立 Amazon EBS CSI 驅動程式：

```
eksctl create addon \
   --name aws-ebs-csi-driver \
```

```
   --region $AWS_REGION \
   --cluster $FLINK_EKS_CLUSTER_NAME \
   --service-account-role-arn arn:aws:iam::${AWS_ACCOUNT_ID}:role/TLR_
${AWS_REGION}_${FLINK_EKS_CLUSTER_NAME}
```

5. 使用下列命令建立 Amazon EBS 儲存類別：

```
cat # EOF # storage-class.yaml
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ebs-sc
provisioner: ebs.csi.aws.com
volumeBindingMode: WaitForFirstConsumer
EOF
```

然後套用該類別：

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

6. Helm 安裝 Amazon EMR Flink Kubernetes Operator，並提供建立服務帳戶的選項。這會建立要在 Flink 部署中使用的 `emr-containers-sa-flink`。

```
helm install flink-kubernetes-operator flink-kubernetes-operator/ \
   --set jobServiceAccount.create=true \
   --set rbac.jobRole.create=true \
   --set rbac.jobRoleBinding.create=true
```

7. 若要提交 Flink 作業並啟用 EBS 磁碟區的自動佈建以進行任務本機復原，請在 `flink-conf.yaml` 檔案中設定下列組態。調整作業狀態大小的大小限制。將 `serviceAccount` 設定為 `emr-containers-sa-flink`。指定檢查點間隔值 (以毫秒為單位)。並省略 `executionRoleArn`。

```
flinkConfiguration:
    task.local-recovery.ebs.enable: true
    kubernetes.taskmanager.local-recovery.persistentVolumeClaim.sizeLimit: 10Gi
    state.checkpoints.dir: s3://BUCKET-PATH/checkpoint
    state.backend.local-recovery: true
    state.backend: hasmap or rocksdb
    state.backend.incremental: "true"
    execution.checkpointing.interval: 15000
```

```
  serviceAccount: emr-containers-sa-flink
```

當您準備好刪除 Amazon EBS CSI 驅動程式外掛程式時，請使用下列命令：

```
  # Detach Attached Policy
  aws iam detach-role-policy --role-name TLR_${$AWS_REGION}_${FLINK_EKS_CLUSTER_NAME}
 --policy-arn arn:aws:iam::aws:policy/service-role/AmazonEBSCSIDriverPolicy
  # Delete the created Role
  aws iam delete-role --role-name TLR_${$AWS_REGION}_${FLINK_EKS_CLUSTER_NAME}
  # Delete the created service account
  eksctl delete iamserviceaccount --name ebs-csi-controller-sa --namespace kube-system
 --cluster $FLINK_EKS_CLUSTER_NAME --region $AWS_REGION
  # Delete Addon
  eksctl delete addon --name aws-ebs-csi-driver --cluster $FLINK_EKS_CLUSTER_NAME --
region $AWS_REGION
  # Delete the EBS storage class
  kubectl delete -f storage-class.yaml
```

## 一般日誌型增量檢查點

> **Note**
>
> Amazon EMR on EKS 6.14.0 及更高版本上的 Flink 支援一般日誌型增量檢查點。

Flink 1.16 中新增了一般日誌型增量檢查點，以提高檢查點的速度。較快的檢查點間隔通常可減少復原工作，因為復原後需要重新處理的事件較少。如需詳細資訊，請參閱 Apache Flink 部落格上的 Improving speed and stability of checkpointing with generic log-based incremental checkpoints。

我們對範例作業的基準測試指出，使用一般日誌型增量檢查點時，檢查點時間已從幾分鐘縮短至幾秒鐘。

若要啟用一般日誌型增量檢查點，請在 `flink-conf.yaml` 檔案中設定下列組態。指定檢查點間隔值 (以毫秒為單位)。

```
    state.backend.changelog.enabled: true
    state.backend.changelog.storage: filesystem
    dstl.dfs.base-path: s3://bucket-path/changelog
    state.backend.local-recovery: true
```

```
    state.backend: rocksdb
    state.checkpoints.dir: s3://bucket-path/checkpoint
    execution.checkpointing.interval: 15000
```

## 精細復原

> **Note**
>
> Amazon EMR on EKS 6.14.0 及更高版本上的 Flink 提供對預設排程器的精細復原支援。Amazon EMR on EKS 6.15.0 及更高版本上的 Flink 提供調整式排程器中的精細復原支援。

當任務在執行過程中失敗時，Flink 會重設整個執行圖表，並從最後一個完成的檢查點觸發完整的重新執行。相較於僅重新執行失敗的任務，此方式的成本更高。精細復原僅會重新啟動失敗任務的管道連接元件。在下列範例中，作業圖表有 5 個頂點 (A 至 E)。頂點之間的所有連接均採用逐點分佈的管道方式，且作業的 `parallelism.default` 設定為 2。

```
A # B # C # D # E
```

在此範例中，總共有 10 個任務正在執行。第一個管道 (a1 至 e1) 在 TaskManager (TM1) 上執行，而第二個管道 (a2 至 e2) 在另一個 TaskManager (TM2) 上執行。

```
a1 # b1 # c1 # d1 # e1
a2 # b2 # c2 # d2 # e2
```

有兩個管道連接的元件：`a1 # e1` 和 `a2 # e2`。如果 TM1 或 TM2 失敗，則失敗僅會影響正在執行 TaskManager 之管道中的 5 個任務。重新啟動策略只會啟動受影響的管道元件。

精細復原僅適用於完全平行的 Flink 作業。`keyBy()` 或 `redistribute()` 操作不支援。如需詳細資訊，請參閱 Flink Improvement Proposal Jira 專案中的 FLIP-1: Fine Grained Recovery from Task Failures。

若要啟用精細復原，請在 `flink-conf.yaml` 檔案中設定下列組態。

```
jobmanager.execution.failover-strategy: region
restart-strategy: exponential-delay or fixed-delay
```

### 調整式排程器中的合併重新啟動機制

> **Note**
>
> Amazon EMR on EKS 6.15.0 及更高版本上的 Flink 支援調整式排程器中的合併重新啟動機制。

調整式排程器可根據可用的插槽調整作業的並行度。如果沒有足夠的插槽以符合設定的作業並行度，則會自動降低並行度。如果有新的插槽可用，作業就會再次縱向擴展至設定的作業並行度。在沒有足夠的可用資源時，調整式排程器可以避免作業停機。這是 Flink Autoscaler 支援的排程器。基於這些原因，我們建議使用 Amazon EMR Flink 搭配調整式排程器。但是，調整式排程器可能會在短時間內進行多次重新啟動，每新增一個新資源就會重新啟動一次。這可能會導致作業效能下降。

在 Amazon EMR 6.15.0 及更高版本中，Flink 在調整式排程器中具有合併重新啟動機制，該機制會在新增第一個資源時開啟重新啟動時段，然後等到設定的預設 1 分鐘時段間隔為止。當有足夠的資源可使用設定的並行度執行作業時，或當間隔逾時，其會執行單次重新啟動。

我們對範例作業的基準測試指出，當您使用調整式排程器和 Flink 自動擴展器時，此功能比預設行為多處理 10% 的記錄。

若要啟用合併重新啟動機制，請在 `flink-conf.yaml` 檔案中設定下列組態。

```
jobmanager.adaptive-scheduler.combined-restart.enabled: true
jobmanager.adaptive-scheduler.combined-restart.window-interval: 1m
```

## 使用 Amazon EMR on EKS 上的 Flink 正常解除委任 Spot 執行個體

Flink 搭配 Amazon EMR on EKS 可以改善任務復原或擴展操作期間的作業重新啟動時間。

### 概觀

Amazon EMR on EKS 版本 6.15.0 及更高版本支援使用 Apache Flink 在 Amazon EMR on EKS 中的 Spot 執行個體上正常解除委任 Task Manager。作為此功能的一部分，Amazon EMR on EKS 與 Flink 提供以下功能：

- 即時檢查點：Flink 串流作業可回應 Spot 執行個體中斷、對正在執行的作業執行即時 (JIT) 檢查點，並防止在這些 Spot 執行個體上排定其他任務。預設和調整式排程器支援 JIT 檢查點。

- 合併重新啟動機制：合併重新啟動機制會在作業達到目標資源並行度或目前設定的時段結束時，盡力嘗試重新啟動作業。這也可以防止因多個 Spot 執行個體終止而可能導致的連續作業重新啟動。組合重新啟動機制僅適用於調整式排程器。

這些功能具有以下優點：

- 您可以利用 Spot 執行個體來執行 Task Manager 並減少叢集支出。
- 改善 Spot 執行個體 Task Manager 的活性，可提高彈性並實現更有效率的作業排程。
- 您的 Flink 作業將具有更長的正常執行時間，因為 Spot 執行個體終止後的重新啟動次數將減少。

## 正常解除委任的運作方式

請考量下列範例：您佈建一個執行 Apache Flink 的 Amazon EMR on EKS 叢集，並為 Job Manager 指定隨需節點，以及為 Task Manager 指定 Spot 執行個體節點。終止前兩分鐘，Task Manager 收到中斷通知。

在此案例中，Job Manager 會處理 Spot 執行個體中斷訊號、封鎖 Spot 執行個體上其他任務的排程，以及為串流作業啟動 JIT 檢查點。

然後，只有在目前的重新啟動間隔時段中有足夠的新資源可用性以滿足目前的作業並行度之後，Job Manager 才會重新啟動作業圖表。重新啟動時段間隔是根據 Spot 執行個體更換持續時間、新 Job Manager Pod 的建立，以及向 Job Manager 註冊來決定。

## 先決條件

若要使用正常的反編譯，請在執行 Apache Flink 的 Amazon EMR on EKS 叢集上建立和執行串流任務。啟用在至少一個 Spot 執行個體上排定的調整式排程器和 Task Manager，如以下範例所示。您應針對 Job Manager 使用隨選節點，並且只要至少有一個 Spot 執行個體，就可以將隨選節點用於 Task Manager。

```
apiVersion: flink.apache.org/v1beta1
kind: FlinkDeployment
metadata:
  name: deployment_name
spec:
  flinkVersion: v1_17
  flinkConfiguration:
    taskmanager.numberOfTaskSlots: "2"
    cluster.taskmanager.graceful-decommission.enabled: "true"
```

```
    execution.checkpointing.interval: "240s"
    jobmanager.adaptive-scheduler.combined-restart.enabled: "true"
    jobmanager.adaptive-scheduler.combined-restart.window-interval : "1m"
  serviceAccount: flink
  jobManager:
    resource:
      memory: "2048m"
      cpu: 1
    nodeSelector:
      'eks.amazonaws.com/capacityType': 'ON_DEMAND'
  taskManager:
    resource:
      memory: "2048m"
      cpu: 1
    nodeSelector:
      'eks.amazonaws.com/capacityType': 'SPOT'
  job:
    jarURI: flink_job_jar_path
```

## 組態

本節涵蓋了您可以根據解除委任需求指定的大部分組態。

| 金鑰 | 描述 | 預設值 | 可接受值 |
| --- | --- | --- | --- |
| cluster.taskmanager.graceful-decommission.enabled | 啟用 Task Manager 的正常解除委任。 | true | true, false |
| jobmanager.adaptive-scheduler.combined-restart.enabled | 在調整式排程器中啟用合併重新啟動機制。 | false | true, false |

| 金鑰 | 描述 | 預設值 | 可接受值 |
| --- | --- | --- | --- |
| jobmanager.adaptive-scheduler.combined-restart.window-interval | 為作業執行合併重新啟動的合併重新啟動時段間隔。未顯示單位的整數即視為以毫秒為單位。 | 1m | 範例：30、60s、3m、1h |

# 針對 Flink 應用程式使用 Autoscaler

運算子自動擴展器可透過從 Flink 作業中收集指標並在作業頂點層級自動調整平行程度來協助減輕背壓。以下為組態具體形式的範例：

```
apiVersion: flink.apache.org/v1beta1
kind: FlinkDeployment
metadata:
  ...
spec:
  ...
  flinkVersion: v1_18
  flinkConfiguration:
    job.autoscaler.enabled: "true"
    job.autoscaler.stabilization.interval: 1m
    job.autoscaler.metrics.window: 5m
    job.autoscaler.target.utilization: "0.6"
    job.autoscaler.target.utilization.boundary: "0.2"
    job.autoscaler.restart.time: 2m
    job.autoscaler.catch-up.duration: 5m
    pipeline.max-parallelism: "720"
  ...
```

此組態會使用 Amazon EMR 最新版本的預設值。如果您使用其他版本，則可能會有不同的值。

> **Note**
>
> 從 Amazon EMR 7.2.0 開始，您不需要在組態`kubernetes.operator`中包含 字首。如果您使用 7.1.0 或更低版本，則必須在每個組態之前使用 字首。例如，您必須指定 `kubernetes.operator.job.autoscaler.scaling.enabled`。

以下是自動擴展器的組態選項。

- `job.autoscaler.scaling.enabled` – 指定是否啟用自動擴展器的頂點擴展執行。預設值為 `true`。如果您停用此組態，自動擴展器只會收集指標並評估每個頂點的建議平行處理，但不會升級任務。
- `job.autoscaler.stabilization.interval` - 不會執行新擴展的穩定期。預設為 5 分鐘。
- `job.autoscaler.metrics.window` - 擴展指標彙總視窗大小。視窗越大，越平滑和穩定，但自動擴展器可能會更慢，以應對突然的負載變化。預設為 15 分鐘。建議您使用 3 到 60 分鐘之間的值進行實驗。
- `job.autoscaler.target.utilization` - 目標頂點利用率，以提供穩定的作業效能和一些負載波動緩衝。預設值為 `0.7`，目標是作業頂點 70% 的使用率/負載。
- `job.autoscaler.target.utilization.boundary` - 作為額外緩衝的目標頂點利用率邊界，以避免負載波動的立即擴展。預設值為 `0.3`，這表示在觸發擴展動作之前，允許偏離目標使用率 30%。
- `ob.autoscaler.restart.time` - 重新啟動應用程式的預期時間。預設為 5 分鐘。
- `job.autoscaler.catch-up.duration` - 追趕的預期時間，這意味著在擴展操作完成後完全處理任何待處理項。預設為 5 分鐘。透過降低追趕時間，自動擴展器必須為擴展動作保留更多額外容量。
- `pipeline.max-parallelism` - 自動擴展器可以使用的最大並行度。如果自動擴展器高於 Flink 組態中或直接在每個運算子上設定的最大並行度，則會忽略此限制。預設值為 -1。請注意，自動擴展器將並行度計算為最大並行數的除數，因此建議選擇具有大量除數的最大並行度設定，而不是依賴 Flink 提供的預設值。建議此組態使用 60 的倍數，例如 120、180、240、360、720 等。

如需更詳細的組態參考頁面，請參閱自動擴展器組態。

## Autoscaler 參數自動調校

本節說明各種 Amazon EMR 版本的自動調校行為。它也會詳細說明不同的自動調整規模組態。

> **Note**
>
> Amazon EMR 7.2.0 及更高版本使用開放原始碼組態`job.autoscaler.restart.time-tracking.enabled`來啟用重新擴展時間估算。重新調整時間估算的功能與 Amazon EMR 自動調校的功能相同，因此您不需要手動將經驗值指派給重新啟動時間。
> 如果您使用的是 Amazon EMR 7.1.0 或更低版本，您仍然可以使用 Amazon EMR 自動調校。

7.2.0 and higher

Amazon EMR 7.2.0 及更高版本會測量套用自動調整規模決策所需的實際重新啟動時間。在 7.1.0 及更低版本中，您必須使用 組態`job.autoscaler.restart.time`來手動設定預估的最長重新啟動時間。透過使用組態 `job.autoscaler.restart.time-tracking.enabled`，您只需輸入第一次擴展的重新啟動時間。之後，運算子會記錄實際重新啟動時間，並將其用於後續擴展。

若要啟用此追蹤，請使用下列命令：

```
job.autoscaler.restart.time-tracking.enabled: true
```

以下是重新調整規模時間估算的相關組態。

| 組態 | 必要 | 預設 | 描述 |
| --- | --- | --- | --- |
| job.autoscaler.restart.time-tracking.enabled | 否 | False | 指出 Flink Autoscaler 是否應該隨著時間自動調整組態，以最佳化擴展決策。請注意，Autoscaler 只能自動調整 Autoscaler 參數 `restart.time` 。 |
| job.autoscaler.restart.time | 否 | 5m | Amazon EMR on EKS 使用的預期重新啟動時間，直到運算子可以判斷先前擴展的實際重新啟動時間為止。 |
| job.autoscaler.restart.time-tracking.limit | 否 | 15m | `job.autoscaler.restart.time-tracking.enabled` 將 設定為 時 |

| 組態 | 必要 | 預設 | 描述 |
| --- | --- | --- | --- |
| | | | 觀察到的重新啟動時間上限true。 |

以下是您可以用來嘗試重新調整規模時間估算的範例部署規格：

```
apiVersion: flink.apache.org/v1beta1
kind: FlinkDeployment
metadata:
  name: autoscaling-example
spec:
  flinkVersion: v1_18
  flinkConfiguration:

    # Autoscaler parameters
    job.autoscaler.enabled: "true"
    job.autoscaler.scaling.enabled: "true"
    job.autoscaler.stabilization.interval: "5s"
    job.autoscaler.metrics.window: "1m"

    job.autoscaler.restart.time-tracking.enabled: "true"
    job.autoscaler.restart.time: "2m"
    job.autoscaler.restart.time-tracking.limit: "10m"

    jobmanager.scheduler: adaptive
    taskmanager.numberOfTaskSlots: "1"
    pipeline.max-parallelism: "12"

  executionRoleArn: <JOB ARN>
  emrReleaseLabel: emr-7.9.0-flink-latest
  jobManager:
    highAvailabilityEnabled: false
    storageDir: s3://<s3_bucket>/flink/autoscaling/ha/
    replicas: 1
    resource:
      memory: "1024m"
      cpu: 0.5
  taskManager:
    resource:
      memory: "1024m"
      cpu: 0.5
```

```
  job:
    jarURI: s3://<s3_bucket>/some-job-with-back-pressure
    parallelism: 1
    upgradeMode: stateless
```

若要模擬背壓，請使用下列部署規格。

```
job:
    jarURI: s3://<s3_bucket>/pyflink-script.py
    entryClass: "org.apache.flink.client.python.PythonDriver"
    args: ["-py", "/opt/flink/usrlib/pyflink-script.py"]
    parallelism: 1
    upgradeMode: stateless
```

將下列 Python 指令碼上傳至 S3 儲存貯體。

```
import logging
import sys
import time
import random

from pyflink.datastream import StreamExecutionEnvironment
from pyflink.table import StreamTableEnvironment

TABLE_NAME="orders"
QUERY=f"""
CREATE TABLE {TABLE_NAME} (
  id INT,
  order_time AS CURRENT_TIMESTAMP,
  WATERMARK FOR order_time AS order_time - INTERVAL '5' SECONDS
)
WITH (
  'connector' = 'datagen',
  'rows-per-second'='10',
  'fields.id.kind'='random',
  'fields.id.min'='1',
  'fields.id.max'='100'
);
"""

def create_backpressure(i):
    time.sleep(2)
    return i
```

```
def autoscaling_demo():
    env = StreamExecutionEnvironment.get_execution_environment()
    t_env = StreamTableEnvironment.create(env)
    t_env.execute_sql(QUERY)
    res_table = t_env.from_path(TABLE_NAME)

    stream =  t_env.to_data_stream(res_table) \
      .shuffle().map(lambda x: create_backpressure(x))\
      .print()
    env.execute("Autoscaling demo")

if __name__ == '__main__':
    logging.basicConfig(stream=sys.stdout, level=logging.INFO, format="%(message)s")
    autoscaling_demo()
```

若要驗證重新調整規模時間估算是否正常運作，請確定 Flink 運算子的DEBUG層級記錄已啟用。以下範例示範如何更新 helm Chart 檔案 `values.yaml`。然後重新安裝更新的 Helm Chart，然後再次執行 Flink 任務。

```
log4j-operator.properties: |+
  # Flink Operator Logging Overrides
  rootLogger.level = DEBUG
```

取得領導 Pod 的名稱。

```
ip=$(kubectl get configmap -n $NAMESPACE <job-name>-cluster-config-map -o json | jq
 -r ".data[\"org.apache.flink.k8s.leader.restserver\"]" | awk -F: '{print $2}' | awk
 -F '/' '{print $3}')

kubectl get pods -n $NAMESPACE -o json | jq -r ".items[] | select(.status.podIP ==
 \"$ip\") | .metadata.name"
```

執行下列命令以取得指標評估中使用的實際重新啟動時間。

```
kubectl logs <FLINK-OPERATOR-POD-NAME> -c flink-kubernetes-operator -n <OPERATOR-
NAMESPACE> -f | grep "Restart time used in scaling summary computation"
```

您應該會看到類似以下的日誌。請注意，只有第一個擴展使用 `job.autoscaler.restart.time`。後續擴展會使用觀察到的重新啟動時間。

```
2024-05-16 17:17:32,590 o.a.f.a.ScalingExecutor        [DEBUG][default/autoscaler-
example] Restart time used in scaling summary computation: PT2M
2024-05-16 17:19:03,787 o.a.f.a.ScalingExecutor        [DEBUG][default/autoscaler-
example] Restart time used in scaling summary computation: PT14S
2024-05-16 17:19:18,976 o.a.f.a.ScalingExecutor        [DEBUG][default/autoscaler-
example] Restart time used in scaling summary computation: PT14S
2024-05-16 17:20:50,283 o.a.f.a.ScalingExecutor        [DEBUG][default/autoscaler-
example] Restart time used in scaling summary computation: PT14S
2024-05-16 17:22:21,691 o.a.f.a.ScalingExecutor        [DEBUG][default/autoscaler-
example] Restart time used in scaling summary computation: PT14S
```

7.0.0 and 7.1.0

開放原始碼內建的 Flink Autoscaler 使用許多指標來做出最佳的擴展決策。不過，它用於計算的預設值適用於大多數工作負載，可能不適用於指定的任務。新增至 Amazon EMR on EKS 版本的 Flink Operator 的自動調校功能會查看在特定擷取指標上觀察到的歷史趨勢，然後嘗試計算針對特定任務量身打造的最佳值。

| 組態 | 必要 | 預設 | 描述 |
| --- | --- | --- | --- |
| kubernetes.operator.job.autoscaler.autotune.enable | 否 | False | 指出 Flink Autoscaler 是否應該隨著時間自動調整組態，以最佳化自動擴展器擴展決策。目前，Autoscaler 只能自動調整 Autoscaler 參數 `restart.time` 。 |
| kubernetes.operator.job.autoscaler.autotune.metrics.history.max.count | 否 | 3 | 指出 Autoscaler 在 Amazon EMR on EKS 指標組態映射中保留多少歷史 Amazon EMR on EKS 指標。 |
| kubernetes.operator.job.autoscaler.autotune.metrics.restart.count | 否 | 3 | 指出 Autoscaler 開始計算指定任務的平均重新啟動時間之前執行的重新啟動次數。 |

若要啟用自動調校，您必須完成下列操作：

- `kubernetes.operator.job.autoscaler.autotune.enable:` 設定為 `true`

- `metrics.job.status.enable`: 設定為 `TOTAL_TIME`
- 遵循為 Flink 應用程式使用 Autoscaler 的設定，以啟用 Autoscaling。

以下是您可以用來嘗試自動調校的範例部署規格。

```
apiVersion: flink.apache.org/v1beta1
kind: FlinkDeployment
metadata:
  name: autoscaling-example
spec:
  flinkVersion: v1_18
  flinkConfiguration:

    # Autotuning parameters
    kubernetes.operator.job.autoscaler.autotune.enable: "true"
    kubernetes.operator.job.autoscaler.autotune.metrics.history.max.count: "2"
    kubernetes.operator.job.autoscaler.autotune.metrics.restart.count: "1"
    metrics.job.status.enable: TOTAL_TIME

    # Autoscaler parameters
    kubernetes.operator.job.autoscaler.enabled: "true"
    kubernetes.operator.job.autoscaler.scaling.enabled: "true"
    kubernetes.operator.job.autoscaler.stabilization.interval: "5s"
    kubernetes.operator.job.autoscaler.metrics.window: "1m"

    jobmanager.scheduler: adaptive

    taskmanager.numberOfTaskSlots: "1"
    state.savepoints.dir: s3://<S3_bucket>/autoscaling/savepoint/
    state.checkpoints.dir: s3://<S3_bucket>/flink/autoscaling/checkpoint/
    pipeline.max-parallelism: "4"

  executionRoleArn: <JOB ARN>
  emrReleaseLabel: emr-6.14.0-flink-latest
  jobManager:
    highAvailabilityEnabled: true
    storageDir: s3://<S3_bucket>/flink/autoscaling/ha/
    replicas: 1
    resource:
      memory: "1024m"
      cpu: 0.5
  taskManager:
```

```
    resource:
      memory: "1024m"
      cpu: 0.5
  job:
    jarURI: s3://<S3_bucket>/some-job-with-back-pressure
    parallelism: 1
    upgradeMode: last-state
```

若要模擬背壓，請使用下列部署規格。

```
  job:
    jarURI: s3://<S3_bucket>/pyflink-script.py
    entryClass: "org.apache.flink.client.python.PythonDriver"
    args: ["-py", "/opt/flink/usrlib/pyflink-script.py"]
    parallelism: 1
    upgradeMode: last-state
```

將下列 Python 指令碼上傳至 S3 儲存貯體。

```
import logging
import sys
import time
import random

from pyflink.datastream import StreamExecutionEnvironment
from pyflink.table import StreamTableEnvironment

TABLE_NAME="orders"
QUERY=f"""
CREATE TABLE {TABLE_NAME} (
  id INT,
  order_time AS CURRENT_TIMESTAMP,
  WATERMARK FOR order_time AS order_time - INTERVAL '5' SECONDS
)
WITH (
  'connector' = 'datagen',
  'rows-per-second'='10',
  'fields.id.kind'='random',
  'fields.id.min'='1',
  'fields.id.max'='100'
);
"""
```

```
def create_backpressure(i):
    time.sleep(2)
    return i

def autoscaling_demo():
    env = StreamExecutionEnvironment.get_execution_environment()
    t_env = StreamTableEnvironment.create(env)
    t_env.execute_sql(QUERY)
    res_table = t_env.from_path(TABLE_NAME)

    stream =  t_env.to_data_stream(res_table) \
      .shuffle().map(lambda x: create_backpressure(x))\
      .print()
    env.execute("Autoscaling demo")

if __name__ == '__main__':
    logging.basicConfig(stream=sys.stdout, level=logging.INFO, format="%(message)s")
    autoscaling_demo()
```

若要驗證您的自動調校器是否正常運作，請使用下列命令。請注意，您必須使用自己的 Flink Operator 領導 Pod 資訊。

首先取得領導 Pod 的名稱。

```
ip=$(kubectl get configmap -n $NAMESPACE <job-name>-cluster-config-map -o json | jq
 -r ".data[\"org.apache.flink.k8s.leader.restserver\"]" | awk -F: '{print $2}' | awk
 -F '/' '{print $3}')

kubectl get pods -n $NAMESPACE -o json | jq -r ".items[] | select(.status.podIP ==
 \"$ip\") | .metadata.name"
```

取得領導 Pod 的名稱後，您可以執行下列命令。

```
kubectl logs -n $NAMESPACE  -c flink-kubernetes-operator --follow <YOUR-FLINK-
OPERATOR-POD-NAME>  | grep -E 'EmrEks|autotun|calculating|restart|autoscaler'
```

您應該會看到類似以下的日誌。

```
[m[33m2023-09-13 20:10:35,941[m [36mc.a.c.f.k.o.a.EmrEksMetricsAutotuner[m
 [36m[DEBUG][flink/autoscaling-example] Using the latest
 Emr Eks Metric for calculating restart.time for autotuning:
 EmrEksMetrics(restartMetric=RestartMetric(restartingTime=65, numRestarts=1))
```

```
[m[33m2023-09-13 20:10:35,941[m [36mc.a.c.f.k.o.a.EmrEksMetricsAutotuner[m
 [32m[INFO ][flink/autoscaling-example] Calculated average restart.time metric via
 autotuning to be: PT0.065S
```

# Amazon EMR on EKS 上 Flink 任務的維護和故障診斷

下列各節概述如何維護長時間執行的 Flink 任務，並提供有關如何對 Flink 任務的一些常見問題進行故障診斷的指導。

## 維護 Flink 應用程式

主題

- 升級模式

Flink 應用程式通常設計為執行很長時間，例如數週、數月甚至數年。與所有長時間執行的服務一樣，Flink 串流應用程式需要進行維護。這包括錯誤修正、改善項目以及遷移至更高版本的 Flink 叢集。

當 `FlinkDeployment` 和 `FlinkSessionJob` 資源的規格發生變化時，您需要升級正在執行的應用程式。為此，操作員會停止正在執行的作業 (除非已暫停)，並使用最新規格重新部署，對於有狀態的應用程式，則使用上次執行的狀態。

使用者可透過 `JobSpec` 的 `upgradeMode` 設定來控制有狀態的應用程式停止和還原時如何管理狀態。

### 升級模式

選用介紹

無狀態

無狀態應用程式從空狀態升級。

最後狀態

在任何應用程式狀態下快速升級 (即使是失敗的作業) 時不需要運作正常的作業，因為一律會使用最新的成功檢查點。如果遺失 HA 中繼資料，可能需要手動復原。若要限制作業在擷取最新檢查點時回退的時間，您可以設定 `kubernetes.operator.job.upgrade.last-state.max.allowed.checkpoint.age`。如果檢查點早於設定的值，則會改為針對運作正常的作業採用儲存點。工作階段模式不支援此功能。

儲存點

使用儲存點進行升級，可提供最大的安全性和作為備份/分叉點的可能性。儲存點將在升級過程中建立。請注意，需執行 Flink 作業才能建立儲存點。如果作業處於運作狀態不佳的狀態，則會使用最後一個檢查點 (除非 kubernetes.operator.job.upgrade.last-state-fallback.enabled 設定為 false)。如果最後一個檢查點無法使用，作業升級將會失敗。

# 疑難排解

本章節描述了如何疑難排解 Amazon EMR on EKS 的問題。如需有關如何疑難排解 Amazon EMR 一般問題的資訊，請參閱《Amazon EMR 管理指南》中的對叢集進行疑難排解。

- 使用 PersistentVolumeClaims (PVC) 對作業進行疑難排解
- 對 Amazon EMR on EKS 垂直自動擴展進行疑難排解
- 對 Amazon EMR on EKS Spark Operator 進行疑難排解

## 對 Amazon EMR on EKS 中的 Apache Flink 進行疑難排解

安裝 Helm Chart 時找不到資源映射

安裝 Helm Chart 時，可能會遇到下列錯誤訊息。

```
Error: INSTALLATION FAILED: pulling from host 1234567890.dkr.ecr.us-
west-2.amazonaws.com failed with status code [manifests 6.13.0]: 403 Forbidden Error:
 INSTALLATION FAILED: unable to build kubernetes objects from release manifest:
 [resource mapping not found for name: "flink-operator-serving-cert" namespace: "<the
 namespace to install your operator>" from "": no matches for kind "Certificate" in
 version "cert-manager.io/v1"

ensure CRDs are installed first, resource mapping not found for name: "flink-operator-
selfsigned-issuer" namespace: "<the namespace to install your operator>" " from "": no
 matches for kind "Issuer" in version "cert-manager.io/v1"

ensure CRDs are installed first].
```

若要解決此錯誤，請安裝 cert-manager 以啟用新增 Webhook 元件。必須將 cert-manager 安裝到您使用的每個 Amazon EKS 叢集。

```
kubectl apply -f https://github.com/cert-manager/cert-manager/releases/download/v1.12.0
```

AWS 服務 存取遭拒錯誤

如果看到 access denied 錯誤，請確認 Helm Chart `values.yaml` 檔案中 `operatorExecutionRoleArn` 的 IAM 角色具有正確許可。此外，請確保 `FlinkDeployment` 規格中 `executionRoleArn` 下的 IAM 角色具有正確許可。

**`FlinkDeployment` 出現問題**

如果您的 `FlinkDeployment` 處於停止狀態，請使用下列步驟強制刪除部署：

1. 編輯部署執行。

```
kubectl edit -n Flink Namespace flinkdeployments/App Name
```

2. 刪除此完成項。

```
finalizers:
  - flinkdeployments.flink.apache.org/finalizer
```

3. 刪除部署。

```
kubectl delete -n Flink Namespace flinkdeployments/App Name
```

在選擇加入中執行 Flink 應用程式時出現 s3a AWSBadRequestException 問題 AWS 區域

如果您在選擇加入 AWS 區域中執行 Flink 應用程式，您可能會看到下列錯誤：

```
Caused by: org.apache.hadoop.fs.s3a.AWSBadRequestException: getFileStatus on
s3://flink.txt: com.amazonaws.services.s3.model.AmazonS3Exception: Bad Request
(Service: Amazon S3; Status Code: 400; Error Code: 400 Bad Request; Request ID:
 ABCDEFGHIJKL; S3 Extended Request ID:
ABCDEFGHIJKLMNOP=; Proxy: null), S3 Extended Request ID: ABCDEFGHIJKLMNOP=:400 Bad
 Request: Bad Request
(Service: Amazon S3; Status Code: 400; Error Code: 400 Bad Request; Request ID:
 ABCDEFGHIJKL; S3 Extended Request ID: ABCDEFGHIJKLMNOP=; Proxy: null)
```

```
Caused by: org.apache.hadoop.fs.s3a.AWSBadRequestException: getS3Region on flink-
application: software.amazon.awssdk.services.s3.model.S3Exception: null
(Service: S3, Status Code: 400, Request ID: ABCDEFGHIJKLMNOP, Extended Request ID:
 ABCDEFGHIJKLMNOPQRST==):null: null
```

```
(Service: S3, Status Code: 400, Request ID: ABCDEFGHIJKLMNOP, Extended Request ID:
 AHl42uDNaTUFOus/5IIVNvSakBcMjMCH7dd37ky0vE6jhABCDEFGHIJKLMNOPQRST==)
```

若要修正這些錯誤，請在您的`FlinkDeployment`定義檔案中使用以下組態。

```
spec:
  flinkConfiguration:
    taskmanager.numberOfTaskSlots: "2"
    fs.s3a.endpoint.region: OPT_IN_AWS_REGION_NAME
```

我們也建議您使用 SDKv2 登入資料提供者：

```
fs.s3a.aws.credentials.provider:
 software.amazon.awssdk.auth.credentials.WebIdentityTokenFileCredentialsProvider
```

如果您想要使用 SDKv1 登入資料提供者，請確定您的 SDK 支援您的選擇加入區域。如需詳細資訊，請參閱 aws-sdk-java GitHub 儲存庫。

如果您在選擇加入區域執行 Flink SQL 陳述式`S3 AWSBadRequestException`時收到 ，請確定您在 flink 組態規格`fs.s3a.endpoint.region: OPT_IN_AWS_REGION_NAME`中設定組態。

S3A AWSBadRequestException

對於 Amazon EMR 6.15.0 - 7.2.0 版，當您在 CN 區域執行 Flink 工作階段任務時，可能會遇到下列錯誤訊息。這些包括中國（北京）和中國（寧夏）：

```
Error:
 {"type":"org.apache.flink.kubernetes.operator.exception.ReconciliationException","message":"or
                    getFileStatus on s3://ABCDPath:
 software.amazon.awssdk.services.s3.model.S3Exception: null (Service: S3, Status Code:
 400, Request ID: ABCDEFGH, Extended Request ID:
                    ABCDEFGH:null: null (Service: S3, Status Code: 400, Request ID:
 ABCDEFGH, Extended Request ID: ABCDEFGH","additionalMetadata":{},"throwableList":

 [{"type":"org.apache.hadoop.fs.s3a.AWSBadRequestException","message":"getFileStatus on
 s3://ABCDPath: software.amazon.awssdk.services.s3.model.S3Exception:
                    null (Service: S3, Status Code: 400, Request ID: ABCDEFGH, Extended
 Request ID: ABCDEFGH:null: null (Service: S3, Status Code: 400, Request ID: ABCDEFGH,
                    Extended Request ID: ABCDEFGH","additionalMetadata":{}},
{"type":"software.amazon.awssdk.services.s3.model.S3Exception","message":"null
 (Service: S3, Status Code: 400,
```

```
                    Request ID: ABCDEFGH, Extended Request ID:
ABCDEFGH","additionalMetadata":{}}]}
```

知道此問題。團隊正在努力修補所有這些發行版本的 flink 運算子。不過，在我們完成修補程式之前，若要修正此錯誤，您需要下載 flink 運算子 Helm Chart、將其解壓縮（擷取壓縮檔案），並在 Helm Chart 中進行組態變更。

特定步驟如下：

1. 將 變更為，特別是將目錄變更為 helm Chart 的本機資料夾，然後執行下列命令列以提取 helm Chart 並解壓縮（擷取）。

   ```
   helm pull oci://public.ecr.aws/emr-on-eks/flink-kubernetes-operator \
   --version $VERSION \
   --namespace $NAMESPACE
   ```

   ```
   tar -zxvf flink-kubernetes-operator-$VERSION.tgz
   ```

2. 前往 helm Chart 資料夾並尋找 `templates/flink-operator.yaml` 檔案。
3. 尋找 `flink-operator-config` ConfigMap，並在 中新增下列`fs.s3a.endpoint.region`組態`flink-conf.yaml`。例如：

   ```
   {{- if .Values.defaultConfiguration.create }}
   apiVersion: v1
   kind: ConfigMap
   metadata:
     name: flink-operator-config
     namespace: {{ .Release.Namespace }}
     labels:
       {{- include "flink-operator.labels" . | nindent 4 }}
   data:
     flink-conf.yaml: |+
   fs.s3a.endpoint.region: {{ .Values.emrContainers.awsRegion }}
   ```

4. 安裝本機 Helm Chart 並執行您的任務。

# 具有 Apache Flink 的 Amazon EMR on EKS 的支援版本

Apache Flink 可用於以下 Amazon EMR on EKS 版本。如需所有可用版本的資訊，請參閱 Amazon EMR on EKS 發行版本。

| 版本標籤 | Java | Flink | Flink 運算子 |
| --- | --- | --- | --- |
| emr-7.2.0-flink-latest | 17 | 1.18.1 | - |
| emr-7.2.0-flink-k8s-operator-latest | 11 | - | 1.8.0 |
| emr-7.1.0-flink-latest | 17 | 1.18.1 | - |
| emr-7.1.0-flink-k8s-operator-latest | 11 | - | 1.6.1 |
| emr-7.0.0-flink-latest | 11 | 1.18.0 | - |
| emr-7.0.0-flink-k8s-operator-latest | 11 | - | 1.6.1 |
| emr-6.15.0-flink-latest | 11 | 1.17.1 | - |
| emr-6.15.0-flink-k8s-operator-latest | 11 | - | 1.6.0 |
| emr-6.14.0-flink-latest | 11 | 1.17.1 | - |
| emr-6.14.0-flink-k8s-operator-latest | 11 | - | 1.6.0 |
| emr-6.13.0-flink-latest | 11 | 1.17.0 | - |
| emr-6.13.0-flink-k8s-operator-latest | 11 | - | 1.5.0 |

# 使用 Amazon EMR on EKS 執行 Spark 任務

作業執行是您提交至 Amazon EMR on EKS 的工作單位，例如 Spark jar、PySpark 指令碼或 SparkSQL 查詢。本主題概述如何使用 管理任務執行 AWS CLI、使用 Amazon EMR 主控台檢視任務執行，以及疑難排解常見的任務執行錯誤。

請注意，您無法在 Amazon EMR on EKS 上執行 IPv6 Spark 任務

> **Note**
>
> 在使用 Amazon EMR on EKS 提交作業執行之前，必須完成 設定 Amazon EMR on EKS 中的步驟。

主題


- 使用 StartJobRun 執行 Spark 作業
- 使用 Spark Operator 執行 Spark 作業
- 使用 spark-submit 執行 Spark 作業
- 搭配 Amazon EMR on EKS 使用 Apache Livy
- 管理 Amazon EMR on EKS 作業執行
- 使用作業範本
- 使用 Pod 範本
- 使用作業重試政策
- 使用 Spark 事件日誌輪換
- 使用 Spark 容器日誌輪換
- 搭配使用垂直自動擴展與 Amazon EMR Spark 作業


## 使用 `StartJobRun` 執行 Spark 作業

本節包含詳細的設定步驟，讓您的環境準備好執行 Spark 任務，然後提供step-by-step說明。

主題


- 設定 Amazon EMR on EKS

- 使用 StartJobRun 提交作業執行
- 使用作業提交器分類
- 使用 Amazon EMR 容器預設分類

## 設定 Amazon EMR on EKS

完成下列任務，為 Amazon EMR on EKS 進行設定。如果已經註冊 Amazon Web Services (AWS) 且已在使用 Amazon EKS，則您幾乎可使用 Amazon EMR on EKS。略過您已完成的任何任務。

> **Note**
>
> 還可以按照 Amazon EMR on EKS 研討會來設定所有必要的資源，以便在 Amazon EMR on EKS 上執行 Spark 作業。該研討會還透過使用 CloudFormation 範本建立入門所需的資源來提供自動化。如需其他範本和最佳實務，請參閱 GitHub 上的 EMR 容器最佳實務指南。

1. 安裝或更新至最新版本的 AWS CLI
2. 設定 kubectl 和 eksctl
3. 開始使用 Amazon EKS – eksctl
4. 啟用 Amazon EMR on EKS 的叢集存取
5. 啟用 EKS 叢集的 IAM 角色
6. 授予使用者對 Amazon EMR on EKS 的存取權
7. 向 Amazon EMR 註冊 Amazon EKS 叢集

### 啟用 Amazon EMR on EKS 的叢集存取權

下列各節顯示啟用叢集存取的幾種方式。第一種是使用 Amazon EKS 叢集存取管理 (CAM)，後一種是示範如何採取手動步驟來啟用叢集存取。

使用 EKS Access Entry 啟用叢集存取 （建議）

> **Note**
>
> ConfigMap `aws-auth` 已棄用。管理 Kubernetes APIs 存取的建議方法是存取項目。

Amazon EMR 已與 Amazon EKS 叢集存取管理 (CAM) 整合，因此您可以自動設定必要的 AuthN 和 AuthZ 政策，以在 Amazon EKS 叢集命名空間中執行 Amazon EMR Spark 任務。當您從 Amazon EKS 叢集命名空間建立虛擬叢集時，Amazon EMR 會自動設定所有必要的許可，因此您不需要在目前的工作流程中新增任何額外的步驟。

> **Note**
>
> Amazon EMR 與 Amazon EKS CAM 整合僅支援新的 Amazon EMR on EKS 虛擬叢集。您無法遷移現有的虛擬叢集以使用此整合。

先決條件

- 請確定您正在執行 2.15.3 版或更新版本的 AWS CLI
- 您的 Amazon EKS 叢集必須位於 1.23 版或更新版本。

設定

若要從 Amazon EKS 設定 Amazon EMR 與 AccessEntry API 操作之間的整合，請確定您已完成下列項目：

- 請確定 Amazon EKS 叢集`authenticationMode`的 設定為 `API_AND_CONFIG_MAP`。

  ```
  aws eks describe-cluster --name <eks-cluster-name>
  ```

  如果尚未設定，請將 `authenticationMode`設定為 `API_AND_CONFIG_MAP`。

  ```
  aws eks update-cluster-config
      --name <eks-cluster-name>
      --access-config authenticationMode=API_AND_CONFIG_MAP
  ```

  如需身分驗證模式的詳細資訊，請參閱叢集身分驗證模式。

- 請確定您用來執行 `CreateVirtualCluster`和 `DeleteVirtualCluster` API 操作的 IAM 角色也具有下列許可：

  ```
  {
    "Effect": "Allow",
    "Action": [
  ```

```
    "eks:CreateAccessEntry"
  ],
  "Resource":
 "arn:<AWS_PARTITION>:eks:<AWS_REGION>:<AWS_ACCOUNT_ID>:cluster/<EKS_CLUSTER_NAME>"
},
{
  "Effect": "Allow",
  "Action": [
    "eks:DescribeAccessEntry",
    "eks:DeleteAccessEntry",
    "eks:ListAssociatedAccessPolicies",
    "eks:AssociateAccessPolicy",
    "eks:DisassociateAccessPolicy"
  ],
  "Resource": "arn:<AWS_PARTITION>:eks:<AWS_REGION>:<AWS_ACCOUNT_ID>:access-entry/
<EKS_CLUSTER_NAME>/role/<AWS_ACCOUNT_ID>/AWSServiceRoleForAmazonEMRContainers/*"
}
```

### 概念和術語

以下是與 Amazon EKS CAM 相關的術語和概念清單。

- 虛擬叢集 (VC) – 在 Amazon EKS 中建立的命名空間邏輯表示法。這是 Amazon EKS 叢集命名空間的 1：1 連結。您可以使用它在指定命名空間內的 Amazon EKS 叢集上執行 Amazon EMR 工作負載。
- 命名空間 – 隔離單一 EKS 叢集內資源群組的機制。
- 存取政策 – 將存取權和動作授予 EKS 叢集中 IAM 角色的許可。
- 存取項目 – 使用角色 arn 建立的項目。您可以將存取項目連結至存取政策，以在 Amazon EKS 叢集中指派特定許可。
- EKS 存取項目整合虛擬叢集 – 使用來自 Amazon EKS 的存取項目 API 操作建立的虛擬叢集。

### 使用 啟用叢集存取 `aws-auth`

必須採取下列動作，以允許 Amazon EMR on EKS 存取叢集中的特定命名空間：建立 Kubernetes 角色、將角色繫結至 Kubernetes 使用者，以及將 Kubernetes 使用者映射至服務連結角色 `AWSServiceRoleForAmazonEMRContainers`。當 IAM 身分映射命令與 `emr-containers` 一起用作服務名稱時，會在 `eksctl` 中自動化這些動作。可以透過使用下列命令輕鬆執行這些操作。

```
eksctl create iamidentitymapping \
```

```
    --cluster my_eks_cluster \
    --namespace kubernetes_namespace \
    --service-name "emr-containers"
```

將 *my_eks_cluster* 取代為 Amazon EKS 叢集的名稱，並將 *kubernetes_namespace* 取代為執行 Amazon EMR 工作負載而建立的 Kubernetes 命名空間。

> **Important**
>
> 您必須使用上一個步驟下載最新的 eksctl 設定 kubectl 和 eksctl，才能使用此功能。

啟用 Amazon EMR on EKS 的叢集存取權的手動步驟

也可使用以下手動步驟來啟用 Amazon EMR on EKS 的叢集存取權。

1. 在特定命名空間中建立 Kubernetes 角色

   Amazon EKS 1.22 - 1.29

   使用 Amazon EKS 1.22 - 1.29，執行下列命令在特定命名空間中建立 Kubernetes 角色。此角色將必要的 RBAC 許可授予 Amazon EMR on EKS。

```
namespace=my-namespace
cat - >>EOF | kubectl apply -f - >>namespace "${namespace}"
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: emr-containers
  namespace: ${namespace}
rules:
  - apiGroups: [""]
    resources: ["namespaces"]
    verbs: ["get"]
  - apiGroups: [""]
    resources: ["serviceaccounts", "services", "configmaps", "events", "pods",
 "pods/log"]
    verbs: ["get", "list", "watch", "describe", "create", "edit", "delete",
 "deletecollection", "annotate", "patch", "label"]
  - apiGroups: [""]
    resources: ["secrets"]
    verbs: ["create", "patch", "delete", "watch"]
```

```
  - apiGroups: ["apps"]
    resources: ["statefulsets", "deployments"]
    verbs: ["get", "list", "watch", "describe", "create", "edit", "delete",
 "annotate", "patch", "label"]
  - apiGroups: ["batch"]
    resources: ["jobs"]
    verbs: ["get", "list", "watch", "describe", "create", "edit", "delete",
 "annotate", "patch", "label"]
  - apiGroups: ["extensions", "networking.k8s.io"]
    resources: ["ingresses"]
    verbs: ["get", "list", "watch", "describe", "create", "edit", "delete",
 "annotate", "patch", "label"]
  - apiGroups: ["rbac.authorization.k8s.io"]
    resources: ["roles", "rolebindings"]
    verbs: ["get", "list", "watch", "describe", "create", "edit", "delete",
 "deletecollection", "annotate", "patch", "label"]
  - apiGroups: [""]
    resources: ["persistentvolumeclaims"]
    verbs: ["get", "list", "watch", "describe", "create", "edit", "delete",
 "deletecollection", "annotate", "patch", "label"]
EOF
```

Amazon EKS 1.21 and below

使用 Amazon EKS 1.21 及以下版本，執行下列命令在特定命名空間中建立 Kubernetes 角色。此角色將必要的 RBAC 許可授予 Amazon EMR on EKS。

```
namespace=my-namespace
cat - >>EOF | kubectl apply -f - >>namespace "${namespace}"
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: emr-containers
  namespace: ${namespace}
rules:
  - apiGroups: [""]
    resources: ["namespaces"]
    verbs: ["get"]
  - apiGroups: [""]
    resources: ["serviceaccounts", "services", "configmaps", "events", "pods",
 "pods/log"]
```

```
    verbs: ["get", "list", "watch", "describe", "create", "edit", "delete",
 "deletecollection", "annotate", "patch", "label"]
  - apiGroups: [""]
    resources: ["secrets"]
    verbs: ["create", "patch", "delete", "watch"]
  - apiGroups: ["apps"]
    resources: ["statefulsets", "deployments"]
    verbs: ["get", "list", "watch", "describe", "create", "edit", "delete",
 "annotate", "patch", "label"]
  - apiGroups: ["batch"]
    resources: ["jobs"]
    verbs: ["get", "list", "watch", "describe", "create", "edit", "delete",
 "annotate", "patch", "label"]
  - apiGroups: ["extensions"]
    resources: ["ingresses"]
    verbs: ["get", "list", "watch", "describe", "create", "edit", "delete",
 "annotate", "patch", "label"]
  - apiGroups: ["rbac.authorization.k8s.io"]
    resources: ["roles", "rolebindings"]
    verbs: ["get", "list", "watch", "describe", "create", "edit", "delete",
 "deletecollection", "annotate", "patch", "label"]
  - apiGroups: [""]
    resources: ["persistentvolumeclaims"]
    verbs: ["get", "list", "watch", "describe", "create", "edit", "delete",
 "deletecollection", "annotate", "patch", "label"]
EOF
```

2. 建立範圍為命名空間的 Kubernetes 角色繫結

   執行下列命令，在指定命名空間中建立 Kubernetes 角色繫結。此角色繫結會將在上一個步驟建立的角色中定義的許可授予名為 `emr-containers` 的使用者。此使用者可識別 Amazon EMR on EKS 的服務連結角色，因此可讓 Amazon EMR on EKS 執行您建立的角色所定義的動作。

```
namespace=my-namespace

cat - <<EOF | kubectl apply -f - --namespace "${namespace}"
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: emr-containers
  namespace: ${namespace}
subjects:
```

```
- kind: User
  name: emr-containers
  apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: Role
  name: emr-containers
  apiGroup: rbac.authorization.k8s.io
EOF
```

3. 更新 Kubernetes **aws-auth** 組態地圖

   可以使用下列其中一個選項，將 Amazon EMR on EKS 服務連結角色與上一個步驟中受 Kubernetes 角色繫結的 `emr-containers` 使用者進行映射。

   選項 1：使用 **eksctl**

   執行下列 `eksctl` 命令，將 Amazon EMR on EKS 服務連結角色與 `emr-containers` 使用者進行映射。

```
eksctl create iamidentitymapping \
    --cluster my-cluster-name \
    --arn "arn:aws:iam::my-account-id:role/AWSServiceRoleForAmazonEMRContainers" \
    --username emr-containers
```

   選項 2：不使用 eksctl

   1. 執行下列命令，在文字編輯器中開啟 `aws-auth` 組態映射。

```
kubectl edit -n kube-system configmap/aws-auth
```

   > **Note**
   >
   > 如果收到錯誤訊息，指出 `Error from server (NotFound): configmaps "aws-auth" not found`，請參閱《Amazon EKS 使用者指南》中的新增使用者角色中的步驟，以套用儲存的 ConfigMap。

   2. 將 Amazon EMR on EKS 服務連結角色詳細資訊新增至 `data` 下 `ConfigMap` 的 `mapRoles` 章節。若此區段在檔案不存在，則將其新增。資料下的已更新 `mapRoles` 章節類似下列範例。

```
apiVersion: v1
```

```
data:
  mapRoles: |
    - rolearn: arn:aws:iam::<your-account-id>:role/
AWSServiceRoleForAmazonEMRContainers
      username: emr-containers
    - ... <other previously existing role entries, if there's any>.
```

3. 儲存檔案並結束您的文字編輯器。

## 啟用 EKS 叢集的 IAM 角色

下列主題詳細說明啟用 IAM 角色的選項。

主題

- 選項 1：在 EKS 叢集上啟用 EKS Pod 身分
- 選項 2：在 EKS 叢集上啟用服務帳戶的 IAM 角色 (IRSA)

### 選項 1：在 EKS 叢集上啟用 EKS Pod 身分

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。

自 emr-7.3.0 發行以來，Amazon EMR on EKS 開始支援 StartJobRun 提交模型的 EKS Pod 身分。

如需 EKS Pod 身分的詳細資訊，請參閱了解 EKS Pod 身分的運作方式。

為什麼選擇 EKS Pod 身分？

作為 EMR 設定的一部分，任務執行角色需要在特定命名空間 (EMR 虛擬叢集） 中的 IAM 角色和服務帳戶之間建立信任界限。使用 IRSA，這是透過更新 EMR 任務執行角色的信任政策來實現的。不過，由於 IAM 信任政策長度有 4096 個字元的硬性限制，在最多十二 (12) 個 EKS 叢集之間共用單一任務執行 IAM 角色有其限制。

EMR 支援 Pod 身分，IAM 角色和服務帳戶之間的信任界限現在由 EKS 團隊透過 EKS Pod 身分的關聯 APIs 管理。

> **Note**
>
> EKS Pod 身分的安全界限仍在服務帳戶層級，而非 Pod 層級。

Pod 身分考量事項

如需 Pod 身分限制的資訊，請參閱 EKS Pod 身分考量事項。

在 EKS 叢集中準備 EKS Pod 身分

檢查 NodeInstanceRole 中是否存在所需的許可

節點角色`NodeInstanceRole`需要代理程式在 EKS 驗證 API 中執行`AssumeRoleForPodIdentity`動作的許可。您可以將以下內容新增至 Amazon EKS 使用者指南中定義的 AmazonEKSWorkerNodePolicy，或使用自訂政策。

如果您的 EKS 叢集建立的 eksctl 版本高於 0.181.0，AmazonEKSWorkerNodePolicy 會自動連接到節點角色，包括必要的`AssumeRoleForPodIdentity`許可。如果許可不存在，請手動將下列許可新增至 AmazonEKSWorkerNodePolicy，以允許 擔任 Pod 身分的角色。EKS Pod 身分代理程式需要此許可，才能擷取 Pod 的登入資料。

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "eks-auth:AssumeRoleForPodIdentity"
            ],
            "Resource": "*"
        }
    ]
}
```

建立 EKS Pod 身分代理程式附加元件

使用下列命令，以最新版本建立 EKS Pod Identity Agent 附加元件：

```
aws eks create-addon --cluster-name cluster-name --addon-name eks-pod-identity-agent

kubectl get pods -n kube-system | grep 'eks-pod-identity-agent'
```

使用下列步驟，從 Amazon EKS 主控台建立 EKS Pod Identity Agent 附加元件：

1. 開啟 Amazon EKS 主控台：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，在版本下拉式清單中選取任何版本。
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. 選擇下一步。
9. 確認代理程式 Pod 透過 CLI 在叢集上執行。

   ```
   kubectl get pods -n kube-system | grep 'eks-pod-identity-agent'
   ```

輸出範例如下：

```
NAME                              READY   STATUS    RESTARTS      AGE
eks-pod-identity-agent-gmqp7      1/1     Running   1 (24h ago)   24h
eks-pod-identity-agent-prnsh      1/1     Running   1 (24h ago)   24h
```

這會在`kube-system`命名空間中設定新的 DaemonSet。在每個 EKS 節點上執行的 Amazon EKS Pod Identity Agent 會使用 AssumeRoleForPodIdentity 動作，從 EKS 身分驗證 API 擷取臨時憑證。然後，這些登入資料可供您在容器內執行 AWS SDKs 使用。

如需詳細資訊，請參閱公有文件中的先決條件：設定 Amazon EKS Pod Identity Agent。

建立任務執行角色

建立或更新允許 EKS Pod Identity 的任務執行角色

若要使用 Amazon EMR on EKS 執行工作負載，您需要建立 IAM 角色。在本文件中，我們將此角色稱為作業執行角色。如需如何建立 IAM 角色的詳細資訊，請參閱《 使用者指南》中的建立 IAM 角色。

此外，您必須建立指定任務執行角色必要許可的 IAM 政策，然後將此政策連接至角色以啟用 EKS Pod Identity。

例如，您有下列任務執行角色。如需詳細資訊，請參閱建立任務執行角色。

```
arn:aws:iam::111122223333:role/PodIdentityJobExecutionRole
```

> **Important**
>
> Amazon EMR on EKS 會根據您的任務執行角色名稱自動建立 Kubernetes 服務帳戶。請確定角色名稱不會太長，因為如果 `cluster_name`、 和 的組合`service_account_name`超過長度限制`pod_name`，您的任務可能會失敗。

任務執行角色組態 – 確保使用下列 EKS Pod 身分的信任許可建立任務執行角色。若要更新現有的任務執行角色，請將其設定為信任下列 EKS 服務委託人，做為信任政策中的額外許可。此信任許可可以與現有的 IRSA 信任政策共存。

```
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
```

使用者許可：使用者需要執行 `StartJobRun` API 呼叫或提交任務的`iam:PassRole`許可。此許可可讓使用者將任務執行角色傳遞至 EMR on EKS。根據預設，任務管理員應該具有 許可。

以下是使用者所需的許可：

```
{
```

```
    "Effect": "Allow",
    "Action": "iam:PassRole",
    "Resource": "arn:aws:iam::111122223333:role/PodIdentityJobExecutionRole",
    "Condition": {
        "StringEquals": {
            "iam:PassedToService": "pods.eks.amazonaws.com"
        }
    }
}
```

若要進一步限制使用者存取特定 EKS 叢集，請將 AssociatedResourceArn 屬性篩選條件新增至 IAM 政策。它將角色假設限制為授權的 EKS 叢集，加強您的資源層級安全控制。

```
"Condition": {
        "ArnLike": {
            "iam:AssociatedResourceARN": [
                "arn:aws:eks:us-west-2:111122223333:cluster/*"
            ]
        }
```

設定 EKS Pod 身分關聯

先決條件

請確定建立 Pod 身分關聯的 IAM 身分，例如 EKS 管理員使用者，具有 許可`eks:CreatePodIdentityAssociation`和 `iam:PassRole`。

JSON

```
    {
        "Effect": "Allow",
        "Action": [
            "eks:CreatePodIdentityAssociation",
        ],
        "Resource": "* or role-arn"
    },
    {
        "Version": "2012-10-17",
        "Statement": [
            {
                "Effect": "Allow",
                "Action": "iam:PassRole",
```

```
                "Resource": "* or role-arn",
                "Condition": {
                    "StringEquals": {
                        "iam:PassedToService": "pods.eks.amazonaws.com"
                    }
                }
        }]
    }
```

建立角色和 EMR 服務帳戶的關聯

Create EMR role associations through the AWS CLI

當您將任務提交至 Kubernetes 命名空間時，管理員必須在任務執行角色與 EMR 受管服務帳戶的身分之間建立關聯。請注意，會在提交作業時自動建立 EMR 受管服務帳戶，範圍限定在提交作業的命名空間。

使用 AWS CLI（高於 2.24.0 版），執行下列命令來建立與 Pod 身分的角色關聯。

執行下列命令來建立與 Pod 身分的角色關聯：

```
aws emr-containers create-role-associations \
        --cluster-name mycluster \
        --namespace mynamespace \
        --role-name JobExecutionRoleIRSAv2
```

請注意：

- 每個叢集最多可以有 1，000 個關聯。每個任務執行角色 - 命名空間映射將需要任務提交者、驅動程式和執行器 Pod 的 3 個關聯。
- 您只能關聯與叢集位於相同 AWS 帳戶中的角色。您可以將存取權從另一個帳戶委派給此帳戶 (您為要使用的 EKS Pod 身分識別所設定) 中的角色。如需委派存取權和 的教學課程`AssumeRole`，請參閱 IAM 教學課程：使用 IAM 角色在 AWS 帳戶之間委派存取權。

Create EMR role associations through Amazon EKS

提交任務時，EMR 會建立具有特定命名模式的服務帳戶。若要建立手動關聯或將此工作流程與 AWS SDK 整合，請遵循下列步驟：

建構服務帳戶名稱：

```
emr-containers-sa-spark-%(SPARK_ROLE)s-%(AWS_ACCOUNT_ID)s-
%(BASE36_ENCODED_ROLE_NAME)s
```

以下範例會為範例任務執行角色 JobExecutionRoleIRSAv2 建立角色關聯。

角色關聯範例：

```
RoleName: JobExecutionRoleIRSAv2
Base36EncodingOfRoleName: 2eum5fah1jc1kwyjc19ikdhdkdegh1n26vbe
```

CLI 命令範例：

```
# setup for the client service account (used by job runner pod)
# emr-containers-sa-spark-client-111122223333-2eum5fah1jc1kwyjc19ikdhdkdegh1n26vbe
aws eks create-pod-identity-association --cluster-name mycluster
 --role-arn arn:aws:iam::111122223333:role/JobExecutionRoleIRSAv2
 --namespace mynamespace --service-account emr-containers-sa-spark-
client-111122223333-2eum5fah1jc1kwyjc19ikdhdkdegh1n26vbe

# driver service account
# emr-containers-sa-spark-driver-111122223333-2eum5fah1jc1kwyjc19ikdhdkdegh1n26vbe

aws eks create-pod-identity-association --cluster-name mycluster
 --role-arn arn:aws:iam::111122223333:role/JobExecutionRoleIRSAv2
 --namespace mynamespace --service-account emr-containers-sa-spark-
driver-111122223333-2eum5fah1jc1kwyjc19ikdhdkdegh1n26vbe

# executor service account
# emr-containers-sa-spark-executor-111122223333-2eum5fah1jc1kwyjc19ikdhdkdegh1n26vbe
aws eks create-pod-identity-association --cluster-name mycluster
 --role-arn arn:aws:iam::111122223333:role/JobExecutionRoleIRSAv2
 --namespace mynamespace --service-account emr-containers-sa-spark-
executor-111122223333-2eum5fah1jc1kwyjc19ikdhdkdegh1n26vbe
```

完成 EKS Pod 身分所需的所有步驟後，您可以略過下列步驟進行 IRSA 設定：

- 在 EKS 叢集上啟用服務帳戶的 IAM 角色 (IRSA)
- 建立任務執行角色
- 更新任務執行角色的信任政策

您可以直接跳至下列步驟：授予使用者對 Amazon EMR on EKS 的存取權

刪除角色關聯

每當您刪除虛擬叢集或任務執行角色，而且您不想再將 EMR 的存取權授予其服務帳戶時，您應該刪除該角色的關聯。這是因為 EKS 允許與不存在的資源（命名空間和服務帳戶）建立關聯。如果命名空間已刪除或角色不再使用，Amazon EMR on EKS 建議刪除關聯，以釋放其他關聯的空間。

> **Note**
>
> 如果您不刪除關聯，保留關聯可能會影響您擴展的能力，因為 EKS 對您可以建立的關聯數量有限制（軟性限制：每個叢集 1000 個關聯）。您可以在指定的命名空間中列出 Pod 身分關聯，以檢查是否有任何需要清除的保留關聯：

```
aws eks list-pod-identity-associations --cluster-name mycluster --namespace mynamespace
```

使用 AWS CLI (2.24.0 版或更新版本），執行下列 emr-containers 命令來刪除 EMR 的角色關聯：

```
aws emr-containers delete-role-associations \
        --cluster-name mycluster \
        --namespace mynamespace \
        --role-name JobExecutionRoleIRSAv2
```

自動將現有的 IRSA 遷移至 Pod 身分

您可以使用工具 eksctl 將服務帳戶 (IRSA) 的現有 IAM 角色遷移至 Pod 身分關聯：

```
eksctl utils migrate-to-pod-identity \
    --cluster mycluster \
    --remove-oidc-provider-trust-relationship \
    --approve
```

在沒有`--approve`旗標的情況下執行命令只會輸出反映遷移步驟的計劃，而且不會發生實際的遷移。

故障診斷

我的任務無法使用登入資料提供者的 NoClassDefinitionFound 或 ClassNotFound 例外狀況，或無法取得登入資料提供者。

EKS Pod 身分使用容器登入資料提供者來擷取必要的登入資料。如果您已指定自訂登入資料提供者，請確保其正常運作。或者，請確定您使用的是支援 EKS Pod Identity 的正確 AWS SDK 版本。如需詳細資訊，請參閱開始使用 Amazon EKS。

任務失敗，因為 eks-pod-identity-agent 日誌中顯示的「因 【x】 大小限制而無法擷取登入資料」錯誤。

EMR on EKS 會根據任務執行角色名稱建立 Kubernetes 服務帳戶。如果角色名稱太長，EKS 身分驗證將無法擷取憑證，因為 `cluster_name`、 `pod_name`和 的組合`service_account_name`超過長度限制。識別哪個元件佔用的空間最多，並相應地調整大小。

任務失敗，且 eks-pod-identity 日誌中顯示「無法擷取登入資料 xxx」錯誤。

此問題的一個可能原因可能是 EKS 叢集是在私有子網路下設定，但未正確設定叢集的 PrivateLink。檢查您的叢集是否位於私有網路中，並設定 AWS PrivateLink 以解決問題。如需詳細說明，請參閱開始使用 Amazon EKS。

選項 2：在 EKS 叢集上啟用服務帳戶的 IAM 角色 (IRSA)

服務帳戶的 IAM 角色功能適用於 Amazon EKS 1.14 版和更高版本，以及在 2019 年 9 月 3 日或之後更新至第 1.13 版或更高版本的 EKS 叢集。若要使用此功能，可將現有 EKS 叢集更新至 1.14 版或更新版本。如需詳細資訊，請參閱更新 Amazon EKS 叢集 Kubernetes 版本。

如果您的叢集支援服務帳戶的 IAM 角色，則會有相關聯的 OpenID Connect 發行者 URL。您可以在 Amazon EKS 主控台中檢視此 URL，也可以使用下列 AWS CLI 命令進行擷取。

> **Important**
>
> 您必須使用最新版本的 AWS CLI ，才能從此命令接收適當的輸出。

```
aws eks describe-cluster --name cluster_name --query "cluster.identity.oidc.issuer" --
output text
```

預期輸出如下。

```
https://oidc.eks.<region-code>.amazonaws.com/id/EXAMPLED539D4633E53DE1B716D3041E
```

若要在叢集中使用服務帳戶的 IAM 角色，則必須使用 eksctl 或 AWS Management Console 來建立 OIDC 身分提供者。

使用 **eksctl** 為您的叢集建立 IAM OIDC 身分提供者

使用以下命令檢查您的 `eksctl` 版本。此程序假設您已安裝 `eksctl`，且您的 `eksctl` 版本為 0.32.0 或更高版本。

```
eksctl version
```

如需有關安裝或升級 eksctl 的詳細資訊，請參閱安裝或升級 eksctl。

使用下列命令為您的叢集建立 OIDC 身分提供者。使用自己的值取代 *`cluster_name`*。

```
eksctl utils associate-iam-oidc-provider --cluster cluster_name --approve
```

使用 為您的叢集建立 IAM OIDC 身分提供者 AWS Management Console

從叢集的 Amazon EKS 主控台描述擷取 OIDC 發行者 URL，或使用下列 AWS CLI 命令。

使用下列命令，從 AWS CLI中擷取 OIDC 發行者 URL。

```
aws eks describe-cluster --name <cluster_name> --query "cluster.identity.oidc.issuer"
 --output text
```

使用下列命令，從 Amazon EKS 主控台中擷取 OIDC 發行者 URL。

1. 前往 https://console.aws.amazon.com/iam/ 開啟 IAM 主控台。
2. 在導覽面板中，選擇身分提供者，然後選擇建立提供者。
    1. 處理 Provider Type (提供者類型) 時，請選擇 Choose a provider type (選擇提供者類型)，然後選擇 OpenID Connect。
    2. 針對 Provider URL (提供者 URL)，貼上叢集的 OIDC 發行者 URL。
    3. 如果為「對象」，則輸入 sts.amazonaws.com，然後選擇下一步。
3. 確認供應商資訊是否正確，然後選擇 Create (建立) 來建立您的身分提供者。

建立作業執行角色

若要在 Amazon EMR on EKS 上執行工作負載，您需要建立 IAM 角色。在本文件中，我們將此角色稱為作業執行角色。如需有關如何建立 IAM 角色的詳細資訊，請參閱《IAM 使用者指南》中的建立 IAM 角色。

您還必須建立 IAM 政策來指定作業執行角色的許可，然後將 IAM 政策附接至作業執行角色。

作業執行角色的下列政策允許存取資源目標、Amazon S3 和 CloudWatch。這些許可是監控作業和存取日誌所必需的。若要使用 遵循相同的程序 AWS CLI，您也可以使用 Amazon EMR on EKS 研討會中建立任務執行的 IAM 角色一節中的步驟來設定角色。

> **Note**
>
> 應適當確定存取權限的範圍，而不是授予給作業執行角色中的所有 S3 物件。

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:PutObject",
                "s3:GetObject",
                "s3:ListBucket"
            ],
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket"
        },
        {
            "Effect": "Allow",
            "Action": [
                "logs:PutLogEvents",
                "logs:CreateLogStream",
                "logs:DescribeLogGroups",
                "logs:DescribeLogStreams"
            ],
            "Resource": [
                "arn:aws:logs:*:*:*"
```

```
            ]
        }
    ]
}
```

如需詳細資訊，請參閱使用作業執行角色、設定作業執行以使用 S3 日誌以及設定作業執行以使用 CloudWatch 日誌。

更新作業執行角色的信任政策

當您使用服務帳戶的 IAM 角色 (IRSA)，在 Kubernetes 命名空間上執行作業時，系統管理員必須在作業執行角色與 EMR 受管服務帳戶的身分之間建立信任關係。可以透過更新作業執行角色的信任政策來建立信任關係。請注意，會在提交作業時自動建立 EMR 受管服務帳戶，範圍限定在提交作業的命名空間。

執行下列命令來更新信任政策。

```
aws emr-containers update-role-trust-policy \
      --cluster-name cluster \
      --namespace namespace \
      --role-name iam_role_name_for_job_execution
```

如需詳細資訊，請參閱搭配使用作業執行角色與 Amazon EMR on EKS。

> **Important**
>
> 執行上述命令的運算子必須具有以下許可：`eks:DescribeCluster`、`iam:GetRole`、`iam:UpdateAssumeRolePolicy`。

## 授予使用者對 Amazon EMR on EKS 的存取權

對於您在 Amazon EMR on EKS 上執行的任何動作，您需要該動作的對應 IAM 許可。必須建立 IAM 政策，以便執行 Amazon EMR on EKS 動作，並將政策附接至您使用的 IAM 使用者或角色。

本主題提供建立新政策並將其附接至使用者的步驟。它還涵蓋了設定 Amazon EMR on EKS 環境所需的基本許可。建議您根據業務需求，盡可能完善特定資源的許可。

在 IAM 主控台中建立新的 IAM 政策並將其附接至使用者

建立新的 IAM 政策

1. 登入 AWS Management Console 並開啟位於 https：//https://console.aws.amazon.com/iam/ 的 IAM 主控台。
2. 在 IAM 主控台的左側導覽窗格中，選擇政策。
3. 在 Policies (政策) 頁面上，選擇 Create a policy (建立政策)。
4. 在建立政策視窗中，導覽至編輯 JSON 索引標籤。建立具有一個或多個 JSON 陳述式的政策文件，如此程序後面的範例所示。接下來，選擇檢閱政策。
5. 在 Review Policy (檢閱政策) 畫面上，輸入 Policy Name (政策名稱)，例如 `AmazonEMROnEKSPolicy`。輸入選用描述，然後選擇建立政策。

將政策附接至使用者或角色

1. 登入 AWS Management Console 並開啟位於 https：//https://console.aws.amazon.com/iam/ 的 IAM 主控台
2. 在導覽窗格中，選擇政策。
3. 在政策清單中，選取在上一節中建立的政策旁的核取方塊。您可用篩選功能表和搜尋方塊來篩選政策清單。
4. 選擇政策動作，再選擇附加。
5. 選擇要與政策附接的使用者或角色。您可用篩選功能表和搜尋方塊來篩選主體實體清單。選擇要與政策附接的使用者或角色後，選擇附接政策。

用於管理虛擬叢集的許可

若要管理 AWS 帳戶中的虛擬叢集，請使用下列許可建立 IAM 政策。這些許可可讓您在 AWS 帳戶中建立、列出、描述和刪除虛擬叢集。

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
```

```
            "Action": [
                "iam:CreateServiceLinkedRole"
            ],
            "Resource": "*",
            "Condition": {
                "StringLike": {
                    "iam:AWSServiceName": "emr-containers.amazonaws.com"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "emr-containers:CreateVirtualCluster",
                "emr-containers:ListVirtualClusters",
                "emr-containers:DescribeVirtualCluster",
                "emr-containers:DeleteVirtualCluster"
            ],
            "Resource": "*"
        }
    ]
}
```

Amazon EMR 已與 Amazon EKS 叢集存取管理 (CAM) 整合，因此您可以自動設定必要的 AuthN 和 AuthZ 政策，以在 Amazon EKS 叢集命名空間中執行 Amazon EMR Spark 任務。若要這樣做，您必須具有下列許可：

```
{
  "Effect": "Allow",
  "Action": [
    "eks:CreateAccessEntry"
  ],
  "Resource":
 "arn:<AWS_PARTITION>:eks:<AWS_REGION>:<AWS_ACCOUNT_ID>:cluster/<EKS_CLUSTER_NAME>"
},
{
  "Effect": "Allow",
  "Action": [
    "eks:DescribeAccessEntry",
    "eks:DeleteAccessEntry",
    "eks:ListAssociatedAccessPolicies",
    "eks:AssociateAccessPolicy",
```

```
    "eks:DisassociateAccessPolicy"
  ],
  "Resource": "arn:<AWS_PARTITION>:eks:<AWS_REGION>:<AWS_ACCOUNT_ID>:access-
entry/<EKS_CLUSTER_NAME>/role/<AWS_ACCOUNT_ID>/AWSServiceRoleForAmazonEMRContainers/*"
}
```

如需詳細資訊，請參閱自動啟用 Amazon EMR on EKS 的叢集存取。

第一次從 AWS 帳戶叫用 `CreateVirtualCluster`操作時，您也需要`CreateServiceLinkedRole`許可才能為 Amazon EMR on EKS 建立服務連結角色。如需詳細資訊，請參閱使用 Amazon EMR on EKS 的服務連結角色。

用於提交作業的許可

若要在 AWS 帳戶中的虛擬叢集上提交任務，請使用下列許可建立 IAM 政策。這些許可可讓您啟動、列出、描述和取消帳戶中所有虛擬叢集的作業執行。應該考慮新增許可以列出或描述虛擬叢集，這可讓您在提交作業之前檢查虛擬叢集的狀態。

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "emr-containers:StartJobRun",
                "emr-containers:ListJobRuns",
                "emr-containers:DescribeJobRun",
                "emr-containers:CancelJobRun"
            ],
            "Resource": "*"
        }
    ]
}
```

用於偵錯和監控的許可

若要存取推送到 Amazon S3 和 CloudWatch 的日誌，或在 Amazon EMR 主控台中檢視應用程式事件日誌，請建立具有下列許可的 IAM 政策。建議您根據業務需求，盡可能完善特定資源的許可。

> **Important**
>
> 如果尚未建立 Amazon S3 儲存貯體，則需要將 `s3:CreateBucket` 許可新增至政策陳述式。如果尚未建立日誌群組，則需要將 `logs:CreateLogGroup` 新增至政策陳述式。

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "emr-containers:DescribeJobRun",
                "elasticmapreduce:CreatePersistentAppUI",
                "elasticmapreduce:DescribePersistentAppUI",
                "elasticmapreduce:GetPersistentAppUIPresignedURL"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:ListBucket"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "logs:Get*",
                "logs:DescribeLogGroups",
                "logs:DescribeLogStreams"
            ],
            "Resource": "*"
        }
    ]
}
```

如需有關如何設定作業執行以將日誌推入 Amazon S3 和 CloudWatch 的詳細資訊，請參閱設定作業執行以使用 S3 日誌和設定作業執行以使用 CloudWatch 日誌。

## 向 Amazon EMR 註冊 Amazon EKS 叢集

註冊叢集是設定 Amazon EMR on EKS 以執行工作負載的最後一個必要步驟。

使用下列命令為您在先前步驟中設定的 Amazon EKS 叢集和命名空間建立具有您所選名稱的虛擬叢集。

> **Note**
>
> 每個虛擬叢集在所有 EKS 叢集中都必須具有唯一的名稱。如果兩個虛擬叢集具有相同名稱，即使兩個虛擬叢集屬於不同的 EKS 叢集，部署程序也會失敗。

```
aws emr-containers create-virtual-cluster \
--name virtual_cluster_name \
--container-provider '{
    "id": "cluster_name",
    "type": "EKS",
    "info": {
        "eksInfo": {
            "namespace": "namespace_name"
        }
    }
}'
```

或者，您可以建立包含虛擬叢集所需參數的 JSON 檔案，然後使用 JSON 檔案的路徑執行 `create-virtual-cluster` 命令。如需詳細資訊，請參閱管理虛擬叢集。

> **Note**
>
> 若要驗證虛擬叢集是否成功建立，請使用 `list-virtual-clusters` 操作或前往 Amazon EMR 主控台中的虛擬叢集頁面來檢視虛擬叢集的狀態。

# 使用 StartJobRun 提交作業執行

使用具有指定參數的 JSON 檔案來提交作業執行

1. 建立 `start-job-run-request.json` 檔案並指定作業執行所需的參數，如下列範例 JSON 檔案所示。如需這些參數的詳細資訊，請參閱 用於設定作業執行的選項。

```
{
  "name": "myjob",
  "virtualClusterId": "123456",
  "executionRoleArn": "iam_role_name_for_job_execution",
  "releaseLabel": "emr-6.2.0-latest",
  "jobDriver": {
    "sparkSubmitJobDriver": {
      "entryPoint": "entryPoint_location",
      "entryPointArguments": ["argument1", "argument2", ...],
       "sparkSubmitParameters": "--class <main_class> --conf
 spark.executor.instances=2 --conf spark.executor.memory=2G --conf
 spark.executor.cores=2 --conf spark.driver.cores=1"
    }
  },
  "configurationOverrides": {
    "applicationConfiguration": [
      {
        "classification": "spark-defaults",
        "properties": {
          "spark.driver.memory":"2G"
         }
      }
    ],
    "monitoringConfiguration": {
      "persistentAppUI": "ENABLED",
      "cloudWatchMonitoringConfiguration": {
        "logGroupName": "my_log_group",
        "logStreamNamePrefix": "log_stream_prefix"
      },
      "s3MonitoringConfiguration": {
        "logUri": "s3://my_s3_log_location"
      }
    }
  }
}
```

2. 搭配使用 `start-job-run` 命令與儲存在本機的 `start-job-run-request.json` 檔案路徑。

```
aws emr-containers start-job-run \
--cli-input-json file://./start-job-run-request.json
```

### 使用 **start-job-run** 命令啟動作業執行

1. 請在 `StartJobRun` 命令中提供所有指定的參數，如下列範例所示。

```
aws emr-containers start-job-run \
--virtual-cluster-id 123456 \
--name myjob \
--execution-role-arn execution-role-arn \
--release-label emr-6.2.0-latest \
--job-driver '{"sparkSubmitJobDriver": {"entryPoint": "entryPoint_location",
 "entryPointArguments": ["argument1", "argument2", ...], "sparkSubmitParameters":
 "--class <main_class> --conf spark.executor.instances=2 --conf
 spark.executor.memory=2G --conf spark.executor.cores=2 --conf
 spark.driver.cores=1"}}' \
--configuration-overrides '{"applicationConfiguration": [{"classification":
 "spark-defaults", "properties": {"spark.driver.memory": "2G"}}],
 "monitoringConfiguration": {"cloudWatchMonitoringConfiguration":
 {"logGroupName": "log_group_name", "logStreamNamePrefix": "log_stream_prefix"},
 "persistentAppUI":"ENABLED",  "s3MonitoringConfiguration": {"logUri":
 "s3://my_s3_log_location" }}}'
```

2. 對於 Spark SQL，請在 `StartJobRun` 命令中提供所有指定的參數，如下列範例所示。

```
aws emr-containers start-job-run \
--virtual-cluster-id 123456 \
--name myjob \
--execution-role-arn execution-role-arn \
--release-label emr-6.7.0-latest \
--job-driver '{"sparkSqlJobDriver": {"entryPoint": "entryPoint_location",
 "sparkSqlParameters": "--conf spark.executor.instances=2 --conf
 spark.executor.memory=2G --conf spark.executor.cores=2 --conf
 spark.driver.cores=1"}}' \
--configuration-overrides '{"applicationConfiguration": [{"classification":
 "spark-defaults", "properties": {"spark.driver.memory": "2G"}}],
 "monitoringConfiguration": {"cloudWatchMonitoringConfiguration":
 {"logGroupName": "log_group_name", "logStreamNamePrefix": "log_stream_prefix"},
```

```
"persistentAppUI":"ENABLED",  "s3MonitoringConfiguration": {"logUri":
"s3://my_s3_log_location" }}}'
```

# 使用作業提交器分類

## 概觀

Amazon EMR on EKS `StartJobRun` 請求會建立作業提交器 Pod (也稱為 job-runner Pod) 以產生 Spark 驅動程式。您可以使用`emr-job-submitter`分類來設定任務提交器 Pod 的節點選擇器，以及設定任務提交器 Pod 記錄容器的影像、CPU 和記憶體。

`emr-job-submitter` 分類下提供下列設定：

**`jobsubmitter.node.selector.[labelKey]`**

新增至作業提交器 Pod 的節點選取器，使用索引鍵 *`labelKey`* 和值作為組態的值。例如，可將 `jobsubmitter.node.selector.identifier` 設定為 `myIdentifier`，且作業提交器 Pod 將擁有一個節點選取器，它具有 `myIdentifier` 的索引鍵識別符值。這可用來指定任務提交器 Pod 可以放置在哪個節點上。要新增多個節點選取器索引鍵，請使用此字首設定多個組態。

**`jobsubmitter.logging.image`**

設定要用於任務提交器 Pod 上記錄容器的自訂映像。

**`jobsubmitter.logging.request.cores`**

設定任務提交器 Pod 上記錄容器CPUs 數量的自訂值，以 CPU 單位為單位。根據預設，此值會設為 100 公尺。

**`jobsubmitter.logging.request.memory`**

為任務提交器 Pod 上的記錄容器設定記憶體量的自訂值，以位元組為單位。根據預設，此值會設為 200Mi。MB 是類似於 MB 的度量單位。

我們建議將任務提交者 Pod 放置在隨需執行個體上。如果任務提交器 Pod 執行的執行個體發生 Spot 執行個體中斷，則將任務提交器 Pod 放置在 Spot 執行個體上可能會導致任務失敗。也可以將作業提交器 Pod 置於單一可用區域中，或使用套用至節點的任何 Kubernetes 標籤。

## 作業提交器分類範例

本節內容

- 適用於作業提交器 Pod 的具有隨需節點放置的 StartJobRun 請求

- 適用於作業提交器 Pod 的具有單一可用區域節點放置的 StartJobRun 請求
- 適用於作業提交器 Pod 的具有單一可用區域和 Amazon EC2 執行個體類型放置的 StartJobRun 請求
- StartJobRun 具有自訂記錄容器映像、CPU 和記憶體的請求

### 適用於作業提交器 Pod 的具有隨需節點放置的 **StartJobRun** 請求

```
cat >spark-python-in-s3-nodeselector-job-submitter.json << EOF
{
  "name": "spark-python-in-s3-nodeselector",
  "virtualClusterId": "virtual-cluster-id",
  "executionRoleArn": "execution-role-arn",
  "releaseLabel": "emr-6.11.0-latest",
  "jobDriver": {
    "sparkSubmitJobDriver": {
      "entryPoint": "s3://S3-prefix/trip-count.py",
       "sparkSubmitParameters": "--conf spark.driver.cores=5  --conf
 spark.executor.memory=20G --conf spark.driver.memory=15G --conf
 spark.executor.cores=6"
    }
  },
  "configurationOverrides": {
    "applicationConfiguration": [
      {
        "classification": "spark-defaults",
        "properties": {
          "spark.dynamicAllocation.enabled":"false"
         }
      },
      {
        "classification": "emr-job-submitter",
        "properties": {
            "jobsubmitter.node.selector.eks.amazonaws.com/capacityType": "ON_DEMAND"
        }
      }
    ],
    "monitoringConfiguration": {
      "cloudWatchMonitoringConfiguration": {
        "logGroupName": "/emr-containers/jobs",
        "logStreamNamePrefix": "demo"
      },
      "s3MonitoringConfiguration": {
        "logUri": "s3://joblogs"
```

```
      }
    }
  }
}
EOF
aws emr-containers start-job-run --cli-input-json file:///spark-python-in-s3-
nodeselector-job-submitter.json
```

適用於作業提交器 Pod 的具有單一可用區域節點放置的 **StartJobRun** 請求

```
cat >spark-python-in-s3-nodeselector-job-submitter-az.json << EOF
{
  "name": "spark-python-in-s3-nodeselector",
  "virtualClusterId": "virtual-cluster-id",
  "executionRoleArn": "execution-role-arn",
  "releaseLabel": "emr-6.11.0-latest",
  "jobDriver": {
    "sparkSubmitJobDriver": {
      "entryPoint": "s3:////S3-prefix/trip-count.py",
       "sparkSubmitParameters": "--conf spark.driver.cores=5  --conf
 spark.executor.memory=20G --conf spark.driver.memory=15G --conf
 spark.executor.cores=6"
    }
  },
  "configurationOverrides": {
    "applicationConfiguration": [
      {
        "classification": "spark-defaults",
        "properties": {
          "spark.dynamicAllocation.enabled":"false"
         }
      },
      {
        "classification": "emr-job-submitter",
        "properties": {
            "jobsubmitter.node.selector.topology.kubernetes.io/zone": "Availability
 Zone"
        }
      }
    ],
    "monitoringConfiguration": {
      "cloudWatchMonitoringConfiguration": {
        "logGroupName": "/emr-containers/jobs",
```

```
        "logStreamNamePrefix": "demo"
      },
      "s3MonitoringConfiguration": {
        "logUri": "s3://joblogs"
      }
    }
  }
}
EOF
aws emr-containers start-job-run --cli-input-json file:///spark-python-in-s3-
nodeselector-job-submitter-az.json
```

適用於作業提交器 Pod 的具有單一可用區域和 Amazon EC2 執行個體類型放置的 **StartJobRun** 請求

```
{
  "name": "spark-python-in-s3-nodeselector",
  "virtualClusterId": "virtual-cluster-id",
  "executionRoleArn": "execution-role-arn",
  "releaseLabel": "emr-6.11.0-latest",
  "jobDriver": {
    "sparkSubmitJobDriver": {
      "entryPoint": "s3:////S3-prefix/trip-count.py",
       "sparkSubmitParameters": "--conf spark.driver.cores=5  --conf
 spark.kubernetes.pyspark.pythonVersion=3 --conf spark.executor.memory=20G
 --conf spark.driver.memory=15G --conf spark.executor.cores=6 --conf
 spark.sql.shuffle.partitions=1000"
    }
  },
  "configurationOverrides": {
    "applicationConfiguration": [
      {
        "classification": "spark-defaults",
        "properties": {
          "spark.dynamicAllocation.enabled":"false",
         }
      },
      {
        "classification": "emr-job-submitter",
        "properties": {
            "jobsubmitter.node.selector.topology.kubernetes.io/zone": "Availability
 Zone",
            "jobsubmitter.node.selector.node.kubernetes.io/instance-type":"m5.4xlarge"
        }
```

```
      }
    ],
    "monitoringConfiguration": {
      "cloudWatchMonitoringConfiguration": {
        "logGroupName": "/emr-containers/jobs",
        "logStreamNamePrefix": "demo"
      },
      "s3MonitoringConfiguration": {
        "logUri": "s3://joblogs"
      }
    }
  }
}
```

### StartJobRun 具有自訂記錄容器映像、CPU 和記憶體的請求

```
{
  "name": "spark-python",
  "virtualClusterId": "virtual-cluster-id",
  "executionRoleArn": "execution-role-arn",
  "releaseLabel": "emr-6.11.0-latest",
  "jobDriver": {
    "sparkSubmitJobDriver": {
      "entryPoint": "s3://S3-prefix/trip-count.py"
    }
  },
  "configurationOverrides": {
    "applicationConfiguration": [
      {
        "classification": "emr-job-submitter",
        "properties": {
            "jobsubmitter.logging.image": "YOUR_ECR_IMAGE_URL",
            "jobsubmitter.logging.request.memory": "200Mi",
            "jobsubmitter.logging.request.cores": "0.5"
        }
      }
    ],
    "monitoringConfiguration": {
      "cloudWatchMonitoringConfiguration": {
        "logGroupName": "/emr-containers/jobs",
        "logStreamNamePrefix": "demo"
      },
      "s3MonitoringConfiguration": {
```

```
        "logUri": "s3://joblogs"
      }
    }
  }
}
```

# 使用 Amazon EMR 容器預設分類

## 概觀

`emr-containers-defaults` 分類下提供下列設定：

**`job-start-timeout`**

根據預設，如果任務無法啟動，而且在 `SUBMITTED` 狀態中等待 15 分鐘，則會逾時。此組態會變更任務逾時前的等待秒數。

**`logging.image`**

設定要用於驅動程式和執行器 Pod 上記錄容器的自訂映像。

**`logging.request.cores`**

設定驅動程式和執行器 Pod 上記錄容器CPUs 數量的自訂值，以 CPU 單位為單位。根據預設，不會設定此項目。

**`logging.request.memory`**

設定驅動程式和執行器 Pod 上記錄容器的記憶體量自訂值，以位元組為單位。根據預設，此值會設為 512Mi。MB 是類似於 MB 的度量單位。

## 作業提交器分類範例

本節內容

- StartJobRun 具有自訂任務逾時的請求
- StartJobRun 具有自訂記錄容器映像、CPU 和記憶體的請求

### **`StartJobRun`** 具有自訂任務逾時的請求

```
{
  "name": "spark-python",
```

```
  "virtualClusterId": "virtual-cluster-id",
  "executionRoleArn": "execution-role-arn",
  "releaseLabel": "emr-6.11.0-latest",
  "jobDriver": {
    "sparkSubmitJobDriver": {
      "entryPoint": "s3://S3-prefix/trip-count.py"
    }
  },
  "configurationOverrides": {
    "applicationConfiguration": [
      {
        "classification": "emr-containers-defaults",
        "properties": {
            "job-start-timeout": "1800"
        }
      }
    ],
    "monitoringConfiguration": {
      "cloudWatchMonitoringConfiguration": {
        "logGroupName": "/emr-containers/jobs",
        "logStreamNamePrefix": "demo"
      },
      "s3MonitoringConfiguration": {
        "logUri": "s3://joblogs"
      }
    }
  }
}
```

### StartJobRun 具有自訂記錄容器映像、CPU 和記憶體的請求

```
{
  "name": "spark-python",
  "virtualClusterId": "virtual-cluster-id",
  "executionRoleArn": "execution-role-arn",
  "releaseLabel": "emr-6.11.0-latest",
  "jobDriver": {
    "sparkSubmitJobDriver": {
      "entryPoint": "s3://S3-prefix/trip-count.py"
    }
  },
  "configurationOverrides": {
    "applicationConfiguration": [
```

```
      {
        "classification": "emr-containers-defaults",
        "properties": {
            "logging.image": "YOUR_ECR_IMAGE_URL",
            "logging.request.memory": "200Mi",
            "logging.request.cores": "0.5"
        }
      }
    ],
    "monitoringConfiguration": {
      "cloudWatchMonitoringConfiguration": {
        "logGroupName": "/emr-containers/jobs",
        "logStreamNamePrefix": "demo"
      },
      "s3MonitoringConfiguration": {
        "logUri": "s3://joblogs"
      }
    }
  }
}
```

# 使用 Spark Operator 執行 Spark 作業

Amazon EMR 6.10.0 版及更高版本支援 Kubernetes Operator for Apache Spark 或 Spark Operator，作為 Amazon EMR on EKS 的作業提交模型。透過 Spark Operator，可以使用 Amazon EMR 發行執行期在您自己的 Amazon EKS 叢集上部署和管理 Spark 應用程式。在 Amazon EKS 叢集中部署 Spark Operator 之後，即可直接向 Operator 提交 Spark 應用程式。Operator 可管理 Spark 應用程式的生命週期。

> **Note**
>
> Amazon EMR 會根據 vCPU 和記憶體使用量計算 Amazon EKS 定價。此計算適用於驅動程式和執行器 Pod。此計算從您下載 Amazon EMR 應用程式映像時開始，直到 Amazon EKS Pod 終止且四捨五入至最接近的秒。

主題


- 為 Amazon EMR on EKS 設定 Spark Operator
- 開始針對 Amazon EMR on EKS 使用 Spark Operator

- 搭配適用於 Amazon EMR on EKS 的 Spark Operator 使用垂直自動擴展
- 為 Amazon EMR on EKS 解除安裝 Spark Operator
- 使用監控組態來監控 Spark Kubernetes 運算子和 Spark 任務
- Amazon EMR on EKS 的安全性和 Spark Operator

## 為 Amazon EMR on EKS 設定 Spark Operator

先完成下列任務，然後在 Amazon EKS 上安裝 Spark Operator。如果已經註冊 Amazon Web Services (AWS) 且已在使用 Amazon EKS，則您幾乎可使用 Amazon EMR on EKS。完成下列任務，即可在 Amazon EKS 上設定 Spark Operator。如果已經完成任何先決條件，則可以跳過這些先決條件，然後繼續進行下一個。

- 安裝或更新至最新版本的 AWS CLI- 如果您已安裝 AWS CLI，請確認您擁有最新版本。
- 設定 kubectl 和 eksctl – eksctl 是您用來與 Amazon EKS 通訊的命令列工具。
- 安裝 Helm – 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。
- 開始使用 Amazon EKS – eksctl – 請依照步驟在 Amazon EKS 中建立具有節點的新 Kubernetes 叢集。
- 選擇 Amazon EMR 基礎映像 URI (6.10.0 版或更高版本) - Amazon EMR 6.10.0 版及更高版本支援 Spark Operator。

## 開始針對 Amazon EMR on EKS 使用 Spark Operator

本主題協助您透過部署 Spark 應用程式和 Schedule Spark 應用程式，開始在 Amazon EKS 上使用 Spark Operator。

### 安裝 Spark Operator

請使用下列步驟來安裝 Kubernetes Operator for Apache Spark。

1. 如果您尚未這麼做，請完成 為 Amazon EMR on EKS 設定 Spark Operator 中的步驟。
2. 向 Amazon ECR 登錄檔驗證 Helm 用戶端。在下列命令中，將 *region-id* 值取代為您偏好的 AWS 區域，以及 按區域劃分的 Amazon ECR 登錄檔帳戶 頁面中區域的對應 *ECR-registry-account* 值。

```
aws ecr get-login-password \
```

```
--region region-id | helm registry login \
--username AWS \
--password-stdin ECR-registry-account.dkr.ecr.region-id.amazonaws.com
```

3. 使用以下命令安裝 Spark Operator。

   對於 Helm Chart `--version` 參數，請使用移除了 `emr-` 字首和日期尾碼的 Amazon EMR 版本標籤。例如，對於 `emr-6.12.0-java17-latest` 發行版本，請指定 `6.12.0-java17`。下列命令中的範例使用 `emr-7.8.0-latest` 版本，因此它會為 Helm Chart `--version` 指定 `7.8.0`。

```
helm install spark-operator-demo \
  oci://895885662937.dkr.ecr.region-id.amazonaws.com/spark-operator \
  --set emrContainers.awsRegion=region-id \
  --version 7.8.0 \
  --namespace spark-operator \
  --create-namespace
```

   根據預設，命令會為 Spark Operator 建立服務帳戶 `emr-containers-sa-spark-operator`。若要使用不同的服務帳戶，請提供引數 `serviceAccounts.sparkoperator.name`。例如：

```
--set serviceAccounts.sparkoperator.name my-service-account-for-spark-operator
```

   如果想要搭配使用垂直自動擴展與 Spark Operator，請在安裝命令中新增以下命令列，以允許 Operator 使用 Webhook：

```
--set webhook.enable=true
```

4. 請確認已使用 `helm list` 命令安裝 Helm Chart：

```
helm list --namespace spark-operator -o yaml
```

   `helm list` 命令應傳回新部署的 Helm Chart 版本資訊：

```
app_version: v1beta2-1.3.8-3.1.1
chart: spark-operator-7.8.0
name: spark-operator-demo
namespace: spark-operator
revision: "1"
status: deployed
updated: 2023-03-14 18:20:02.721638196 +0000 UTC
```

5. 使用您需要的任何其他選項完成安裝。如需詳細資訊，請參閱 GitHub 上的 spark-on-k8s-operator 文件。

## 執行 Spark 應用程式

Amazon EMR 6.10.0 或更高版本支援 Spark Operator。當您安裝 Spark Operator 時，它會預設建立服務帳戶 `emr-containers-sa-spark` 以執行 Spark 應用程式。使用以下步驟在 Amazon EMR on EKS 6.10.0 或更高版本上透過 Spark Operator 執行 Spark 應用程式。

1. 在使用 Spark Operator 執行 Spark 應用程式之前，請先完成 為 Amazon EMR on EKS 設定 Spark Operator 和 安裝 Spark Operator 中的步驟。
2. 使用以下範例內容，建立 `SparkApplication` 定義檔案 `spark-pi.yaml`：

```
apiVersion: "sparkoperator.k8s.io/v1beta2"
kind: SparkApplication
metadata:
  name: spark-pi
  namespace: spark-operator
spec:
  type: Scala
  mode: cluster
  image: "895885662937.dkr.ecr.us-west-2.amazonaws.com/spark/emr-6.10.0:latest"
  imagePullPolicy: Always
  mainClass: org.apache.spark.examples.SparkPi
  mainApplicationFile: "local:///usr/lib/spark/examples/jars/spark-examples.jar"
  sparkVersion: "3.3.1"
  restartPolicy:
    type: Never
  volumes:
    - name: "test-volume"
      hostPath:
        path: "/tmp"
        type: Directory
  driver:
    cores: 1
    coreLimit: "1200m"
    memory: "512m"
    labels:
      version: 3.3.1
    serviceAccount: emr-containers-sa-spark
    volumeMounts:
```

```
      - name: "test-volume"
        mountPath: "/tmp"
  executor:
    cores: 1
    instances: 1
    memory: "512m"
    labels:
      version: 3.3.1
    volumeMounts:
      - name: "test-volume"
        mountPath: "/tmp"
```

3. 現在，使用下列命令提交 Spark 應用程式。這也將建立名為 `spark-pi` 的 `SparkApplication` 物件：

```
kubectl apply -f spark-pi.yaml
```

4. 使用下列命令檢查 `SparkApplication` 物件的事件：

```
kubectl describe sparkapplication spark-pi --namespace spark-operator
```

如需有關透過 Spark Operator 將應用程式提交至 Spark 的詳細資訊，請參閱 GitHub 上 `spark-on-k8s-operator` 文件中的使用 `SparkApplication`。

## 使用 Amazon S3 進行儲存

若要使用 Amazon S3 做為檔案儲存選項，請將下列組態新增至 YAML 檔案。

```
hadoopConf:
# EMRFS filesystem
  fs.s3.customAWSCredentialsProvider:
 com.amazonaws.auth.WebIdentityTokenCredentialsProvider
  fs.s3.impl: com.amazon.ws.emr.hadoop.fs.EmrFileSystem
  fs.AbstractFileSystem.s3.impl: org.apache.hadoop.fs.s3.EMRFSDelegate
  fs.s3.buffer.dir: /mnt/s3
  fs.s3.getObject.initialSocketTimeoutMilliseconds: "2000"
  mapreduce.fileoutputcommitter.algorithm.version.emr_internal_use_only.EmrFileSystem:
 "2"
  mapreduce.fileoutputcommitter.cleanup-
failures.ignored.emr_internal_use_only.EmrFileSystem: "true"
sparkConf:
 # Required for EMR Runtime
```

```
 spark.driver.extraClassPath: /usr/lib/hadoop-lzo/lib/*:/usr/lib/hadoop/hadoop-
aws.jar:/usr/share/aws/aws-java-sdk/*:/usr/share/aws/emr/emrfs/conf:/usr/share/aws/
emr/emrfs/lib/*:/usr/share/aws/emr/emrfs/auxlib/*:/usr/share/aws/emr/security/conf:/
usr/share/aws/emr/security/lib/*:/usr/share/aws/hmclient/lib/aws-glue-datacatalog-
spark-client.jar:/usr/share/java/Hive-JSON-Serde/hive-openx-serde.jar:/usr/share/aws/
sagemaker-spark-sdk/lib/sagemaker-spark-sdk.jar:/home/hadoop/extrajars/*
 spark.driver.extraLibraryPath: /usr/lib/hadoop/lib/native:/usr/lib/hadoop-lzo/lib/
native:/docker/usr/lib/hadoop/lib/native:/docker/usr/lib/hadoop-lzo/lib/native
 spark.executor.extraClassPath: /usr/lib/hadoop-lzo/lib/*:/usr/lib/hadoop/hadoop-
aws.jar:/usr/share/aws/aws-java-sdk/*:/usr/share/aws/emr/emrfs/conf:/usr/share/aws/
emr/emrfs/lib/*:/usr/share/aws/emr/emrfs/auxlib/*:/usr/share/aws/emr/security/conf:/
usr/share/aws/emr/security/lib/*:/usr/share/aws/hmclient/lib/aws-glue-datacatalog-
spark-client.jar:/usr/share/java/Hive-JSON-Serde/hive-openx-serde.jar:/usr/share/aws/
sagemaker-spark-sdk/lib/sagemaker-spark-sdk.jar:/home/hadoop/extrajars/*
 spark.executor.extraLibraryPath: /usr/lib/hadoop/lib/native:/usr/lib/hadoop-lzo/lib/
native:/docker/usr/lib/hadoop/lib/native:/docker/usr/lib/hadoop-lzo/lib/native
```

如果您使用 Amazon EMR 7.2.0 版和更新版本，預設會包含組態。在這種情況下，您可以將檔案路徑設定為 ，`s3://<bucket_name>/<file_path>`而不是`local://<file_path>`在 Spark 應用程式 YAML 檔案中。

然後正常提交 Spark 應用程式。

## 搭配適用於 Amazon EMR on EKS 的 Spark Operator 使用垂直自動擴展

從 Amazon EMR 7.0 開始，您可以使用 Amazon EMR on EKS 垂直自動擴展來簡化資源管理。它會自動調整記憶體和 CPU 資源，以適應您為 Amazon EMR Spark 應用程式提供的工作負載需求。如需詳細資訊，請參閱搭配使用垂直自動擴展與 Amazon EMR Spark 作業。

本章節描述了如何設定 Spark Operator 以使用垂直自動擴展。

### 先決條件

設定監控之前，請務必完成下列設定任務：

- 完成「為 Amazon EMR on EKS 設定 Spark Operator」中的步驟。
- （選用） 如果您先前已安裝較舊版本的 Spark 運算子，請刪除 SparkApplication/ScheduledSparkApplication CRD。

```
kubectl delete crd sparkApplication
kubectl delete crd scheduledSparkApplication
```

- 完成「安裝 Spark Operator」中的步驟。在步驟 3 中，將以下命令列新增至安裝命令，以允許該 Operator 使用 Webhook：

```
--set webhook.enable=true
```

- 完成「設定 Amazon EMR on EKS 的垂直自動擴展」中的步驟。
- 允許存取 Amazon S3 位置中的檔案：

1. 使用具有 S3 許可`JobExecutionRole`的 標註您的驅動程式和運算子服務帳戶。

```
kubectl annotate serviceaccount -n spark-operator emr-containers-sa-spark
 eks.amazonaws.com/role-arn=JobExecutionRole
kubectl annotate serviceaccount -n spark-operator emr-containers-sa-spark-
operator eks.amazonaws.com/role-arn=JobExecutionRole
```

2. 更新該命名空間中任務執行角色的信任政策。

```
aws emr-containers update-role-trust-policy \
--cluster-name cluster \
--namespace ${Namespace}\
--role-name iam_role_name_for_job_execution
```

3. 編輯任務執行角色的 IAM 角色信任政策，並將 `serviceaccount` 從 更新`emr-containers-sa-spark-*-*-xxxx`為 `emr-containers-sa-*`。

```
{
    "Effect": "Allow",
    "Principal": {
        "Federated": "OIDC-provider"
    },
    "Action": "sts:AssumeRoleWithWebIdentity",
    "Condition": {
        "StringLike": {
            "OIDC": "system:serviceaccount:${Namespace}:emr-containers-sa-*"
        }
    }
}
```

4. 如果您使用 Amazon S3 做為檔案儲存體，請將下列預設值新增至 yaml 檔案。

```
hadoopConf:
# EMRFS filesystem
```

```
  fs.s3.customAWSCredentialsProvider:
 com.amazonaws.auth.WebIdentityTokenCredentialsProvider
  fs.s3.impl: com.amazon.ws.emr.hadoop.fs.EmrFileSystem
  fs.AbstractFileSystem.s3.impl: org.apache.hadoop.fs.s3.EMRFSDelegate
  fs.s3.buffer.dir: /mnt/s3
  fs.s3.getObject.initialSocketTimeoutMilliseconds: "2000"

 mapreduce.fileoutputcommitter.algorithm.version.emr_internal_use_only.EmrFileSystem:
 "2"
  mapreduce.fileoutputcommitter.cleanup-
failures.ignored.emr_internal_use_only.EmrFileSystem: "true"
sparkConf:
 # Required for EMR Runtime
 spark.driver.extraClassPath: /usr/lib/hadoop-lzo/lib/*:/usr/lib/hadoop/hadoop-
aws.jar:/usr/share/aws/aws-java-sdk/*:/usr/share/aws/emr/emrfs/conf:/usr/share/
aws/emr/emrfs/lib/*:/usr/share/aws/emr/emrfs/auxlib/*:/usr/share/aws/emr/
security/conf:/usr/share/aws/emr/security/lib/*:/usr/share/aws/hmclient/lib/aws-
glue-datacatalog-spark-client.jar:/usr/share/java/Hive-JSON-Serde/hive-openx-
serde.jar:/usr/share/aws/sagemaker-spark-sdk/lib/sagemaker-spark-sdk.jar:/home/
hadoop/extrajars/*
 spark.driver.extraLibraryPath: /usr/lib/hadoop/lib/native:/usr/lib/hadoop-lzo/
lib/native:/docker/usr/lib/hadoop/lib/native:/docker/usr/lib/hadoop-lzo/lib/
native
 spark.executor.extraClassPath: /usr/lib/hadoop-lzo/lib/*:/usr/lib/hadoop/
hadoop-aws.jar:/usr/share/aws/aws-java-sdk/*:/usr/share/aws/emr/emrfs/conf:/usr/
share/aws/emr/emrfs/lib/*:/usr/share/aws/emr/emrfs/auxlib/*:/usr/share/aws/emr/
security/conf:/usr/share/aws/emr/security/lib/*:/usr/share/aws/hmclient/lib/aws-
glue-datacatalog-spark-client.jar:/usr/share/java/Hive-JSON-Serde/hive-openx-
serde.jar:/usr/share/aws/sagemaker-spark-sdk/lib/sagemaker-spark-sdk.jar:/home/
hadoop/extrajars/*
 spark.executor.extraLibraryPath: /usr/lib/hadoop/lib/native:/usr/lib/hadoop-
lzo/lib/native:/docker/usr/lib/hadoop/lib/native:/docker/usr/lib/hadoop-lzo/lib/
native
```

## 在 Spark Operator 上使用垂直自動擴展來執行作業

在使用 Spark Operator 執行 Spark 應用程式之前，必須先完成 先決條件 中的步驟。

若要搭配 Spark 運算子使用垂直自動調整規模，請將下列組態新增至 Spark 應用程式規格的驅動程式，以開啟垂直自動調整規模：

```
dynamicSizing:
```

```
  mode: Off
  signature: "my-signature"
```

此組態會啟用垂直自動擴展，並且是必要的簽章組態，可讓您為任務選擇簽章。

如需組態和參數值的詳細資訊，請參閱設定 Amazon EMR on EKS 的垂直自動調整規模。依預設，您的作業在垂直自動擴展的僅監控關閉模式下提交。此監控狀態可讓您計算和檢視資源建議，而無需執行自動擴展。如需詳細資訊，請參閱垂直自動調整規模模式。

以下是名為 的範例`SparkApplication`定義檔案，`spark-pi.yaml`其中包含使用垂直自動擴展所需的組態。

```
apiVersion: "sparkoperator.k8s.io/v1beta2"
kind: SparkApplication
metadata:
  name: spark-pi
  namespace: spark-operator
spec:
  type: Scala
  mode: cluster
  image: "895885662937.dkr.ecr.us-west-2.amazonaws.com/spark/emr-7.8.0:latest"
  imagePullPolicy: Always
  mainClass: org.apache.spark.examples.SparkPi
  mainApplicationFile: "local:///usr/lib/spark/examples/jars/spark-examples.jar"
  sparkVersion: "3.4.1"
  dynamicSizing:
    mode: Off
    signature: "my-signature"
  restartPolicy:
    type: Never
  volumes:
    - name: "test-volume"
      hostPath:
        path: "/tmp"
        type: Directory
  driver:
    cores: 1
    coreLimit: "1200m"
    memory: "512m"
    labels:
      version: 3.4.1
    serviceAccount: emr-containers-sa-spark
    volumeMounts:
```

```
      - name: "test-volume"
        mountPath: "/tmp"
  executor:
    cores: 1
    instances: 1
    memory: "512m"
    labels:
      version: 3.4.1
    volumeMounts:
      - name: "test-volume"
        mountPath: "/tmp"
```

現在，使用下列命令提交 Spark 應用程式。這也將建立名為 `spark-pi` 的 `SparkApplication` 物件：

```
kubectl apply -f spark-pi.yaml
```

如需有關透過 Spark Operator 將應用程式提交至 Spark 的詳細資訊，請參閱 GitHub 上 `spark-on-k8s-operator` 文件中的使用 `SparkApplication`。

## 驗證垂直自動擴展功能

若要確認垂直自動擴展適用於已提交的作業，請使用 kubectl 取得 `verticalpodautoscaler` 自訂資源並檢視您的擴展建議。

```
kubectl get verticalpodautoscalers --all-namespaces \
-l=emr-containers.amazonaws.com/dynamic.sizing.signature=my-signature
```

此查詢的輸出應如下所示：

```
NAMESPACE        NAME                                                        MODE
 CPU   MEM         PROVIDED   AGE
spark-operator   ds-p73j6mkosvc4xeb3gr7x4xol2bfcw5evqimzqojrlysvj3giozuq-vpa   Off
     580026651   True       15m
```

如果您的輸出看起來不相似或包含錯誤碼，請參閱 對 Amazon EMR on EKS 垂直自動擴展進行疑難排解 以取得協助解決問題的步驟。

若要移除 Pod 和應用程式，請執行下列命令：

```
kubectl delete sparkapplication spark-pi
```

# 為 Amazon EMR on EKS 解除安裝 Spark Operator

使用下列步驟解除安裝 Spark Operator。

1. 使用正確的命名空間刪除 Spark Operator。針對此範例，命名空間為 `spark-operator-demo`。

```
helm uninstall spark-operator-demo -n spark-operator
```

2. 刪除 Spark Operator 服務帳戶：

```
kubectl delete sa emr-containers-sa-spark-operator -n spark-operator
```

3. 刪除 Spark Operator `CustomResourceDefinitions` (CRD)：

```
kubectl delete crd sparkapplications.sparkoperator.k8s.io
kubectl delete crd scheduledsparkapplications.sparkoperator.k8s.io
```

# 使用監控組態來監控 Spark Kubernetes 運算子和 Spark 任務

監控組態可讓您輕鬆地將 Spark 應用程式和運算子日誌的日誌封存設定到 Amazon S3 或 Amazon CloudWatch。您可以選擇其中一個或兩者。這樣做會將日誌代理程式附屬項目新增至您的 Spark Operator Pod、驅動程式和執行器 Pod，然後將這些元件的日誌轉送到您設定的接收器。

## 先決條件

設定監控之前，請務必完成下列設定任務：

1. （選用）如果您先前已安裝較舊版本的 Spark 運算子，請刪除 SparkApplication/ScheduledSparkApplication CRD。

```
kubectl delete crd scheduledsparkapplications.sparkoperator.k8s.io
kubectl delete crd sparkapplications.sparkoperator.k8s.io
```

2. 如果您還沒有運算子/任務執行角色，請在 IAM 中建立該角色。
3. 執行下列命令來更新您剛建立之運算子/任務執行角色的信任政策：

```
aws emr-containers update-role-trust-policy \
--cluster-name cluster \
--namespace namespace \
```

```
--role-name iam_role_name_for_operator/job_execution_role
```

4. 將運算子/任務執行角色的 IAM 角色信任政策編輯如下：

```
{
    "Effect": "Allow",
    "Principal": {
        "Federated": "${OIDC-provider}"
    },
    "Action": "sts:AssumeRoleWithWebIdentity",
    "Condition": {
        "StringLike": {
            "OIDC_PROVIDER:sub": "system:serviceaccount:${Namespace}:emr-
containers-sa-*"
        }
    }
}
```

5. 在具有下列許可的 IAM 中建立 monitoringConfiguration 政策：

   JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogStreams",
                "logs:CreateLogStream",
                "logs:CreateLogGroup",
                "logs:PutLogEvents"
            ],
            "Resource": [
                "arn:aws:logs:us-east-1:account_id:log-group:log_group_name",
                "arn:aws:logs:us-east-1:account_id:log-
group:log_group_name:*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": "logs:DescribeLogGroups",
            "Resource": "*"
```

```
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:PutObject",
                "s3:GetObject",
                "s3:ListBucket"
            ],
            "Resource": [
                "arn:aws:s3:::bucket_name",
                "arn:aws:s3:::bucket_name/*"
            ]
        }
    ]
}
```

6. 將上述政策連接至您的運算子/任務執行角色。

## Spark 運算子日誌

您可以在執行 時，以下列方式定義監控組態`helm install`：

```
helm install spark-operator spark-operator \
--namespace namespace \
--set emrContainers.awsRegion=aws_region \
--set emrContainers.monitoringConfiguration.image=log_agent_image_url \
--set
 emrContainers.monitoringConfiguration.s3MonitoringConfiguration.logUri=S3_bucket_uri \
--set
 emrContainers.monitoringConfiguration.cloudWatchMonitoringConfiguration.logGroupName=log_group
 \
--set
 emrContainers.monitoringConfiguration.cloudWatchMonitoringConfiguration.logStreamNamePrefix=lo
 \
--set emrContainers.monitoringConfiguration.sideCarResources.limits.cpuLimit=500m \
--set emrContainers.monitoringConfiguration.sideCarResources.limits.memoryLimit=512Mi \
--set
 emrContainers.monitoringConfiguration.containerLogRotationConfiguration.rotationSize=2GB
 \
--set
 emrContainers.monitoringConfiguration.containerLogRotationConfiguration.maxFilesToKeep=10
 \
--set webhook.enable=true \
```

```
--set emrContainers.operatorExecutionRoleArn=operator_execution_role_arn
```

監控組態

以下是 monitoringConfiguration 下的可用組態選項。

- 映像 （選用） – 日誌代理程式映像 URL。如果未提供， 將依 emrReleaseLabel 擷取。
- s3MonitoringConfiguration – 將此選項設定為封存至 Amazon S3。
  - logUri –（必要） – 您要存放日誌的 Amazon S3 儲存貯體路徑。
  - 以下是上傳日誌後 Amazon S3 儲存貯體路徑的範例格式。第一個範例顯示未啟用日誌輪換。

    ```
    s3://${logUri}/${POD NAME}/operator/stdout.gz
    s3://${logUri}/${POD NAME}/operator/stderr.gz
    ```

    預設啟用日誌輪換。您可以同時看到具有遞增索引的輪換檔案，以及與上一個範例相同的目前檔案。

    ```
    s3://${logUri}/${POD NAME}/operator/stdout_YYYYMMDD_index.gz
    s3://${logUri}/${POD NAME}/operator/stderr_YYYYMMDD_index.gz
    ```
- cloudWatchMonitoringConfiguration – 要設定轉送的組態金鑰 Amazon CloudWatch。
  - logGroupName （必要） – 您要傳送 Amazon CloudWatch 日誌的日誌群組名稱。如果群組不存在，則會自動建立群組。
  - logStreamNamePrefix （選用） – 您要將日誌傳送到其中的日誌串流名稱。預設值為空字串。中的格式 Amazon CloudWatch 如下：

    ```
    ${logStreamNamePrefix}/${POD NAME}/STDOUT or STDERR
    ```
- sideCarResources （選用） – 在已啟動的 Fluentd 附屬容器上設定資源限制的組態金鑰。
  - memoryLimit （選用） – 記憶體限制。根據需要進行調整。預設值為 512Mi。
  - cpuLimit （選用） – CPU 限制。根據需要進行調整。預設值為 500 公尺。
- containerLogRotationConfiguration （選用） – 控制容器日誌輪換行為。依預設會啟用此功能。
  - rotationSize （必要） – 指定日誌輪換的檔案大小。可能的值範圍為 2KB 至 2GB。rotationSize 參數的數值單位部分會以整數形式傳遞。由於不支援小數值，因此可以使用值 1500MB 來指定 1.5GB 的輪換大小。預設值為 2GB。
  - maxFilesToKeep （必要） – 指定輪換發生後要保留在容器中的檔案數量上限。下限值是 1，上限值是 50。預設為 10。

設定 monitoringConfiguration 之後，您應該能夠檢查 Amazon S3 儲存貯體 Amazon CloudWatch 或或兩者上的 Spark Operator Pod 日誌。對於 Amazon S3 儲存貯體，您需要等待 2 分鐘，第一個日誌檔案才能排清。

若要尋找 中的日誌 Amazon CloudWatch，您可以導覽至下列項目：CloudWatch > 日誌群組 > ***#####*** > *Pod ##*/操作員/stderr

或者，您可以導覽至：CloudWatch > 日誌群組 > ***######*** > *Pod ##*/操作員/stdout

## Spark 應用程式日誌

可採用以下方法定義此設定。

```
apiVersion: "sparkoperator.k8s.io/v1beta2"
kind: SparkApplication
metadata:
  name: spark-pi
  namespace: namespace
spec:
  type: Scala
  mode: cluster
  imagePullPolicy: Always
  mainClass: org.apache.spark.examples.SparkPi
  mainApplicationFile: "local:///usr/lib/spark/examples/jars/spark-examples.jar"
  sparkVersion: "3.3.1"
  emrReleaseLabel: emr_release_label
  executionRoleArn: job_execution_role_arn
  restartPolicy:
    type: Never
  volumes:
    - name: "test-volume"
      hostPath:
        path: "/tmp"
        type: Directory
  driver:
    cores: 1
    coreLimit: "1200m"
    memory: "512m"
    labels:
      version: 3.3.1
    volumeMounts:
      - name: "test-volume"
        mountPath: "/tmp"
```

```
  executor:
    cores: 1
    instances: 1
    memory: "512m"
    labels:
      version: 3.3.1
    volumeMounts:
      - name: "test-volume"
        mountPath: "/tmp"
  monitoringConfiguration:
    image: "log_agent_image"
    s3MonitoringConfiguration:
      logUri: "S3_bucket_uri"
    cloudWatchMonitoringConfiguration:
      logGroupName: "log_group_name"
      logStreamNamePrefix: "log_stream_prefix"
    sideCarResources:
      limits:
        cpuLimit: "500m"
        memoryLimit: "250Mi"
    containerLogRotationConfiguration:
      rotationSize: "2GB"
      maxFilesToKeep: "10"
```

以下是 monitoringConfiguration 下的可用組態選項。

- 映像（選用）– 日誌代理程式映像 URL。如果未提供，將依 emrReleaseLabel 擷取。
- s3MonitoringConfiguration – 將此選項設定為封存至 Amazon S3。
  - logUri（必要）– 您要存放日誌的 Amazon S3 儲存貯體路徑。第一個範例顯示未啟用日誌輪換：

    ```
    s3://${logUri}/${APPLICATION NAME}-${APPLICATION UID}/${POD NAME}/stdout.gz
    s3://${logUri}/${APPLICATION NAME}-${APPLICATION UID}/${POD NAME}/stderr.gz
    ```

    日誌輪換預設為啟用。您可以同時使用輪換的檔案（具有遞增索引）和目前的檔案（沒有日期戳記的檔案）。

    ```
    s3://${logUri}/${APPLICATION NAME}-${APPLICATION UID}/${POD NAME}/
    stdout_YYYYMMDD_index.gz
    s3://${logUri}/${APPLICATION NAME}-${APPLICATION UID}/${POD NAME}/
    stderr_YYYYMMDD_index.gz
    ```

- cloudWatchMonitoringConfiguration – 要設定轉送的組態金鑰 Amazon CloudWatch。
  - logGroupName（必要）– 您要傳送日誌的 Cloudwatch 日誌群組名稱。如果群組不存在，則會自動建立該群組。
  - logStreamNamePrefix（選用）– 您要將日誌傳送到其中的日誌串流名稱。預設值為空字串。CloudWatch 的格式如下所示：

```
${logStreamNamePrefix}/${APPLICATION NAME}-${APPLICATION UID}/${POD NAME}/stdout
${logStreamNamePrefix}/${APPLICATION NAME}-${APPLICATION UID}/${POD NAME}/stderr
```

- sideCarResources（選用）– 在已啟動的 Fluentd 附屬容器上設定資源限制的組態金鑰。
  - memoryLimit（選用）– 記憶體限制。根據需要進行調整。預設值為 250Mi。
  - cpuLimit – CPU 限制。根據需要進行調整。預設值為 500 公尺。
- containerLogRotationConfiguration（選用）– 控制容器日誌輪換行為。依預設會啟用此功能。
  - rotationSize（必要）– 指定日誌輪換的檔案大小。可能的值範圍為 2KB 至 2GB。rotationSize 參數的數值單位部分會以整數形式傳遞。由於不支援小數值，因此可以使用值 1500MB 來指定 1.5GB 的輪換大小。預設值為 2GB。
  - maxFilesToKeep（必要）– 指定輪換發生後要保留在容器中的檔案數量上限。最小值為 1。最大值為 50。預設為 10。

設定 monitoringConfiguration 之後，您應該能夠在 Amazon S3 儲存貯體或 CloudWatch 或兩者上檢查 Spark 應用程式驅動程式和執行器日誌。對於 Amazon S3 儲存貯體，您需要等待 2 分鐘才能排清第一個日誌檔案。例如，在 Amazon S3 中，儲存貯體路徑如下所示：

Amazon S3 > 儲存貯體 > ****** > *Spark ###### - UUID* > *Pod ##* > stderr.gz

或者：

Amazon S3 > 儲存貯體 > ****** > *Spark ###### - UUID* > *Pod ##* > stdout.gz

在 CloudWatch 中，路徑如下所示：

CloudWatch > 日誌群組 > ****** > *Spark ###### - UUID*/*Pod ##*/stderr

或者：

CloudWatch > 日誌群組 > ****** > *Spark ###### - UUID*/*Pod ##*/stdout

# Amazon EMR on EKS 的安全性和 Spark Operator

當您使用 Spark 運算子時，有幾種方式可以設定叢集存取許可。首先是使用角色型存取控制，角色型存取控制 (RBAC) 會根據人員在組織內的角色來限制存取。它已成為處理存取的主要方式。第二個存取方法是擔任 AWS Identity and Access Management 角色，透過特定指派的許可提供資源存取。

主題


- 使用角色型存取控制 (RBAC) 設定叢集存取許可
- 使用服務帳戶的 IAM 角色 (IRSA) 來設定叢集存取許可


## 使用角色型存取控制 (RBAC) 設定叢集存取許可

為了部署 Spark Operator，Amazon EMR on EKS 會為 Spark Operator 和 Spark 應用程式建立了兩個角色和服務帳戶。

主題


- Operator 服務帳戶和角色
- Spark 服務帳戶和角色


### Operator 服務帳戶和角色

Amazon EMR on EKS 會建立 Operator 服務帳戶和角色，以管理適用於 Spark 作業和其他資源 (例如服務) 的 `SparkApplications`。

此服務帳戶的預設名稱為 `emr-containers-sa-spark-operator`。

下列規則適用於此服務角色：

```
 rules:
- apiGroups:
  - ""
  resources:
  - pods
  verbs:
  - "*"
- apiGroups:
  - ""
  resources:
  - services
  - configmaps
```

```
  - secrets
  verbs:
  - create
  - get
  - delete
  - update
- apiGroups:
  - extensions
  - networking.k8s.io
  resources:
  - ingresses
  verbs:
  - create
  - get
  - delete
- apiGroups:
  - ""
  resources:
  - nodes
  verbs:
  - get
- apiGroups:
  - ""
  resources:
  - events
  verbs:
  - create
  - update
  - patch
- apiGroups:
  - ""
  resources:
  - resourcequotas
  verbs:
  - get
  - list
  - watch
- apiGroups:
  - apiextensions.k8s.io
  resources:
  - customresourcedefinitions
  verbs:
  - create
  - get
```

```
  - update
  - delete
- apiGroups:
  - admissionregistration.k8s.io
  resources:
  - mutatingwebhookconfigurations
  - validatingwebhookconfigurations
  verbs:
  - create
  - get
  - update
  - delete
- apiGroups:
  - sparkoperator.k8s.io
  resources:
  - sparkapplications
  - sparkapplications/status
  - scheduledsparkapplications
  - scheduledsparkapplications/status
  verbs:
  - "*"
  {{- if .Values.batchScheduler.enable }}
  # required for the `volcano` batch scheduler
- apiGroups:
  - scheduling.incubator.k8s.io
  - scheduling.sigs.dev
  - scheduling.volcano.sh
  resources:
  - podgroups
  verbs:
  - "*"
  {{- end }}
  {{ if .Values.webhook.enable }}
- apiGroups:
  - batch
  resources:
  - jobs
  verbs:
  - delete
  {{- end }}
```

Spark 服務帳戶和角色

Spark 驅動程式 Pod 需要與該 Pod 位於相同命名空間的 Kubernetes 服務帳戶。此服務帳戶需要許可才能建立、取得、列出、修補和刪除執行程式 Pod，以及為驅動程式建立 Kubernetes 無頭服務。如果沒有服務帳戶，驅動程式會失敗並結束，除非 Pod 命名空間中的預設服務帳戶具有所需許可。

此服務帳戶的預設名稱為 `emr-containers-sa-spark`。

下列規則適用於此服務角色：

```
 rules:
- apiGroups:
  - ""
  resources:
  - pods
  verbs:
  - "*"
- apiGroups:
  - ""
  resources:
  - services
  verbs:
  - "*"
- apiGroups:
  - ""
  resources:
  - configmaps
  verbs:
  - "*"
- apiGroups:
  - ""
  resources:
  - persistentvolumeclaims
  verbs:
  - "*"
```

## 使用服務帳戶的 IAM 角色 (IRSA) 來設定叢集存取許可

本節使用範例示範如何設定 Kubernetes 服務帳戶以擔任 AWS Identity and Access Management 角色。然後，使用服務帳戶的 Pod 可以存取該角色有權存取的任何 AWS 服務。

下列範例會執行 Spark 應用程式，以計算 Amazon S3 中檔案的字數。為此，您可以設定服務帳戶的 IAM 角色 (IRSA)，以驗證和授權 Kubernetes 服務帳戶。

> **Note**
>
> 此範例將 "spark-operator" 命名空間用於 Spark Operator 以及您在其中提交 Spark 應用程式的命名空間。

先決條件

嘗試此頁面的範例之前，請先完成下列先決條件：

- 設定 Spark Operator。
- 安裝 Spark Operator.
- 建立 Amazon S3 儲存貯體。
- 將您最喜愛的詩歌儲存在名為 `poem.txt` 的文字檔案中，然後將檔案上傳到 S3 儲存貯體。在此頁面中建立的 Spark 應用程式將讀取文字檔案的內容。如需有關將檔案上傳到 S3 的詳細資訊，請參閱 Amazon Simple Storage Service 使用者指南中的上傳物件至儲存貯體。

設定要擔任 IAM 角色的 Kubernetes 服務帳戶

使用下列步驟來設定 Kubernetes 服務帳戶，以擔任 IAM 角色，Pod 可以使用該角色存取該角色有權存取 AWS 的服務。

1. 完成 後先決條件，請使用 AWS Command Line Interface 建立 `example-policy.json` 檔案，允許唯讀存取您上傳至 Amazon S3 的檔案：

```
cat >example-policy.json <<EOF
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:ListBucket"
            ],
            "Resource": [
                "arn:aws:s3:::my-pod-bucket",
                "arn:aws:s3:::my-pod-bucket/*"
            ]
        }
```

```
    ]
}
EOF
```

2. 然後，建立 IAM 政策 `example-policy`：

```
aws iam create-policy --policy-name example-policy --policy-document file:// 
example-policy.json
```

3. 接下來，建立 IAM 角色 `example-role`，並將其與 Spark 驅動程式的 Kubernetes 服務帳戶建立關聯：

```
eksctl create iamserviceaccount --name driver-account-sa --namespace spark-operator 
 \
--cluster my-cluster --role-name "example-role" \
--attach-policy-arn arn:aws:iam::111122223333:policy/example-policy --approve
```

4. 使用 Spark 驅動程式服務帳戶所需的叢集角色連結來建立 yaml 檔案：

```
cat >spark-rbac.yaml <<EOF
apiVersion: v1
kind: ServiceAccount
metadata:
  name: driver-account-sa
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: spark-role
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: edit
subjects:
  - kind: ServiceAccount
    name: driver-account-sa
    namespace: spark-operator
EOF
```

5. 套用叢集角色連結組態：

```
kubectl apply -f spark-rbac.yaml
```

kubectl 命令應該確認成功建立帳戶：

```
serviceaccount/driver-account-sa created
clusterrolebinding.rbac.authorization.k8s.io/spark-role configured
```

從 Spark Operator 中執行應用程式

設定 Kubernetes 服務帳戶之後，可以執行 Spark 應用程式來計算作為 先決條件 的一部分上傳的文字檔案中的字數。

1. 建立新檔案 `word-count.yaml`，其中包含字數統計應用程式的 `SparkApplication` 定義。

```
cat >word-count.yaml <<EOF
apiVersion: "sparkoperator.k8s.io/v1beta2"
kind: SparkApplication
metadata:
  name: word-count
  namespace: spark-operator
spec:
  type: Java
  mode: cluster
  image: "895885662937.dkr.ecr.us-west-2.amazonaws.com/spark/emr-6.10.0:latest"
  imagePullPolicy: Always
  mainClass: org.apache.spark.examples.JavaWordCount
  mainApplicationFile: local:///usr/lib/spark/examples/jars/spark-examples.jar
  arguments:
    - s3://my-pod-bucket/poem.txt
  hadoopConf:
   # EMRFS filesystem
    fs.s3.customAWSCredentialsProvider:
 com.amazonaws.auth.WebIdentityTokenCredentialsProvider
    fs.s3.impl: com.amazon.ws.emr.hadoop.fs.EmrFileSystem
    fs.AbstractFileSystem.s3.impl: org.apache.hadoop.fs.s3.EMRFSDelegate
    fs.s3.buffer.dir: /mnt/s3
    fs.s3.getObject.initialSocketTimeoutMilliseconds: "2000"

 mapreduce.fileoutputcommitter.algorithm.version.emr_internal_use_only.EmrFileSystem:
 "2"
    mapreduce.fileoutputcommitter.cleanup-
failures.ignored.emr_internal_use_only.EmrFileSystem: "true"
  sparkConf:
    # Required for EMR Runtime
```

```
    spark.driver.extraClassPath: /usr/lib/hadoop-lzo/lib/*:/usr/lib/hadoop/
hadoop-aws.jar:/usr/share/aws/aws-java-sdk/*:/usr/share/aws/emr/emrfs/conf:/usr/
share/aws/emr/emrfs/lib/*:/usr/share/aws/emr/emrfs/auxlib/*:/usr/share/aws/emr/
security/conf:/usr/share/aws/emr/security/lib/*:/usr/share/aws/hmclient/lib/aws-
glue-datacatalog-spark-client.jar:/usr/share/java/Hive-JSON-Serde/hive-openx-
serde.jar:/usr/share/aws/sagemaker-spark-sdk/lib/sagemaker-spark-sdk.jar:/home/
hadoop/extrajars/*
    spark.driver.extraLibraryPath: /usr/lib/hadoop/lib/native:/usr/lib/hadoop-lzo/
lib/native:/docker/usr/lib/hadoop/lib/native:/docker/usr/lib/hadoop-lzo/lib/native
    spark.executor.extraClassPath: /usr/lib/hadoop-lzo/lib/*:/usr/lib/hadoop/
hadoop-aws.jar:/usr/share/aws/aws-java-sdk/*:/usr/share/aws/emr/emrfs/conf:/usr/
share/aws/emr/emrfs/lib/*:/usr/share/aws/emr/emrfs/auxlib/*:/usr/share/aws/emr/
security/conf:/usr/share/aws/emr/security/lib/*:/usr/share/aws/hmclient/lib/aws-
glue-datacatalog-spark-client.jar:/usr/share/java/Hive-JSON-Serde/hive-openx-
serde.jar:/usr/share/aws/sagemaker-spark-sdk/lib/sagemaker-spark-sdk.jar:/home/
hadoop/extrajars/*
    spark.executor.extraLibraryPath: /usr/lib/hadoop/lib/native:/usr/lib/hadoop-
lzo/lib/native:/docker/usr/lib/hadoop/lib/native:/docker/usr/lib/hadoop-lzo/lib/
native
  sparkVersion: "3.3.1"
  restartPolicy:
    type: Never
  driver:
    cores: 1
    coreLimit: "1200m"
    memory: "512m"
    labels:
      version: 3.3.1
    serviceAccount: my-spark-driver-sa
  executor:
    cores: 1
    instances: 1
    memory: "512m"
    labels:
      version: 3.3.1
EOF
```

2. 提交 Spark 應用程式。

```
kubectl apply -f word-count.yaml
```

kubectl 命令應該傳回您已成功建立名為 `word-count` 的 `SparkApplication` 物件的確認資訊。

```
sparkapplication.sparkoperator.k8s.io/word-count configured
```

3. 若要檢查 SparkApplication 物件的事件，請執行下列命令：

```
kubectl describe sparkapplication word-count -n spark-operator
```

kubectl 命令應傳回 SparkApplication 的描述和事件：

```
Events:
  Type     Reason                                Age                    From
    Message
  ----     ------                                ----                   ----
    -------
  Normal   SparkApplicationSpecUpdateProcessed  3m2s (x2 over 17h)     spark-
operator  Successfully processed spec update for SparkApplication word-count
  Warning  SparkApplicationPendingRerun         3m2s (x2 over 17h)     spark-
operator  SparkApplication word-count is pending rerun
  Normal   SparkApplicationSubmitted            2m58s (x2 over 17h)    spark-
operator  SparkApplication word-count was submitted successfully
  Normal   SparkDriverRunning                   2m56s (x2 over 17h)    spark-
operator  Driver word-count-driver is running
  Normal   SparkExecutorPending                 2m50s                  spark-
operator  Executor [javawordcount-fdd1698807392c66-exec-1] is pending
  Normal   SparkExecutorRunning                 2m48s                  spark-
operator  Executor [javawordcount-fdd1698807392c66-exec-1] is running
  Normal   SparkDriverCompleted                 2m31s (x2 over 17h)    spark-
operator  Driver word-count-driver completed
  Normal   SparkApplicationCompleted            2m31s (x2 over 17h)    spark-
operator  SparkApplication word-count completed
  Normal   SparkExecutorCompleted               2m31s (x2 over 2m31s)  spark-
operator  Executor [javawordcount-fdd1698807392c66-exec-1] completed
```

應用程式現在正在計算 S3 檔案中的單詞。要尋找字數，請參閱驅動程式的日誌檔案：

```
kubectl logs pod/word-count-driver -n spark-operator
```

kubectl 命令應傳回日誌檔案的內容及字數統計應用程式的結果。

```
INFO DAGScheduler: Job 0 finished: collect at JavaWordCount.java:53, took 5.146519 s
```

```
                Software: 1
```

如需有關如何透過 Spark Operator 將應用程式提交至 Spark 的詳細資訊，請參閱 GitHub 上 Kubernetes Operator for Apache Spark (spark-on-k8s-operator) 文件中的使用 SparkApplication。

# 使用 spark-submit 執行 Spark 作業

Amazon EMR 6.10.0 版及更高版本支援 `spark-submit` 作為命令列工具，可以使用它向 Amazon EMR on EKS 叢集提交 Spark 應用程式並執行。

> **Note**
>
> Amazon EMR 會根據 vCPU 和記憶體消耗量計算 Amazon EKS 定價。此計算適用於驅動程式和執行器 Pod。此計算從您下載 Amazon EMR 應用程式映像時開始，直到 Amazon EKS Pod 終止且四捨五入至最接近的秒數為止。

主題


- 為 Amazon EMR on EKS 設定 spark-submit
- 開始針對 Amazon EMR on EKS 使用 spark-submit
- 驗證 spark-submit 的 Spark 驅動程式服務帳戶安全需求


## 為 Amazon EMR on EKS 設定 spark-submit

完成下列任務進行設定，然後才能在 Amazon EMR on EKS 上使用 spark-submit 執行應用程式。如果已經註冊 Amazon Web Services (AWS) 且已在使用 Amazon EKS，則您幾乎可使用 Amazon EMR on EKS。如果已經完成任何先決條件，則可以跳過這些先決條件，然後繼續進行下一個。

- 安裝或更新至最新版本的 AWS CLI- 如果您已安裝 AWS CLI，請確認您擁有最新版本。
- 設定 kubectl 和 eksctl – eksctl 是用來與 Amazon EKS 通訊的命令列工具。
- 開始使用 Amazon EKS – eksctl – 請依照步驟，在 Amazon EKS 中建立具有節點的新 Kubernetes 叢集。
- 選擇 Amazon EMR 基礎映像 URI (6.10.0 版或更高版本) - Amazon EMR 6.10.0 版及更高版本支援 `spark-submit` 命令。
- 確認驅動程式服務帳戶具有建立和監控執行程式 Pod 的適當許可。如需詳細資訊，請參閱驗證 spark-submit 的 Spark 驅動程式服務帳戶安全需求。

- 設定本機 AWS 憑證設定檔。
- 從 Amazon EKS 主控台，選擇您的 EKS 叢集，然後尋找位於概觀、詳細資訊，以及 API 伺服器端點下的 EKS 叢集端點。

## 開始針對 Amazon EMR on EKS 使用 spark-submit

Amazon EMR 6.10.0 及更高版本支援 spark-submit，可在 Amazon EKS 叢集上執行 Spark 應用程式。以下章節說明如何提交 Spark 應用程式的命令。

### 執行 Spark 應用程式

完成以下步驟，以執行 Spark 應用程式：

1. 在使用 `spark-submit` 命令執行 Spark 應用程式之前，請先完成 為 Amazon EMR on EKS 設定 spark-submit 中的步驟。
2. 使用 Amazon EMR on EKS 基礎映像執行容器。如需詳細資訊，請參閱如何選取基礎映像 URI。

```
kubectl run -it containerName --image=EMRonEKSImage --command -n namespace /bin/
bash
```

3. 設定以下環境變數的值：

```
export SPARK_HOME=spark-home
export MASTER_URL=k8s://Amazon EKS-cluster-endpoint
```

4. 現在，使用下列命令提交 Spark 應用程式：

```
$SPARK_HOME/bin/spark-submit \
 --class org.apache.spark.examples.SparkPi \
 --master $MASTER_URL \
 --conf spark.kubernetes.container.image=895885662937.dkr.ecr.us-
west-2.amazonaws.com/spark/emr-6.10.0:latest \
 --conf spark.kubernetes.authenticate.driver.serviceAccountName=spark \
 --deploy-mode cluster \
 --conf spark.kubernetes.namespace=spark-operator \
 local:///usr/lib/spark/examples/jars/spark-examples.jar 20
```

如需有關將應用程式提交到 Spark 的詳細資訊，請參閱 Apache Spark 文件中的提交應用程式。

> **Important**
>
> `spark-submit` 僅支援叢集模式作為提交機制。

## 驗證 spark-submit 的 Spark 驅動程式服務帳戶安全需求

Spark 驅動程式 Pod 使用 Kubernetes 服務帳戶來存取 Kubernetes API 伺服器，以建立和監控執行程式 Pod。驅動程式服務帳戶必須具有適當的許可，才能列出、建立、編輯、修補及刪除叢集中的 Pod。透過執行以下命令，可確認您可列出這些資源：

```
kubectl auth can-i list|create|edit|delete|patch pods
```

透過執行每個命令來驗證您具有必要的許可。

```
kubectl auth can-i list pods
kubectl auth can-i create pods
kubectl auth can-i edit pods
kubectl auth can-i delete pods
kubectl auth can-i patch pods
```

下列規則適用於此服務角色：

```
 rules:
- apiGroups:
  - ""
  resources:
  - pods
  verbs:
  - "*"
- apiGroups:
  - ""
  resources:
  - services
  verbs:
  - "*"
- apiGroups:
  - ""
  resources:
  - configmaps
  verbs:
```

```
  - "*"
- apiGroups:
  - ""
  resources:
  - persistentvolumeclaims
  verbs:
  - "*"
```

## 為 spark-submit 設定服務帳戶 (IRSA) 的 IAM 角色

以下各節說明如何設定服務帳戶 (IRSA) 的 IAM 角色來驗證和授權 Kubernetes 服務帳戶，以便您可以執行存放在 Amazon S3 中的 Spark 應用程式。

先決條件

在嘗試本文件中的任何範例之前，請確定您已完成下列先決條件：

- 已完成設定 spark-submit
- 建立 S3 儲存貯體並上傳 Spark 應用程式 jar

設定 Kubernetes 服務帳戶以擔任 IAM 角色

下列步驟說明如何設定 Kubernetes 服務帳戶以擔任 AWS Identity and Access Management (IAM) 角色。將 Pod 設定為使用服務帳戶後，他們可以存取角色具有存取許可的任何 AWS 服務 。

1. 建立政策檔案，以允許唯讀存取您上傳的 Amazon S3 物件：

```
cat >my-policy.json <<EOF
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:ListBucket"
            ],
            "Resource": [
                "arn:aws:s3:::<my-spark-jar-bucket>",
                "arn:aws:s3:::<my-spark-jar-bucket>/*"
            ]
        }
```

```
    ]
}
EOF
```

2. 建立 IAM 政策。

```
aws iam create-policy --policy-name my-policy --policy-document file://my-
policy.json
```

3. 建立 IAM 角色，並將其與 Spark 驅動程式的 Kubernetes 服務帳戶建立關聯

```
eksctl create iamserviceaccount --name my-spark-driver-sa --namespace spark-
operator \
--cluster my-cluster --role-name "my-role" \
--attach-policy-arn arn:aws:iam::111122223333:policy/my-policy --approve
```

4. 建立具有 Spark 驅動程式服務帳戶所需許可的 YAML 檔案：

```
cat >spark-rbac.yaml <<EOF
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  namespace: default
  name: emr-containers-role-spark
rules:
- apiGroups:
  - ""
  resources:
  - pods
  verbs:
  - "*"
- apiGroups:
  - ""
  resources:
  - services
  verbs:
  - "*"
- apiGroups:
  - ""
  resources:
  - configmaps
  verbs:
  - "*"
```

```
- apiGroups:
  - ""
  resources:
  - persistentvolumeclaims
  verbs:
  - "*"
---
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: spark-role-binding
  namespace: default
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: Role
  name: emr-containers-role-spark
subjects:
- kind: ServiceAccount
  name: emr-containers-sa-spark
  namespace: default
EOF
```

5. 套用叢集角色繫結組態。

```
kubectl apply -f spark-rbac.yaml
```

6. `kubectl` 命令應傳回所建立帳戶的確認。

```
serviceaccount/emr-containers-sa-spark created
clusterrolebinding.rbac.authorization.k8s.io/emr-containers-role-spark configured
```

### 執行 Spark 應用程式

Amazon EMR 6.10.0 及更高版本支援 spark-submit，可在 Amazon EKS 叢集上執行 Spark 應用程式。完成以下步驟，以執行 Spark 應用程式：

1. 請確定您已完成設定 Amazon EMR on EKS 的 spark-submit 中的步驟。
2. 設定以下環境變數的值：

```
export SPARK_HOME=spark-home
export MASTER_URL=k8s://Amazon EKS-cluster-endpoint
```

3. 現在，使用下列命令提交 Spark 應用程式：

```
$SPARK_HOME/bin/spark-submit \
 --class org.apache.spark.examples.SparkPi \
 --master $MASTER_URL \
 --conf spark.kubernetes.container.image=895885662937.dkr.ecr.us-
west-2.amazonaws.com/spark/emr-6.15.0:latest \
 --conf spark.kubernetes.authenticate.driver.serviceAccountName=emr-containers-sa-
spark \
 --deploy-mode cluster \
 --conf spark.kubernetes.namespace=default \
 --conf "spark.driver.extraClassPath=/usr/lib/hadoop-lzo/lib/*:/usr/lib/hadoop/
hadoop-aws.jar:/usr/share/aws/aws-java-sdk/*:/usr/share/aws/emr/emrfs/conf:/usr/
share/aws/emr/emrfs/lib/*:/usr/share/aws/emr/emrfs/auxlib/*:/usr/share/aws/emr/
security/conf:/usr/share/aws/emr/security/lib/*:/usr/share/aws/hmclient/lib/aws-
glue-datacatalog-spark-client.jar:/usr/share/java/Hive-JSON-Serde/hive-openx-
serde.jar:/usr/share/aws/sagemaker-spark-sdk/lib/sagemaker-spark-sdk.jar:/home/
hadoop/extrajars/*" \
 --conf "spark.driver.extraLibraryPath=/usr/lib/hadoop/lib/native:/usr/lib/hadoop-
lzo/lib/native:/docker/usr/lib/hadoop/lib/native:/docker/usr/lib/hadoop-lzo/lib/
native" \
 --conf "spark.executor.extraClassPath=/usr/lib/hadoop-lzo/lib/*:/usr/lib/hadoop/
hadoop-aws.jar:/usr/share/aws/aws-java-sdk/*:/usr/share/aws/emr/emrfs/conf:/usr/
share/aws/emr/emrfs/lib/*:/usr/share/aws/emr/emrfs/auxlib/*:/usr/share/aws/emr/
security/conf:/usr/share/aws/emr/security/lib/*:/usr/share/aws/hmclient/lib/aws-
glue-datacatalog-spark-client.jar:/usr/share/java/Hive-JSON-Serde/hive-openx-
serde.jar:/usr/share/aws/sagemaker-spark-sdk/lib/sagemaker-spark-sdk.jar:/home/
hadoop/extrajars/*" \
 --conf "spark.executor.extraLibraryPath=/usr/lib/hadoop/lib/native:/usr/lib/
hadoop-lzo/lib/native:/docker/usr/lib/hadoop/lib/native:/docker/usr/lib/hadoop-lzo/
lib/native" \
 --conf
 spark.hadoop.fs.s3.customAWSCredentialsProvider=com.amazonaws.auth.WebIdentityTokenCredent
 \
 --conf spark.hadoop.fs.s3.impl=com.amazon.ws.emr.hadoop.fs.EmrFileSystem \
 --conf
 spark.hadoop.fs.AbstractFileSystem.s3.impl=org.apache.hadoop.fs.s3.EMRFSDelegate \
 --conf spark.hadoop.fs.s3.buffer.dir=/mnt/s3 \
 --conf spark.hadoop.fs.s3.getObject.initialSocketTimeoutMilliseconds="2000" \
 --conf
 spark.hadoop.mapreduce.fileoutputcommitter.algorithm.version.emr_internal_use_only.EmrFile
 \
```

```
 --conf spark.hadoop.mapreduce.fileoutputcommitter.cleanup-
failures.ignored.emr_internal_use_only.EmrFileSystem="true" \
 s3://my-pod-bucket/spark-examples.jar 20
```

4. 在 Spark 驅動程式完成 Spark 任務後，您應該會在提交結束時看到日誌行，指出 Spark 任務已完成。

```
23/11/24 17:02:14 INFO LoggingPodStatusWatcherImpl: Application
 org.apache.spark.examples.SparkPi with submission ID default:org-apache-spark-
examples-sparkpi-4980808c03ff3115-driver finished
23/11/24 17:02:14 INFO ShutdownHookManager: Shutdown hook called
```

清除

執行應用程式完成後，您可以使用下列命令執行清除。

```
kubectl delete -f spark-rbac.yaml
```

# 搭配 Amazon EMR on EKS 使用 Apache Livy

透過 Amazon EMR 7.1.0 版和更新版本，您可以使用 Apache Livy 在 Amazon EMR on EKS 上提交任務。使用 Apache Livy，您可以設定自己的 Apache Livy REST 端點，並使用它在 Amazon EKS 叢集上部署和管理 Spark 應用程式。在 Amazon EKS 叢集中安裝 Livy 之後，您可以使用 Livy 端點將 Spark 應用程式提交到您的 Livy 伺服器。伺服器會管理 Spark 應用程式的生命週期。

> **Note**
>
> Amazon EMR 會根據 vCPU 和記憶體使用量計算 Amazon EKS 定價。此計算適用於驅動程式和執行器 Pod。此計算會從您下載 Amazon EMR 應用程式映像時開始，直到 Amazon EKS Pod 終止且四捨五入至最接近的秒數為止。

主題


- 為 Amazon EMR on EKS 設定 Apache Livy
- Amazon EMR on EKS 上的 Apache Livy 入門
- 使用適用於 Amazon EMR on EKS 的 Apache Livy 執行 Spark 應用程式
- 使用 Amazon EMR on EKS 解除安裝 Apache Livy

- Apache Livy 搭配 Amazon EMR on EKS 的安全性
- Amazon EMR on EKS 版本上 Apache Livy 的安裝屬性
- 疑難排解常見的環境變數格式錯誤

## 為 Amazon EMR on EKS 設定 Apache Livy

您必須先安裝和設定一組先決條件工具，才能在 Amazon EKS 叢集上安裝 Apache Livy。其中包括 AWS CLI，這是使用 AWS 資源的基本命令列工具、使用 Amazon EKS 的命令列工具，以及在此使用案例中使用的控制器，讓您的叢集應用程式可供網際網路使用，以及路由網路流量。

- 安裝或更新至最新版本的 AWS CLI- 如果您已安裝 AWS CLI，請確認您擁有最新版本。
- 設定 kubectl 和 eksctl – eksctl 是您用來與 Amazon EKS 通訊的命令列工具。
- 安裝 Helm – 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。
- 開始使用 Amazon EKS – eksctl – 請依照步驟在 Amazon EKS 中建立具有節點的新 Kubernetes 叢集。
- 選取 Amazon EMR 發行標籤 – Amazon EMR 7.1.0 及更高版本支援 Apache Livy。
- 安裝 ALB 控制器 – ALB 控制器管理 Kubernetes 叢集的 AWS Elastic Load Balancing。當您在設定 Apache Livy 時建立 Kubernetes 輸入時，它會建立 AWS Network Load Balancer (NLB)。

## Amazon EMR on EKS 上的 Apache Livy 入門

完成下列步驟以安裝 Apache Livy。其中包括設定套件管理員、建立執行 Spark 工作負載的命名空間、安裝 Livy、設定負載平衡和驗證步驟。您必須完成這些步驟，才能使用 Spark 執行批次任務。

1. 如果您尚未設定 Amazon EMR on EKS 的 Apache Livy。
2. 向 Amazon ECR 登錄檔驗證 Helm 用戶端。您可以從 AWS 區域 Amazon ECR 登錄帳戶依區域尋找 的對應`ECR-registry-account`值。

```
aws ecr get-login-password \--region <AWS_REGION> | helm registry login \
--username AWS \
--password-stdin <ECR-registry-account>.dkr.ecr.<region-id>.amazonaws.com
```

3. 設定 Livy 會為 Livy 伺服器建立服務帳戶，並為 Spark 應用程式建立另一個帳戶。若要設定服務帳戶的 IRSA，請參閱使用服務帳戶 (IRSA) 的 IAM 角色設定存取許可。

4. 建立命名空間來執行 Spark 工作負載。

```
kubectl create ns <spark-ns>
```

5. 使用以下命令來安裝 Livy。

此 Livy 端點只能在內部供 EKS 叢集中的 VPC 使用。若要啟用 VPC 以外的存取，請在 Helm 安裝命令`--set loadbalancer.internal=false`中設定 。

> **Note**
>
> 根據預設，在此 Livy 端點內不會啟用 SSL，而且端點僅在 EKS 叢集的 VPC 內可見。如果您設定 `loadbalancer.internal=false`和 `ssl.enabled=false`，則會將不安全的端點公開到 VPC 外部。若要設定安全 Livy 端點，請參閱使用 TLS/SSL 設定安全 Apache Livy 端點。

```
helm install livy-demo \
  oci://895885662937.dkr.ecr.region-id.amazonaws.com/livy \
  --version 7.9.0 \
  --namespace livy-ns \
  --set image=ECR-registry-account.dkr.ecr.region-id.amazonaws.com/livy/
emr-7.9.0:latest \
  --set sparkNamespace=<spark-ns> \
  --create-namespace
```

您應該會看到下列輸出。

```
NAME: livy-demo
LAST DEPLOYED: Mon Mar 18 09:23:23 2024
NAMESPACE: livy-ns
STATUS: deployed
REVISION: 1
TEST SUITE: None
NOTES:
The Livy server has been installed.
Check installation status:
1. Check Livy Server pod is running
  kubectl --namespace livy-ns get pods -l "app.kubernetes.io/instance=livy-demo"
2. Verify created NLB is in Active state and it's target groups are healthy (if
 loadbalancer.enabled is true)
```

```
Access LIVY APIs:
    # Ensure your NLB is active and healthy
    # Get the Livy endpoint using command:
    LIVY_ENDPOINT=$(kubectl get svc -n livy-ns -l app.kubernetes.io/
instance=livy-demo,emr-containers.amazonaws.com/type=loadbalancer -o
 jsonpath='{.items[0].status.loadBalancer.ingress[0].hostname}' |  awk '{printf
 "%s:8998\n", $0}')
    # Access Livy APIs using http://$LIVY_ENDPOINT or https://$LIVY_ENDPOINT (if
 SSL is enabled)
    # Note: While uninstalling Livy, makes sure the ingress and NLB are deleted
 after running the helm command to avoid dangling resources
```

Livy 伺服器和 Spark 工作階段的預設服務帳戶名稱為 `emr-containers-sa-livy`和 `emr-containers-sa-spark-livy`。若要使用自訂名稱，請使用 `serviceAccounts.name`和 `sparkServiceAccount.name` 參數。

```
--set serviceAccounts.name=my-service-account-for-livy
--set sparkServiceAccount.name=my-service-account-for-spark
```

6. 確認您已安裝 Helm Chart。

```
helm list -n livy-ns -o yaml
```

`helm list` 命令應該會傳回新 Helm Chart 的相關資訊。

```
app_version: 0.7.1-incubating
chart: livy-emr-7.9.0
name: livy-demo
namespace: livy-ns
revision: "1"
status: deployed
updated: 2024-02-08 22:39:53.539243 -0800 PST
```

7. 確認 Network Load Balancer 處於作用中狀態。

```
LIVY_NAMESPACE=<livy-ns>
LIVY_APP_NAME=<livy-app-name>
AWS_REGION=<AWS_REGION>

# Get the NLB Endpoint URL
```

```
NLB_ENDPOINT=$(kubectl --namespace $LIVY_NAMESPACE get svc -l "app.kubernetes.io/
instance=$LIVY_APP_NAME,emr-containers.amazonaws.com/type=loadbalancer" -o
 jsonpath='{.items[0].status.loadBalancer.ingress[0].hostname}')

# Get all the load balancers in the account's region
ELB_LIST=$(aws elbv2 describe-load-balancers --region $AWS_REGION)

# Get the status of the NLB that matching the endpoint from the Kubernetes service
NLB_STATUS=$(echo $ELB_LIST | grep -A 8 "\"DNSName\": \"$NLB_ENDPOINT\"" | awk '/
Code/{print $2}/}/' | tr -d '"},\n')
echo $NLB_STATUS
```

8. 現在驗證 Network Load Balancer 中的目標群組是否正常運作。

```
LIVY_NAMESPACE=<livy-ns>
LIVY_APP_NAME=<livy-app-name>
AWS_REGION=<AWS_REGION>

# Get the NLB endpoint
NLB_ENDPOINT=$(kubectl --namespace $LIVY_NAMESPACE get svc -l "app.kubernetes.io/
instance=$LIVY_APP_NAME,emr-containers.amazonaws.com/type=loadbalancer" -o
 jsonpath='{.items[0].status.loadBalancer.ingress[0].hostname}')

# Get all the load balancers in the account's region
ELB_LIST=$(aws elbv2 describe-load-balancers --region $AWS_REGION)

# Get the NLB ARN from the NLB endpoint
NLB_ARN=$(echo $ELB_LIST | grep -B 1 "\"DNSName\": \"$NLB_ENDPOINT\"" | awk
 '/"LoadBalancerArn":/,/"/'| awk '/:/{print $2}' | tr -d \",)

# Get the target group from the NLB. Livy setup only deploys 1 target group
TARGET_GROUP_ARN=$(aws elbv2 describe-target-groups --load-balancer-arn $NLB_ARN
 --region $AWS_REGION | awk '/"TargetGroupArn":/,/"/'| awk '/:/{print $2}' | tr -d
 \",)

# Get health of target group
aws elbv2 describe-target-health --target-group-arn $TARGET_GROUP_ARN
```

以下是顯示目標群組狀態的範例輸出：

```
{
    "TargetHealthDescriptions": [
        {
```

```
            "Target": {
                "Id": "<target IP>",
                "Port": 8998,
                "AvailabilityZone": "us-west-2d"
            },
            "HealthCheckPort": "8998",
            "TargetHealth": {
                "State": "healthy"
            }
        }
    ]
}
```

一旦 NLB 狀態變為 `active` 且目標群組為 `healthy`，您就可以繼續。可能需要幾分鐘的時間。

9. 從 Helm 安裝擷取 Livy 端點。您的 Livy 端點是否安全取決於您是否啟用 SSL。

```
LIVY_NAMESPACE=<livy-ns>
 LIVY_APP_NAME=livy-app-name
 LIVY_ENDPOINT=$(kubectl get svc -n livy-ns -l app.kubernetes.io/
instance=livy-app-name,emr-containers.amazonaws.com/type=loadbalancer -o
 jsonpath='{.items[0].status.loadBalancer.ingress[0].hostname}' |  awk '{printf
 "%s:8998\n", $0}')
 echo "$LIVY_ENDPOINT"
```

10. 從 Helm 安裝擷取 Spark 服務帳戶

```
SPARK_NAMESPACE=spark-ns
LIVY_APP_NAME=<livy-app-name>
SPARK_SERVICE_ACCOUNT=$(kubectl --namespace $SPARK_NAMESPACE
 get sa -l "app.kubernetes.io/instance=$LIVY_APP_NAME" -o
 jsonpath='{.items[0].metadata.name}')
echo "$SPARK_SERVICE_ACCOUNT"
```

您應該會看到類似下列的輸出內容：

```
emr-containers-sa-spark-livy
```

11. 如果您`internalALB=true`將 設定為從 VPC 外部啟用存取，請建立 Amazon EC2 執行個體，並確保 Network Load Balancer 允許來自 EC2 執行個體的網路流量。您必須這樣做，執行個體才能

存取您的 Livy 端點。如需在 VPC 外部安全地公開端點的詳細資訊，請參閱使用 TLS/SSL 設定安全的 Apache Livy 端點。

12. 安裝 Livy 會建立服務帳戶`emr-containers-sa-spark`以執行 Spark 應用程式。如果您的 Spark 應用程式使用任何 AWS 資源，例如 S3 或呼叫 AWS API 或 CLI 操作，您必須將具有必要許可的 IAM 角色連結至您的 Spark 服務帳戶。如需詳細資訊，請參閱使用服務帳戶 (IRSA) 的 IAM 角色設定存取許可。

Apache Livy 支援您可以在安裝 Livy 時使用的其他組態。如需詳細資訊，請參閱 Apache Livy on Amazon EMR on EKS 版本的安裝屬性。

## 使用適用於 Amazon EMR on EKS 的 Apache Livy 執行 Spark 應用程式

在使用 Apache Livy 執行 Spark 應用程式之前，請確定您已完成設定適用於 Amazon EMR on EKS 的 Apache Livy 和適用於 Amazon EMR on EKS 的 Apache Livy 入門中的步驟。

您可以使用 Apache Livy 來執行兩種類型的應用程式：

- 批次工作階段 – 一種 Livy 工作負載，用於提交 Spark 批次任務。
- 互動式工作階段 – 一種 Livy 工作負載，提供程式設計和視覺化界面來執行 Spark 查詢。

> **Note**
> 來自不同工作階段的驅動程式和執行器 Pod 可以互相通訊。命名空間不保證 Pod 之間的任何安全性。Kubernetes 不允許特定命名空間內 Pod 子集的選擇性許可。

### 執行批次工作階段

若要提交批次任務，請使用下列命令。

```
curl -s -k -H 'Content-Type: application/json' -X POST \
      -d '{
            "name": "my-session",
            "file": "entryPoint_location (S3 or local)",
            "args": ["argument1", "argument2", ...],
            "conf": {
                "spark.kubernetes.namespace": "<spark-namespace>",
                "spark.kubernetes.container.image": "public.ecr.aws/emr-on-eks/spark/
emr-7.9.0:latest",
```

```
                "spark.kubernetes.authenticate.driver.serviceAccountName": "<spark-
service-account>"
            }
        }' <livy-endpoint>/batches
```

若要監控您的批次任務，請使用下列命令。

```
curl -s -k -H 'Content-Type: application/json' -X GET <livy-endpoint>/batches/my-
session
```

## 執行互動式工作階段

若要使用 Apache Livy 執行互動式工作階段，請參閱下列步驟。

1. 請確定您能夠存取自我託管或受管 Jupyter 筆記本，例如 SageMaker AI Jupyter 筆記本。您的 jupyter 筆記本必須已安裝 Sparkmagic。
2. 建立 Spark 組態 的儲存貯體`spark.kubernetes.file.upload.path`。確定 Spark 服務帳戶具有對儲存貯體的讀取和寫入存取權。如需如何設定 Spark 服務帳戶的詳細資訊，請參閱使用服務帳戶的 IAM 角色設定存取許可 (IRSA)
3. 使用命令 在 Jupyter 筆記本中載入 Sparkmagic`%load_ext sparkmagic.magics`。
4. 執行 命令`%manage_spark`，使用 Jupyter 筆記本設定您的 Livy 端點。選擇新增端點索引標籤，選擇設定的身分驗證類型，將 Livy 端點新增至筆記本，然後選擇新增端點。
5. `%manage_spark` 再次執行 以建立 Spark 內容，然後前往建立工作階段。選擇 Livy 端點，指定唯一的工作階段名稱選擇語言，然後新增下列屬性。

   ```
   {
     "conf": {
       "spark.kubernetes.namespace": "livy-namespace",
       "spark.kubernetes.container.image": "public.ecr.aws/emr-on-eks/spark/
   emr-7.9.0:latest",
       "spark.kubernetes.authenticate.driver.serviceAccountName": "<spark-service-
   account>",
       "spark.kubernetes.file.upload.path": "<URI_TO_S3_LOCATION_>"
     }
   }
   ```

6. 提交應用程式並等待它建立 Spark 內容。
7. 若要監控互動式工作階段的狀態，請執行下列命令。

```
curl -s -k -H 'Content-Type: application/json' -X GET livy-endpoint/sessions/my-
interactive-session
```

### 監控 Spark 應用程式

若要使用 Livy UI 監控 Spark 應用程式的進度，請使用連結 `http://<livy-endpoint>/ui`。

## 使用 Amazon EMR on EKS 解除安裝 Apache Livy

請依照下列步驟解除安裝 Apache Livy。

1. 使用命名空間名稱和應用程式名稱刪除 Livy 設定。在此範例中，應用程式名稱為 `livy-demo`，命名空間為 `livy-ns`。

   ```
   helm uninstall livy-demo -n livy-ns
   ```

2. 解除安裝時，Amazon EMR on EKS 會刪除 Livy 中的 Kubernetes 服務、 AWS 負載平衡器，以及您在安裝期間建立的目標群組。刪除資源可能需要幾分鐘的時間。在命名空間上再次安裝 Livy 之前，請確定已刪除資源。
3. 刪除 Spark 命名空間。

   ```
   kubectl delete namespace spark-ns
   ```

## Apache Livy 搭配 Amazon EMR on EKS 的安全性

請參閱下列主題，進一步了解如何使用 Amazon EMR on EKS 設定 Apache Livy 的安全性。這些選項包括使用傳輸層安全性、角色型存取控制，這是根據人員在組織內的角色進行存取，以及使用根據授予的許可提供資源存取權的 IAM 角色。

主題

- 使用 TLS/SSL 設定安全的 Apache Livy 端點
- 使用角色型存取控制 (RBAC) 設定 Apache Livy 和 Spark 應用程式許可
- 使用服務帳戶的 IAM 角色設定存取許可 (IRSA)

## 使用 TLS/SSL 設定安全的 Apache Livy 端點

請參閱下列各節，進一步了解如何使用end-to-end TLS 和 SSL 加密為 Amazon EMR on EKS 設定 Apache Livy。

### 設定 TLS 和 SSL 加密

若要在 Apache Livy 端點上設定 SSL 加密，請遵循下列步驟。

- 安裝 Secrets Store CSI 驅動程式和 AWS Secrets and Configuration Provider (ASCP) – Secrets Store CSI 驅動程式和 ASCP 安全地存放 Livy 伺服器 Pod 啟用 SSL 所需的 Livy 的 JKS 憑證和密碼。您也可以只安裝 Secrets Store CSI Driver，並使用任何其他支援的秘密提供者。
- 建立 ACM 憑證 – 需要此憑證才能保護用戶端與 ALB 端點之間的連線。
- 為 AWS Secrets Manager – 設定 JKS 憑證、金鑰密碼和金鑰存放區密碼，以保護 ALB 端點與 Livy 伺服器之間的連線。
- 將許可新增至 Livy 服務帳戶以從中擷取秘密 AWS Secrets Manager – Livy 伺服器需要這些許可才能從 ASCP 擷取秘密，並新增 Livy 組態以保護 Livy 伺服器。若要將 IAM 許可新增至服務帳戶，請參閱使用服務帳戶 (IRSA) 的 IAM 角色設定存取許可。

### 使用 的金鑰和金鑰存放區密碼設定 JKS 憑證 AWS Secrets Manager

請依照下列步驟，使用金鑰和金鑰存放區密碼來設定 JKS 憑證。

1. 產生 Livy 伺服器的金鑰存放區檔案。

```
keytool -genkey -alias <host> -keyalg RSA -keysize 2048 -dname
 CN=<host>,OU=hw,O=hw,L=<your_location>,ST=<state>,C=<country> -
keypass <keyPassword> -keystore <keystore_file> -storepass <storePassword> --
validity 3650
```

2. 建立憑證。

```
keytool -export -alias <host> -keystore mykeystore.jks -rfc -
file mycertificate.cert -storepass <storePassword>
```

3. 建立信任存放區檔案。

```
keytool -import -noprompt -alias <host>-file <cert_file> -
keystore <truststore_file> -storepass <truststorePassword>
```

4. 將 JKS 憑證儲存在其中 AWS Secrets Manager。將 取代`livy-jks-secret`為您的秘密，並將`fileb://mykeystore.jks`取代為金鑰存放區 JKS 憑證的路徑。

```
aws secretsmanager create-secret \
--name livy-jks-secret \
--description "My Livy keystore JKS secret" \
--secret-binary fileb://mykeystore.jks
```

5. 在 Secrets Manager 中儲存金鑰存放區和金鑰密碼。請務必使用您自己的參數。

```
aws secretsmanager create-secret \
--name livy-jks-secret \
--description "My Livy key and keystore password secret" \
--secret-string "{\"keyPassword\":\"<test-key-password>\",\"keyStorePassword\":
\"<test-key-store-password>\"}"
```

6. 使用下列命令建立 Livy 伺服器命名空間。

```
kubectl create ns <livy-ns>
```

7. 為具有 JKS 憑證和密碼的 Livy 伺服器建立 `ServiceProviderClass` 物件。

```
cat >livy-secret-provider-class.yaml << EOF
apiVersion: secrets-store.csi.x-k8s.io/v1
kind: SecretProviderClass
metadata:
  name: aws-secrets
spec:
  provider: aws
  parameters:
    objects: |
        - objectName: "livy-jks-secret"
          objectType: "secretsmanager"
        - objectName: "livy-passwords"
          objectType: "secretsmanager"

EOF
kubectl apply -f livy-secret-provider-class.yaml -n <livy-ns>
```

啟用 SSL 的 Apache Livy 入門

在 Livy 伺服器上啟用 SSL 之後，您必須設定 `serviceAccount` 才能存取 `keyStore`和 `keyPasswords`秘密 AWS Secrets Manager。

1. 建立 Livy 伺服器命名空間。

```
kubectl create namespace <livy-ns>
```

2. 設定 Livy 服務帳戶以存取 Secrets Manager 中的秘密。如需設定 IRSA 的詳細資訊，請參閱在安裝 Apache Livy 時設定 IRSA。

```
aws ecr get-login-password \--region region-id | helm registry login \
--username AWS \
--password-stdin ECR-registry-account.dkr.ecr.region-id.amazonaws.com
```

3. 安裝 Livy。對於 Helm Chart --version 參數，請使用您的 Amazon EMR 發行標籤，例如 `7.1.0`。您也必須將 Amazon ECR 登錄帳戶 ID 和區域 ID 取代為您自己的 IDs。您可以從 AWS 區域 Amazon ECR 登錄帳戶依區域尋找 的對應`ECR-registry-account`值。

```
helm install <livy-app-name> \
  oci://895885662937.dkr.ecr.region-id.amazonaws.com/livy \
  --version 7.9.0 \
  --namespace livy-namespace-name \
  --set image=<ECR-registry-account.dkr.ecr>.<region>.amazonaws.com/livy/
emr-7.9.0:latest \
  --set sparkNamespace=spark-namespace \
  --set ssl.enabled=true
  --set ssl.CertificateArn=livy-acm-certificate-arn
  --set ssl.secretProviderClassName=aws-secrets
  --set ssl.keyStoreObjectName=livy-jks-secret
  --set ssl.keyPasswordsObjectName=livy-passwords
  --create-namespace
```

4. 從在 Amazon EMR on EKS 上安裝 Apache Livy 的步驟 5 繼續。

## 使用角色型存取控制 (RBAC) 設定 Apache Livy 和 Spark 應用程式許可

若要部署 Livy，Amazon EMR on EKS 會建立伺服器服務帳戶和角色，以及 Spark 服務帳戶和角色。這些角色必須具備必要的 RBAC 許可，才能完成設定並執行 Spark 應用程式。

伺服器服務帳戶和角色的 RBAC 許可

Amazon EMR on EKS 會建立 Livy 伺服器服務帳戶和角色，以管理 Spark 任務的 Livy 工作階段，以及將流量路由到傳入和其他資源或從中路由流量。

此服務帳戶的預設名稱為 `emr-containers-sa-livy`。它必須具有下列許可。

```
rules:
- apiGroups:
  - ""
  resources:
  - "namespaces"
  verbs:
  - "get"
- apiGroups:
  - ""
  resources:
  - "serviceaccounts"
    "services"
    "configmaps"
    "events"
    "pods"
    "pods/log"
  verbs:
  - "get"
    "list"
    "watch"
    "describe"
    "create"
    "edit"
    "delete"
    "deletecollection"
    "annotate"
    "patch"
    "label"
 - apiGroups:
   - ""
   resources:
   - "secrets"
   verbs:
   - "create"
     "patch"
     "delete"
     "watch"
```

```
 - apiGroups:
   - ""
   resources:
   - "persistentvolumeclaims"
   verbs:
   - "get"
     "list"
     "watch"
     "describe"
     "create"
     "edit"
     "delete"
     "annotate"
     "patch"
     "label"
```

### Spark 服務帳戶和角色的 RBAC 許可

Spark 驅動程式 Pod 需要與該 Pod 位於相同命名空間的 Kubernetes 服務帳戶。此服務帳戶需要管理執行器 Pod 和驅動程式 Pod 所需任何資源的許可。除非命名空間中的預設服務帳戶具有所需的許可，否則驅動程式會失敗並結束。需要下列 RBAC 許可。

```
rules:
- apiGroups:
  - ""
    "batch"
    "extensions"
    "apps"
  resources:
  - "configmaps"
    "serviceaccounts"
    "events"
    "pods"
    "pods/exec"
    "pods/log"
    "pods/portforward"
    "secrets"
    "services"
    "persistentvolumeclaims"
    "statefulsets"
  verbs:
  - "create"
    "delete"
```

```
    "get"
    "list"
    "patch"
    "update"
    "watch"
    "describe"
    "edit"
    "deletecollection"
    "patch"
    "label"
```

## 使用服務帳戶的 IAM 角色設定存取許可 (IRSA)

根據預設，Livy 伺服器和 Spark 應用程式的驅動程式和執行程式無法存取 AWS 資源。伺服器服務帳戶和 Spark 服務帳戶控制對 Livy 伺服器和 Spark 應用程式 Pod AWS 資源的存取。若要授予存取權，您需要將服務帳戶對應至具有必要 AWS 許可的 IAM 角色。

您可以在安裝 Apache Livy 之前、安裝期間或完成安裝之後設定 IRSA 映射。

在安裝 Apache Livy 時設定 IRSA（適用於伺服器服務帳戶）

> **Note**
>
> 只有伺服器服務帳戶才支援此映射。

1. 請確定您已完成為 Amazon EMR on EKS 設定 Apache Livy，並且正在搭配 Amazon EMR on EKS 安裝 Apache Livy。
2. 為 Livy 伺服器建立 Kubernetes 命名空間。在此範例中，命名空間的名稱為 `livy-ns`。
3. 建立 IAM 政策，其中包含您希望 Pod 存取 AWS 服務 的 許可。下列範例會建立 IAM 政策，以取得 Spark 進入點的 Amazon S3 資源。

```
cat >my-policy.json <<EOF{
"Version": "2012-10-17",
    "Statement": [
        {
"Effect": "Allow",
            "Action": "s3:GetObject",
            "Resource": "arn:aws:s3:::my-spark-entrypoint-bucket"
        }
```

```
    ]
}
EOF

aws iam create-policy --policy-name my-policy --policy-document file://my-
policy.json
```

4. 使用下列命令將您的 AWS 帳戶 ID 設定為變數。

```
account_id=$(aws sts get-caller-identity --query "Account" --output text)
```

5. 將叢集的 OpenID Connect (OIDC) 身分提供者設定為 環境變數。

```
oidc_provider=$(aws eks describe-cluster --name my-cluster --region $AWS_REGION --
query "cluster.identity.oidc.issuer" --output text | sed -e "s/^https:\/\///")
```

6. 設定命名空間和服務帳戶名稱的變數。請務必使用您自己的值。

```
export namespace=default
export service_account=my-service-account
```

7. 使用下列命令建立信任政策檔案。如果您想要將角色的存取權授予命名空間中的所有服務帳戶，請複製下列命令，並將 取代`StringEquals`為 `StringLike`，並將 取代`$service_account`為 `*`。

```
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
```

8. 建立角色。

```
aws iam create-role --role-name my-role --assume-role-policy-document file://trust-
relationship.json --description "my-role-description"
```

9. 使用下列 Helm 安裝命令，將 設定為`serviceAccount.executionRoleArn`映射 IRSA。以下是 Helm 安裝命令的範例。您可以從 Amazon ECR AWS 區域 登錄帳戶依區域尋找 的對應`ECR-registry-account`值。 https://docs.aws.amazon.com/emr/latest/EMR-on-EKS-DevelopmentGuide/docker-custom-images-tag.html#docker-custom-images-ECR

```
helm install livy-demo \
  oci://895885662937.dkr.ecr.us-west-2.amazonaws.com/livy \
  --version 7.9.0 \
  --namespace livy-ns \
  --set image=ECR-registry-account.dkr.ecr.region-id.amazonaws.com/livy/
emr-7.9.0:latest \
  --set sparkNamespace=spark-ns \
  --set serviceAccount.executionRoleArn=arn:aws:iam::123456789012:role/my-role
```

將 IRSA 映射至 Spark 服務帳戶

將 IRSA 映射至 Spark 服務帳戶之前，請確定您已完成下列項目：

- 請確定您已完成為 Amazon EMR on EKS 設定 Apache Livy，並且正在搭配 Amazon EMR on EKS 安裝 Apache Livy。
- 您必須為叢集擁有現有的 IAM OpenID Connect (OIDC) 提供者。若要查看您是否已有 或如何建立，請參閱為您的叢集建立 IAM OIDC 提供者。
- 請確定您已安裝 或 的 `eksctl` CLI 版本 0.171.0 或更新版本 AWS CloudShell。若要安裝或更新 `eksctl`，請參閱安裝 `eksctl` 文件。

請依照下列步驟將 IRSA 映射到您的 Spark 服務帳戶：

1. 使用下列命令來取得 Spark 服務帳戶。

```
SPARK_NAMESPACE=<spark-ns>
LIVY_APP_NAME=<livy-app-name>
```

```
kubectl --namespace $SPARK_NAMESPACE describe sa -l "app.kubernetes.io/instance=
$LIVY_APP_NAME" | awk '/^Name:/ {print $2}'
```

2. 為服務帳戶的命名空間和名稱設定變數。

```
export namespace=default
export service_account=my-service-account
```

3. 使用下列命令為 IAM 角色建立信任政策檔案。下列範例授予 命名空間內所有服務帳戶的許可，以使用 角色。若要這樣做，請將 取代`StringEquals`為 `StringLike`，並將 取代`$service_account`為 *。

```
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
```

4. 建立角色。

```
aws iam create-role --role-name my-role --assume-role-policy-document file://trust-
relationship.json --description "my-role-description"
```

5. 使用下列`eksctl`命令對應伺服器或 Spark 服務帳戶。請務必使用您自己的值。

```
 eksctl create iamserviceaccount --name spark-sa \
 --namespace spark-namespace --cluster livy-eks-cluster \
 --attach-role-arn arn:aws:iam::0123456789012:role/my-role \
```

```
--approve --override-existing-serviceaccounts
```

# Amazon EMR on EKS 版本上 Apache Livy 的安裝屬性

Apache Livy 安裝可讓您選取 Livy Helm Chart 的版本。Helm Chart 提供各種屬性，可自訂您的安裝和設定體驗。Amazon EMR on EKS 7.1.0 版及更新版本支援這些屬性。

主題

- Amazon EMR 7.1.0 安裝屬性

## Amazon EMR 7.1.0 安裝屬性

下表說明所有支援的 Livy 屬性。安裝 Apache Livy 時，您可以選擇 Livy Helm Chart 版本。若要在安裝期間設定屬性，請使用命令 `--set <property>=<value>`。

| 屬性 | 描述 | 預設 |
| --- | --- | --- |
| image | Livy 伺服器的 Amazon EMR 版本 URI。這是必要的組態。 | "" |
| sparkNamespace | 執行 Livy Spark 工作階段的命名空間。例如，指定「livy」。這是必要的組態。 | "" |
| nameOverride | 提供名稱而非 `livy`。名稱會設定為所有 Livy 資源的標籤 | "livy" |
| fullnameOverride | 提供要使用的名稱，而非資源的完整名稱。 | "" |
| ssl.enabled | 啟用從 Livy 端點到 Livy 伺服器的end-to-end SSL。 | FALSE |
| ssl.certificateArn | 如果啟用 SSL，這是服務所建立 NLB 的 ACM 憑證 ARN。 | "" |

| 屬性 | 描述 | 預設 |
| --- | --- | --- |
| ssl.secretProviderClassName | 如果啟用 SSL，這是使用 SSL 保護 Livy 伺服器連線 NLB 的秘密提供者類別名稱。 | "" |
| ssl.keyStoreObjectName | 如果已啟用 SSL，則秘密提供者類別中金鑰存放區憑證的物件名稱。 | "" |
| ssl.keyPasswordsObjectName | 如果已啟用 SSL，則具有金鑰存放區和金鑰密碼之秘密的物件名稱。 | "" |
| rbac.create | 如果為 true， 會建立 RBAC 資源。 | FALSE |
| serviceAccount.create | 如果為 true， 會建立 Livy 服務帳戶。 | TRUE |
| serviceAccount.name | 用於 Livy 的服務帳戶名稱。如果您未設定此屬性並建立服務帳戶，Amazon EMR on EKS 會使用`fullname`覆寫屬性自動產生名稱。 | 「emr-containers-sa-livy」 |
| serviceAccount.executionRoleArn | Livy 服務帳戶的執行角色 ARN。 | "" |
| sparkServiceAccount.create | 如果為 true， 會在 中建立 Spark 服務帳戶 `.Release.Namespace` | TRUE |

| 屬性 | 描述 | 預設 |
| --- | --- | --- |
| sparkServiceAccount.name | 用於 Spark 的服務帳戶名稱。如果您未設定此屬性並建立 Spark 服務帳戶，則 Amazon EMR on EKS `-spark-livy` 會自動產生名稱，其中包含帶有尾碼的 `fullnameOverride` 屬性。 | 「emr-containers-sa-spark-livy」 |
| service.name | Livy 服務的名稱 | `"emr-containers-livy"` |
| service.annotations | Livy 服務註釋 | `{}` |
| loadbalancer.enabled | 是否要為用於在 Amazon EKS 叢集外部公開 Livy 端點的 Livy 服務建立負載平衡器。 | `FALSE` |
| loadbalancer.internal | 是否要將 Livy 端點設定為 VPC 內部或外部。<br><br>將此屬性設定為向 VPC 外部的來源`FALSE`公開端點。我們建議您使用 TLS/SSL 保護您的端點。如需詳細資訊，請參閱設定 TLS 和 SSL 加密。 | `FALSE` |
| imagePullSecrets | 用於從私有儲存庫提取 Livy 映像`imagePullSecret` 的名稱清單。 | [] |
| resources | Livy 容器的資源請求和限制。 | {} |
| nodeSelector | 排程 Livy Pod 的節點。 | {} |
| 容差 | 包含要定義的 Livy Pod 容錯的清單。 | [] |

| 屬性 | 描述 | 預設 |
| --- | --- | --- |
| 親和性 | Livy Pod 親和性規則。 | {} |
| persistence.enabled | 如果為 true， 會啟用工作階段目錄的持久性。 | FALSE |
| persistence.subPath | 掛載至工作階段目錄的 PVC 子路徑。 | "" |
| persistence.existingClaim | 要使用的 PVC，而不是建立新的 PVC。 | {} |
| persistence.storageClass | 要使用的儲存體方案。若要定義此參數，請使用 格式`storageClassName: <storageClass>` 。將此參數設定為 會`"-"`停用動態佈建。如果您將此參數設定為 null 或未指定任何項目，Amazon EMR on EKS 不會設定 storageClassName 並使用預設佈建器。 | "" |
| persistence.accessMode | PVC 存取模式。 | `ReadWriteOnce` |
| persistence.size | PVC 大小。 | 20Gi |
| persistence.annotations | PVC 的其他註釋。 | {} |
| env.* | 要設定為 Livy 容器的其他 env。如需詳細資訊，請參閱在安裝 Livy 時輸入您自己的 Livy 和 Spark 組態。 | {} |
| envFrom.* | 從 Kubernetes 組態映射或秘密設定為 Livy 的其他 env。 | [] |

| 屬性 | 描述 | 預設 |
| --- | --- | --- |
| livyConf.* | 從掛載的 Kubernetes 組態映射或秘密設定的其他 livy.conf 項目。 | {} |
| sparkDefaultsConf.* | 從掛載的 Kubernetes 組態映射或秘密設定的其他`spark-defaults.conf` 項目。 | {} |

## 疑難排解常見的環境變數格式錯誤

當您輸入 Livy 和 Spark 組態時，有不支援的環境變數格式，並可能導致錯誤。此程序會引導您完成一系列步驟，以協助確保您使用正確的格式。

在安裝 Livy 時輸入您自己的 Livy 和 Spark 組態

您可以使用 `env.*` Helm 屬性設定任何 Apache Livy 或 Apache Spark 環境變數。請依照下列步驟，將範例組態轉換為`example.config.with-dash.withUppercase`支援的環境變數格式。

1. 使用 1 和小寫字母取代大寫字母。例如，`example.config.with-dash.withUppercase` 會變成 `example.config.with-dash.with1uppercase`。
2. 將破折號 (-) 取代為 0。例如， `example.config.with-dash.with1uppercase`會變成 `example.config.with0dash.with1uppercase`
3. 以底線 (_) 取代點 (.)。例如，`example.config.with0dash.with1uppercase` 會變成 `example_config_with0dash_with1uppercase`。
4. 以大寫字母取代所有小寫字母。
5. 將 字首`LIVY_`新增至變數名稱。
6. 使用格式 --set env.*YOUR_VARIABLE_NAME*.value=*yourvalue* 透過 Helm Chart 安裝 Livy 時，請使用 變數

例如，若要設定 Livy 和 Spark 組態 `livy.server.recovery.state-store = filesystem` 和 `spark.kubernetes.executor.podNamePrefix = my-prefix`，請使用下列 Helm 屬性：

```
–set env.LIVY_LIVY_SERVER_RECOVERY_STATE0STORE.value=filesystem
–set env.LIVY_SPARK_KUBERNETES_EXECUTOR_POD0NAME0PREFIX.value=myprefix
```

# 管理 Amazon EMR on EKS 作業執行

以下各章節涵蓋可協助您管理 Amazon EMR on EKS 作業執行的主題。這包括在使用 時設定任務執行參數 AWS CLI、設定日誌資料的儲存方式、執行 Spark SQL 指令碼來執行查詢、了解任務執行狀態，以及了解如何監控任務。如果您想要設定並完成任務執行來處理資料，通常可以依序處理這些主題。

主題


- 使用 管理任務執行 AWS CLI
- 透過 StartJobRun API 執行 Spark SQL 指令碼
- 作業執行狀態
- 在 Amazon EMR 主控台中檢視作業
- 執行作業時的常見錯誤


## 使用 管理任務執行 AWS CLI

本主題說明如何使用 AWS Command Line Interface () 管理任務執行AWS CLI。它詳細介紹了屬性，例如安全參數、驅動程式和各種覆寫設定。它還包含子主題，涵蓋各種設定記錄的方式。

主題


- 用於設定作業執行的選項
- 設定作業執行以使用 Amazon S3 日誌
- 設定作業執行以使用 Amazon CloudWatch Logs
- 停止作業執行
- 描述作業執行
- 取消作業執行


### 用於設定作業執行的選項

使用下列選項來設定作業執行參數：

- `--execution-role-arn`：必須提供用於執行作業的 IAM 角色。如需詳細資訊，請參閱搭配使用作業執行角色與 Amazon EMR on EKS。
- `--release-label`：可以使用 Amazon EMR 5.32.0 和 6.2.0 及更高版本來部署 Amazon EMR on EKS。舊版 Amazon EMR 不支援 Amazon EMR on EKS。如需詳細資訊，請參閱Amazon EMR on EKS 發行版本。

- `--job-driver`：作業驅動程式用於提供主要作業的輸入。這是一個聯合類型欄位，只能在其中傳遞您要執行之作業類型的其中一個值。支援的作業類型包括：
  - Spark 提交作業 - 用於透過 Spark 提交來執行命令。可以使用此作業類型，透過 Spark Submit 來執行 Scala、PySpark、SparkR、SparkSQL 以及其他支援的作業。此作業類型具有下列參數：
    - Entrypoint - 這是對您要執行的主要 jar/py 檔案的 HCFS (Hadoop 相容檔案系統) 參考。
    - EntryPointArguments - 這是您要傳遞給主要 jar/py 檔案的引數陣列。應使用 entrypoint 程式碼讀取這些參數。陣列中的每個引數應該以逗號分隔。EntryPointArguments 不能包含括號或圓括號，例如 ()、{} 或 []。
    - SparkSubmitParameters - 這些是您要傳送到作業的其他 spark 參數。使用此參數可覆寫預設 Spark 屬性，例如驅動程式記憶體或 -conf 或 -class 等執行程式的數量。如需其他資訊，請參閱透過 spark-submit 啟動應用程式。
  - Spark SQL 作業 - 用於透過 Spark SQL 執行 SQL 查詢檔案。可以使用此作業類型來執行 SparkSQL 作業。此作業類型具有下列參數：
    - Entrypoint - 這是對您要執行的 SQL 查詢檔案的 HCFS (Hadoop 相容檔案系統) 參考。

      如需可用於 Spark SQL 作業的其他 Spark 參數清單，請參閱 透過 StartJobRun API 執行 Spark SQL 指令碼。
- `--configuration-overrides`：可以透過提供組態物件來覆寫應用程式的預設組態。您可以使用速記語法，以提供組態或參考 JSON 檔案中物件的組態。組態物件是由分類、屬性和選用的巢狀組態所組成。屬性由您想要在檔案中覆寫的設定組成。您可以在單一 JSON 物件中，為多個應用程式指定多個分類。可用的組態分類隨 Amazon EMR 發行版本而有所不同。如需每個 Amazon EMR 發行版本可用的組態分類清單，請參閱 Amazon EMR on EKS 發行版本。

  如果在應用程式中覆寫和 Spark 提交參數中傳遞相同的組態，會優先採用 Spark 提交參數。完整的組態優先順序清單如下，以最高優先順序到最低優先順序排列。

  - 建立 `SparkSession` 時提供的組態。
  - 使用 `–conf`，作為 `sparkSubmitParameters` 的一部分提供的組態。
  - 作為應用程式覆寫的一部分提供的組態。
  - 由 Amazon EMR 針對發行版本選擇的優化組態。
  - 應用程式的預設開放原始碼組態。

  若要使用 Amazon CloudWatch 或 Amazon S3 監控作業執行，必須提供 CloudWatch 的組態詳細資訊。如需詳細資訊，請參閱設定作業執行以使用 Amazon S3 日誌及設定作業執行以使用 Amazon CloudWatch Logs。如果 S3 儲存貯體或 CloudWatch 日誌群組不存在，則 Amazon EMR 會在將日誌上傳到儲存貯體之前先建立。

- 如需 Kubernetes 組態選項的其他清單，請參閱 Kubernetes 上的 Spark 屬性。

  不支援以下 Spark 組態。

  - `spark.kubernetes.authenticate.driver.serviceAccountName`
  - `spark.kubernetes.authenticate.executor.serviceAccountName`
  - `spark.kubernetes.namespace`
  - `spark.kubernetes.driver.pod.name`
  - `spark.kubernetes.container.image.pullPolicy`
  - `spark.kubernetes.container.image`

  > **Note**
  >
  > 可以將 `spark.kubernetes.container.image` 用於自訂 Docker 映像檔。如需詳細資訊，請參閱自訂 Amazon EMR on EKS 的 Docker 映像檔。

## 設定作業執行以使用 Amazon S3 日誌

為了能夠監控作業進度並對失敗進行疑難排解，必須設定作業，以便將日誌資訊傳送到 Amazon S3、Amazon CloudWatch Logs 或兩者。本主題可協助您開始在透過 Amazon EMR on EKS 啟動的作業上將應用程式日誌發布到 Amazon S3。

S3 日誌 IAM 政策

在您的作業可以傳送日誌資料到 Amazon S3 之前，必須在作業執行角色的許可政策中包含下列許可。將 *`amzn-s3-demo-logging-bucket`* 取代為您的記錄儲存貯體名稱。

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:PutObject",
                "s3:GetObject",
                "s3:ListBucket"
            ],
```

```
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-logging-bucket",
                "arn:aws:s3:::amzn-s3-demo-logging-bucket/*"
            ]
        }
    ]
}
```

> **Note**
>
> Amazon EMR on EKS 也可以建立 Amazon S3 儲存貯體。如果無法使用 Amazon S3 儲存貯體，請在 IAM 政策中包含 “`s3:CreateBucket`” 許可。

在授予執行角色適當許可以便將日誌傳送到 Amazon S3 之後，當在 `start-job-run` 請求的 `monitoringConfiguration` 區段中傳遞 `s3MonitoringConfiguration` 時，會將日誌資料傳送到以下 Amazon S3 位置，如 使用 管理任務執行 AWS CLI 中所示。

- 提交者日誌 - /*logUri*/*virtual-cluster-id*/jobs/*job-id*/containers/*pod-name*/(stderr.gz/stdout.gz)
- 驅動程式日誌 - /*logUri*/*virtual-cluster-id*/jobs/*job-id*/containers/*spark-application-id*/spark-*job-id*-driver/(stderr.gz/stdout.gz)
- 執行程式日誌 - /*logUri*/*virtual-cluster-id*/jobs/*job-id*/containers/*spark-application-id*/*executor-pod-name*/(stderr.gz/stdout.gz)

## 設定作業執行以使用 Amazon CloudWatch Logs

若要監控作業進度並對失敗進行疑難排解，必須設定作業，以便將日誌資訊傳送到 Amazon S3、Amazon CloudWatch Logs 或兩者。本主題可協助您開始在透過 Amazon EMR on EKS 啟動的作業上使用 CloudWatch Logs。如需有關 CloudWatch Logs 的詳細資訊，請參閱《Amazon CloudWatch 使用者指南》中的監控日誌檔案。

### CloudWatch Logs IAM 政策

為了讓作業將日誌資料傳送到 CloudWatch Logs，必須在作業執行角色的許可政策中包含下列許可。將 *my_log_group_name* 和 *my_log_stream_prefix* 分別取代為 CloudWatch 日誌群組名稱和日誌串流名稱。如果日誌群組和日誌串流不存在，只要執行角色 ARN 具有適當的許可，Amazon EMR 就會建立它們。

JSON

```
{
     "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogStream",
                "logs:DescribeLogGroups",
                "logs:DescribeLogStreams"
            ],
            "Resource": [
                "arn:aws:logs:*:*:*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "logs:PutLogEvents"
            ],
            "Resource": [
                "arn:aws:logs:*:*:log-group:my_log_group_name:log-
stream:my_log_stream_prefix/*"
            ]
        }
    ]
}
```

> **Note**
>
> Amazon EMR on EKS 也可以建立日誌串流。如果日誌串流不存在，IAM 政策應包含 `"logs:CreateLogGroup"` 許可。

在為執行角色提供適當的許可之後，當在 `start-job-run` 請求的 `monitoringConfiguration` 區段中傳遞 `cloudWatchMonitoringConfiguration` 時，應用程式會將其日誌資料傳送至 CloudWatch Logs，如 使用 管理任務執行 AWS CLI 中所示。

在 `StartJobRun` API 中，*`log_group_name`* 是 CloudWatch 的日誌群組名稱，而 *`log_stream_prefix`* 是 CloudWatch 的日誌串流名稱字首。您可以在 AWS Management Console 中檢視及搜尋這些日誌。

- 提交者日誌 - *`logGroup`*/*`logStreamPrefix`*/*`virtual-cluster-id`*/jobs/*`job-id`*/containers/*`pod-name`*/(stderr/stdout)
- 驅動程式日誌 - *`logGroup`*/*`logStreamPrefix`*/*`virtual-cluster-id`*/jobs/*`job-id`*/containers/*`spark-application-id`*/spark-*`job-id`*-driver/(stderrstdout)
- 執行程式日誌 - *`logGroup`*/*`logStreamPrefix`*/*`virtual-cluster-id`*/jobs/*`job-id`*/containers/*`spark-application-id`*/*`executor-pod-name`*/(stderr/stdout)

### 停止作業執行

可以執行 `list-job-run` 以顯示作業執行的狀態，如下列範例所示。

```
aws emr-containers list-job-runs --virtual-cluster-id <cluster-id>
```

### 描述作業執行

可以執行 `describe-job-run` 以取得有關作業的詳細資訊，例如作業狀態、狀態詳細資料和作業名稱，如下列範例所示。

```
aws emr-containers describe-job-run --virtual-cluster-id cluster-id --id job-run-id
```

### 取消作業執行

可以執行 `cancel-job-run` 以取消執行中的作業，如下列範例所示。

```
aws emr-containers cancel-job-run --virtual-cluster-id cluster-id --id job-run-id
```

## 透過 StartJobRun API 執行 Spark SQL 指令碼

Amazon EMR on EKS 6.7.0 版及更高版本包含 Spark SQL 作業驅動程式，以便可以透過 `StartJobRun` API 來執行 Spark SQL 指令碼。您可以提供 SQL 進入點檔案，以使用 `StartJobRun` API 在 Amazon EMR on EKS 上執行 Spark SQL 查詢，而無需對現有的 Spark SQL 指令碼進行任何修改。下表列出了 Spark SQL 作業透過 StartJobRun API 支援的 Spark 參數。

可以從下面的 Spark 參數中選擇，以傳送到 Spark SQL 作業。使用這些參數來覆寫預設的 Spark 屬性。

| 選項 | 描述 |
| --- | --- |
| --name NAME | Application Name (應用程式名稱) |
| --jars JARS | 驅動程式和執行程式 classpath 隨附的以逗號分隔的 jar 清單。 |
| --packages | 驅動程式和執行程式 classpath 中包含以逗號分隔的 jar 的 maven 座標清單。 |
| --exclude-packages | 以逗號分隔的 groupId:artifactId 清單，在解決 –packages 中提供的相依性時排除，以避免相依性衝突。 |
| --repositories | 以逗號分隔的其他遠端儲存庫清單，用於搜尋 –packages 提供的 maven 座標。 |
| --files FILES | 要放置在每個執行程式的工作目錄中的以逗號分隔的檔案清單。 |
| --conf PROP=VALUE | Spark 組態屬性。 |
| --properties-file FILE | 要從中載入額外屬性的檔案路徑。 |
| --driver-memory MEM | 驅動程式的記憶體。預設值為 1024MB。 |
| --driver-java-options | 用於傳遞給驅動程式的額外 Java 選項。 |
| --driver-library-path | 用於傳遞給驅動程式的額外程式庫路徑條目。 |
| --driver-class-path | 用於傳遞給驅動程式的額外 classpath 條目。 |
| --executor-memory MEM | 每個執行程式的記憶體。預設值為 1 GB。 |
| --driver-cores NUM | 驅動程式使用的核心數目。 |
| --total-executor-cores NUM | 所有執行程式的核心總數。 |

| 選項 | 描述 |
| --- | --- |
| --executor-cores NUM | 每個執行程式使用的核心數量。 |
| --num-executors NUM | 要啟動的執行程式數量。 |
| -hivevar <key=value> | 套用於 Hive 命令的變數替換，例如，`-hivevar A=B` |
| -hiveconf <property=value> | 用於指定屬性的值。 |

若為 Spark SQL 作業，請建立 start-job-run-request.json 檔案，並指定作業執行所需的參數，如下列範例所示：

```
{
  "name": "myjob",
  "virtualClusterId": "123456",
  "executionRoleArn": "iam_role_name_for_job_execution",
  "releaseLabel": "emr-6.7.0-latest",
  "jobDriver": {
    "sparkSqlJobDriver": {
      "entryPoint": "entryPoint_location",
       "sparkSqlParameters": "--conf spark.executor.instances=2 --conf
 spark.executor.memory=2G --conf spark.executor.cores=2 --conf spark.driver.cores=1"
    }
  },
  "configurationOverrides": {
    "applicationConfiguration": [
      {
        "classification": "spark-defaults",
        "properties": {
          "spark.driver.memory":"2G"
         }
      }
    ],
    "monitoringConfiguration": {
      "persistentAppUI": "ENABLED",
      "cloudWatchMonitoringConfiguration": {
        "logGroupName": "my_log_group",
        "logStreamNamePrefix": "log_stream_prefix"
      },
```

```
      "s3MonitoringConfiguration": {
        "logUri": "s3://my_s3_log_location"
      }
    }
  }
}
```

## 作業執行狀態

將作業執行提交到 Amazon EMR on EKS 作業佇列時，作業執行會進入 PENDING 狀態。工作將經過以下狀態，直到其失敗 (以 0 代碼結束) 或失敗 (以與非零代碼結束) 為止。

作業執行可能有以下狀態：

- PENDING – 將作業執行提交至 Amazon EMR on EKS 時的初始作業狀態。作業正在等待提交至虛擬叢集，而 Amazon EMR on EKS 正在提交此作業。
- SUBMITTED – 已成功提交至虛擬叢集的作業執行。然後，叢集排程器會嘗試在叢集上執行此作業。
- RUNNING – 在虛擬叢集中執行的作業執行。在 Spark 應用程式中，這意味著 Spark 驅動程式進程處於 running 狀態。
- FAILED – 無法提交至虛擬叢集或未成功完成的作業執行。查看 StateDetails 和 FailureReason，以尋找有關此作業失敗的其他資訊。
- COMPLETED – 已成功完成的作業執行。
- CANCEL_PENDING – 已請求取消的作業執行。Amazon EMR on EKS 正在嘗試取消虛擬叢集上的作業。
- CANCELLED – 已成功取消的作業執行。

## 在 Amazon EMR 主控台中檢視作業

任務執行資料可供檢視，因此您可以在每個任務通過狀態時進行監控。若要在 Amazon EMR 主控台中檢視作業，請執行以下步驟。

1. 在 Amazon EMR 主控台左側功能表中，在 Amazon EMR on EKS 下，選擇虛擬叢集。
2. 從虛擬叢集清單中，選取要檢視其作業的虛擬叢集。
3. 在作業執行資料表中，選取檢視日誌以檢視作業執行的詳細資訊。

> **Note**
>
> 預設啟用對單鍵體驗的支援 在作業提交期間，在 `monitoringConfiguration` 中將 `persistentAppUI` 設定為 `DISABLED` 可關閉此功能。如需詳細資訊，請參閱檢視持續應用程式使用者界面。

## 執行作業時的常見錯誤

執行 `StartJobRun` API 時，可能會發生下列錯誤。資料表列出每個錯誤並提供緩解步驟，讓您可以快速解決問題。

| 錯誤訊息 | 錯誤情況 | 建議的後續步驟 |
| --- | --- | --- |
| error: argument --*argument* is required | 缺少必要參數。 | 將缺少的引數新增到 API 請求。 |
| 呼叫 StartJobRun 操作時發生錯誤 (AccessDeniedException)：User: *ARN* is not authorized to perform: emr-containers:StartJobRun | 缺少執行角色。 | 請參閱「使用 搭配使用作業執行角色與 Amazon EMR on EKS」。 |
| 呼叫 StartJobRun 操作時發生錯誤 (AccessDeniedException)：User: *ARN* is not authorized to perform: emr-containers:StartJobRun | 呼叫者沒有透過條件金鑰存取執行角色的許可 [有效/無效格式]。 | 請參閱 搭配使用作業執行角色與 Amazon EMR on EKS。 |
| 呼叫 StartJobRun 操作時發生錯誤 (AccessDeniedException)：User: *ARN* is not authorized to perform: emr-containers:StartJobRun | 作業提交者和執行角色 ARN 來自不同的帳戶。 | 確定作業提交者和執行角色 ARN 來自相同的 AWS 帳戶。 |
| 偵測到 1 個驗證錯誤：'executionRoleArn' 的*##*值無法滿足 ARN 規則表達式模式： | 呼叫者透過條件金鑰擁有執行角色的許可，但角色不符合 ARN 格式的限制。 | 提供遵循 ARN 格式的執行角色。請參閱 搭配使用作業執 |

| 錯誤訊息 | 錯誤情況 | 建議的後續步驟 |
| --- | --- | --- |
| ^arn:(aws[a-zA-Z0-9-]*):iam::(\d{12})?:(role((\u002F)\|(\u002F[\u0021-\u007F]+\u002F))[\w+=,.@-]+) | | 行角色與 Amazon EMR on EKS。 |
| 呼叫 StartJobRun 操作時發生錯誤 (ResourceNotFoundException)：虛擬叢集 *Virtual Cluster ID* 不存在。 | 找不到虛擬叢集 ID。 | 提供向 Amazon EMR on EKS 註冊的虛擬叢集 ID。 |
| 呼叫 StartJobRun 操作時發生錯誤 (ValidationException)：虛擬叢集 *state* 無效，無法建立資源 JobRun。 | 虛擬叢集尚未準備好執行作業。 | 請參閱 虛擬叢集狀態。 |
| 呼叫 StartJobRun 操作時發生錯誤 (ResourceNotFoundException)：版本 *RELEASE* 不存在。 | 作業提交中指定的版本不正確。 | 請參閱 Amazon EMR on EKS 發行版本。 |
| 呼叫 StartJobRun 操作時發生錯誤 (AccessDeniedException)：User: *ARN* is not authorized to perform: emr-containers:StartJobRun on resource: *ARN* with an explicit deny。<br><br>呼叫 StartJobRun 操作時發生錯誤 (AccessDeniedException)：User: *ARN* is not authorized to perform: emr-containers:StartJobRun on resource: *ARN* | 未授權使用者呼叫 StartJobRun。 | 請參閱 搭配使用作業執行角色與 Amazon EMR on EKS。 |

| 錯誤訊息 | 錯誤情況 | 建議的後續步驟 |
| --- | --- | --- |
| 呼叫 StartJobRun 操作時發生錯誤 (ValidationException)：configurationOverrides.monitoringConfiguration.s3MonitoringConfiguration.logUri failed to satisfy constraint : %s | S3 路徑 URI 語法無效。 | logUri 應採用 s3://... 的格式 |

在作業執行之前執行 `DescribeJobRun` API 時，可能會發生下列錯誤。

| 錯誤訊息 | 錯誤情況 | 建議的後續步驟 |
| --- | --- | --- |
| 狀態詳細資訊：JobRun 提交失敗。<br><br>不支援 *classification* 分類。<br><br>失敗原因：VALIDATION_ERROR<br><br>狀態：FAILED。 | StartJobRun 中的參數無效。 | 請參閱 Amazon EMR on EKS 發行版本。 |
| 狀態詳細資訊：叢集 *EKS Cluster ID* 不存在。<br><br>失敗原因：CLUSTER_UNAVAILABLE<br><br>狀態：FAILED | EKS 叢集無法使用。 | 檢查 EKS 叢集是否存在並具有正確的許可。如需詳細資訊，請參閱設定 Amazon EMR on EKS。 |
| 狀態詳細資訊：叢集 *EKS Cluster ID* 沒有足夠的許可。<br><br>失敗原因：CLUSTER_UNAVAILABLE | Amazon EMR 沒有存取 EKS 叢集的許可。 | 確認已在註冊的命名空間中為 Amazon EMR 設定許可。如需詳細資訊，請參閱設定 Amazon EMR on EKS。 |

| 錯誤訊息 | 錯誤情況 | 建議的後續步驟 |
| --- | --- | --- |
| 狀態：FAILED | | |
| 狀態詳細資訊：叢集 *EKS Cluster ID* 目前無法連線。<br><br>失敗原因：CLUSTER_UNAVAILABLE<br><br>狀態：FAILED | 無法連線到 EKS 叢集。 | 檢查 EKS 叢集是否存在並具有正確的許可。如需詳細資訊，請參閱設定 Amazon EMR on EKS。 |
| 狀態詳細資訊：由於內部錯誤，JobRun 提交失敗。<br><br>失敗原因：INTERNAL_ERROR<br><br>狀態：FAILED | EKS 叢集發生內部錯誤。 | N/A |
| 狀態詳細資訊：叢集 *EKS Cluster ID* 沒有足夠的資源。<br><br>失敗原因：USER_ERROR<br><br>狀態：FAILED | EKS 叢集中的資源不足，無法執行作業。 | 為 EKS 節點群組新增更多容量，或設定 EKS Autoscaler。如需詳細資訊，請參閱 Cluster Autoscaler。 |

在作業執行之後執行 `DescribeJobRun` API 時，可能會發生下列錯誤。

| 錯誤訊息 | 錯誤情況 | 建議的後續步驟 |
| --- | --- | --- |
| 狀態詳細資訊：監控 JobRun 時出現問題。<br><br>叢集 *EKS Cluster ID* 不存在。<br><br>失敗原因：CLUSTER_UNAVAILABLE | EKS 叢集不存在。 | 檢查 EKS 叢集是否存在並具有正確的許可。如需詳細資訊，請參閱設定 Amazon EMR on EKS。 |

| 錯誤訊息 | 錯誤情況 | 建議的後續步驟 |
| --- | --- | --- |
| 狀態：FAILED | | |
| 狀態詳細資訊：監控 JobRun 時出現問題。<br><br>叢集 *EKS Cluster ID* 沒有足夠的許可。<br><br>失敗原因：CLUSTER_UNAVAILABLE<br><br>狀態：FAILED | Amazon EMR 沒有存取 EKS 叢集的許可。 | 確認已在註冊的命名空間中為 Amazon EMR 設定許可。如需詳細資訊，請參閱設定 Amazon EMR on EKS。 |
| 狀態詳細資訊：監控 JobRun 時出現問題。<br><br>叢集 *EKS Cluster ID* 目前無法連線。<br><br>失敗原因：CLUSTER_UNAVAILABLE<br><br>狀態：FAILED | 無法連線到 EKS 叢集。 | 檢查 EKS 叢集是否存在並具有正確的許可。如需詳細資訊，請參閱設定 Amazon EMR on EKS。 |
| 狀態詳細資訊：由於內部錯誤，無法監控 JobRun<br><br>失敗原因：INTERNAL_ERROR<br><br>狀態：FAILED | 發生內部錯誤，正在阻止 JobRun 監控。 | N/A |

當作業無法啟動且作業處於 SUBMITTED 狀態 15 分鐘時，可能會發生下列錯誤。這可能是由於缺少叢集資源所致。

| 錯誤訊息 | 錯誤情況 | 建議的後續步驟 |
| --- | --- | --- |
| 叢集逾時 | 作業已處於 SUBMITTED 狀態 15 分鐘或更長時間。 | 可以使用如下所示的組態來覆寫此參數的 15 分鐘預設設定。 |

使用下列組態將叢集逾時設定變更為 30 分鐘。請注意，提供的新 `job-start-timeout` 值的單位應為秒：

```
{
"configurationOverrides": {
  "applicationConfiguration": [{
      "classification": "emr-containers-defaults",
      "properties": {
          "job-start-timeout":"1800"
      }
  }]
}
```

# 使用作業範本

作業範本會儲存在開始作業執行時可跨 `StartJobRun` API 調用來共用的值。它支援兩種使用案例：

- 防止重複出現 `StartJobRun` API 請求值。
- 強制執行必須透過 `StartJobRun` API 請求提供特定值的規則。

作業範本可讓您為作業執行定義可重複使用的範本，以套用其他自訂項目，例如：

- 設定執行程式和驅動程式運算容量
- 設定安全性和控管屬性，例如 IAM 角色
- 自訂要跨多個應用程式和資料管道使用的 Docker 映像檔

下列主題提供使用 範本的詳細資訊，包括如何使用它們來啟動任務執行，以及如何變更範本參數。

主題


- 建立並使用任務範本來啟動任務執行
- 定義作業範本參數

- 控制對作業範本的存取

## 建立並使用任務範本來啟動任務執行

本節說明建立任務範本，以及使用 範本以 AWS Command Line Interface () 開始任務執行AWS CLI。

建立作業範本

1. 建立 `create-job-template-request.json` 檔案並指定作業範本所需的參數，如下列範例 JSON 檔案所示。如需所有可用參數的資訊，請參閱 CreateJobTemplate API。

   `StartJobRun` API 所需的大多數值也是 `jobTemplateData` 所必需的。如果您想在使用作業範本調用 StartJobRun 時為任何參數使用預留位置並提供值，請參閱作業範本參數的下一節。

```
{
   "name": "mytemplate",
   "jobTemplateData": {
        "executionRoleArn": "iam_role_arn_for_job_execution",
        "releaseLabel": "emr-6.7.0-latest",
        "jobDriver": {
            "sparkSubmitJobDriver": {
                "entryPoint": "entryPoint_location",
                "entryPointArguments": [ "argument1","argument2",...],
                "sparkSubmitParameters": "--class <main_class> --conf
 spark.executor.instances=2 --conf spark.executor.memory=2G --conf
 spark.executor.cores=2 --conf spark.driver.cores=1"
            }
        },
        "configurationOverrides": {
            "applicationConfiguration": [
                {
                    "classification": "spark-defaults",
                    "properties": {
                         "spark.driver.memory":"2G"
                    }
                }
            ],
            "monitoringConfiguration": {
                "persistentAppUI": "ENABLED",
                "cloudWatchMonitoringConfiguration": {
                    "logGroupName": "my_log_group",
                    "logStreamNamePrefix": "log_stream_prefix"
```

```
                },
                "s3MonitoringConfiguration": {
                    "logUri": "s3://my_s3_log_location/"
                }
            }
        }
    }
}
```

2. 搭配使用 `create-job-template` 命令與儲存在本機的 `create-job-template-request.json` 檔案路徑。

```
aws emr-containers create-job-template \
--cli-input-json file://./create-job-template-request.json
```

使用作業範本來啟動作業執行

在 `StartJobRun` 命令中提供虛擬叢集 ID、作業範本 ID 以及作業名稱，如以下範例所示。

```
aws emr-containers start-job-run \
--virtual-cluster-id 123456 \
--name myjob \
--job-template-id 1234abcd
```

## 定義作業範本參數

作業範本參數可讓您指定作業範本中的變數。使用該作業範本啟動作業執行時，需要指定這些參數變數的值。作業範本參數會以 `${parameterName}` 格式指定。可以選擇將 `jobTemplateData` 欄位中的任何值指定為作業範本參數。針對每個作業範本參數變數，請指定其資料類型 (`STRING` 或 `NUMBER`)，並選擇性地指定預設值。以下範例顯示如何為進入點位置、主類別和 S3 日誌位置值指定作業範本參數。

將進入點位置、主類別和 Amazon S3 日誌位置指定為作業範本參數

1. 建立 `create-job-template-request.json` 檔案並指定作業範本所需的參數，如下列範例 JSON 檔案所示。如需有關參數的詳細資訊，請參閱 CreateJobTemplate API。

```
{
   "name": "mytemplate",
   "jobTemplateData": {
```

```
        "executionRoleArn": "iam_role_arn_for_job_execution",
        "releaseLabel": "emr-6.7.0-latest",
        "jobDriver": {
            "sparkSubmitJobDriver": {
                "entryPoint": "${EntryPointLocation}",
                "entryPointArguments": [ "argument1","argument2",...],
                "sparkSubmitParameters": "--class ${MainClass} --conf
spark.executor.instances=2 --conf spark.executor.memory=2G --conf
spark.executor.cores=2 --conf spark.driver.cores=1"
            }
        },
        "configurationOverrides": {
            "applicationConfiguration": [
                {
                    "classification": "spark-defaults",
                    "properties": {
                         "spark.driver.memory":"2G"
                    }
                }
            ],
            "monitoringConfiguration": {
                "persistentAppUI": "ENABLED",
                "cloudWatchMonitoringConfiguration": {
                    "logGroupName": "my_log_group",
                    "logStreamNamePrefix": "log_stream_prefix"
                },
                "s3MonitoringConfiguration": {
                    "logUri": "${LogS3BucketUri}"
                }
            }
        },
        "parameterConfiguration": {
            "EntryPointLocation": {
                "type": "STRING"
            },
            "MainClass": {
                "type": "STRING",
                "defaultValue":"Main"
            },
            "LogS3BucketUri": {
                "type": "STRING",
                "defaultValue":"s3://my_s3_log_location/"
            }
        }
```

```
    }
}
```

2. 搭配使用 `create-job-template` 命令與儲存在本機或 Amazon S3 中的 `create-job-template-request.json` 檔案路徑。

```
aws emr-containers create-job-template \
--cli-input-json file://./create-job-template-request.json
```

使用具有作業範本參數的作業範本開始作業執行

若要使用包含作業範本參數的作業範本開始作業執行，請在 `StartJobRun` API 請求中指定作業範本 ID 以及作業範本參數值，如下所示。

```
aws emr-containers start-job-run \
--virtual-cluster-id 123456 \
--name myjob \
--job-template-id 1234abcd \
--job-template-parameters '{"EntryPointLocation": "entry_point_location","MainClass":
 "ExampleMainClass","LogS3BucketUri": "s3://example_s3_bucket/"}'
```

## 控制對作業範本的存取

`StartJobRun` 政策可讓您強制使用者或角色只能使用您指定的作業範本來執行作業，而且如果不使用指定的作業範本，就無法執行 `StartJobRun` 操作。為此，首先確保為使用者或角色授予對指定作業範本的讀取許可，如下所示。

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "emr-containers:DescribeJobTemplate",
            "Resource": [
                "job_template_1_arn",
                "job_template_2_arn"
            ]
```

```
        }
    ]
}
```

若要強制使用者或角色只能在使用指定的作業範本時調用 `StartJobRun` 操作，可將下列 `StartJobRun` 政策許可指派給指定的使用者或角色。

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "emr-containers:StartJobRun",
            "Resource": [
                "virtual_cluster_arn",
            ],
            "Condition": [
                "StringEquals": {
                    "emr-containers:JobTemplateArn": [
                        "job_template_1_arn",
                        "job_template_2_arn",
                        ...
                    ]
                 }
                ]
            }
        }
    ]
}
```

如果作業範本在執行角色 ARN 欄位中指定作業範本參數，則使用者將能夠提供此參數的值，因此可以使用任意執行角色來調用 `StartJobRun`。若要限制使用者可以提供的執行角色，請參閱 搭配使用作業執行角色與 Amazon EMR on EKS 中的控制對執行角色的存取。

如果上述 `StartJobRun` 動作政策中未針對指定的使用者或角色指定任何條件，則將允許使用者或角色使用他們具有讀取權限的任意作業範本或使用任意執行角色，在指定虛擬叢集上調用 `StartJobRun` 動作。

# 使用 Pod 範本

從 Amazon EMR 5.33.0 或 6.3.0 版開始，Amazon EMR on EKS 支援 Spark 的 Pod 範本功能。Pod 是一個或多個容器群組，其中包含共用儲存和網路資源，以及如何執行容器的規範。Pod 範本是決定如何執行每個 Pod 的規範。可以使用 Pod 範本檔案來定義 Spark 組態不支援的驅動程式或執行程式 Pod 組態。如需有關 Spark 的 Pod 範本功能的詳細資訊，請參閱 Pod 範本。

> **Note**
>
> Pod 範本功能僅適用於驅動程式和執行程式 Pod。您不能使用 Pod 範本設定任務提交者 Pod。

## 常用案例

透過搭配使用 Pod 範本與 Amazon EMR on EKS，可定義如何在共用的 EKS 叢集上執行 Spark 作業，並節省成本及改善資源使用率和效能。

- 為了降低成本，可以排程 Spark 驅動程式任務在 Amazon EC2 隨需執行個體上執行，同時排程 Spark 執行程式任務在 Amazon EC2 Spot 執行個體上執行。
- 若要提高資源使用率，您可以支援多個團隊在相同 EKS 叢集上執行其工作負載。每個團隊都會獲得一個指定的 Amazon EC2 節點群組，以便在其中執行工作負載。可以使用 Pod 範本，將對應的容差套用至其工作負載。
- 若要改善監控，可以執行單獨的日誌容器，將日誌轉寄至現有的監控應用程式。

例如，下列 Pod 範本檔案示範常見的使用案例。

```
apiVersion: v1
kind: Pod
spec:
  volumes:
    - name: source-data-volume
      emptyDir: {}
    - name: metrics-files-volume
      emptyDir: {}
  nodeSelector:
    eks.amazonaws.com/nodegroup: emr-containers-nodegroup
  containers:
  - name: spark-kubernetes-driver # This will be interpreted as driver Spark main
 container
```

```
    env:
      - name: RANDOM
        value: "random"
    volumeMounts:
      - name: shared-volume
        mountPath: /var/data
      - name: metrics-files-volume
        mountPath: /var/metrics/data
  - name: custom-side-car-container # Sidecar container
    image: <side_car_container_image>
    env:
      - name: RANDOM_SIDECAR
        value: random
    volumeMounts:
      - name: metrics-files-volume
        mountPath: /var/metrics/data
    command:
      - /bin/sh
      - '-c'
      -  <command-to-upload-metrics-files>
  initContainers:
  - name: spark-init-container-driver # Init container
    image: <spark-pre-step-image>
    volumeMounts:
      - name: source-data-volume # Use EMR predefined volumes
        mountPath: /var/data
    command:
      - /bin/sh
      - '-c'
      -  <command-to-download-dependency-jars>
```

Pod 範本會完成下列任務：

- 新增一個在 Spark 主容器啟動之前執行的初始化容器。初始化容器與 Spark 主容器共用稱為 `source-data-volume` 的 EmptyDir 磁碟區。可以讓初始化容器執行初始化步驟，例如下載相依性或產生輸入資料。然後，Spark 主容器會消耗資料。
- 新增與 Spark 主容器一起執行的另一個附屬容器。這兩個容器正在共用另一個稱為的 `metrics-files-volume` 的 `EmptyDir` 磁碟區。您的 Spark 作業可以產生指標，例如 Prometheus 指標。然後，Spark 作業可以將指標放入檔案中，並讓附屬容器將檔案上傳到您自己的 BI 系統，以供將來分析。
- 將新環境變數新增至 Spark 主容器。可以讓作業消耗環境變數。

- 定義節點選取器，以便僅在 `emr-containers-nodegroup` 節點群組上排程 Pod。這有助於隔離作業和團隊的運算資源。

## 使用 Amazon EMR on EKS 啟用 Pod 範本

若要使用 Amazon EMR on EKS 啟用 Pod 範本功能，請設定 Spark 屬性 `spark.kubernetes.driver.podTemplateFile` 和 `spark.kubernetes.executor.podTemplateFile`，以指向 Amazon S3 中的 Pod 範本檔案。然後，Spark 會下載 Pod 範本檔案，並使用它來建構驅動程式和執行程式 Pod。

> **Note**
>
> Spark 使用作業執行角色來載入 Pod 範本，因此作業執行角色必須有權存取 Amazon S3 以載入 Pod 範本。如需詳細資訊，請參閱建立作業執行角色。

您可以使用 `SparkSubmitParameters` 來指定 Pod 範本的 Amazon S3 路徑，如下列作業執行 JSON 檔案所示。

```
{
  "name": "myjob",
  "virtualClusterId": "123456",
  "executionRoleArn": "iam_role_name_for_job_execution",
  "releaseLabel": "release_label",
  "jobDriver": {
    "sparkSubmitJobDriver": {
      "entryPoint": "entryPoint_location",
      "entryPointArguments": ["argument1", "argument2", ...],
       "sparkSubmitParameters": "--class <main_class> \
         --conf
 spark.kubernetes.driver.podTemplateFile=s3://path_to_driver_pod_template \
         --conf
 spark.kubernetes.executor.podTemplateFile=s3://path_to_executor_pod_template \
         --conf spark.executor.instances=2 \
         --conf spark.executor.memory=2G \
         --conf spark.executor.cores=2 \
         --conf spark.driver.cores=1"
    }
  }
}
```

或者，可以使用 `configurationOverrides` 來指定 Pod 範本的 Amazon S3 路徑，如下列作業執行 JSON 檔案所示。

```
{
  "name": "myjob",
  "virtualClusterId": "123456",
  "executionRoleArn": "iam_role_name_for_job_execution",
  "releaseLabel": "release_label",
  "jobDriver": {
    "sparkSubmitJobDriver": {
      "entryPoint": "entryPoint_location",
      "entryPointArguments": ["argument1", "argument2", ...],
       "sparkSubmitParameters": "--class <main_class> \
         --conf spark.executor.instances=2 \
         --conf spark.executor.memory=2G \
         --conf spark.executor.cores=2 \
         --conf spark.driver.cores=1"
    }
  },
  "configurationOverrides": {
    "applicationConfiguration": [
      {
        "classification": "spark-defaults",
        "properties": {
          "spark.driver.memory":"2G",
          "spark.kubernetes.driver.podTemplateFile":"s3://path_to_driver_pod_template",

 "spark.kubernetes.executor.podTemplateFile":"s3://path_to_executor_pod_template"
         }
      }
    ]
  }
}
```

> **Note**
>
> 1. 搭配使用 Pod 範本功能與 Amazon EMR on EKS 時，需要遵循安全準則，例如隔離不受信任的應用程式碼。如需詳細資訊，請參閱Amazon EMR on EKS 安全最佳實務。
> 2. 無法使用 `spark.kubernetes.driver.podTemplateContainerName` 和 `spark.kubernetes.executor.podTemplateContainerName` 來變更 Spark 主容器名稱 ，因為這些名稱硬編碼為 `spark-kubernetes-driver` 和 `spark-kubernetes-`

executors。如果想要自訂 Spark 主容器，則必須使用這些硬編碼名稱在 Pod 範本中指定容器。

## Pod 範本欄位

使用 Amazon EMR on EKS 設定 Pod 範本時，請考慮下列欄位限制。

- Amazon EMR on EKS 僅允許 Pod 範本中的下列欄位啟用適當的作業排程。

  以下是允許的 Pod 層級欄位：

  - `apiVersion`
  - `kind`
  - `metadata`
  - `spec.activeDeadlineSeconds`
  - `spec.affinity`
  - `spec.containers`
  - `spec.enableServiceLinks`
  - `spec.ephemeralContainers`
  - `spec.hostAliases`
  - `spec.hostname`
  - `spec.imagePullSecrets`
  - `spec.initContainers`
  - `spec.nodeName`
  - `spec.nodeSelector`
  - `spec.overhead`
  - `spec.preemptionPolicy`
  - `spec.priority`
  - `spec.priorityClassName`
  - `spec.readinessGates`
  - `spec.runtimeClassName`
  - `spec.schedulerName`
  - `spec.subdomain`

- `spec.terminationGracePeriodSeconds`
- `spec.tolerations`
- `spec.topologySpreadConstraints`
- `spec.volumes`

以下是允許的 Spark 主容器層級欄位：

- `env`
- `envFrom`
- `name`
- `lifecycle`
- `livenessProbe`
- `readinessProbe`
- `resources`
- `startupProbe`
- `stdin`
- `stdinOnce`
- `terminationMessagePath`
- `terminationMessagePolicy`
- `tty`
- `volumeDevices`
- `volumeMounts`
- `workingDir`

當您在 Pod 範本中使用任何不允許的欄位時，Spark 會擲出例外狀況，且作業失敗。下列範例會顯示 Spark 控制器日誌中的錯誤訊息，因為存在不允許的欄位。

```
Executor pod template validation failed.
Field container.command in Spark main container not allowed but specified.
```

- Amazon EMR on EKS 會在 Pod 範本中預先定義下列參數。在 Pod 範本中指定的欄位不得與這些欄位重疊。

  以下是預先定義的磁碟區名稱：

  - `emr-container-communicate`

- config-volume
- emr-container-application-log-dir
- emr-container-event-log-dir
- temp-data-dir
- mnt-dir
- home-dir
- emr-container-s3

以下是僅適用於 Spark 主容器的預定義磁碟區掛載：

- 名稱：emr-container-communicate；掛載路徑：/var/log/fluentd
- 名稱：emr-container-application-log-dir；掛載路徑：/var/log/spark/user
- 名稱：emr-container-event-log-dir；掛載路徑：/var/log/spark/apps
- 名稱：mnt-dir；掛載路徑：/mnt
- 名稱：temp-data-dir；掛載路徑：/tmp
- 名稱：home-dir；掛載路徑：/home/hadoop

這些是僅適用於 Spark 主容器的預定義環境變數：

- SPARK_CONTAINER_ID
- K8S_SPARK_LOG_URL_STDERR
- K8S_SPARK_LOG_URL_STDOUT
- SIDECAR_SIGNAL_FILE

> **Note**
>
> 您仍然可以使用這些預先定義的磁碟區，並將其掛載至額外的附屬容器中。例如，您可以使用 emr-container-application-log-dir 並將其掛接到 Pod 範本中定義的您自己的附屬容器。

如果您指定的欄位與 Pod 範本中的任何預定義欄位衝突，Spark 會擲出例外狀況，而作業會失敗。下列範例會顯示 Spark 應用程式日誌中的錯誤訊息，因為與預定義的欄位衝突。

```
Defined volume mount path on main container must not overlap with reserved mount
 paths: [<reserved-paths>]
```

# 附屬容器的考量

Amazon EMR 控制由 Amazon EMR on EKS 佈建的 Pod 的生命週期。附屬容器應遵循 Spark 主容器的相同生命週期。如果將額外的附屬容器插入至 Pod，建議與 Amazon EMR 定義的 Pod 生命週期管理整合，以便 Spark 主容器結束時，附屬容器可以自行停止。

為了降低成本，建議您實施一個程序，以防止具有附屬容器的驅動程式 Pod 在作業完成後繼續執行。當執行程式完成時，Spark 驅動程式會刪除執行程式 Pod。但是，當驅動程式完成時，其他附屬容器會繼續執行。Pod 會計費，直到 Amazon EMR on EKS 清理驅動程式 Pod 為止，通常在驅動程式 Spark 主容器完成後不到一分鐘。為了降低成本，可以將其他附屬容器與 Amazon EMR on EKS 為驅動程式和執行程式 Pod 定義的生命週期管理機制整合，如下節所述。

驅動程式和執行程式 Pod 中的 Spark 主容器每兩秒向檔案 `/var/log/fluentd/main-container-terminated` 傳送一次 `heartbeat`。透過將 Amazon EMR 預先定義的 `emr-container-communicate` 磁碟區掛載新增至附屬容器，可以定義附屬容器的子流程，以定期追蹤此檔案的上次修改時間。然後，如果子流程發現 Spark 主容器長時間停止 `heartbeat`，則子流程會自行停止。

下列範例示範追蹤活動訊號檔案並自行停止的子流程。將 *your_volume_mount* 取代為掛載預定義磁碟區的路徑。指令碼綁定在附屬容器使用的映像內。在 Pod 範本檔案中，可以使用下列命令 `sub_process_script.sh` 和 `main_command` 來指定附屬容器。

```
MOUNT_PATH="your_volume_mount"
FILE_TO_WATCH="$MOUNT_PATH/main-container-terminated"
INITIAL_HEARTBEAT_TIMEOUT_THRESHOLD=60
HEARTBEAT_TIMEOUT_THRESHOLD=15
SLEEP_DURATION=10

function terminate_main_process() {
  # Stop main process
}

# Waiting for the first heartbeat sent by Spark main container
echo "Waiting for file $FILE_TO_WATCH to appear..."
start_wait=$(date +%s)
while ! [[ -f "$FILE_TO_WATCH" ]]; do
    elapsed_wait=$(expr $(date +%s) - $start_wait)
    if [ "$elapsed_wait" -gt "$INITIAL_HEARTBEAT_TIMEOUT_THRESHOLD" ]; then
        echo "File $FILE_TO_WATCH not found after $INITIAL_HEARTBEAT_TIMEOUT_THRESHOLD
 seconds; aborting"
        terminate_main_process
        exit 1
```

```
    fi
    sleep $SLEEP_DURATION;
done;
echo "Found file $FILE_TO_WATCH; watching for heartbeats..."

while [[ -f "$FILE_TO_WATCH" ]]; do
    LAST_HEARTBEAT=$(stat -c %Y $FILE_TO_WATCH)
    ELAPSED_TIME_SINCE_AFTER_HEARTBEAT=$(expr $(date +%s) - $LAST_HEARTBEAT)
    if [ "$ELAPSED_TIME_SINCE_AFTER_HEARTBEAT" -gt "$HEARTBEAT_TIMEOUT_THRESHOLD" ];
 then
        echo "Last heartbeat to file $FILE_TO_WATCH was more than
 $HEARTBEAT_TIMEOUT_THRESHOLD seconds ago at $LAST_HEARTBEAT; terminating"
        terminate_main_process
        exit 0
    fi
    sleep $SLEEP_DURATION;
done;
echo "Outside of loop, main-container-terminated file no longer exists"

# The file will be deleted once the fluentd container is terminated

echo "The file $FILE_TO_WATCH doesn't exist any more;"
terminate_main_process
exit 0
```

# 使用作業重試政策

在 Amazon EMR on EKS 6.9.0 版及更新版本中，可為作業執行設定重試政策。重試政策會導致作業驅動程式 Pod 在失敗或遭到刪除時自動重新啟動。這讓長時間執行的 Spark 串流作業對故障更具彈性。

## 設定作業的重試政策

若要設定重試政策，可以使用 StartJobRun API 提供 `RetryPolicyConfiguration` 欄位。`retryPolicyConfiguration` 範例如下所示：

```
aws emr-containers start-job-run \
--virtual-cluster-id cluster_id \
--name sample-job-name \
--execution-role-arn execution-role-arn \
--release-label emr-6.9.0-latest \
--job-driver '{
  "sparkSubmitJobDriver": {
```

```
    "entryPoint": "local:///usr/lib/spark/examples/src/main/python/pi.py",
    "entryPointArguments": [ "2" ],
    "sparkSubmitParameters": "--conf spark.executor.instances=2 --conf
 spark.executor.memory=2G --conf spark.executor.cores=2 --conf spark.driver.cores=1"
  }
}' \
--retry-policy-configuration '{
    "maxAttempts": 5
  }' \
--configuration-overrides '{
  "monitoringConfiguration": {
    "cloudWatchMonitoringConfiguration": {
      "logGroupName": "my_log_group_name",
      "logStreamNamePrefix": "my_log_stream_prefix"
    },
    "s3MonitoringConfiguration": {
       "logUri": "s3://amzn-s3-demo-logging-bucket"
    }
  }
}'
```

> **Note**
>
> `retryPolicyConfiguration` 僅從 AWS CLI 1.27.68 版開始提供。若要將 更新 AWS CLI 至最新版本，請參閱安裝或更新最新版本的 AWS CLI

將 `maxAttempts` 欄位設定為您希望作業驅動程式 Pod 在失敗或遭到刪除時重新啟動的次數上限。兩次作業驅動程式重試嘗試之間的執行間隔為指數重試間隔 (10 秒、20 秒、40 秒...)，上限為 6 分鐘，如 Kubernetes 文件所述。

> **Note**
>
> 每個額外的作業驅動程式執行將按照另一個作業執行計費，且將受到 Amazon EMR on EKS 定價的約束。

## 重試政策組態值

- 作業的預設重試政策：根據預設，`StartJobRun` 包含設定為最多 1 次的重試政策。可以視需要設定重試政策。

> **Note**
>
> 如果 `retryPolicyConfiguration` 的 `maxAttempts` 設定為 1，則表示在失敗時不會進行重試來啟動驅動程式 Pod。

- 停用作業的重試政策：若要停用重試政策，請將 retryPolicyConfiguration 中的嘗試次數上限設為 1。

```
"retryPolicyConfiguration": {
    "maxAttempts": 1
}
```

- 將作業的 maxAttempts 設定在有效範圍內：如果 `maxAttempts` 值超出有效範圍，`StartJobRun` 呼叫將失敗。有效 `maxAttempts` 範圍介於 1 到 2,147,483,647 (32 位元整數) 之間，這是 Kubernetes 的 `backOffLimit` 組態設定所支援的範圍。如需詳細資訊，請參閱 Kubernetes 文件的 Pod 退避失敗政策。如果 `maxAttempts` 值無效，則會傳回下列錯誤訊息：

```
{
 "message": "Retry policy configuration's parameter value of maxAttempts is invalid"
}
```

## 擷取作業的重試政策狀態

可以使用 `ListJobRuns` 和 `DescribeJobRun` API 檢視作業的重試嘗試狀態。一旦請求具有已啟用重試政策組態的作業，`ListJobRun` 和 `DescribeJobRun` 回應就會在 `RetryPolicyExecution` 欄位中包含重試政策的狀態。此外，`DescribeJobRun` 回應將包含在作業的 `StartJobRun` 請求中輸入的 `RetryPolicyConfiguration`。

回應範例

ListJobRuns response

```
{
  "jobRuns": [
    ...
    ...
    "retryPolicyExecution" : {
      "currentAttemptCount": 2
    }
    ...
```

```
    ...
  ]
}
```

DescribeJobRun response

```
{
  ...
  ...
  "retryPolicyConfiguration": {
    "maxAttempts": 5
   },
   "retryPolicyExecution" : {
    "currentAttemptCount": 2
  },
  ...
  ...
}
```

在作業中停用重試政策後，將不會顯示這些欄位，如下面的 重試政策組態值 中所述。

## 使用重試政策監控作業

啟用重試政策時，會為建立的每個作業驅動程式產生 CloudWatch 事件。若要訂閱這些事件，請使用下列命令設定 CloudWatch 事件規則：

```
aws events put-rule \
--name cwe-test \
--event-pattern '{"detail-type": ["EMR Job Run New Driver Attempt"]}'
```

此事件將傳回作業驅動程式的 `newDriverPodName`、`newDriverCreatedAt` 時間戳記、`previousDriverFailureMessage` 和 `currentAttemptCount` 的相關資訊。如果停用重試政策，將不會建立這些事件。

如需有關如何使用 CloudWatch 事件監控作業的詳細資訊，請參閱 使用 Amazon CloudWatch Events 監控作業。

## 尋找驅動程式和執行程式的日誌

驅動程式 Pod 名稱遵循 `spark-<job id>-driver-<random-suffix>` 格式。相同的 `random-suffix` 會新增至驅動程式產生的執行程式 Pod 名稱。使用此 `random-suffix` 時，可尋找驅動

程式及其關聯執行程式的日誌。只有在為作業啟用重試政策時，才會顯示 `random-suffix`；否則，`random-suffix` 就不存在。

如需有關如何使用日誌的監控組態來設定作業的詳細資訊，請參閱 執行 Spark 應用程式。

# 使用 Spark 事件日誌輪換

使用 Amazon EMR 6.3.0 及更高版本，可以為 Amazon EMR on EKS 開啟 Spark 事件日誌輪換功能。此功能不會產生單一事件日誌檔案，而是會根據您設定的時間間隔來輪換檔案，並移除最舊的事件日誌檔案。

輪換 Spark 事件日誌可協助您避免長時間執行或串流作業所產生的大型 Spark 事件日誌檔案可能發生的問題。例如，使用透過 `persistentAppUI` 參數啟用的事件日誌來開始長時間執行的 Spark 作業。Spark 驅動程式會產生事件日誌檔案。如果作業執行數小時或數天，且 Kubernetes 節點上的磁碟空間有限，則事件日誌檔案可能會耗用所有可用的磁碟空間。開啟 Spark 事件日誌輪換功能可解決此問題，方法是將日誌檔案分割為多個檔案並移除最舊的檔案。

> **Note**
>
> 此功能僅適用於 Amazon EMR on EKS。在 Amazon EC2 執行的 Amazon EMR 不支援 Spark 事件日誌輪換。

若要開啟 Spark 事件日誌輪換功能，請設定下列 Spark 參數：

- `spark.eventLog.rotation.enabled`-開啟日誌輪換。依預設，它在 Spark 組態檔案中被停用。設定為 true 可開啟此功能。
- `spark.eventLog.rotation.interval`-指定日誌輪換的時間間隔。最小值為 60 秒。預設值為 300 秒。
- `spark.eventLog.rotation.minFileSize`-指定最小檔案大小以輪換日誌檔案。最小且預設值為 1 MB。
- `spark.eventLog.rotation.maxFilesToRetain`-指定在清理期間要保留多少個已輪換的日誌檔案。有效範圍為 1 到 10。預設值為 2。

可以在 `StartJobRun` API 的 `sparkSubmitParameters` 區段中指定這些參數，如下列範例所示。

```
"sparkSubmitParameters": "--class org.apache.spark.examples.SparkPi --conf
 spark.eventLog.rotation.enabled=true --conf spark.eventLog.rotation.interval=300 --
```

```
conf spark.eventLog.rotation.minFileSize=1m --conf
 spark.eventLog.rotation.maxFilesToRetain=2"
```

# 使用 Spark 容器日誌輪換

使用 Amazon EMR 6.11.0 及更高版本，可以為 Amazon EMR on EKS 開啟 Spark 容器日誌輪換功能。此功能不會產生單一 `stdout` 或 `stderr` 日誌檔案，而是會根據您設定的輪換大小來輪換檔案，並從容器中移除最舊的日誌檔案。

輪換 Spark 容器日誌可協助您避免長時間執行或串流作業所產生的大型 Spark 日誌檔案可能發生的問題。例如，可以啟動長時間執行的 Spark 作業，而 Spark 驅動程式會產生容器日誌檔案。如果作業執行數小時或數天，且 Kubernetes 節點上的磁碟空間有限，則容器日誌檔案可能會耗用所有可用的磁碟空間。當您開啟 Spark 容器日誌輪換時，您會將日誌檔案分割成多個檔案，然後移除最舊的檔案。

若要開啟 Spark 容器日誌輪換功能，請設定下列 Spark 參數：

**`containerLogRotationConfiguration`**

在 `monitoringConfiguration` 中包含此參數以開啟日誌輪換。預設為停用。除了 `s3MonitoringConfiguration` 之外，還必須使用 `containerLogRotationConfiguration`。

**`rotationSize`**

`rotationSize` 參數指定日誌輪換的檔案大小。可能的值範圍為 2KB 至 2GB。`rotationSize` 參數的數值單位部分會以整數形式傳遞。由於不支援小數值，因此可以使用值 `1500MB` 來指定 1.5GB 的輪換大小。

**`maxFilesToKeep`**

`maxFilesToKeep` 參數會指定輪換發生後，要在容器中保留的檔案數上限。下限值是 1，上限值是 50。

可以在 `StartJobRun` API 的 `monitoringConfiguration` 區段中指定這些參數，如下列範例所示。在此範例中，使用 `rotationSize = "10 MB"` 和 `maxFilesToKeep = 3`，Amazon EMR on EKS 在 10 MB 時輪換日誌，產生新的日誌檔案，然後在日誌檔案數量達到 3 時清除最舊的日誌檔案。

```
{
  "name": "my-long-running-job",
```

```
  "virtualClusterId": "123456",
  "executionRoleArn": "iam_role_name_for_job_execution",
  "releaseLabel": "emr-6.11.0-latest",
  "jobDriver": {
    "sparkSubmitJobDriver": {
      "entryPoint": "entryPoint_location",
      "entryPointArguments": ["argument1", "argument2", ...],
       "sparkSubmitParameters": "--class main_class --conf spark.executor.instances=2
 --conf spark.executor.memory=2G --conf spark.executor.cores=2 --conf
 spark.driver.cores=1"
    }
  },
  "configurationOverrides": {
    "applicationConfiguration": [
      {
        "classification": "spark-defaults",
        "properties": {
          "spark.driver.memory":"2G"
         }
      }
    ],
    "monitoringConfiguration": {
      "persistentAppUI": "ENABLED",
      "cloudWatchMonitoringConfiguration": {
        "logGroupName": "my_log_group",
        "logStreamNamePrefix": "log_stream_prefix"
      },
      "s3MonitoringConfiguration": {
        "logUri": "s3://my_s3_log_location"
      },
      "containerLogRotationConfiguration": {
        "rotationSize":"10MB",
        "maxFilesToKeep":"3"
      }
    }
  }
}
```

若要使用 Spark 容器日誌輪換開始執行作業，請在 StartJobRun 命令中包含使用這些參數設定的 json 檔案的路徑。

```
aws emr-containers start-job-run \
--cli-input-json file://path-to-json-request-file
```

# 搭配使用垂直自動擴展與 Amazon EMR Spark 作業

Amazon EMR on EKS 垂直自動擴展功能會自動調整記憶體和 CPU 資源，以適應您為 Amazon EMR Spark 應用程式提供的工作負載需求。這可簡化資源管理。

為了追蹤 Amazon EMR Spark 應用程式的即時和歷史資源使用情況，垂直自動擴展功能可利用 Kubernetes Vertical Pod Autoscaler (VPA)。垂直自動擴展功能使用 VPA 收集的資料，自動調整指派給 Spark 應用程式的記憶體和 CPU 資源。這種簡化流程可提高可靠性並優化成本。

主題


- 設定 Amazon EMR on EKS 的垂直自動擴展
- Amazon EMR on EKS 垂直自動擴展入門
- 設定 Amazon EMR on EKS 的垂直自動擴展
- 監控 Amazon EMR on EKS 的垂直自動擴展
- 解除安裝 Amazon EMR on EKS 垂直自動擴展 Operator


## 設定 Amazon EMR on EKS 的垂直自動擴展

本主題可協助您讓 Amazon EKS 叢集準備好提交具有垂直自動擴展功能的 Amazon EMR Spark 作業。設定程序會要求您確認或完成下列各章節中的任務：

主題


- 先決條件
- 在 Amazon EKS 叢集上安裝 Operator Lifecycle Manager (OLM)
- 安裝 Amazon EMR on EKS 垂直自動擴展運算子


### 先決條件

在叢集上安裝可垂直自動擴展的 Kubernetes Operator 之前，請先完成下列任務。如果已經完成任何先決條件，則可以跳過這些先決條件，然後繼續進行下一個。

- 安裝或更新至最新版本的 AWS CLI- 如果您已安裝 AWS CLI，請確認您擁有最新版本。
- 安裝 kubectl – kubectl 是一種命令列工具，可用來與 Kubernetes API 伺服器進行通訊。您需要 kubectl 才能在 Amazon EKS 叢集上安裝和監控垂直自動擴展相關成品。
- 安裝 Operator SDK – Amazon EMR on EKS 使用 Operator SDK 作為您在叢集上安裝的垂直自動擴展運算子使用壽命的套件管理工具。

- 安裝 Docker – 您需要存取 Docker CLI 來驗證和擷取要安裝在 Amazon EKS 叢集上的垂直自動擴展相關 Docker 映像檔。
- 安裝 Kubernetes 指標伺服器 – 您必須先安裝指標伺服器，垂直 Pod 自動擴展器才能從 Kubernetes API 伺服器擷取指標。
- Amazon EKS – eksctl (1.24 版或更新版本） 入門 – Amazon EKS 1.24 版及更新版本支援垂直自動擴展。建立叢集後，請進行註冊以與 Amazon EMR 搭配使用。
- 選擇 Amazon EMR 基礎映像 URI (6.10.0 版或更高版本) - Amazon EMR 6.10.0 版及更高版本支援垂直自動擴展。

## 在 Amazon EKS 叢集上安裝 Operator Lifecycle Manager (OLM)

使用 Operator SDK CLI 在想要設定垂直自動擴展的 Amazon EMR on EKS 叢集上安裝 Operator Lifecycle Manager (OLM)，如下列範例所示。設定完成後，可以使用 OLM 來安裝和管理 Amazon EMR 垂直自動擴展運算子的生命週期。

```
operator-sdk olm install
```

若要驗證安裝，請執行 `olm status` 命令：

```
operator-sdk olm status
```

請確認命令傳回成功結果，與以下範例輸出類似：

```
INFO[0007] Successfully got OLM status for version X.XX
```

如果安裝未成功，請參閱 對 Amazon EMR on EKS 垂直自動擴展進行疑難排解。

## 安裝 Amazon EMR on EKS 垂直自動擴展運算子

使用下列步驟在 Amazon EKS 叢集上安裝垂直自動擴展運算子：

1. 設定下列將用於完成安裝的環境變數：
   - **`$REGION`** 指向叢集的 AWS 區域 。例如：`us-west-2`。
   - **`$ACCOUNT_ID`** 指向您所在區域的 Amazon ECR 帳戶 ID。如需詳細資訊，請參閱按區域劃分的 Amazon ECR 登錄檔帳戶。

- **$RELEASE** 指向要用於叢集的 Amazon EMR 版本。對於垂直自動擴展，必須使用 Amazon EMR 6.10.0 或更高版本。

2. 接下來，取得運算子的 Amazon ECR 登錄檔驗證字符。

```
aws ecr get-login-password \
 --region region-id | docker login \
 --username AWS \
 --password-stdin $ACCOUNT_ID.dkr.ecr.region-id.amazonaws.com
```

3. 使用下列命令安裝 Amazon EMR on EKS 垂直自動擴展運算子：

```
ECR_URL=$ACCOUNT_ID.dkr.ecr.$REGION.amazonaws.com && \
REPO_DEST=dynamic-sizing-k8s-operator-olm-bundle && \
BUNDLE_IMG=emr-$RELEASE-dynamic-sizing-k8s-operator && \
operator-sdk run bundle \
$ECR_URL/$REPO_DEST/$BUNDLE_IMG\:latest
```

這會在 Amazon EKS 叢集的預設命名空間中建立垂直自動擴展運算子的版本。使用此命令在不同的命名空間中安裝：

```
operator-sdk run bundle \
$ACCOUNT_ID.dkr.ecr.$REGION.amazonaws.com/dynamic-sizing-k8s-operator-olm-bundle/
emr-$RELEASE-dynamic-sizing-k8s-operator:latest \
-n operator-namespace
```

> **Note**
>
> 如果指定的命名空間不存在，則 OLM 將不會安裝運算子。如需詳細資訊，請參閱找不到 Kubernetes 命名空間。

4. 確認已使用 kubectl Kubernetes 命令列工具成功安裝 Operator。

```
kubectl get csv -n operator-namespace
```

`kubectl` 命令應傳回新部署的垂直自動擴展器 Operator，其階段狀態為已成功。如果遇到安裝或設定問題，請參閱 對 Amazon EMR on EKS 垂直自動擴展進行疑難排解。

# Amazon EMR on EKS 垂直自動擴展入門

當您想要自動調校記憶體和 CPU 資源以適應 Amazon EMR Spark 應用程式工作負載時，請使用 Amazon EMR on EKS 的垂直自動擴展。如需詳細資訊，請參閱搭配 Amazon EMR Spark 任務使用垂直自動擴展。

## 使用垂直自動擴展功能提交 Spark 作業

透過 StartJobRun API 提交作業時，請將下列兩個組態新增至驅動程式，以便 Spark 作業開啟垂直自動擴展：

```
"spark.kubernetes.driver.annotation.emr-containers.amazonaws.com/
dynamic.sizing":"true",
"spark.kubernetes.driver.annotation.emr-containers.amazonaws.com/
dynamic.sizing.signature":"YOUR_JOB_SIGNATURE"
```

在上面的代碼中，第一列啟用垂直自動擴展功能。下一列是必要的簽章組態，可讓您為作業選擇簽章。

如需這些組態和可接受參數值的詳細資訊，請參閱 設定 Amazon EMR on EKS 的垂直自動擴展。依預設，您的作業在垂直自動擴展的僅監控關閉模式下提交。此監控狀態可讓您計算和檢視資源建議，而無需執行自動擴展。如需詳細資訊，請參閱垂直自動擴展模式。

以下範例示範如何使用垂直自動擴展來完成範例 `start-job-run` 命令：

```
aws emr-containers start-job-run \
--virtual-cluster-id $VIRTUAL_CLUSTER_ID \
--name $JOB_NAME \
--execution-role-arn $EMR_ROLE_ARN \
--release-label emr-6.10.0-latest \
--job-driver '{
  "sparkSubmitJobDriver": {
     "entryPoint": "local:///usr/lib/spark/examples/src/main/python/pi.py"
   }
 }' \
--configuration-overrides '{
    "applicationConfiguration": [{
        "classification": "spark-defaults",
        "properties": {
          "spark.kubernetes.driver.annotation.emr-containers.amazonaws.com/
dynamic.sizing": "true",
          "spark.kubernetes.driver.annotation.emr-containers.amazonaws.com/
dynamic.sizing.signature": "test-signature"
```

```
        }
    }]
  }'
```

### 驗證垂直自動擴展功能

若要確認垂直自動擴展適用於已提交的作業，請使用 kubectl 取得 `verticalpodautoscaler` 自訂資源並檢視您的擴展建議。例如，下列命令會查詢 使用垂直自動擴展功能提交 Spark 作業 區段中範例作業的建議：

```
kubectl get verticalpodautoscalers --all-namespaces \
-l=emr-containers.amazonaws.com/dynamic.sizing.signature=test-signature
```

此查詢的輸出應如下所示：

```
NAME                                                          MODE   CPU         MEM
 PROVIDED   AGE
ds-jceyefkxnhrvdzw6djum3naf2abm6o63a6dvjkkedqtkhlrf25eq-vpa   Off    3304504865  True
         87m
```

如果您的輸出看起來不相似或包含錯誤碼，請參閱 對 Amazon EMR on EKS 垂直自動擴展進行疑難排解 以取得協助解決問題的步驟。

## 設定 Amazon EMR on EKS 的垂直自動擴展

當您透過 StartJobRun API 提交 Amazon EMR Spark 作業時，可設定垂直自動擴展。在 Spark 驅動程式 Pod 中設定自動擴展相關組態參數，如 使用垂直自動擴展功能提交 Spark 作業 中的範例所示。

Amazon EMR on EKS 垂直自動擴展 Operator 會偵聽具有自動擴展功能的驅動程式 Pod，然後透過驅動程式 Pod 中的設定進行與 Kubernetes Vertical Pod Autoscaler (VPA) 的整合。這有助於 Spark 執行程式 Pod 的資源追蹤和自動擴展。

以下各章節描述了為 Amazon EKS 叢集設定垂直自動擴展時可以使用的參數。

> **Note**
>
> 將功能切換參數設定為標籤，並將其餘參數設定為 Spark 驅動程式 Pod 中的註釋。自動擴展參數屬於 `emr-containers.amazonaws.com/` 域，並具有 `dynamic.sizing` 字首。

## 必要參數

提交作業時，必須在 Spark 作業驅動程式中包含下列兩個參數：

| 金鑰 | 描述 | 接受的值 | 預設值 | Type | Spark 參數[1] |
| --- | --- | --- | --- | --- | --- |
| `dynamic.sizing` | 功能切換 | `true`, `false` | 未設定 | label | `spark.kubernetes.driver.label.emr-containers.amazonaws.com/dynamic.sizing` |
| `dynamic.sizing.signature` | 作業簽章 | string | 未設定 | 註釋 | `spark.kubernetes.driver.annotation.emr-containers.amazonaws.com/dynamic.sizing.signature` |

[1] `ConfigurationOverride` 在 `StartJobRun` API 中使用此參數作為 `SparkSubmitParameter` 或 。

- **`dynamic.sizing`** - 可以使用 `dynamic.sizing` 標籤開啟和關閉垂直自動擴展功能。若要開啟垂直自動擴展，請在 Spark 驅動程式 Pod 中將 `dynamic.sizing` 設定為 `true`。如果省略此標籤或將其設定為除 `true` 以外的任何值，垂直自動擴展功能會關閉。
- **`dynamic.sizing.signature`** - 在驅動程式 Pod 中設定含有 `dynamic.sizing.signature` 注釋的作業簽章。垂直自動擴展可跨 Amazon EMR Spark 作業的不同執行來彙總資源使用量資料，以衍生資源建議。您可以提供唯一識別符，以便將作業繫結在一起。

> **Note**
>
> 如果作業以固定間隔 (例如每日或每週) 重複出現，則作業簽章對於作業的每個新執行個體都應保持不變。這可確保垂直自動擴展功能可以計算和彙總作業的不同執行中的建議。

[1] `ConfigurationOverride` 在 `StartJobRun` API 中使用此參數作為 `SparkSubmitParameter` 或 。

## 選用的參數

垂直自動擴展也支援下列選用參數。將它們設定為驅動程式 Pod 中的注釋。

| 金鑰 | 描述 | 接受的值 | 預設值 | Type | Spark 參數[1] |
| --- | --- | --- | --- | --- | --- |
| `dynamic.sizing.mode` | 垂直自動擴展模式 | `Off`, `Initial`, `Auto` | `Off` | 註釋 | `spark.kubernetes.driver.annotation.emr-containers.amazonaws.com/dynamic.sizing.mode` |
| `dynamic.sizing.scale.memory` | 啟用記憶體擴展 | *true*, *false* | `true` | 註釋 | `spark.kubernetes.driver.annotation.emr-containers.amazonaws.com/dynamic.sizing.scale.memory` |

| 金鑰 | 描述 | 接受的值 | 預設值 | Type | Spark 參數[1] |
| --- | --- | --- | --- | --- | --- |
| dynamic.sizing.scale.cpu | 開啟或關閉 CPU 擴展 | *true*, *false* | false | 註釋 | spark.kubernetes.driver.annotation.emr-containers.amazonaws.com/dynamic.sizing.scale.cpu |
| dynamic.sizing.scale.memory.min | 記憶體擴展的下限 | 字串，K8s 資源數量例如：1G | 未設定 | 註釋 | spark.kubernetes.driver.annotation.emr-containers.amazonaws.com/dynamic.sizing.scale.memory.min |
| dynamic.sizing.scale.memory.max | 記憶體擴展的上限 | 字串，K8s 資源數量例如：4G | 未設定 | 註釋 | spark.kubernetes.driver.annotation.emr-containers.amazonaws.com/dynamic.sizing.scale.memory.max |

| 金鑰 | 描述 | 接受的值 | 預設值 | Type | Spark 參數[1] |
| --- | --- | --- | --- | --- | --- |
| dynamic.sizing.scale.cpu.min | CPU 擴展的下限 | 字串，K8s 資源數量例如：1 | 未設定 | 註釋 | spark.kubernetes.driver.annotation.emr-containers.amazonaws.com/dynamic.sizing.scale.cpu.min |
| dynamic.sizing.scale.cpu.max | CPU 擴展的上限 | 字串，K8s 資源數量例如：2 | 未設定 | 註釋 | spark.kubernetes.driver.annotation.emr-containers.amazonaws.com/dynamic.sizing.scale.cpu.max |

垂直自動擴展模式

mode 參數會映射至 VPA 支援的不同自動擴展模式。使用驅動程式 Pod 上的 dynamic.sizing.mode 註釋來設定模式。此參數支援下列值：

- 關閉 – 試轉模式，您可以在該模式中監控建議，但不會執行自動擴展。這是垂直自動擴展的預設模式。在此模式中，相關聯的垂直 Pod 自動擴展器資源會計算建議，您可以透過 kubectl、Prometheus 和 Grafana 等工具監控建議。
- 初始 - 在此模式中，如果根據作業的歷史執行 (例如週期性作業) 提供建議，則 VPA 會在作業開始時自動擴展資源。

- 自動 - 在此模式中，VPA 會移出 Spark 執行程式 Pod，並在 Spark 驅動程式 Pod 重新啟動它們時，使用建議的資源設定來自動擴展它們。有時，VPA 會移出執行中的 Spark 執行程式 Pod，因此當它重試中斷的執行程式時，可能會導致額外的延遲。

資源擴展

當您設定垂直自動擴展時，可選擇是否擴展 CPU 和記憶體資源。將 `dynamic.sizing.scale.cpu` 和 `dynamic.sizing.scale.memory` 註釋設定為 `true` 或 `false`。根據預設，CPU 擴展設定為 `false`，記憶體擴展設定為 `true`。

資源最小值和最大值 (邊界)

或者，也可以在 CPU 和記憶體資源上設定邊界。啟用自動擴展時，請為這些具有 `dynamic.sizing.[memory/cpu].[min/max]` 註釋的資源選擇最小值和最大值。根據預設，資源沒有限制。將註釋設定為代表 Kubernetes 資源數量的字串值。例如，將 `dynamic.sizing.memory.max` 設定為 `4G`，表示 4 GB。

## 監控 Amazon EMR on EKS 的垂直自動擴展

您可以使用 kubectl Kubernetes 命令列工具，列出叢集上作用中的垂直自動擴展相關建議。也可以檢視追蹤的作業簽章，並清除與簽章相關聯的任何不需要的資源。

### 列出叢集的垂直自動擴展建議

使用 kubectl 取得 `verticalpodautoscaler` 資源，並檢視目前的狀態和建議。下列範例查詢會傳回 Amazon EKS 叢集中的所有作用中資源。

```
kubectl get verticalpodautoscalers \
-o custom-columns="NAME:.metadata.name,"\
"SIGNATURE:.metadata.labels.emr-containers\.amazonaws\.com/dynamic\.sizing
\.signature,"\
"MODE:.spec.updatePolicy.updateMode,"\
"MEM:.status.recommendation.containerRecommendations[0].target.memory" \
--all-namespaces
```

此查詢的輸出如下所示：

```
NAME                    SIGNATURE                MODE      MEM
ds-example-id-1-vpa     job-signature-1          Off       none
```

```
ds-example-id-2-vpa   job-signature-2          Initial   12936384283
```

### 查詢並刪除叢集的垂直自動擴展建議

刪除 Amazon EMR 垂直自動擴展作業執行資源時，它會自動刪除追蹤並儲存建議的關聯 VPA 物件。

下列範例使用 kubectl 來清除由簽章識別之作業的建議：

```
kubectl delete jobrun -n emr -l=emr-containers\.amazonaws\.com/dynamic\.sizing
\.signature=integ-test
jobrun.dynamicsizing.emr.services.k8s.aws "ds-job-signature" deleted
```

如果您不知道特定作業簽章，或想要清除叢集中的所有資源，則可以在命令中使用 `--all` 或 `--all-namespaces`，而非唯一的作業 ID，如下列範例所示：

```
kubectl delete jobruns --all --all-namespaces
jobrun.dynamicsizing.emr.services.k8s.aws "ds-example-id" deleted
```

## 解除安裝 Amazon EMR on EKS 垂直自動擴展 Operator

如果想要從 Amazon EKS 叢集中移除垂直自動擴展 Operator，請搭配使用 `cleanup` 命令與 Operator SDK CLI，如以下範例所示。這也會刪除與 Operator 一起搭配安裝的上游相依性，例如 Vertical Pod Autoscaler。

```
operator-sdk cleanup emr-dynamic-sizing
```

刪除 Operator 時，如果叢集中有任何正在執行的作業，則這些作業會繼續執行，而不會進行垂直自動擴展。如果在刪除 Operator 後在叢集中提交作業，則 Amazon EMR on EKS 將忽略您在設定期間定義的任何垂直自動擴展相關參數。

# 在 Amazon EMR on EKS 上執行互動式工作負載

互動端點是一個閘道，它將 Amazon EMR Studio 連接到 Amazon EMR on EKS，以便您可執行互動式工作負載。可以將互動端點與 EMR Studio 搭配使用，在 Amazon S3 和 Amazon DynamoDB 等資料存放區中使用資料集來執行互動式分析。

使用案例

- 使用 EMR Studio IDE 體驗建立 ETL 指令碼。IDE 會擷取內部部署資料，並在轉換後將其儲存在 Amazon S3 中，以供後續分析。
- 使用筆記本探索資料集，並訓練機器學習模型，以偵測資料集中的異常情況。
- 建立指令碼，為分析應用程式 (例如，商業儀表板) 產生每日報告。

主題


- 互動端點概觀
- 在 Amazon EMR on EKS 上建立互動端點的先決條件
- 為虛擬叢集建立互動端點
- 配置互動端點的設定
- 監控互動端點
- 使用自助託管的 Jupyter 筆記本
- 使用 CLI 命令取得互動式端點的相關資訊


## 互動端點概觀

互動端點可讓 Amazon EMR Studio 等互動式用戶端連線到 Amazon EMR on EKS 叢集，以執行互動式工作負載。互動端點由 Jupyter Enterprise Gateway 提供支援，可提供互動式用戶端所需的遠端核心生命週期管理功能。核心是特定於語言的程序，可與 Jupyter 型 Amazon EMR Studio 用戶端互動，以執行互動式工作負載。

互動端點支援下列核心：

- Python 3
- PySpark on Kubernetes
- 帶 Scala 的 Apache Spark

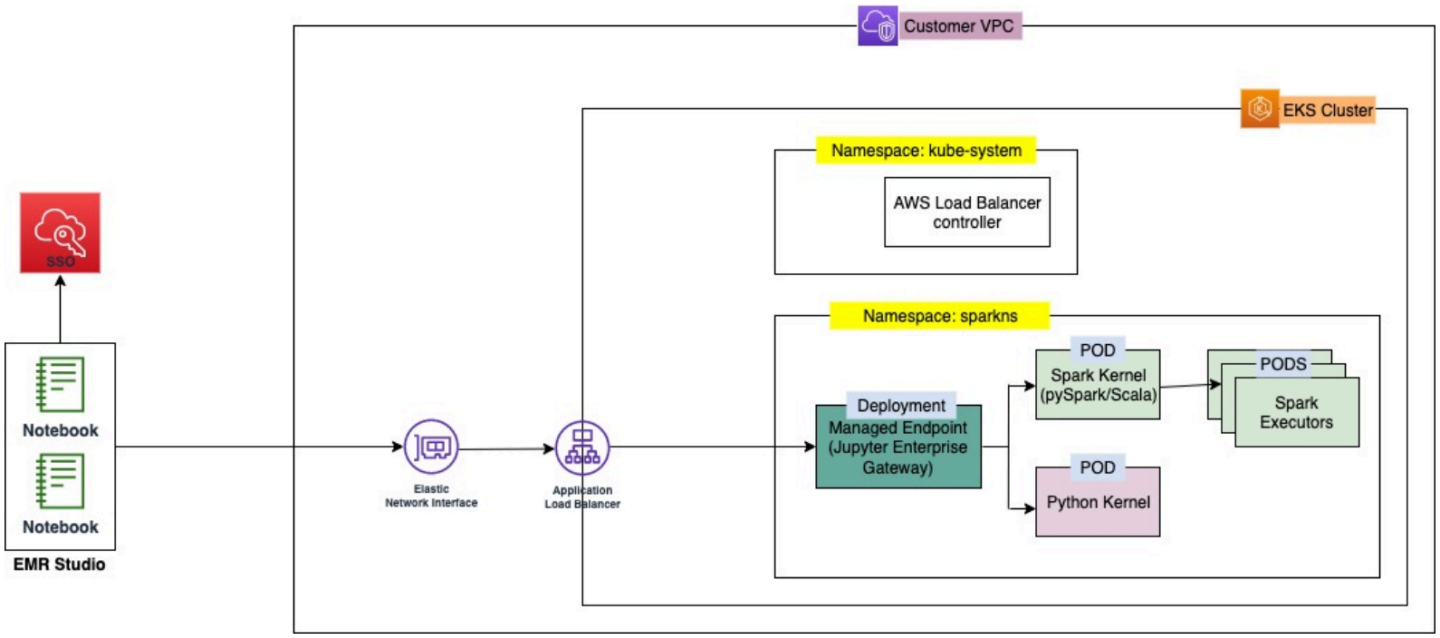
> **Note**
>
> Amazon EMR on EKS 定價適用於互動端點和核心。如需詳細資訊，請參閱 Amazon EMR on EKS 定價頁面。

EMR Studio 需要以下實體才能與 Amazon EMR on EKS 連接。

- Amazon EMR on EKS 虛擬叢集 - 虛擬叢集是您向其註冊 Amazon EMR 的 Kubernetes 命名空間。Amazon EMR 使用虛擬叢集來執行作業和託管端點。可使用相同實體叢集來支援多個虛擬叢集。不過，每個虛擬叢集都會映射 Amazon EKS 叢集上的一個命名空間。虛擬叢集不會建立任何增加帳單或需要在服務之外進行生命週期管理的作用中資源。
- Amazon EMR on EKS 互動端點 - 互動端點是一個 HTTPS 端點，EMR Studio 使用者可以將工作區與其相連。只能從 EMR Studio 中存取 HTTPS 端點，並在 Amazon EKS 叢集的 Amazon Virtual Private Cloud (Amazon VPC) 的私有子網路中建立這些端點。

  Python、PySpark 和 Spark Scala 核心使用在 Amazon EMR on EKS 作業執行角色中定義的許可來調用其他 AWS 服務。連線到互動端點的所有核心和使用者都會利用您在建立端點時指定的角色。我們建議您為不同的使用者建立單獨的端點，並且使用者具有不同的 AWS Identity and Access Management (IAM) 角色。
- AWS Application Load Balancer 控制器 – AWS Application Load Balancer 控制器會管理 Amazon EKS Kubernetes 叢集的 Elastic Load Balancing。當您建立 Kubernetes Ingress 資源時，控制器會佈建 Application Load Balancer (ALB)。ALB 會在 Amazon EKS 叢集之外但在相同 Amazon VPC 內公開 Kubernetes 服務 (例如互動端點)。當您建立互動端點時，也會部署 Ingress 資源，透過 ALB 公開互動端點，以供互動式用戶端連線。您只需要為每個 Amazon EKS 叢集安裝 one AWS Application Load Balancer 控制器。

下圖說明 Amazon EMR on EKS 中的互動端點架構。Amazon EKS 叢集包含用於執行分析工作負載的運算和互動端點。Application Load Balancer 控制器會在 `kube-system` 命名空間中執行；工作負載和互動端點會在您建立虛擬叢集時指定的命名空間中執行。建立互動端點時，Amazon EMR on EKS 控制平面會在 Amazon EKS 叢集中建立互動端點部署。此外，應用程式負載平衡器傳入的執行個體是由 AWS 負載平衡器控制器建立。Application Load Balancer 可為 EMR Studio 等用戶端提供外部介面，已連接到 Amazon EMR 叢集並執行互動式工作負載。




# 在 Amazon EMR on EKS 上建立互動端點的先決條件

本章節描述了設定互動端點的先決條件，EMR Studio 可使用該端點連線到 Amazon EMR on EKS 叢集並執行互動式工作負載。

## AWS CLI

請依照安裝或更新至最新版本 AWS CLI中的步驟，安裝最新版本的 AWS Command Line Interface (AWS CLI)。

## 安裝 eksctl

請依照安裝 kubectl 中的步驟安裝最新版本的 eksctl。如果為 Amazon EKS 叢集使用 Kubernetes 版本 1.22 或更新版本，請使用 eksctl 0.117.0 以後的版本。

## Amazon EKS 叢集

建立 Amazon EKS 叢集。使用 Amazon EMR on EKS ，將叢集註冊為虛擬叢集。以下是此叢集的要求和考量事項。

- 叢集必須與 EMR Studio 處於相同的 Amazon Virtual Private Cloud (VPC)。
- 叢集必須擁有至少一個私有子網路，以啟動互動端點、連結 Git 型儲存庫以及以私有模式啟動 Application Load Balancer。

- EMR Studio 和用於註冊虛擬叢集的 Amazon EKS 叢集之間必須至少有一個共同的私有子網路。這可確保互動端點在 Studio 工作區中顯示為選項，並啟用從 Studio 到 Application Load Balancer 的連線。

  您可以選擇兩種方法來連接 Studio 和 Amazon EKS 叢集：

  - 建立 Amazon EKS 叢集，並將其與屬於 EMR Studio 的子網路產生關聯。
  - 或者，建立 EMR Studio，並為 Amazon EKS 叢集指定私有子網路。
- Amazon EMR on EKS 互動端點不支援 Amazon EKS 優化的 ARM Amazon Linux AMI。
- 互動式端點可與使用 Kubernetes 版本高達 1.30 的 Amazon EKS 叢集搭配使用。
- 僅支援 Amazon EKS 受管節點群組。

## 為 Amazon EMR on EKS 授予叢集存取權

使用為 Amazon EMR on EKS 授予叢集存取權中的步驟，將 Amazon EMR on EKS 存取權授予給叢集中的特定命名空間。

## 在 Amazon EKS 叢集上啟用 IRSA

若要在 Amazon EKS 叢集上啟用服務帳戶的 IAM 角色 (IRSA)，請按照啟用服務帳戶的 IAM 角色 (IRSA) 中的步驟進行操作。

## 建立 IAM 作業執行角色

必須建立 IAM 角色，才能在 Amazon EMR on EKS 互動端點上執行工作負載。在本文件中，我們將 IAM 角色稱為作業執行角色。此 IAM 角色會同時指派給互動端點容器和您使用 EMR Studio 提交作業時所建立的實際執行容器。您將需要 Amazon EMR on EKS 之作業執行角色的 Amazon Resource Name (ARN)。這需要兩個步驟：

- 建立適用於作業執行的 IAM 角色。
- 更新作業執行角色的信任政策。

## 授予使用者對 Amazon EMR on EKS 的存取權

提出建立互動端點請求的 IAM 實體 (使用者或角色) 也必須擁有下列 Amazon EC2 和 `emr-containers` 許可。請遵循 授予使用者對 Amazon EMR on EKS 的存取權 中描述的步驟，授予這些許可，以允許 Amazon EMR on EKS 建立、管理和刪除安全群組，這些安全群組會將傳入流量限制到互動端點的負載平衡器。

下列 `emr-containers` 許可允許使用者執行基本的互動端點操作：

```
"ec2:CreateSecurityGroup",
"ec2:DeleteSecurityGroup",
"ec2:AuthorizeSecurityGroupEgress",
"ec2:AuthorizeSecurityGroupIngress",
"ec2:RevokeSecurityGroupEgress",
"ec2:RevokeSecurityGroupIngress"

"emr-containers:CreateManagedEndpoint",
"emr-containers:ListManagedEndpoints",
"emr-containers:DescribeManagedEndpoint",
"emr-containers:DeleteManagedEndpoint"
```

## 向 Amazon EMR 註冊 Amazon EKS 叢集

設定虛擬叢集，並將其映射至 Amazon EKS 叢集中您要執行作業的命名空間。僅限 AWS Fargate叢集，請針對 Amazon EMR on EKS 虛擬叢集和 Fargate 設定檔使用相同的命名空間。

如需有關設定 Amazon EMR on EKS 虛擬叢集的資訊，請參閱 向 Amazon EMR 註冊 Amazon EKS 叢集。

## 部署 AWS Load Balancer控制器至 Amazon EKS 叢集

Amazon EKS 叢集需要 An AWS Application Load Balancer。只需為每個 Amazon EKS 叢集設定一個 Application Load Balancer 控制器。如需有關設定 AWS Application Load Balancer控制器的資訊，請參閱《Amazon EKS 使用者指南》中的安裝 AWS Load Balancer控制器附加元件。

# 為虛擬叢集建立互動端點

本主題說明使用 AWS 命令列界面 (AWS CLI) 建立互動式端點的幾種方式，並包含可用組態參數的詳細資訊。

## 使用 `create-managed-endpoint` 命令建立互動端點

在 `create-managed-endpoint` 命令中指定參數，如下所示。Amazon EMR on EKS 支援使用 Amazon EMR 6.7.0 及更高版本建立互動端點。

```
aws emr-containers create-managed-endpoint \
--type JUPYTER_ENTERPRISE_GATEWAY \
```

```
--virtual-cluster-id 1234567890abcdef0xxxxxxx \
--name example-endpoint-name \
--execution-role-arn arn:aws:iam::444455556666:role/JobExecutionRole \
--release-label emr-6.9.0-latest \
--configuration-overrides '{
    "applicationConfiguration": [{
        "classification": "spark-defaults",
        "properties": {
            "spark.driver.memory": "2G"
        }
    }],
    "monitoringConfiguration": {
        "cloudWatchMonitoringConfiguration": {
            "logGroupName": "log_group_name",
            "logStreamNamePrefix": "log_stream_prefix"
        },
        "persistentAppUI": "ENABLED",
        "s3MonitoringConfiguration": {
            "logUri": "s3://my_s3_log_location"
        }
    }
}'
```

如需詳細資訊，請參閱用於建立互動端點的參數。

## 在 JSON 檔案中使用指定參數建立互動端點

1. 建立 `create-managed-endpoint-request.json` 檔案並指定端點所需的參數，如下列 JSON 檔案所示：

```
{
    "name": "MY_TEST_ENDPOINT",
    "virtualClusterId": "MY_CLUSTER_ID",
    "type": "JUPYTER_ENTERPRISE_GATEWAY",
    "releaseLabel": "emr-6.9.0-latest",
    "executionRoleArn": "arn:aws:iam::444455556666:role/JobExecutionRole",
    "configurationOverrides":
    {
        "applicationConfiguration":
        [
            {
                "classification": "spark-defaults",
                "properties":
```

```
                {
                    "spark.driver.memory": "8G"
                }
            }
        ],
        "monitoringConfiguration":
        {
            "persistentAppUI": "ENABLED",
            "cloudWatchMonitoringConfiguration":
            {
                "logGroupName": "my_log_group",
                "logStreamNamePrefix": "log_stream_prefix"
            },
            "s3MonitoringConfiguration":
            {
                "logUri": "s3:///my_s3_log_location"
            }
        }
    }
}
```

2. 搭配使用 `create-managed-endpoint` 命令與儲存在本機或 Amazon S3 中的 `create-managed-endpoint-request.json` 檔案路徑。

```
aws emr-containers create-managed-endpoint \
--cli-input-json  file:///./create-managed-endpoint-request.json --region AWS-Region
```

## 建立互動端點的輸出

在終端中應能看到下列輸出。輸出包括新互動端點的名稱和識別符：

```
{
    "id": "1234567890abcdef0",
    "name": "example-endpoint-name",
    "arn": "arn:aws:emr-containers:us-west-2:111122223333:/
virtualclusters/444455556666/endpoints/444455556666",
    "virtualClusterId": "111122223333xxxxxxxx"
}
```

執行中的 `aws emr-containers create-managed-endpoint` 會建立自我簽署的憑證，它允許 EMR Studio 和互動端點伺服器之間的 HTTPS 通訊。

如果您執行 `create-managed-endpoint` 但尚未完成先決條件，Amazon EMR 會傳回錯誤訊息，其中包含為了繼續而必須採取的動作。

## 用於建立互動端點的參數

主題


- 互動端點的必要參數
- 互動端點的選用參數


### 互動端點的必要參數

建立互動端點時，必須指定下列參數：

**`--type`**

請使用 `JUPYTER_ENTERPRISE_GATEWAY`。這是唯一支援的類型。

**`--virtual-cluster-id`**

您 Amazon EMR on EKS 註冊的虛擬叢集的識別符。

**`--name`**

互動端點的描述性名稱，它可幫助 EMR Studio 使用者從下拉式清單中選擇。

**`--execution-role-arn`**

適用於 Amazon EMR on EKS 的 IAM 作業執行角色的 Amazon Resource Name (ARN)，該角色作為先決條件的一部分而建立。

**`--release-label`**

用於端點的 Amazon EMR 版本的版本標籤。例如 `emr-6.9.0-latest`。透過 Amazon EMR 6.7.0 及更高版本，Amazon EMR on EKS 支援互動端點。

### 互動端點的選用參數

建立互動端點時，也可選擇性地指定下列參數：

**`--configuration-overrides`**

若要覆寫應用程式的預設組態，請提供組態物件。可以使用速記語法，以提供組態或參考 JSON 檔案中物件的組態。

組態物件是由分類、屬性和選用的巢狀組態所組成。屬性由您想要在檔案中覆寫的設定組成。您可以在單一 JSON 物件中，為多個應用程式指定多個分類。可用的組態分類因 Amazon EMR on EKS 版本而異。如需每個 Amazon EMR on EKS 版本可用的組態分類清單，請參閱 Amazon EMR on EKS 發行版本。除了針對每個版本列出的組態分類之外，互動端點還會引入其他分類 `jeg-config`。如需詳細資訊，請參閱Jupyter Enterprise Gateway (JEG) 組態選項。

# 配置互動端點的設定

本節包含一系列主題，涵蓋互動式端點和 Pod 設定的各種組態。這些可讓您監控故障並進行故障診斷、將日誌資訊傳送至 Amazon S3 或 Amazon CloudWatch Logs，或建立互動式端點以指定自訂 Pod 範本。

主題


- 監控 Spark 任務
- 使用互動端點指定自訂 Pod 範本
- 將 JEG Pod 部署到節點群組
- Jupyter Enterprise Gateway (JEG) 組態選項
- 修改 PySpark 工作階段參數
- 具有互動端點的自訂核心映像


## 監控 Spark 任務

如此一來，您就可以監控故障並進行疑難排解，設定互動端點，以便透過端點啟動的作業可以將日誌資訊傳送到 Amazon S3、Amazon CloudWatch Logs 或兩者。以下各章節描述了針對您透過 Amazon EMR on EKS 互動端點啟動的 Spark 作業，如何將 Spark 應用程式日誌傳送至 Amazon S3。

設定 Amazon S3 日誌的 IAM 政策

在您的核心將日誌資料傳送到 Amazon S3 之前，作業執行角色的許可政策必須包含下列許可。將 *`amzn-s3-demo-destination-bucket`* 取代為您的記錄儲存貯體名稱。

JSON

```
{
    "Version": "2012-10-17",
```

```
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:PutObject",
                "s3:GetObject",
                "s3:ListBucket"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-destination-bucket",
                "arn:aws:s3:::amzn-s3-demo-logging-bucket/*"
            ]
        }
    ]
}
```

> **Note**
>
> Amazon EMR on EKS 也可以建立 S3 儲存貯體。如果無法使用 S3 儲存貯體，請在 IAM 政策中包含 `s3:CreateBucket` 許可。

將所需許可授予給執行角色以便將日誌傳送到 S3 儲存貯體之後，您的日誌資料會傳送到以下 Amazon S3 位置。在 `create-managed-endpoint` 請求的 `monitoringConfiguration` 區段中傳遞 `s3MonitoringConfiguration` 時，會發生這種情況。

- 驅動程式日誌 – `logUri/virtual-cluster-id/endpoints/endpoint-id/containers/spark-application-id/spark-application-id-driver/(stderr.gz/stdout.gz)`
- 執行程式日誌 – `logUri/virtual-cluster-id/endpoints/endpoint-id/containers/spark-application-id/executor-pod-name-exec-<Number>/(stderr.gz/stdout.gz)`

> **Note**
>
> Amazon EMR on EKS 不會將端點日誌上傳到 S3 儲存貯體。

## 使用互動端點指定自訂 Pod 範本

您可以建立互動端點，在其中指定驅動程式和執行程式的自訂 Pod 範本。Pod 範本是決定如何執行每個 Pod 的規範。可以使用 Pod 範本檔案來定義 Spark 組態不支援的驅動程式或執行程式 Pod 的組態。Amazon EMR 6.3.0 版本及更高版本目前支援 Pod 範本。

如需有關 Pod 範本的詳細資訊，請參閱《Amazon EMR on EKS 開發指南》中的使用 Pod 範本。

以下範例會示範如何使用 Pod 範本建立互動端點：

```
aws emr-containers create-managed-endpoint \
    --type JUPYTER_ENTERPRISE_GATEWAY \
    --virtual-cluster-id virtual-cluster-id \
    --name example-endpoint-name \
    --execution-role-arn arn:aws:iam::aws-account-id:role/EKSClusterRole \
    --release-label emr-6.9.0-latest \
    --configuration-overrides '{
        "applicationConfiguration": [
        {
            "classification": "spark-defaults",
            "properties": {
                "spark.kubernetes.driver.podTemplateFile": "path/to/driver/
template.yaml",
                "spark.kubernetes.executor.podTemplateFile": "path/to/executor/
template.yaml"
            }
        }]
    }'
```

## 將 JEG Pod 部署到節點群組

JEG (Jupyter Enterprise Gateway) Pod 放置是一項功能，可讓您在特定節點群組上部署互動端點。使用此功能，可以設定互動端點的設定，例如 `instance type`。

### 將 JEG Pod 關聯至受管節點群組

下列組態屬性可讓您指定要部署 JEG Pod 之 Amazon EKS 叢集上的受管節點群組名稱。

```
//payload
--configuration-overrides '{
      "applicationConfiguration": [
```

```
            {
                "classification": "endpoint-configuration",
                "properties": {
                    "managed-nodegroup-name": NodeGroupName
                }
            }
        ]
    }'
```

節點群組必須已將 Kubernetes 標籤 `for-use-with-emr-containers-managed-endpoint-ng=`*`NodeGroupName`* 附接至屬於節點群組的所有節點。若要列出節點群組中擁有此標籤的所有節點，請使用下列命令：

```
kubectl get nodes --show-labels | grep for-use-with-emr-containers-managed-endpoint-ng=NodeGroupName
```

如果上述命令輸出未傳回屬於受管節點群組的節點，則節點群組中沒有附接 `for-use-with-emr-containers-managed-endpoint-ng=`*`NodeGroupName`* Kubernetes 標籤的節點。在這種情況下，遵循以下步驟將標籤附接至節點群組中的節點。

1. 使用下列命令，將 `for-use-with-emr-containers-managed-endpoint-ng=`*`NodeGroupName`* Kubernetes 標籤新增至受管節點群組 *`NodeGroupName`* 中的所有節點：

   ```
   kubectl label nodes --selector eks:nodegroup-name=NodeGroupName for-use-with-emr-containers-managed-endpoint-ng=NodeGroupName
   ```

2. 使用下列命令，確認已正確標記節點：

   ```
   kubectl get nodes --show-labels | grep for-use-with-emr-containers-managed-endpoint-ng=NodeGroupName
   ```

受管節點群組必須與 Amazon EKS 叢集的安全群組相關聯，這通常發生在您使用 `eksctl` 建立叢集和受管節點群組時。您可以使用下列步驟在 AWS 主控台中驗證此項目。

1. 轉至 Amazon EKS 主控台中的叢集。
2. 轉至叢集的「聯網」索引標籤，並記下叢集安全群組。
3. 轉至叢集的「運算」索引標籤，然後按一下受管節點群組名稱。
4. 在受管節點群組的詳細資訊索引標籤下，確認您先前記下的叢集安全群組列在安全群組下。

如果受管節點群組未附接至 Amazon EKS 叢集安全群組，則需要將 `for-use-with-emr-containers-managed-endpoint-sg=ClusterName/NodeGroupName` 標籤附接至節點群組安全群組。使用下列步驟來附接此標籤。

1. 轉至 Amazon EC2 主控台並按一下左側導覽窗格中的安全群組。
2. 按一下核取方塊，選取受管節點群組的安全群組。
3. 在標籤索引標籤下，使用管理標籤按鈕新增標籤 `for-use-with-emr-containers-managed-endpoint-sg=ClusterName/NodeGroupName`。

## 將 JEG Pod 關聯至自我管理節點群組

下列組態屬性可讓您指定要部署 JEG Pod 之 Amazon EKS 叢集上的自我管理或未受管節點群組名稱。

```
//payload
--configuration-overrides '{
      "applicationConfiguration": [
            {
                "classification": "endpoint-configuration",
                "properties": {
                    "self-managed-nodegroup-name": NodeGroupName
                }
            }
        ]
    }'
```

節點群組必須已將 `for-use-with-emr-containers-managed-endpoint-ng=NodeGroupName` Kubernetes 標籤附接至屬於節點群組的所有節點。若要列出節點群組中擁有此標籤的所有節點，請使用下列命令：

```
kubectl get nodes --show-labels | grep for-use-with-emr-containers-managed-endpoint-ng=NodeGroupName
```

如果上述命令輸出未傳回屬於自我管理節點群組的節點，則節點群組中沒有附接 `for-use-with-emr-containers-managed-endpoint-ng=NodeGroupName` Kubernetes 標籤的節點。在這種情況下，遵循以下步驟將標籤附接至節點群組中的節點。

1. 如果使用 `eksctl` 建立自我管理節點群組，請使用下列命令將 `for-use-with-emr-containers-managed-endpoint-ng=`*`NodeGroupName`* Kubernetes 標籤一次新增至自我管理節點群組 *`NodeGroupName`* 中的所有節點。

```
kubectl label nodes --selector alpha.eksctl.io/nodegroup-name=NodeGroupName for-use-with-emr-containers-managed-endpoint-ng=NodeGroupName
```

   如果您未使用 `eksctl` 建立自我管理節點群組，則需要將上述命令中的選取器取代為附接至節點群組中所有節點的不同 Kubernetes 標籤。

2. 使用下列命令，確認已正確標記節點：

```
kubectl get nodes --show-labels | grep for-use-with-emr-containers-managed-endpoint-ng=NodeGroupName
```

自我管理節點群組的安全群組必須已附接 `for-use-with-emr-containers-managed-endpoint-sg=`*`ClusterName`*`/`*`NodeGroupName`* 標籤。使用下列步驟將標籤從 AWS Management Console中附接至安全群組。

1. 導覽至 Amazon EC2 主控台。在左側導覽窗格中，選取安全群組。
2. 選取自我管理節點群組中安全群組旁邊的核取方塊。
3. 在標籤索引標籤下，使用管理標籤按鈕新增標籤 `for-use-with-emr-containers-managed-endpoint-sg=`*`ClusterName`*`/`*`NodeGroupName`*。用適當的值取代 *`ClusterName`* 和 *`NodeGroupName`*。

## 將 JEG Pod 關聯至具有隨需執行個體的受管節點群組

也可以定義其他標籤 (稱為 Kubernetes 標籤選取器)，以指定其他約束或限制，以便在指定節點或節點群組上執行互動端點。以下範例說明如何將隨需 Amazon EC2 執行個體用於 JEG Pod。

```
--configuration-overrides '{
      "applicationConfiguration": [
            {
                "classification": "endpoint-configuration",
                "properties": {
                    "managed-nodegroup-name": NodeGroupName,
                    "node-labels": "eks.amazonaws.com/capacityType:ON_DEMAND"
                }
```

```
            }
        ]
    }'
```

> **Note**
>
> 只能將 `node-labels` 屬性與 `managed-nodegroup-name` 或 `self-managed-nodegroup-name` 屬性搭配使用。

## Jupyter Enterprise Gateway (JEG) 組態選項

Amazon EMR on EKS 使用 Jupyter Enterprise Gateway (JEG) 開啟互動端點。建立端點時，可以為允許列出的 JEG 組態設定下列值。

- **`RemoteMappingKernelManager.cull_idle_timeout`** - 以秒為單位的逾時值 (整數)，在此時間之後核心會被視為閒置並準備好被剔除。`0` 或更低的值會停用剔除。對於網路連線不佳的使用者而言，短暫的逾時可能會導致核心被剔除。
- **`RemoteMappingKernelManager.cull_interval`** - 以秒為單位的時間間隔 (整數)，會根據該時間值檢查超過剔除逾時值的閒置核心。

## 修改 PySpark 工作階段參數

從 Amazon EMR on EKS 6.9.0 開始，可透過在 EMR 筆記本儲存格中執行 `%%configure` 魔術命令，在 Amazon EMR Studio 中調整與 PySpark 工作階段相關聯的 Spark 組態。

下列範例顯示了範例承載，可用來修改 Spark 驅動程式和執行程式的記憶體、核心和其他屬性。對於 `conf` 設定，可設定 Apache Spark 組態文件中提到的任何 Spark 組態。

```
%%configure -f
{
  "driverMemory": "16G",
  "driverCores" 4,
  "executorMemory" : "32G"
  "executorCores": 2,
  "conf": {
     "spark.dynamicAllocation.maxExecutors" : 10,
     "spark.dynamicAllocation.minExecutors": 1
  }
```

```
}
```

下列範例顯示範例承載，可用於將檔案、PyFiles 和 jar 相依性新增至 Spark 執行期。

```
%%configure -f
{
  "files": "s3://amzn-s3-demo-bucket-emr-eks/sample_file.txt",
  "pyFiles": : "path-to-python-files",
  "jars" : "path-to-jars
}
```

## 具有互動端點的自訂核心映像

為了確保在 Amazon EMR Studio 中執行互動式工作負載時具有應用程式的正確相依性，您可以為互動端點自訂 Docker 映像檔，並執行自訂的基礎核心映像。若要建立互動端點，並將其與自訂 Docker 映像檔相連，請執行以下步驟。

> **Note**
>
> 只能覆寫基礎映像。無法新增核心映像類型。

1. 建立並發布自訂的 Docker 映像檔。基礎映像包含 Spark 執行期和隨之一起執行的筆記本核心。若要建立映像，可遵循 如何自訂 Docker 映像檔 中的步驟 1 到 4。在步驟 1 中，Docker 檔案中的基礎映像 URI 必須使用 `notebook-spark` 代替 `spark`。

   ```
   ECR-registry-account.dkr.ecr.Region.amazonaws.com/notebook-spark/container-image-tag
   ```

   如需如何選取 AWS 區域 和容器映像標籤的詳細資訊，請參閱 選取基礎映像 URI 的詳細資訊。
2. 建立可與自訂映像搭配使用的互動端點。

   a. 使用下列內容建立 JSON 檔案 `custom-image-managed-endpoint.json`。此範例使用 Amazon EMR 6.9.0 版。

      Example

      ```
      {
          "name": "endpoint-name",
      ```

```
    "virtualClusterId": "virtual-cluster-id",
    "type": "JUPYTER_ENTERPRISE_GATEWAY",
    "releaseLabel": "emr-6.9.0-latest",
    "executionRoleArn": "execution-role-arn",
    "configurationOverrides": {
        "applicationConfiguration": [
            {
                "classification": "jupyter-kernel-overrides",
                "configurations": [
                    {
                        "classification": "python3",
                        "properties": {
                            "container-image": "123456789012.dkr.ecr.us-
west-2.amazonaws.com/custom-notebook-python:latest"
                        }
                    },
                    {
                        "classification": "spark-python-kubernetes",
                        "properties": {
                            "container-image": "123456789012.dkr.ecr.us-
west-2.amazonaws.com/custom-notebook-spark:latest"
                        }
                    }
                ]
            }
        ]
    }
}
```

b. 使用 JSON 檔案中指定的組態建立互動端點，如下列範例所示。如需詳細資訊，請參閱使用 `create-managed-endpoint` 命令建立互動端點。

```
aws emr-containers create-managed-endpoint --cli-input-json custom-image-
managed-endpoint.json
```

3. 透過 EMR Studio 連接至互動端點。如需詳細資訊和完成的步驟，請參閱 AWS Workshop Studio 文件的 Amazon EMR on EKS 區段中的從 Studio 連線。

# 監控互動端點

使用 Amazon EMR on EKS 6.10 版及更高版本，互動端點會發出 Amazon CloudWatch 指標，用於監控核心生命週期操作並進行疑難排解。指標是由互動式用戶端 (例如 EMR Studio 或自行託管的

Jupyter 筆記本) 觸發。互動端點支援的每個操作都有與其相關聯的指標。操作會建模為每個指標的維度，如下表所示。互動端點發出的指標會顯示在您帳戶中的自訂命名空間 EMRContainers 下方。

| 指標 | 描述 | 單位 |
| --- | --- | --- |
| RequestCount | 互動端點所處理之操作的請求累計數目。 | 計數 |
| RequestLatency | 從請求到達互動端點到互動端點傳送回應的時間。 | 毫秒 |
| 4XXError | 當操作的請求導致在處理過程中出現 4xx 錯誤時發出。 | 計數 |
| 5XXError | 當操作的請求導致 5Xxx 伺服器端錯誤時發出。 | 計數 |
| KernelLaunchSuccess | 僅適用於 CreateKernel 操作。它表示成功執行且包含此請求的核心啟動累計次數。 | 計數 |
| KernelLaunchFailure | 僅適用於 CreateKernel 操作。它表示發生失敗且包含此請求的核心啟動累計次數。 | 計數 |

每個互動端點指標都附接有下列兩個維度：

- **`ManagedEndpointId`** - 互動端點的識別符
- **`OperationName`** - 互動式用戶端觸發的操作

**`OperationName`** 維度的可能值如下表所示：

| **`operationName`** | 操作說明 |
| --- | --- |
| `CreateKernel` | 互動端點啟動核心的請求。 |

| operationName | 操作說明 |
| --- | --- |
| ListKernels | 互動端點列出核心 (先前已使用相同的工作階段字符啟動這些核心) 的請求。 |
| GetKernel | 互動端點取得先前已啟動之特定核心詳細資訊的請求。 |
| ConnectKernel | 互動端點在筆記本用戶端與核心之間建立連線的請求。 |
| ConfigureKernel | 在 pyspark 核心上發布 %%configure magic request。 |
| ListKernelSpecs | 互動端點列出可用核心規範的請求。 |
| GetKernelSpec | 互動端點取得先前已啟動之核心規範的請求。 |
| GetKernelSpecResource | 互動端點取得先前已啟動之核心規範相關聯特定資源的請求。 |

## 範例

### 若要存取在特定日期為互動端點啟動的核心總數：

1. 選取自訂命名空間：EMRContainers
2. 選取您的 ManagedEndpointId、OperationName – CreateKernel。
3. 具有統計值 SUM 和期限 1 day 的 RequestCount 指標將提供過去 24 小時內發出的所有核心啟動請求。
4. 具有統計值 SUM 和期限 1 day 的 KernelLaunchSuccess 指標將提供過去 24 小時內發出的所有成功的核心啟動請求。

### 若要存取特定日期互動端點的核心失敗次數：

1. 選取自訂命名空間：EMRContainers
2. 選取您的 ManagedEndpointId、OperationName – CreateKernel。

3. 具有統計值 SUM 和期限 1 day 的 KernelLaunchFailure 指標將提供過去 24 小時內發出的所有失敗的核心啟動請求。也可選擇 4XXError 和 5XXError 指標來了解發生了什麼類型的核心啟動失敗。

# 使用自助託管的 Jupyter 筆記本

可以在 Amazon EC2 執行個體或您自己的 Amazon EKS 叢集上託管和管理 Jupyter 或 JupyterLab 筆記本，作為自助託管的 Jupyter 筆記本。然後，使用自助託管的 Jupyter 筆記本執行互動式工作負載。以下各章節將逐步介紹在 Amazon EKS 叢集上設定和部署自助託管的 Jupyter 筆記本的程序。

在 EKS 叢集上建立自助託管的 Jupyter 筆記本

- 建立安全群組
- 建立 Amazon EMR on EKS 互動端點
- 擷取互動端點的閘道伺服器 URL
- 擷取驗證字符以連接到互動端點
- 範例：部署 JupyterLab 筆記本
- 刪除自助託管的 Jupyter 筆記本

## 建立安全群組

必須先建立安全群組來控制筆記本與互動端點之間的流量，才能建立互動端點並執行自助託管的 Jupyter 或 JupyterLab 筆記本。若要使用 Amazon EC2 主控台或 Amazon EC2 SDK 建立安全群組，請參閱《Amazon EC2 使用者指南》中的建立安全群組中的步驟。應該在要部署筆記本伺服器的 VPC 中建立安全群組。

若要遵循本指南中的範例，請使用與 Amazon EKS 叢集相同的 VPC。如果想要將筆記本託管在與 Amazon EKS 叢集的 VPC 不同的 VPC 中，則可能需要在這兩個 VPC 之間建立對等互連。如需在兩個 VPC 之間建立對等互連的步驟，請參閱《Amazon VPC 入門指南》中的建立 VPC 對等互連。

您需要安全群組的 ID，才能在下一個步驟中建立 Amazon EMR on EKS 互動端點。

## 建立 Amazon EMR on EKS 互動端點

為筆記本建立安全群組之後，請使用 為虛擬叢集建立互動端點 中提供的步驟建立互動端點。必須提供在 建立安全群組 中為筆記本建立的安全群組 ID 。

在下列組態覆寫設定中插入安全 ID 來取代 *your-notebook-security-group-id*：

```
--configuration-overrides '{
    "applicationConfiguration": [
        {
            "classification": "endpoint-configuration",
            "properties": {
                "notebook-security-group-id": "your-notebook-security-group-id"
            }
        }
    ],
    "monitoringConfiguration": {
    ...'
```

## 擷取互動端點的閘道伺服器 URL

建立互動端點後，請使用 AWS CLI中的 `describe-managed-endpoint` 命令來擷取閘道伺服器 URL。需要此 URL 才能將筆記本連接到端點。閘道伺服器 URL 是私有端點。

```
aws emr-containers describe-managed-endpoint \
--region region \
--virtual-cluster-id virtualClusterId \
--id endpointId
```

最初，您的端點處於 CREATING 狀態。幾分鐘後，它會轉換為 ACTIVE 狀態。當端點為 ACTIVE 時，即已準備好可供使用。

記下 `aws emr-containers describe-managed-endpoint` 命令從作用中端點傳回的 `serverUrl` 屬性。當您部署自助託管的 Jupyter 或 JupyterLab 筆記本時，需要此 URL 才能將筆記本連接到端點。

## 擷取驗證字符以連接到互動端點

若要從 Jupyter 或 JupyterLab 筆記本連接到互動端點，必須使用 `GetManagedEndpointSessionCredentials` API 產生工作階段字符。字符充當驗證證明，以連接到互動端點伺服器。

下面的輸出範例將更詳細地說明以下命令。

```
aws emr-containers get-managed-endpoint-session-credentials \
```

```
--endpoint-identifier endpointArn \
--virtual-cluster-identifier virtualClusterArn \
--execution-role-arn executionRoleArn \
--credential-type "TOKEN" \
--duration-in-seconds durationInSeconds \
--region region
```

***endpointArn***

端點的 ARN。可以在 `describe-managed-endpoint` 呼叫的結果中尋找 ARN。

***virtualClusterArn***

虛擬叢集的 ARN。

***executionRoleArn***

執行角色的 ARN。

***durationInSeconds***

字符有效的持續時間 (以秒為單位)。預設持續時間為 15 分鐘 (`900`)，最長為 12 小時 (`43200`)。

***region***

與端點相同的區域。

輸出應類似以下範例。請記下您在部自助託管的 Jupyter 或 JupyterLab 筆記本時將使用的 *session-token* 值。

```
{
    "id": "credentialsId",
    "credentials": {
        "token": "session-token"
    },
    "expiresAt": "2022-07-05T17:49:38Z"
}
```

## 範例：部署 JupyterLab 筆記本

完成上述步驟後，可以嘗試此範例程序，將 JupyterLab 筆記本部署到具有互動端點的 Amazon EKS 叢集中。

1. 建立命名空間，以執行筆記本伺服器。
2. 在本機建立檔案 `notebook.yaml`，其中具有以下內容。檔案內容如下所述。

```
apiVersion: v1
kind: Pod
metadata:
  name: jupyter-notebook
  namespace: namespace
spec:
  containers:
  - name: minimal-notebook
    image: jupyter/all-spark-notebook:lab-3.1.4 # open source image
    ports:
    - containerPort: 8888
    command: ["start-notebook.sh"]
    args: ["--LabApp.token=''"]
    env:
    - name: JUPYTER_ENABLE_LAB
      value: "yes"
    - name: KERNEL_LAUNCH_TIMEOUT
      value: "400"
    - name: JUPYTER_GATEWAY_URL
      value: "serverUrl"
    - name: JUPYTER_GATEWAY_VALIDATE_CERT
      value: "false"
    - name: JUPYTER_GATEWAY_AUTH_TOKEN
      value: "session-token"
```

如果您要將 Jupyter 筆記本部署到僅限 Fargate 的叢集，請使用 `role` 標籤來標記 Jupyter Pod，如下列範例所示：

```
...
metadata:
  name: jupyter-notebook
  namespace: default
  labels:
    role: example-role-name-label
spec:
            ...
```

***namespace***

在其中部署筆記本的 Kubernetes 命名空間。

***serverUrl***

`describe-managed-endpoint` 命令在 [擷取互動端點的閘道伺服器 URL](#) 中傳回的 `serverUrl` 屬性。

***session-token***

`get-managed-endpoint-session-credentials` 命令在 [擷取驗證字符以連接到互動端點](#) 中傳回的 `session-token` 屬性。

**KERNEL_LAUNCH_TIMEOUT**

互動端點等待核心進入 RUNNING 狀態的時間 (以秒為單位)。將核心啟動逾時設定為適當的值 (最多 400 秒)，以確保有足夠的時間來完成核心啟動。

**KERNEL_EXTRA_SPARK_OPTS**

或者，可以為 Spark 核心傳遞額外的 Spark 組態。將具有值的此環境變數設定為 Spark 組態屬性，如以下範例所示：

```
- name: KERNEL_EXTRA_SPARK_OPTS
  value: "--conf spark.driver.cores=2
          --conf spark.driver.memory=2G
          --conf spark.executor.instances=2
          --conf spark.executor.cores=2
          --conf spark.executor.memory=2G
          --conf spark.dynamicAllocation.enabled=true
          --conf spark.dynamicAllocation.shuffleTracking.enabled=true
          --conf spark.dynamicAllocation.minExecutors=1
          --conf spark.dynamicAllocation.maxExecutors=5
          --conf spark.dynamicAllocation.initialExecutors=1
          "
```

3. 將 Pod 規範部署至 Amazon EKS 叢集。

```
kubectl apply -f notebook.yaml -n namespace
```

這將啟動已連接到 Amazon EMR on EKS 互動端點的最小 JupyterLab 筆記本。請等待，直到 Pod 處於 RUNNING 狀態。可以使用下列命令來檢查其狀態：

```
kubectl get pod jupyter-notebook -n namespace
```

當 Pod 準備就緒時，`get pod` 命令會傳回類似以下輸出：

```
NAME              READY  STATUS   RESTARTS  AGE
jupyter-notebook  1/1    Running  0         46s
```

4. 將筆記本安全群組附接至排程筆記本所在的節點。

   a. 首先，使用 `describe pod` 命令識別在其中排程 `jupyter-notebook` Pod 的節點。

   ```
   kubectl describe pod jupyter-notebook -n namespace
   ```

   b. 在以下網址開啟 Amazon EKS 主控台：https://console.aws.amazon.com/eks/home#/clusters。

   c. 導覽至 Amazon EKS 叢集的運算索引標籤，然後選取 `describe pod` 命令識別的節點。選取節點的執行個體 ID。

   d. 從動作功能表中，選取安全性 > 變更安全群組，以附接您在 建立安全群組 中建立的安全群組。

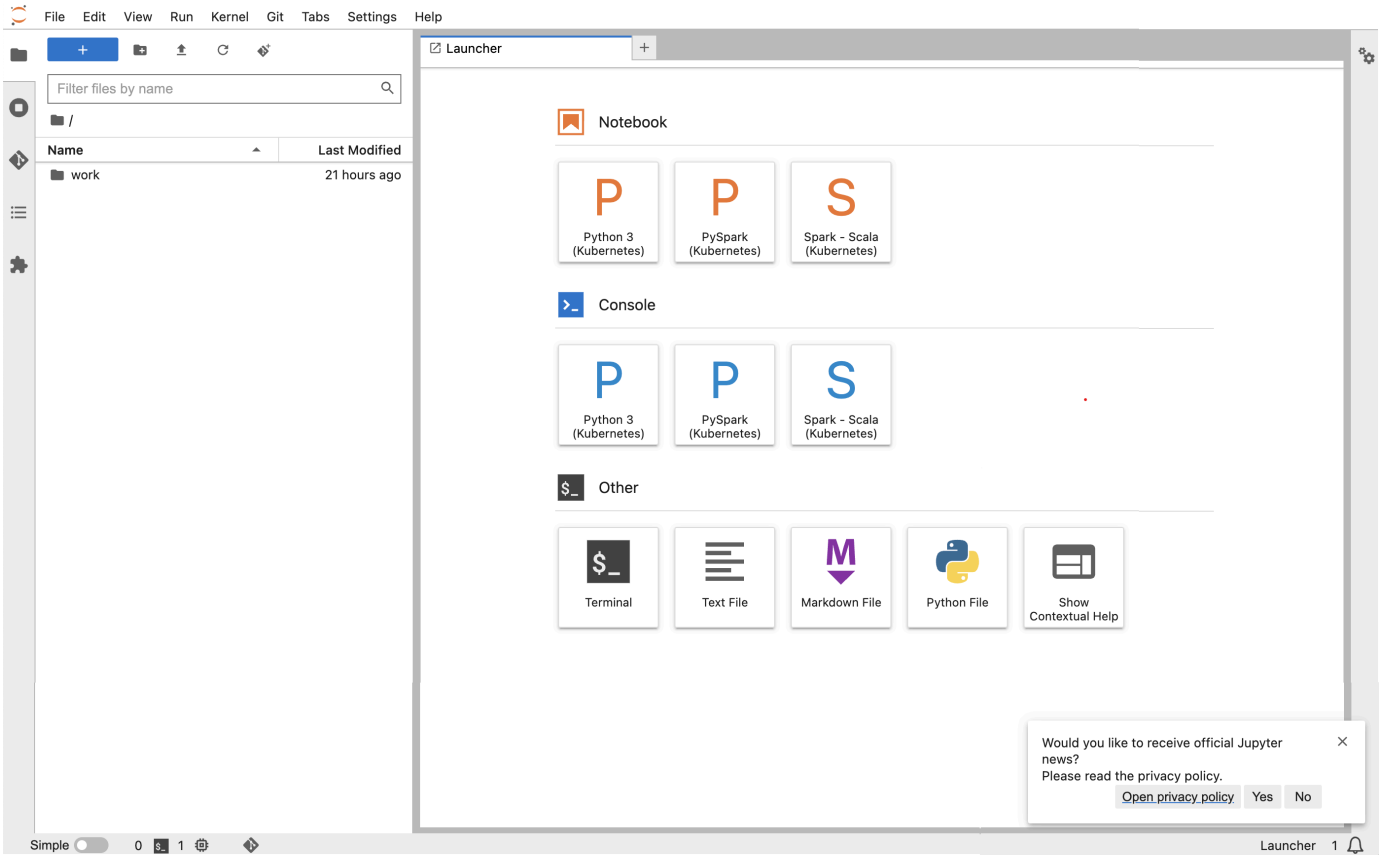
   e. 如果您要在 上部署 Jupyter 筆記本 Pod AWS Fargate，請建立 `SecurityGroupPolicy` 以套用到具有角色標籤的 Jupyter 筆記本 Pod：

   ```
   cat >my-security-group-policy.yaml <<EOF
   apiVersion: vpcresources.k8s.aws/v1beta1
   kind: SecurityGroupPolicy
   metadata:
     name: example-security-group-policy-name
     namespace: default
   spec:
     podSelector:
       matchLabels:
         role: example-role-name-label
     securityGroups:
       groupIds:
         - your-notebook-security-group-id
   EOF
   ```

5. 現在，執行 port-forward，以便您可以在本機存取 JupyterLab 介面：

```
kubectl port-forward jupyter-notebook 8888:8888 -n namespace
```

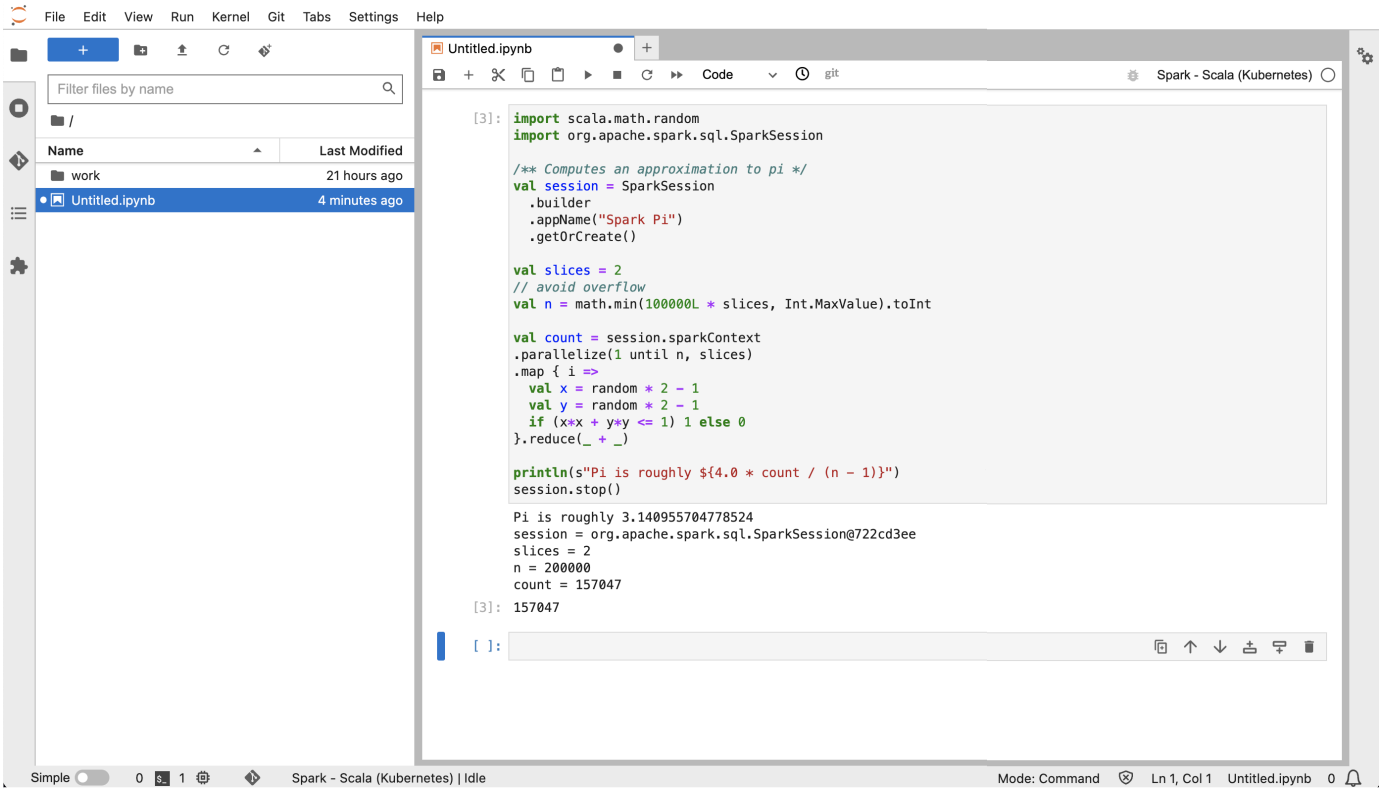
執行後，導覽至您的本機瀏覽器並造訪 `localhost:8888` 以查看 JupyterLab 介面：




6. 從 JupyterLab 中建立新的 Scala 筆記本。以下是程式碼片段範例，您可以執行它以接近 Pi 的值：

```
import scala.math.random
import org.apache.spark.sql.SparkSession

/** Computes an approximation to pi */
val session = SparkSession
  .builder
  .appName("Spark Pi")
  .getOrCreate()

val slices = 2
// avoid overflow
val n = math.min(100000L * slices, Int.MaxValue).toInt

val count = session.sparkContext
.parallelize(1 until n, slices)
```

```
.map { i =>
  val x = random * 2 - 1
  val y = random * 2 - 1
  if (x*x + y*y <= 1) 1 else 0
}.reduce(_ + _)

println(s"Pi is roughly ${4.0 * count / (n - 1)}")
session.stop()
```




## 刪除自助託管的 Jupyter 筆記本

當您準備好刪除自助託管的筆記本時，也可以刪除互動端點和安全群組。請依下列順序執行動作：

1. 使用下列命令來刪除 `jupyter-notebook` Pod：

```
kubectl delete pod jupyter-notebook -n namespace
```

2. 然後，使用 `delete-managed-endpoint` 命令刪除互動端點。如需有關刪除互動端點的步驟，請參閱 [刪除互動端點]。最初，您的端點將處於 TERMINATING 狀態。清除所有資源後，它會轉換為 TERMINATED 狀態。

3. 如果不打算將您在 建立安全群組 中建立的筆記本安全群組用於其他 Jupyter 筆記本部署，則可以將其刪除。如需詳細資訊，請參閱《Amazon EC2 使用者指南》中的刪除安全群組。

# 使用 CLI 命令取得互動式端點的相關資訊

本主題涵蓋除了 `create-managed-endpoint` 之外的互動端點上的受支援操作。

## 擷取互動端點詳細資訊

建立互動式端點之後，您可以使用 `describe-managed-endpoint` AWS CLI 命令擷取其詳細資訊。插入您自己的 *`managed-endpoint-id`*、*`virtual-cluster-id`* 以及 *`region`* 的值：

```
aws emr-containers describe-managed-endpoint --id managed-endpoint-id \
 --virtual-cluster-id virtual-cluster-id --region region
```

輸出看起來類似於以下內容，包含指定的端點，例如 ARN、ID 和名稱。

```
{
   "id": "as3ys2xxxxxxx",
   "name": "endpoint-name",
    "arn": "arn:aws:emr-containers:us-east-1:1828xxxxxxxx:/virtualclusters/
lbhl6kwwyoxxxxxxxxxxxxxxx/endpoints/as3ysxxxxxxxx",
    "virtualClusterId": "lbhl6kwwyoxxxxxxxxxxxxxxx",
    "type": "JUPYTER_ENTERPRISE_GATEWAY",
    "state": "ACTIVE",
    "releaseLabel": "emr-6.9.0-latest",
   "executionRoleArn": "arn:aws:iam::1828xxxxxxxx:role/RoleName",
    "certificateAuthority": {
        "certificateArn": "arn:aws:acm:us-east-1:1828xxxxxxxx:certificate/zzzzzzzz-
e59b-4ed0-aaaa-bbbbbbbbbbbb",
        "certificateData": "certificate-data"
    },
    "configurationOverrides": {
        "applicationConfiguration": [
            {
                "classification": "spark-defaults",
                "properties": {
                    "spark.driver.memory": "8G"
                }
            }
        ],
```

```
        "monitoringConfiguration": {
            "persistentAppUI": "ENABLED",
            "cloudWatchMonitoringConfiguration": {
                "logGroupName": "log-group-name",
                "logStreamNamePrefix": "log-stream-name-prefix"
            },
            "s3MonitoringConfiguration": {
                "logUri": "s3-bucket-name"
            }
        }
    },
   "serverUrl": "https://internal-k8s-namespace-ingressa-aaaaaaaaaa-
zzzzzzzzzz.us-east-1.elb.amazonaws.com:18888 (https://internal-k8s-nspluto-
ingressa-51e860abbd-1620715833.us-east-1.elb.amazonaws.com:18888/)",
    "createdAt": "2022-09-19T12:37:49+00:00",
    "securityGroup": "sg-aaaaaaaaaaaaaa",
    "subnetIds": [
        "subnet-11111111111",
        "subnet-22222222222",
        "subnet-33333333333"
    ],
    "stateDetails": "Endpoint created successfully. It took 3 Minutes 15 Seconds",
    "tags": {}
 }
```

## 列出與虛擬叢集關聯的所有互動端點

使用 `list-managed-endpoints` AWS CLI 命令來擷取與指定虛擬叢集相關聯的所有互動式端點清單。將 `virtual-cluster-id` 取代為虛擬叢集 ID。

```
aws emr-containers list-managed-endpoints --virtual-cluster-id virtual-cluster-id
```

`list-managed-endpoint` 命令的輸出如下所示：

```
{
    "endpoints": [{
        "id": "as3ys2xxxxxxx",
        "name": "endpoint-name",
        "arn": "arn:aws:emr-containers:us-east-1:1828xxxxxxxx:/virtualclusters/
lbhl6kwwyoxxxxxxxxxxxxxxx/endpoints/as3ysxxxxxxxx",
        "virtualClusterId": "lbhl6kwwyoxxxxxxxxxxxxxxx",
        "type": "JUPYTER_ENTERPRISE_GATEWAY",
```

```
        "state": "ACTIVE",
        "releaseLabel": "emr-6.9.0-latest",
        "executionRoleArn": "arn:aws:iam::1828xxxxxxxx:role/RoleName",
        "certificateAuthority": {
            "certificateArn": "arn:aws:acm:us-east-1:1828xxxxxxxx:certificate/zzzzzzzz-
e59b-4ed0-aaaa-bbbbbbbbbbbb",
            "certificateData": "certificate-data"
        },
        "configurationOverrides": {
            "applicationConfiguration": [{
                "classification": "spark-defaults",
                "properties": {
                    "spark.driver.memory": "8G"
                }
            }],
            "monitoringConfiguration": {
                "persistentAppUI": "ENABLED",
                "cloudWatchMonitoringConfiguration": {
                    "logGroupName": "log-group-name",
                    "logStreamNamePrefix": "log-stream-name-prefix"
                },
                "s3MonitoringConfiguration": {
                    "logUri": "s3-bucket-name"
                }
            }
        },
        "serverUrl": "https://internal-k8s-namespace-ingressa-aaaaaaaaaa-
zzzzzzzzzz.us-east-1.elb.amazonaws.com:18888 (https://internal-k8s-nspluto-
ingressa-51e860abbd-1620715833.us-east-1.elb.amazonaws.com:18888/)",
        "createdAt": "2022-09-19T12:37:49+00:00",
        "securityGroup": "sg-aaaaaaaaaaaaaa",
        "subnetIds": [
            "subnet-11111111111",
            "subnet-22222222222",
            "subnet-33333333333"
        ],
        "stateDetails": "Endpoint created successfully. It took 3 Minutes 15 Seconds",
        "tags": {}
    }]
}
```

# 刪除互動端點

若要刪除與 Amazon EMR on EKS 虛擬叢集相關聯的互動式端點，請使用 `delete-managed-endpoint` AWS CLI 命令。刪除互動端點時，Amazon EMR on EKS 會移除為該端點建立的預設安全群組。

為命令指定下列參數的值：

- --id: 您要刪除之互動端點的識別符。
- --virtual-cluster-id – 與您要刪除之互動端點相關聯的虛擬叢集識別符。這與建立互動端點時指定的虛擬叢集 ID 相同。

```
aws emr-containers delete-managed-endpoint --id managed-endpoint-id --virtual-cluster-id virtual-cluster-id
```

該命令會傳回類似以下內容的輸出，以確認您已刪除互動端點：

```
{
    "id":"8gai4l4exxxxx",
    "virtualClusterId":"0b0qvauoy3ch1nqodxxxxxxxx"
}
```

# 使用 Amazon EMR on EKS 將資料上傳至 Amazon S3 Express One Zone

透過 Amazon EMR 7.2.0 版及更高版本，您可以在執行任務和工作負載時搭配 Amazon Amazon S3 EMR on EKS，以提高效能。S3 Express One Zone 是一種高效能的單區域 Amazon S3 儲存類別，可為對大多數延遲敏感的應用程式提供一致的單一位數毫秒資料存取。在發布時，S3 Express One Zone 提供 Amazon S3 中最低延遲和最高效能的雲端物件儲存。

## 先決條件

您必須先具備下列先決條件，才能將 S3 Express One Zone 與 Amazon EMR on EKS 搭配使用：

- 已完成設定 Amazon EMR on EKS。
- 設定 Amazon EMR on EKS 之後，請建立虛擬叢集。

## 開始使用 S3 Express One Zone

請依照下列步驟開始使用 S3 Express One Zone

1. 將 `CreateSession` 許可新增至您的任務執行角色。當 S3 Express One Zone 最初在 S3 物件PUT上執行 `GET`、 `LIST`或 等動作時，儲存類別`CreateSession`會代表您呼叫 。以下是如何授予 `CreateSession`許可的範例。

   JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Resource":
 "arn:aws:s3express:<AWS_REGION>:<ACCOUNT_ID>:bucket/DOC-EXAMPLE-BUCKET",
            "Action": [
                "s3express:CreateSession"
            ]
        }
    ]
```

```
}
```

2. 您必須使用 Apache Hadoop 連接器 S3A 來存取 S3 Express 儲存貯體，因此請將 Amazon S3 URIs 變更為使用s3a結構描述來使用連接器。如果他們不使用 結構描述，您可以變更用於 s3和結構s3n描述的檔案系統實作。

   若要變更 s3 結構描述，請指定下列叢集組態：

```
[
  {
    "Classification": "core-site",
    "Properties": {
      "fs.s3.impl": "org.apache.hadoop.fs.s3a.S3AFileSystem",
      "fs.AbstractFileSystem.s3.impl": "org.apache.hadoop.fs.s3a.S3A"
    }
  }
]
```

   若要變更 s3n 配置，請指定下列叢集組態：

```
[
  {
    "Classification": "core-site",
    "Properties": {
      "fs.s3n.impl": "org.apache.hadoop.fs.s3a.S3AFileSystem",
      "fs.AbstractFileSystem.s3n.impl": "org.apache.hadoop.fs.s3a.S3A"
    }
  }
]
```

3. 在您的 spark-submit 組態中，使用 Web 身分憑證提供者。

```
"spark.hadoop.fs.s3a.aws.credentials.provider=com.amazonaws.auth.WebIdentityTokenCredential
```

# 監控任務

您可以使用 Amazon CloudWatch Events 來追蹤在 Amazon EMR on EKS 虛擬叢集上執行的任務。您可以使用事件來追蹤在虛擬叢集上執行之作業的活動和運作狀態。以下主題會示範如何有效設定監控，以維護資源的運作狀態。

主題


- 使用 Amazon CloudWatch Events 監控作業
- 使用 CloudWatch Events 來自動化 Amazon EMR on EKS
- 範例：設定調用 Lambda 的規則
- 使用 Amazon CloudWatch Events 透過重試政策來監控作業的驅動程式 Pod


## 使用 Amazon CloudWatch Events 監控作業

當作業執行狀態變更時，Amazon EMR on EKS 會發出事件。每個事件都會提供資訊，例如事件發生的日期和時間，以及有關事件的進一步詳細資訊，例如虛擬叢集 ID 和受影響的作業執行 ID。

您可以使用事件來追蹤在虛擬叢集上執行之作業的活動和運作狀態。也可以使用 Amazon CloudWatch Events 件來定義當作業執行產生符合您指定模式的事件時要採取的動作。事件對於在作業執行的生命週期中監控特定事件非常有用。例如，可以監控作業執行狀態從 `submitted` 變更為 `running` 的時間。如需 CloudWatch Events 的詳細資訊，請參閱 Amazon EventBridge 使用者指南。

下表會列出 Amazon EMR on EKS 事件，以及該事件的狀態或狀態變更、該事件嚴重性以及事件訊息。每個事件表示做為自動傳送到事件串流的 JSON 物件。JSON 物件包含有關該事件進一步的詳細資訊。當您使用 CloudWatch Events 為事件處理設定規則時，JSON 物件尤其重要，因為規則要試圖符合 JSON 物件中的模式。如需詳細資訊，請參閱《Amazon EventBridge 使用者指南》中的 Amazon EventBridge 事件模式和 Amazon EMR on EKS 事件。 EventBridge

作業執行狀態變更事件

| State | 嚴重性 | 訊息 |
| --- | --- | --- |
| SUBMITTED | INFO | 作業執行 *JobRunId* (*JobRunName* ) 已於 UTC 時間 *Time* 順利提交至虛擬叢集 *VirtualClusterId* 。 |

| State | 嚴重性 | 訊息 |
| --- | --- | --- |
| RUNNING (執行中) | INFO | 虛擬叢集 *VirtualClusterId* 中的作業執行 *JobRunId* (*JobRunName* ) 已於 *Time* 開始執行。 |
| COMPLETED (已完成) | INFO | 虛擬叢集 *VirtualClusterId* 中的作業執行 *jobRunId* (*JobRunName* ) 已於 *Time* 完成。作業執行於 *Time* 開始執行並花費 *Num* 分鐘完成。 |
| CANCELLED (已取消) | WARN | 虛擬叢集 *VirtualClusterId* 中的作業執行 *JobRunId* (*JobRunName* ) 的取消請求已於 *Time* 成功，且作業執行現已取消。 |
| 失敗 | ERROR | 虛擬叢集 *VirtualClusterId* 中的作業執行 *JobRunId* (*JobRunName* ) 已於 *Time* 失敗。 |

# 使用 CloudWatch Events 來自動化 Amazon EMR on EKS

您可以使用 Amazon CloudWatch Events 自動化您的 AWS 服務，以回應系統事件，例如應用程式可用性問題或資源變更。來自 AWS 服務的事件會以近乎即時的方式交付至 CloudWatch Events。您可編寫簡單的規則，來指示您在意的事件，以及當事件符合規則時所要自動執行的動作。可以自動觸發的動作如下：

- 叫用 AWS Lambda 函數
- 調用 Amazon EC2 執行命令
- 將事件轉傳至 Amazon Kinesis Data Streams
- 啟用 AWS Step Functions 狀態機器
- 通知 Amazon Simple Notification Service (SNS) 主題或 Amazon Simple Queue Service (SQS) 佇列

下列是 CloudWatch Events 與 Amazon EMR on EKS 搭配使用的部分範例：

- 在作業執行成功時啟動 Lambda 函數
- 在作業執行失敗時通知 Amazon SNS 主題

針對 SUBMITTED、RUNNING、CANCELLED、FAILED 和 COMPLETED 狀態變更，Amazon EMR on EKS 會產生 "detail-type:" "EMR Job Run State Change" CloudWatch Events 事件。

## 範例：設定調用 Lambda 的規則

使用下列步驟設定 CloudWatch Events 規則，該規則會在發生「EMR 作業執行狀態變更」事件時調用 Lambda。

```
aws events put-rule \
--name cwe-test \
--event-pattern '{"detail-type": ["EMR Job Run State Change"]}'
```

新增您擁有的 Lambda 函數作為新目標，並授 CloudWatch Events 叫用 Lambda 函數的許可，如下所示。使用您的帳戶 ID 取代 *123456789012*。

```
aws events put-targets \
--rule cwe-test \
--targets Id=1,Arn=arn:aws:lambda:us-east-1:123456789012:function:MyFunction
```

```
aws lambda add-permission \
--function-name MyFunction \
--statement-id MyId \
--action 'lambda:InvokeFunction' \
--principal events.amazonaws.com
```

> **Note**
>
> 無法撰寫依賴於通知事件的順序或存在情況的程式，因為它們可能會被移出序列或遺漏。盡可能發出事件。

## 使用 Amazon CloudWatch Events 透過重試政策來監控作業的驅動程式 Pod

使用 CloudWatch Events，可監控已在具有重試政策的作業中建立的驅動程式 Pod。如需詳細資訊，請參閱本指南中的 使用重試政策監控作業。

# 管理虛擬叢集

虛擬叢集是 Amazon EMR 註冊的 Kubernetes 命名空間。您可以建立、描述、列出和刪除虛擬叢集。它們不會耗用系統中的任何其他資源。單一虛擬叢集映射至單一 Kubernetes 命名空間。鑑於此關係，您可以使用與建立 Kubernetes 命名空間模型相同的方式來建立虛擬叢集的模型，以符合您的需求。請參閱 Kubernetes 概念概觀文件中的可能使用案例。

若要使用 Amazon EKS 叢集上的 Kubernetes 命名空間註冊 Amazon EMR，您需要 EKS 叢集的名稱，以及為執行工作負載而設定的命名空間。Amazon EMR 中的這些已註冊叢集稱為虛擬叢集，因為它們不會管理實體運算或儲存，而是指向在其中排程工作負載的 Kubernetes 命名空間。

> **Note**
>
> 在建立虛擬叢集之前，必須先完成 設定 Amazon EMR on EKS 中的步驟 1-8。

主題


- 建立虛擬叢集
- 列出虛擬叢集
- 描述虛擬叢集
- 刪除虛擬叢集
- 虛擬叢集狀態


## 建立虛擬叢集

透過使用 EKS 叢集上的命名空間註冊 Amazon EMR，執行下列命令來建立虛擬叢集。使用您為虛擬叢集提供的名稱取代 *virtual_cluster_name*。將 *eks_cluster_name* 取代為 EKS 叢集的名稱。將 *namespace_name* 取代為您要註冊 Amazon EMR 的命名空間。

```
aws emr-containers create-virtual-cluster \
--name virtual_cluster_name \
--container-provider '{
    "id": "eks_cluster_name",
    "type": "EKS",
    "info": {
```

```
        "eksInfo": {
            "namespace": "namespace_name"
        }
    }
}'
```

或者，可以建立包含虛擬叢集所需參數的 JSON 檔案，如下列範例所示。

```
{
    "name": "virtual_cluster_name",
    "containerProvider": {
        "type": "EKS",
        "id": "eks_cluster_name",
        "info": {
            "eksInfo": {
                "namespace": "namespace_name"
            }
        }
    }
}
```

然後使用 JSON 檔案的路徑來執行下列 `create-virtual-cluster` 命令。

```
aws emr-containers create-virtual-cluster \
--cli-input-json file:///./create-virtual-cluster-request.json
```

> **Note**
>
> 若要驗證虛擬叢集是否成功建立，請檢視虛擬叢集的狀態，方法是執行 `list-virtual-clusters` 命令或前往 Amazon EMR 主控台中的虛擬叢集頁面。

# 列出虛擬叢集

執行以下命令以檢視虛擬叢集狀態。

```
aws emr-containers list-virtual-clusters
```

# 描述虛擬叢集

執行下列命令，以取得有關虛擬叢集的詳細資訊，例如命名空間、狀態和註冊日期。將 *123456* 取代為虛擬叢集 ID。

```
aws emr-containers describe-virtual-cluster --id 123456
```

# 刪除虛擬叢集

執行下列命令以刪除虛擬叢集。將 *123456* 取代為虛擬叢集 ID。

```
aws emr-containers delete-virtual-cluster --id 123456
```

# 虛擬叢集狀態

下表描述四個可能的虛擬叢集狀態。

| State | 描述 |
| --- | --- |
| RUNNING | 虛擬叢集處於 RUNNING 狀態。 |
| TERMINATING | 正在請求終止虛擬叢集。 |
| TERMINATED | 請求的終止已完成。 |
| ARRESTED | 請求的終止失敗，因為許可不足。 |

# Amazon EMR on EKS 的教學課程

本章節描述了當您使用 Amazon EMR on EKS 應用程式時的常見使用案例。每個應用程式都是專門的，可以採取唯一步驟進行設定。這些主題提供使用每個應用程式的指示。

主題


- 搭配使用 Delta Lake 與 Amazon EMR on EKS
- 搭配使用 Apache Iceberg 與 Amazon EMR on EKS
- 使用 PyFlink
- 搭配 Flink 使用 AWS Glue
- 將 Apache Hudi 與 Apache Flink 搭配使用
- 搭配使用 RAPIDS Accelerator for Apache Spark 和 Amazon EMR on EKS
- 針對 Apache Spark on Amazon EMR on EKS 使用 Amazon Redshift 整合
- 使用 Volcano 作為 Amazon EMR on EKS 上 Apache Spark 的自訂排程器
- 使用 YuniKorn 作為 Amazon EMR on EKS 上 Apache Spark 的自訂排程器


## 搭配使用 Delta Lake 與 Amazon EMR on EKS

Delta Lake 是用於建置 Lakehouse 架構的開放原始碼儲存架構。以下說明如何將其設定為使用。

搭配使用 Delta Lake 與 Amazon EMR on EKS 應用程式

1. 當您啟動作業執行以提交應用程式組態中的 Spark 作業時，請包含 Delta Lake JAR 檔案：

```
--job-driver '{"sparkSubmitJobDriver" : {
      "sparkSubmitParameters" : "--jars local:///usr/share/aws/delta/lib/delta-core.jar,local:///usr/share/aws/delta/lib/delta-storage.jar,local:///usr/share/aws/delta/lib/delta-storage-s3-dynamodb.jar"}}'
```

> **Note**
>
> Amazon EMR 7.0.0 版及更新版本使用 Delta Lake 3.0，將其重新命名`delta-core.jar`為 `delta-spark.jar`。如果您使用 Amazon EMR 7.0.0 版或更新版本，請務必使用正確的檔案名稱，例如下列範例：

```
--jars local:///usr/share/aws/delta/lib/delta-spark.jar
```

2. 包含 Delta Lake 其他組態，並使用 AWS Glue Data Catalog 做為中繼存放區。

```
--configuration-overrides '{
        "applicationConfiguration": [
        {
          "classification" : "spark-defaults",
          "properties" : {
            "spark.sql.extensions" : "io.delta.sql.DeltaSparkSessionExtension",

 "spark.sql.catalog.spark_catalog":"org.apache.spark.sql.delta.catalog.DeltaCatalog",
"spark.hadoop.hive.metastore.client.factory.class":"com.amazonaws.glue.catalog.metastore.AW
           }
        }]}'
```

# 搭配使用 Apache Iceberg 與 Amazon EMR on EKS

Iceberg 的執行期 JAR 包含 Spark 執行期支援所需的 Iceberg 類別。下列程序說明如何使用 Iceberg 火花執行時間開始任務執行。

搭配使用 Apache Iceberg 與 Amazon EMR on EKS 應用程式

1. 當您啟動作業執行以提交應用程式組態中的 Spark 作業時，請包含 Iceberg Spark 執行期 JAR 檔案：

```
--job-driver '{"sparkSubmitJobDriver" : {"sparkSubmitParameters" : "--jars
 local:///usr/share/aws/iceberg/lib/iceberg-spark3-runtime.jar"}}'
```

2. 包括 Iceberg 其他組態：

```
--configuration-overrides '{
    "applicationConfiguration": [
    "classification" : "spark-defaults",
    "properties" : {
        "spark.sql.catalog.dev.warehouse" : "s3://amzn-s3-demo-bucket/EXAMPLE-
PREFIX/ ",
        "spark.sql.extensions ":"
 org.apache.iceberg.spark.extensions.IcebergSparkSessionExtensions ",
```

```
        "spark.sql.catalog.dev" : "org.apache.iceberg.spark.SparkCatalog",
        "spark.sql.catalog.dev.catalog-impl" :
 "org.apache.iceberg.aws.glue.GlueCatalog",
        "spark.sql.catalog.dev.io-impl": "org.apache.iceberg.aws.s3.S3FileIO"
        }
    ]
}'
```

若要了解有關 EMR 的 Apache Iceberg 版本的詳細資訊，請參閱 Iceberg 版本歷史記錄。

# 目錄整合的 Spark 工作階段組態

## Iceberg AWS Glue 目錄整合的 Spark 工作階段組態

此範例說明如何將 Iceberg 與 整合 AWS Glue 編目程式：

```
spark-sql \
  --conf spark.sql.catalog.rms = org.apache.iceberg.spark.SparkCatalog \
  --conf spark.sql.catalog.rms.type = glue \
  --conf spark.sql.catalog.rms.glue.id = glue RMS catalog ID \
  --conf spark.sql.catalog.rms.glue.account-id = AWS account ID \

  --conf spark.sql.extensions=
    org.apache.iceberg.spark.extensions.IcebergSparkSessionExtensions
```

下面會顯示範例查詢：

```
SELECT * FROM rms.rmsdb.table1
```

## Iceberg REST Glue AWS 目錄整合的 Spark 工作階段組態

此範例說明如何將 Iceberg REST 與 整合 AWS Glue 編目程式：

```
spark-sql \
  --conf spark.sql.catalog.rms = org.apache.iceberg.spark.SparkCatalog \
  --conf spark.sql.catalog.rms.type = rest \
  --conf spark.sql.catalog.rms.warehouse = glue RMS catalog ID \
  --conf spark.sql.catalog.rms.uri = glue endpoint URI/iceberg \
  --conf spark.sql.catalog.rms.rest.sigv4-enabled = true \
  --conf spark.sql.catalog.rms.rest.signing-name = glue \
```

```
  --conf spark.sql.extensions=
    org.apache.iceberg.spark.extensions.IcebergSparkSessionExtensions
```

下面會顯示範例查詢：

```
SELECT * FROM rms.rmsdb.table1
```

此組態僅適用於 Redshift Managed Storage。不支援 Amazon S3 的 FGAC。

# 使用 PyFlink

Amazon EMR on EKS 6.15.0 版及更新版本支援 PyFlink。如果您已有 PyFlink 指令碼，您可以執行下列其中一項操作：

- 建立包含 PyFlink 指令碼的自訂映像。
- 將指令碼上傳至 Amazon S3 位置

如果您還沒有指令碼，您可以使用下列範例來啟動 PyFlink 任務。此範例會從 S3 擷取指令碼。如果您使用自訂映像搭配映像中已包含的指令碼，則必須將指令碼路徑更新為您存放指令碼的位置。如果指令碼位於 S3 位置，Amazon EMR on EKS 將擷取指令碼並將其放在 Flink 容器中的`/opt/flink/usrlib/`目錄下。

```
apiVersion: flink.apache.org/v1beta1
kind: FlinkDeployment
metadata:
  name: python-example
spec:
  flinkVersion: v1_17
  flinkConfiguration:
    taskmanager.numberOfTaskSlots: "1"
  executionRoleArn: job-execution-role
  emrReleaseLabel: "emr-6.15.0-flink-latest"
  jobManager:
    highAvailabilityEnabled: false
    replicas: 1
    resource:
      memory: "2048m"
      cpu: 1
  taskManager:
    resource:
```

```
      memory: "2048m"
      cpu: 1
  job:
    jarURI: s3://S3 bucket with your script/pyflink-script.py
    entryClass: "org.apache.flink.client.python.PythonDriver"
    args: ["-py", "/opt/flink/usrlib/pyflink-script.py"]
    parallelism: 1
    upgradeMode: stateless
```

# 搭配 Flink 使用 AWS Glue

Amazon EMR on EKS 搭配 Apache Flink 6.15.0 版和更新版本支援使用 AWS Glue Data Catalog 做為串流和批次 SQL 工作流程的中繼資料存放區。

您必須先建立名為 AWS `default`的 Glue 資料庫，做為 Flink SQL Catalog。此 Flink Catalog 會存放中繼資料，例如資料庫、資料表、分割區、檢視、函數，以及存取其他外部系統中資料所需的其他資訊。

```
aws glue create-database \
    --database-input "{\"Name\":\"default\"}"
```

若要啟用 AWS Glue 支援，請使用`FlinkDeployment`規格。此範例規格使用 Python 指令碼來快速發行一些 Flink SQL 陳述式，以與 Glue AWS 目錄互動。

```
apiVersion: flink.apache.org/v1beta1
kind: FlinkDeployment
metadata:
  name: python-example
spec:
  flinkVersion: v1_17
  flinkConfiguration:
    taskmanager.numberOfTaskSlots: "1"
    aws.glue.enabled: "true"
  executionRoleArn: job-execution-role-arn;
  emrReleaseLabel: "emr-6.15.0-flink-latest"
  jobManager:
    highAvailabilityEnabled: false
    replicas: 1
    resource:
      memory: "2048m"
      cpu: 1
```

```
  taskManager:
    resource:
      memory: "2048m"
      cpu: 1
  job:
    jarURI: s3://<S3_bucket_with_your_script/pyflink-glue-script.py
    entryClass: "org.apache.flink.client.python.PythonDriver"
    args: ["-py", "/opt/flink/usrlib/pyflink-glue-script.py"]
    parallelism: 1
    upgradeMode: stateless
```

以下是 PyFlink 指令碼可能看起來的範例。

```
import logging
import sys
from pyflink.datastream import StreamExecutionEnvironment
from pyflink.table import StreamTableEnvironment

def glue_demo():
    env = StreamExecutionEnvironment.get_execution_environment()
    t_env = StreamTableEnvironment.create(stream_execution_environment=env)
    t_env.execute_sql("""
          CREATE CATALOG glue_catalog WITH (
          'type' = 'hive',
          'default-database' = 'default',
          'hive-conf-dir' = '/glue/confs/hive/conf',
          'hadoop-conf-dir' = '/glue/confs/hadoop/conf'
          )
                      """)
    t_env.execute_sql("""
          USE CATALOG glue_catalog;
                      """)
    t_env.execute_sql("""
          DROP DATABASE IF EXISTS eks_flink_db CASCADE;
                      """)
    t_env.execute_sql("""
          CREATE DATABASE IF NOT EXISTS eks_flink_db WITH ('hive.database.location-
uri'= 's3a://S3-bucket-to-store-metadata/flink/flink-glue-for-hive/warehouse/');
                      """)
    t_env.execute_sql("""
          USE eks_flink_db;
                  """)
    t_env.execute_sql("""
```

```
          CREATE TABLE IF NOT EXISTS eksglueorders (
            order_number BIGINT,
            price        DECIMAL(32,2),
            buyer        RO first_name STRING, last_name STRING,
            order_time   TIMESTAMP(3)
          ) WITH (
            'connector' = 'datagen'
          );
                      """)
    t_env.execute_sql("""
          CREATE TABLE IF NOT EXISTS eksdestglueorders (
            order_number BIGINT,
            price        DECIMAL(32,2),
            buyer        ROW first_name STRING, last_name STRING,
            order_time   TIMESTAMP(3)
          ) WITH (
            'connector' = 'filesystem',
            'path' = 's3://S3-bucket-to-store-metadata/flink/flink-glue-for-hive/
warehouse/eksdestglueorders',
            'format' = 'json'
          );
                 """)
    t_env.execute_sql("""
          CREATE TABLE IF NOT EXISTS print_table (
            order_number BIGINT,
            price        DECIMAL(32,2),
            buyer        ROW first_name STRING, last_name STRING,
            order_time   TIMESTAMP(3)
          ) WITH (
            'connector' = 'print'
          );
                """)
    t_env.execute_sql("""
          EXECUTE STATEMENT SET
          BEGIN
          INSERT INTO eksdestglueorders SELECT * FROM  eksglueorders LIMIT 10;
          INSERT INTO print_table SELECT * FROM eksdestglueorders;
          END;
            """)


if __name__ == '__main__':
    logging.basicConfig(stream=sys.stdout, level=logging.INFO, format="%(message)s")
```

```
    glue_demo()
```

# 將 Apache Hudi 與 Apache Flink 搭配使用

Apache Hudi 是一種開放原始碼資料管理架構，具有插入、更新、 upsert 和刪除等記錄層級操作，可用於簡化資料管理和資料管道開發。與 Amazon S3 中的高效資料管理結合時，Hudi 可讓您即時擷取和更新資料。Hudi 會維護您在資料集上執行之所有操作的中繼資料，因此所有動作都保持原子和一致性。

Apache Hudi 可在 Amazon EMR on EKS 上使用 Apache Flink 搭配 Amazon EMR 7.2.0 版和更新版本。請參閱下列步驟，了解如何開始使用並提交 Apache Hudi 任務。

## 提交 Apache Hudi 任務

請參閱下列步驟，了解如何提交 Apache Hudi 任務。

1. 建立名為 的 AWS Glue 資料庫`default`。

```
aws glue create-database --database-input "{\"Name\":\"default\"}"
```

2. 遵循 Flink Kubernetes Operator SQL 範例來建置 `flink-sql-runner.jar` 檔案。
3. 建立 Hudi SQL 指令碼，如下所示。

```
CREATE CATALOG hudi_glue_catalog WITH (
'type' = 'hudi',
'mode' = 'hms',
'table.external' = 'true',
'default-database' = 'default',
'hive.conf.dir' = '/glue/confs/hive/conf/',
'catalog.path' = 's3://<hudi-example-bucket>/FLINK_HUDI/warehouse/'
);

USE CATALOG hudi_glue_catalog;
CREATE DATABASE IF NOT EXISTS hudi_db;
use hudi_db;

CREATE TABLE IF NOT EXISTS hudi-flink-example-table(
    uuid VARCHAR(20),
    name VARCHAR(10),
    age INT,
    ts TIMESTAMP(3),
```

```
    `partition` VARCHAR(20)
)
PARTITIONED BY (`partition`)
WITH (
  'connector' = 'hudi',
  'path' = 's3://<hudi-example-bucket>/hudi-flink-example-table',
  'hive_sync.enable' = 'true',
  'hive_sync.mode' = 'glue',
  'hive_sync.table' = 'hudi-flink-example-table',
  'hive_sync.db' = 'hudi_db',
  'compaction.delta_commits' = '1',
  'hive_sync.partition_fields' = 'partition',
  'hive_sync.partition_extractor_class' =
 'org.apache.hudi.hive.MultiPartKeysValueExtractor',
  'table.type' = 'COPY_ON_WRITE'
);

EXECUTE STATEMENT SET
BEGIN

INSERT INTO hudi-flink-example-table VALUES
    ('id1','Alex',23,TIMESTAMP '1970-01-01 00:00:01','par1'),
    ('id2','Stephen',33,TIMESTAMP '1970-01-01 00:00:02','par1'),
    ('id3','Julian',53,TIMESTAMP '1970-01-01 00:00:03','par2'),
    ('id4','Fabian',31,TIMESTAMP '1970-01-01 00:00:04','par2'),
    ('id5','Sophia',18,TIMESTAMP '1970-01-01 00:00:05','par3'),
    ('id6','Emma',20,TIMESTAMP '1970-01-01 00:00:06','par3'),
    ('id7','Bob',44,TIMESTAMP '1970-01-01 00:00:07','par4'),
    ('id8','Han',56,TIMESTAMP '1970-01-01 00:00:08','par4');

END;
```

4. 將 Hudi SQL 指令碼和 `flink-sql-runner.jar` 檔案上傳至 S3 位置。
5. 在您的 `FlinkDeployments` YAML 檔案中，將 `hudi.enabled` 設定為 `true`。

```
spec:
  flinkConfiguration:
    hudi.enabled: "true"
```

6. 建立 YAML 檔案以執行您的組態。此範例檔案名為 `hudi-write.yaml`。

```
apiVersion: flink.apache.org/v1beta1
kind: FlinkDeployment
```

```
metadata:
  name: hudi-write-example
spec:
  flinkVersion: v1_18
  flinkConfiguration:
    taskmanager.numberOfTaskSlots: "2"
    hudi.enabled: "true"
  executionRoleArn: "<JobExecutionRole>"
  emrReleaseLabel: "emr-7.9.0-flink-latest"
  jobManager:
    highAvailabilityEnabled: false
    replicas: 1
    resource:
      memory: "2048m"
      cpu: 1
  taskManager:
    resource:
      memory: "2048m"
      cpu: 1
  job:
    jarURI: local:///opt/flink/usrlib/flink-sql-runner.jar
    args: ["/opt/flink/scripts/hudi-write.sql"]
    parallelism: 1
    upgradeMode: stateless
  podTemplate:
    spec:
      initContainers:
        - name: flink-sql-script-download
          args:
            - s3
            - cp
            - s3://<s3_location>/hudi-write.sql
            - /flink-scripts
          image: amazon/aws-cli:latest
          imagePullPolicy: Always
          resources: {}
          terminationMessagePath: /dev/termination-log
          terminationMessagePolicy: File
          volumeMounts:
            - mountPath: /flink-scripts
              name: flink-scripts
        - name: flink-sql-runner-download
          args:
            - s3
```

```
            - cp
            - s3://<s3_location>/flink-sql-runner.jar
            - /flink-artifacts
          image: amazon/aws-cli:latest
          imagePullPolicy: Always
          resources: {}
          terminationMessagePath: /dev/termination-log
          terminationMessagePolicy: File
          volumeMounts:
            - mountPath: /flink-artifacts
              name: flink-artifact
      containers:
        - name: flink-main-container
          volumeMounts:
            - mountPath: /opt/flink/scripts
              name: flink-scripts
            - mountPath: /opt/flink/usrlib
              name: flink-artifact
      volumes:
        - emptyDir: {}
          name: flink-scripts
        - emptyDir: {}
          name: flink-artifact
```

7. 將 Flink Hudi 任務提交至 Flink Kubernetes Operator。

```
kubectl apply -f hudi-write.yaml
```

# 搭配使用 RAPIDS Accelerator for Apache Spark 和 Amazon EMR on EKS

使用 Amazon EMR on EKS，可執行 Nvidia RAPIDS Accelerator for Apache Spark 的作業。本教學課程說明如何使用 RAPIDS on EC2 圖形處理單元 (GPU) 執行個體類型來執行 Spark 作業。此教學課程使用下列版本：

- Amazon EMR on EKS 發行版本 6.9.0 及更高版本
- Apache Spark 3.x

透過使用 Nvidia RAPIDS Accelerator for Apache Spark 外掛程式，可藉助 Amazon EC2 GPU 執行個體類型來加速 Spark。當您一起使用這些技術時，可以加速資料科學管道，而無需進行任何程式碼變更。這樣可減少資料處理和模型訓練所需的執行時間。在更短的時間內完成更多工作，您可以減少在基礎設施成本上的支出。

開始之前，請確保擁有下列資源。

- Amazon EMR on EKS 虛擬叢集
- Amazon EKS 叢集，具有已啟用 GPU 的節點群組

Amazon EKS 虛擬叢集是 Amazon EKS 叢集上 Kubernetes 命名空間的已註冊控制碼，由 Amazon EMR on EKS 管理。此控點可讓 Amazon EMR 使用 Kubernetes 命名空間作為執行中作業的目的地。如需有關如何設定虛擬叢集的詳細資訊，請參閱本指南中的 設定 Amazon EMR on EKS。

必須使用具有 GPU 執行個體的節點群組來設定 Amazon EKS 虛擬叢集。必須使用 Nvidia 裝置外掛程式來設定節點。如需進一步了解，請參閱受管節點群組。

若要設定 Amazon EKS 叢集以新增啟用 GPU 的節點群組，請執行下列程序：

新增已啟用 GPU 的節點群組

1. 使用下列 create-nodegroup 命令建立啟用 GPU 的節點群組。請務必使用正確的參數來取代 Amazon EKS 叢集。使用支援 Spark RAPIDS 的執行個體類型，例如 P4、P3、G5 或 G4dn。

```
aws eks create-nodegroup \
 --cluster-name EKS_CLUSTER_NAME \
 --nodegroup-name NODEGROUP_NAME \
 --scaling-config minSize=0,maxSize=5,desiredSize=2 CHOOSE_APPROPRIATELY \
 --ami-type AL2_x86_64_GPU \
 --node-role NODE_ROLE \
 --subnets SUBNETS_SPACE_DELIMITED  \
 --remote-access ec2SshKey= SSH_KEY \
 --instance-types GPU_INSTANCE_TYPE \
 --disk-size DISK_SIZE \
 --region AWS_REGION
```

2. 在叢集中安裝 Nvidia 裝置外掛程式，以便在叢集的每個節點上發出 GPU 數量，並在叢集中執行已啟用 GPU 的容器。執行下列程式碼安裝外掛程式。

```
kubectl apply -f https://raw.githubusercontent.com/NVIDIA/k8s-device-plugin/v0.9.0/
nvidia-device-plugin.yml
```

3. 若要驗證叢集的每個節點上有多少 GPU 可用，請執行下列命令：

```
kubectl get nodes  "-o=custom-
columns=NAME:.metadata.name,GPU:.status.allocatable.nvidia\.com/gpu"
```

## 執行 Spark RAPIDS 作業

1. 將 Spark RAPIDS 作業提交至 Amazon EMR on EKS 叢集。下列程式碼顯示啟動作業的命令範例。第一次執行作業時，可能需要幾分鐘的時間下載映像並將其快取到節點上。

```
aws emr-containers start-job-run \
--virtual-cluster-id VIRTUAL_CLUSTER_ID \
--execution-role-arn JOB_EXECUTION_ROLE \
--release-label emr-6.9.0-spark-rapids-latest \
--job-driver '{"sparkSubmitJobDriver": {"entryPoint": "local:///usr/lib/
spark/examples/jars/spark-examples.jar","entryPointArguments":  ["10000"],
 "sparkSubmitParameters":"--class org.apache.spark.examples.SparkPi "}}' \
---configuration-overrides '{"applicationConfiguration": [{"classification":
 "spark-defaults","properties": {"spark.executor.instances":
 "2","spark.executor.memory": "2G"}}],"monitoringConfiguration":
 {"cloudWatchMonitoringConfiguration": {"logGroupName": "LOG_GROUP
 _NAME"},"s3MonitoringConfiguration": {"logUri": "LOG_GROUP_STREAM"}}}'
```

2. 若要驗證 Spark RAPIDS Accelerator 已啟用，請檢查 Spark 驅動程式日誌。這些日誌會儲存在 CloudWatch 或您執行 `start-job-run` 命令時指定的 S3 位置。下列範例通常會顯示日誌行的樣式：

```
22/11/15 00:12:44 INFO RapidsPluginUtils: RAPIDS Accelerator build:
 {version=22.08.0-amzn-0, user=release, url=, date=2022-11-03T03:32:45Z, revision=,
 cudf_version=22.08.0, branch=}
22/11/15 00:12:44 INFO RapidsPluginUtils: RAPIDS Accelerator JNI build:
 {version=22.08.0, user=, url=https://github.com/NVIDIA/spark-rapids-jni.git,
 date=2022-08-18T04:14:34Z, revision=a1b23cd_sample, branch=HEAD}
22/11/15 00:12:44 INFO RapidsPluginUtils: cudf build: {version=22.08.0,
 user=, url=https://github.com/rapidsai/cudf.git, date=2022-08-18T04:14:34Z,
 revision=a1b23ce_sample, branch=HEAD}
```

```
22/11/15 00:12:44 WARN RapidsPluginUtils: RAPIDS Accelerator 22.08.0-amzn-0 using
 cudf 22.08.0.
22/11/15 00:12:44 WARN RapidsPluginUtils:
 spark.rapids.sql.multiThreadedRead.numThreads is set to 20.
22/11/15 00:12:44 WARN RapidsPluginUtils: RAPIDS Accelerator is enabled, to disable
 GPU support set `spark.rapids.sql.enabled` to false.
22/11/15 00:12:44 WARN RapidsPluginUtils: spark.rapids.sql.explain is set to
 `NOT_ON_GPU`. Set it to 'NONE' to suppress the diagnostics logging about the query
 placement on the GPU.
```

3. 若要查看將在 GPU 上執行的操作，請執行下列步驟以啟用額外的日誌。請注意 "spark.rapids.sql.explain : ALL" 組態。

```
aws emr-containers start-job-run \
--virtual-cluster-id VIRTUAL_CLUSTER_ID \
--execution-role-arn JOB_EXECUTION_ROLE \
--release-label emr-6.9.0-spark-rapids-latest \
--job-driver '{"sparkSubmitJobDriver": {"entryPoint": "local:///usr/lib/
spark/examples/jars/spark-examples.jar","entryPointArguments":  ["10000"],
 "sparkSubmitParameters":"--class org.apache.spark.examples.SparkPi "}}' \
---configuration-overrides '{"applicationConfiguration":
 [{"classification": "spark-defaults","properties":
 {"spark.rapids.sql.explain":"ALL","spark.executor.instances":
 "2","spark.executor.memory": "2G"}}],"monitoringConfiguration":
 {"cloudWatchMonitoringConfiguration": {"logGroupName":
 "LOG_GROUP_NAME"},"s3MonitoringConfiguration": {"logUri": "LOG_GROUP_STREAM"}}}'
```

上一個命令是使用 GPU 的作業範例。其輸出如以下範例所示。如需了解輸出，請參閱此說明圖例：

- * - 表示在 GPU 上運作的操作
- ! - 表示無法在 GPU 上執行的操作
- @ - 表示在 GPU 上運作的操作，但無法執行，因為它位於無法在 GPU 上執行的計畫內

```
 22/11/15 01:22:58 INFO GpuOverrides: Plan conversion to the GPU took 118.64 ms
 22/11/15 01:22:58 INFO GpuOverrides: Plan conversion to the GPU took 4.20 ms
 22/11/15 01:22:58 INFO GpuOverrides: GPU plan transition optimization took 8.37 ms
 22/11/15 01:22:59 WARN GpuOverrides:
    *Exec <ProjectExec> will run on GPU
```

```
      *Expression <Alias> substring(cast(date#149 as string), 0, 7) AS month#310
 will run on GPU
        *Expression <Substring> substring(cast(date#149 as string), 0, 7) will run
 on GPU
          *Expression <Cast> cast(date#149 as string) will run on GPU
      *Exec <SortExec> will run on GPU
        *Expression <SortOrder> date#149 ASC NULLS FIRST will run on GPU
        *Exec <ShuffleExchangeExec> will run on GPU
          *Partitioning <RangePartitioning> will run on GPU
            *Expression <SortOrder> date#149 ASC NULLS FIRST will run on GPU
          *Exec <UnionExec> will run on GPU
            !Exec <ProjectExec> cannot run on GPU because not all expressions can
 be replaced
              @Expression <AttributeReference> customerID#0 could run on GPU
              @Expression <Alias> Charge AS kind#126 could run on GPU
                @Expression <Literal> Charge could run on GPU
              @Expression <AttributeReference> value#129 could run on GPU
              @Expression <Alias> add_months(2022-11-15, cast(-(cast(_we0#142 as
 bigint) + last_month#128L) as int)) AS date#149 could run on GPU
                ! <AddMonths> add_months(2022-11-15, cast(-
(cast(_we0#142 as bigint) + last_month#128L) as int)) cannot run
 on GPU because GPU does not currently support the operator class
 org.apache.spark.sql.catalyst.expressions.AddMonths
                  @Expression <Literal> 2022-11-15 could run on GPU
                  @Expression <Cast> cast(-(cast(_we0#142 as bigint) +
 last_month#128L) as int) could run on GPU
                    @Expression <UnaryMinus> -(cast(_we0#142 as bigint) +
 last_month#128L) could run on GPU
                      @Expression <Add> (cast(_we0#142 as bigint) +
 last_month#128L) could run on GPU
                        @Expression <Cast> cast(_we0#142 as bigint) could run on
 GPU
                          @Expression <AttributeReference> _we0#142 could run on
 GPU
                        @Expression <AttributeReference> last_month#128L could run
 on GPU
```

# 針對 Apache Spark on Amazon EMR on EKS 使用 Amazon Redshift 整合

使用 Amazon EMR 版本 6.9.0 及更高版本，每個版本映像都包括 Apache Spark 和 Amazon Redshift 之間的連接器。這樣，就可以在 Amazon EMR on EKS 上使用 Spark 來處理儲存在 Amazon Redshift 中的資料。整合是以 `spark-redshift` 開放原始碼連接器為基礎。對於 Amazon EMR on EKS，會包含 Apache Spark 的 Amazon Redshift 整合作為原生整合。

主題


- 使用 Apache Spark 的 Amazon Redshift 整合，啟動 Spark 應用程式
- 使用 Apache Spark 的 Amazon Redshift 整合進行身分驗證
- 在 Amazon Redshift 中讀取和寫入
- 使用 Spark 連接器時的考量和限制


## 使用 Apache Spark 的 Amazon Redshift 整合，啟動 Spark 應用程式

若要使用整合，必須在 Spark 作業中傳遞必要的 Spark Redshift 相依性。您必須使用 `--jars` 來包含與 Redshift 連接器相關的程式庫。若要查看 `--jars` 選項支援的其他檔案位置，請參閱 Apache Spark 說明文件的進階相依性管理一節。

- `spark-redshift.jar`
- `spark-avro.jar`
- `RedshiftJDBC.jar`
- `minimal-json.jar`

若要使用 Apache Spark on Amazon EMR on EKS 版本 6.9.0 或更高版本的 Amazon Redshift 整合來啟動 Spark 應用程式，請使用以下範例命令。請注意，與 `--conf spark.jars` 選項一起列出的路徑是 JAR 檔案的預設路徑。

```
aws emr-containers start-job-run \

--virtual-cluster-id cluster_id \
--execution-role-arn arn \
--release-label emr-6.9.0-latest\
--job-driver '{
```

```
    "sparkSubmitJobDriver": {
        "entryPoint": "s3://script_path",
            "sparkSubmitParameters":
            "--conf spark.kubernetes.file.upload.path=s3://upload_path
             --conf spark.jars=
                /usr/share/aws/redshift/jdbc/RedshiftJDBC.jar,
                /usr/share/aws/redshift/spark-redshift/lib/spark-redshift.jar,
                /usr/share/aws/redshift/spark-redshift/lib/spark-avro.jar,
                /usr/share/aws/redshift/spark-redshift/lib/minimal-json.jar"
                            }
            }'
```

## 使用 Apache Spark 的 Amazon Redshift 整合進行身分驗證

下列各節顯示當您與 Apache Spark 整合時，Amazon Redshift 的身分驗證選項。各節說明如何擷取登入憑證，以及使用 JDBC 驅動程式搭配 IAM 身分驗證的詳細資訊。

### 使用 AWS Secrets Manager 擷取登入資料並連線至 Amazon Redshift

可以將憑證儲存在 Secrets Manager 中，以便安全地向 Amazon Redshift 進行身分驗證。可以讓您的 Spark 作業呼叫 `GetSecretValue` API 以獲取憑證：

```
from pyspark.sql import SQLContextimport boto3

sc = # existing SparkContext
sql_context = SQLContext(sc)

secretsmanager_client = boto3.client('secretsmanager',
 region_name=os.getenv('AWS_REGION'))
secret_manager_response = secretsmanager_client.get_secret_value(
    SecretId='string',
    VersionId='string',
    VersionStage='string'
)
username = # get username from secret_manager_response
password = # get password from secret_manager_response
url = "jdbc:redshift://redshifthost:5439/database?user=" + username + "&password="
 + password

# Access to Redshift cluster using Spark
```

## 搭配使用基於 IAM 的身分驗證與 Amazon EMR on EKS 作業執行角色

從 Amazon EMR on EKS 6.9.0 版開始，Amazon Redshift JDBC 驅動器 2.1 版或更高版本已封裝到環境中。使用 JDBC 驅動器 2.1 及更高版本，可以指定 JDBC URL，而不包含原始使用者名稱和密碼。相反地，可以指定 `jdbc:redshift:iam:// 配置`。這會命令 JDBC 驅動器使用 Amazon EMR on EKS 作業執行角色來自動擷取憑證。

如需詳細資訊，請參閱《Amazon Redshift 管理指南》中的設定 JDBC 或 ODBC 連線以使用 IAM 憑證。

下列範例 URL 使用 `jdbc:redshift:iam:// 配置`。

```
jdbc:redshift:iam://examplecluster.abc123xyz789.us-west-2.redshift.amazonaws.com:5439/
dev
```

當作業執行角色符合提供的條件時，需要下列許可。

| 權限 | 作業執行角色所需的條件 |
| --- | --- |
| `redshift:GetClusterCredentials` | JDBC 驅動器從 Amazon Redshift 獲取憑證時所需的條件 |
| `redshift:DescribeCluster` | 如果在 JDBC URL 中而非端點中指定 Amazon Redshift 叢集和 AWS 區域 時所需的條件 |
| `redshift-serverless:GetCredentials` | JDBC 驅動器從 Amazon Redshift Serverless 獲取憑證時所需的條件 |
| `redshift-serverless:GetWorkgroup` | 如果您使用的是 Amazon Redshift Serverless 並且根據工作群組名稱和區域指定 URL 時所需的條件 |

您的作業執行角色政策應具有下列許可。

```
{
            "Effect": "Allow",
            "Action": [
                "redshift:GetClusterCredentials",
                "redshift:DescribeCluster",
```

```
                "redshift-serverless:GetCredentials",
                "redshift-serverless:GetWorkgroup"
            ],
            "Resource": [

 "arn:aws:redshift:AWS_REGION:ACCOUNT_ID:dbname:CLUSTER_NAME/DATABASE_NAME",
                "arn:aws:redshift:AWS_REGION:ACCOUNT_ID:dbuser:DATABASE_NAME/USER_NAME"
            ]
        }
```

## 使用 JDBC 驅動器對 Amazon Redshift 進行身分驗證

### 在 JDBC URL 中設定使用者名稱和密碼

若要對 Amazon Redshift 叢集驗證 Spark 作業，可以在 JDBC URL 中指定 Amazon Redshift 資料庫名稱和密碼。

> **Note**
>
> 如果在 URL 中傳遞資料庫憑證，則擁有 URL 存取權的任何人也可以存取憑證。通常不建議使用此方法，因為它不安全。

如果您的應用程式不考慮安全性，可以使用下列格式在 JDBC URL 中設定使用者名稱和密碼：

```
jdbc:redshift://redshifthost:5439/database?user=username&password=password
```

# 在 Amazon Redshift 中讀取和寫入

下列程式碼範例使用 PySpark，透過資料來源 API 和 SparkSQL，在 Amazon Redshift 資料庫中讀取和寫入範例資料。

### Data source API

使用 PySpark，透過資料來源 API，在 Amazon Redshift 資料庫中讀取和寫入範例資料。

```
import boto3
from pyspark.sql import SQLContext

sc = # existing SparkContext
```

```
sql_context = SQLContext(sc)

url = "jdbc:redshift:iam://redshifthost:5439/database"
aws_iam_role_arn = "arn:aws:iam::accountID:role/roleName"

df = sql_context.read \
    .format("io.github.spark_redshift_community.spark.redshift") \
    .option("url", url) \
    .option("dbtable", "tableName") \
    .option("tempdir", "s3://path/for/temp/data") \
    .option("aws_iam_role", "aws_iam_role_arn") \
    .load()

df.write \
    .format("io.github.spark_redshift_community.spark.redshift") \
    .option("url", url) \
    .option("dbtable", "tableName_copy") \
    .option("tempdir", "s3://path/for/temp/data") \
    .option("aws_iam_role", "aws_iam_role_arn") \
    .mode("error") \
    .save()
```

### SparkSQL

使用 PySpark，透過 SparkSQL，在 Amazon Redshift 資料庫中讀取和寫入範例資料。

```
import boto3
import json
import sys
import os
from pyspark.sql import SparkSession

spark = SparkSession \
    .builder \
    .enableHiveSupport() \
    .getOrCreate()

url = "jdbc:redshift:iam://redshifthost:5439/database"
aws_iam_role_arn = "arn:aws:iam::accountID:role/roleName"

bucket = "s3://path/for/temp/data"
tableName = "tableName" # Redshift table name

s = f"""CREATE TABLE IF NOT EXISTS {tableName} (country string, data string)
```

```
    USING io.github.spark_redshift_community.spark.redshift
    OPTIONS (dbtable '{tableName}', tempdir '{bucket}', url '{url}', aws_iam_role
 '{aws_iam_role_arn}' ); """

spark.sql(s)

columns = ["country" ,"data"]
data = [("test-country","test-data")]
df = spark.sparkContext.parallelize(data).toDF(columns)

# Insert data into table
df.write.insertInto(tableName, overwrite=False)
df = spark.sql(f"SELECT * FROM {tableName}")
df.show()
```

## 使用 Spark 連接器時的考量和限制

Spark 連接器支援各種方法來管理登入資料、設定安全性，以及與其他 AWS 服務連線。熟悉此清單中的建議，以設定功能和彈性連線。

- 建議您激活 SSL，進行從 Spark on Amazon EMR 到 Amazon Redshift 的 JDBC 連接。
- 作為最佳實務，建議您在 AWS Secrets Manager 中管理 Amazon Redshift 叢集的憑證。如需範例，請參閱使用 AWS Secrets Manager 擷取憑證以連線至 Amazon Redshift。
- 建議使用 Amazon Redshift 身分驗證參數的 `aws_iam_role` 參數傳遞 IAM 角色。
- 參數 `tempformat` 目前不支援 Parquet 格式。
- `tempdir` URI 指向 Amazon S3 位置。此暫時目錄不會自動清理，因此可能會增加額外的費用。
- 請考慮下列針對 Amazon Redshift 的建議：
  - 建議您封鎖對 Amazon Redshift 叢集的公開存取。
  - 建議開啟 Amazon Redshift 稽核日誌。
  - 建議開啟 Amazon Redshift 靜態加密。
- 請考慮下列針對 Amazon S3 的建議：
  - 建議阻止 Amazon S3 儲存貯體的公有存取。
  - 建議使用 Amazon S3 伺服器端加密來加密您使用的 S3 儲存貯體。
  - 建議使用 Amazon S3 生命週期政策來定義 S3 儲存貯體的保留規則。
  - Amazon EMR 一律會驗證從開放原始碼匯入到映像的程式碼。為了安全起見，我們不支援將 `tempdir` URI 中的 AWS 存取金鑰編碼為從 Spark 到 Amazon S3 的身分驗證方法。

如需有關使用連接器及其支援參數的詳細資訊，請參閱下列資源：

- 《Amazon Redshift 管理指南》中的 Apache Spark 的 Amazon Redshift 整合
- Github 上的 `spark-redshift` 社群儲存庫

# 使用 Volcano 作為 Amazon EMR on EKS 上 Apache Spark 的自訂排程器

透過 Amazon EMR on EKS，可以將 Spark 運算子或 spark-submit 與 Kubernetes 自訂排程器搭配使用，以執行 Spark 作業。本教學課程介紹了如何在自訂佇列上使用 Volcano 排程器來執行 Spark 作業。

## 概觀

Volcano 可以透過諸如佇列排程、公平共用排程以及資源保留等進階功能來幫助管理 Spark 排程。如需有關 Volcano 優勢的詳細資訊，請參閱 The Linux Foundation 的 CNCF 部落格上的為何 Spark 選擇 Volcano 作為 Kubernetes 上的內建批次排程器。

## 安裝並設定 Volcano

1. 根據您的架構需求，選擇下列其中一個 kubectl 命令來安裝 Volcano：

```
# x86_64
kubectl apply -f https://raw.githubusercontent.com/volcano-sh/volcano/v1.5.1/
installer/volcano-development.yaml
# arm64:
kubectl apply -f https://raw.githubusercontent.com/volcano-sh/volcano/v1.5.1/
installer/volcano-development-arm64.yaml
```

2. 準備 Volcano 佇列範例。佇列是 PodGroups 的集合。佇列採用 FIFO，是資源劃分的基礎。

```
cat << EOF > volcanoQ.yaml
apiVersion: scheduling.volcano.sh/v1beta1
kind: Queue
metadata:
  name: sparkqueue
spec:
  weight: 4
  reclaimable: false
  capability:
```

```
    cpu: 10
    memory: 20Gi
EOF

kubectl apply -f volcanoQ.yaml
```

3. 將 PodGroup 清單檔案範例上傳到 Amazon S3。PodGroup 是一組具有強大關聯性的 Pod。您通常會使用 PodGroup 進行批次排程。將下列 PodGroup 範例提交到您在先前步驟中定義的佇列。

```
cat << EOF > podGroup.yaml
apiVersion: scheduling.volcano.sh/v1beta1
kind: PodGroup
spec:
  # Set minMember to 1 to make a driver pod
  minMember: 1
  # Specify minResources to support resource reservation.
  # Consider the driver pod resource and executors pod resource.
  # The available resources should meet the minimum requirements of the Spark job
  # to avoid a situation where drivers are scheduled, but they can't schedule
  # sufficient executors to progress.
  minResources:
    cpu: "1"
    memory: "1Gi"
  # Specify the queue. This defines the resource queue that the job should be
 submitted to.
  queue: sparkqueue
EOF

aws s3 mv podGroup.yaml s3://bucket-name
```

## 使用 Volcano 排程器和 Spark Operator 來執行 Spark 應用程式

1. 如果您尚未完成，請先完成下節中的步驟進行設定：

   a. 安裝並設定 Volcano

   b. 為 Amazon EMR on EKS 設定 Spark Operator

   c. 安裝 Spark Operator

   當您執行 `helm install spark-operator-demo` 命令時，請包含下列引數：

   ```
   --set batchScheduler.enable=true
   ```

```
--set webhook.enable=true
```

2. 建立已設定 batchScheduler 的 SparkApplication 定義檔案 spark-pi.yaml。

```
apiVersion: "sparkoperator.k8s.io/v1beta2"
kind: SparkApplication
metadata:
  name: spark-pi
  namespace: spark-operator
spec:
  type: Scala
  mode: cluster
  image: "895885662937.dkr.ecr.us-west-2.amazonaws.com/spark/emr-6.10.0:latest"
  imagePullPolicy: Always
  mainClass: org.apache.spark.examples.SparkPi
  mainApplicationFile: "local:///usr/lib/spark/examples/jars/spark-examples.jar"
  sparkVersion: "3.3.1"
  batchScheduler: "volcano"   #Note: You must specify the batch scheduler name as
 'volcano'
  restartPolicy:
    type: Never
  volumes:
    - name: "test-volume"
      hostPath:
        path: "/tmp"
        type: Directory
  driver:
    cores: 1
    coreLimit: "1200m"
    memory: "512m"
    labels:
      version: 3.3.1
    serviceAccount: emr-containers-sa-spark
    volumeMounts:
      - name: "test-volume"
        mountPath: "/tmp"
  executor:
    cores: 1
    instances: 1
    memory: "512m"
    labels:
      version: 3.3.1
    volumeMounts:
      - name: "test-volume"
```

```
        mountPath: "/tmp"
```

3. 使用下列命令提交 Spark 應用程式。這也會建立 `SparkApplication` 物件，名為 `spark-pi`：

```
kubectl apply -f spark-pi.yaml
```

4. 使用下列命令檢查 `SparkApplication` 物件的事件：

```
kubectl describe pods spark-pi-driver --namespace spark-operator
```

第一個 Pod 事件會顯示 Volcano 已排程 Pod：

```
Type    Reason     Age   From                Message
----    ------     ----  ----                -------
Normal  Scheduled  23s   volcano             Successfully assigned default/spark-
pi-driver to integration-worker2
```

## 使用 Volcano 排程器和 **`spark-submit`** 來執行 Spark 應用程式

1. 首先，請先完成 為 Amazon EMR on EKS 設定 spark-submit 一節中的步驟。必須在 Volcano 支援下建立自己的 `spark-submit` 分發。如需詳細資訊，請參閱 Apache Spark 文件中的使用 Volcano 作為 Spark on Kubernetes 的自訂排程器的建置一節。
2. 設定以下環境變數的值：

```
export SPARK_HOME=spark-home
export MASTER_URL=k8s://Amazon-EKS-cluster-endpoint
```

3. 使用下列命令提交 Spark 應用程式：

```
$SPARK_HOME/bin/spark-submit \
 --class org.apache.spark.examples.SparkPi \
 --master $MASTER_URL \
 --conf spark.kubernetes.container.image=895885662937.dkr.ecr.us-
west-2.amazonaws.com/spark/emr-6.10.0:latest \
 --conf spark.kubernetes.authenticate.driver.serviceAccountName=spark \
 --deploy-mode cluster \
 --conf spark.kubernetes.namespace=spark-operator \
 --conf spark.kubernetes.scheduler.name=volcano \
 --conf spark.kubernetes.scheduler.volcano.podGroupTemplateFile=/path/to/podgroup-
template.yaml \
```

```
 --conf
 spark.kubernetes.driver.pod.featureSteps=org.apache.spark.deploy.k8s.features.VolcanoFeatu
 \
 --conf
 spark.kubernetes.executor.pod.featureSteps=org.apache.spark.deploy.k8s.features.VolcanoFea
 \
 local:///usr/lib/spark/examples/jars/spark-examples.jar 20
```

4. 使用下列命令檢查 `SparkApplication` 物件的事件：

```
kubectl describe pod spark-pi --namespace spark-operator
```

第一個 Pod 事件會顯示 Volcano 已排程 Pod：

```
Type    Reason     Age   From               Message
----    ------     ----  ----               -------
Normal  Scheduled  23s   volcano            Successfully assigned default/spark-
pi-driver to integration-worker2
```

# 使用 YuniKorn 作為 Amazon EMR on EKS 上 Apache Spark 的自訂排程器

透過 Amazon EMR on EKS，可以將 Spark 運算子或 spark-submit 與 Kubernetes 自訂排程器搭配使用，以執行 Spark 作業。本教學課程介紹了如何在自訂佇列上使用 YuniKorn 排程器和群排程來執行 Spark 作業。

## 概觀

Apache YuniKorn 可以透過應用程式感知排程來協助管理 Spark 排程，讓您可以對資源配額和優先順序進行精細控制。透過群排程，YuniKorn 僅在滿足應用程式的最小資源請求時才會對應用程式進行排程。如需詳細資訊，請參閱 Apache YuniKorn 文件網站中的什麼是在群排程。

## 建立您的叢集並設定 YuniKorn

使用下列步驟來部署 Amazon EKS 叢集。可以變更 AWS 區域 (`region`) 和可用區域 (`availabilityZones`)。

1. 定義 Amazon EKS 叢集：

```
cat <<EOF >eks-cluster.yaml
---
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: emr-eks-cluster
  region: eu-west-1

vpc:
  clusterEndpoints:
    publicAccess: true
    privateAccess: true

iam:
  withOIDC: true

nodeGroups:
  - name: spark-jobs
    labels: { app: spark }
    instanceType: m5.xlarge
    desiredCapacity: 2
    minSize: 2
    maxSize: 3
    availabilityZones: ["eu-west-1a"]
EOF
```

2. 建立叢集：

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

3. 建立您將在其中執行 Spark 作業的命名空間 `spark-job`：

```
kubectl create namespace spark-job
```

4. 接下來，建立 Kubernetes 角色和角色連結。這是 Spark 作業執行使用的服務帳戶所必需的。

   a. 定義 Spark 作業的服務帳戶、角色和角色連結。

```
cat <<EOF >emr-job-execution-rbac.yaml
---
apiVersion: v1
```

```
kind: ServiceAccount
metadata:
  name: spark-sa
  namespace: spark-job
automountServiceAccountToken: false
---
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: spark-role
  namespace: spark-job
rules:
  - apiGroups: ["", "batch","extensions"]
    resources: ["configmaps","serviceaccounts","events","pods","pods/
exec","pods/log","pods/
portforward","secrets","services","persistentvolumeclaims"]
    verbs: ["create","delete","get","list","patch","update","watch"]
---
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: spark-sa-rb
  namespace: spark-job
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: Role
  name: spark-role
subjects:
  - kind: ServiceAccount
    name: spark-sa
    namespace: spark-job
EOF
```

b. 使用下列命令套用 Kubernetes 角色和角色連結定義：

```
kubectl apply -f emr-job-execution-rbac.yaml
```

## 安裝與設定 YuniKorn

1. 使用下列 kubectl 命令建立命名空間 `yunikorn`，以部署 Yunikorn 排程器：

```
kubectl create namespace yunikorn
```

2. 若要安裝排程器，請執行下列 Helm 命令：

```
helm repo add yunikorn https://apache.github.io/yunikorn-release
```

```
helm repo update
```

```
helm install yunikorn yunikorn/yunikorn --namespace yunikorn
```

## 使用 YuniKorn 排程器和 Spark Operator 來執行 Spark 應用程式

1. 如果您尚未完成，請先完成下節中的步驟進行設定：

   a. 建立您的叢集並設定 YuniKorn
   b. 安裝與設定 YuniKorn
   c. 為 Amazon EMR on EKS 設定 Spark Operator
   d. 安裝 Spark Operator

   當您執行 `helm install spark-operator-demo` 命令時，請包含下列引數：

   ```
   --set batchScheduler.enable=true
   --set webhook.enable=true
   ```

2. 建立 `SparkApplication` 定義檔案 `spark-pi.yaml`。

   若要使用 YuniKorn 作為作業的排程器，必須將某些註釋和標籤新增至應用程式定義。註釋和標籤會指定作業的佇列，以及您要使用的排程策略。

   在下列範例中，註釋 `schedulingPolicyParameters` 會設定應用程式的群排程。然後，此範例會建立作業群組或「成群」作業，以指定在排程 Pod 開始作業執行之前必須具備的最小容量。最後，它會在任務群組定義中進行指定，以使用帶有 `"app": "spark"` 標籤的節點群組，如 建立您的叢集並設定 YuniKorn 區段中所定義。

   ```
   apiVersion: "sparkoperator.k8s.io/v1beta2"
   kind: SparkApplication
   metadata:
   ```

```
  name: spark-pi
  namespace: spark-job
spec:
  type: Scala
  mode: cluster
  image: "895885662937.dkr.ecr.us-west-2.amazonaws.com/spark/emr-6.10.0:latest"
  imagePullPolicy: Always
  mainClass: org.apache.spark.examples.SparkPi
  mainApplicationFile: "local:///usr/lib/spark/examples/jars/spark-examples.jar"
  sparkVersion: "3.3.1"
  restartPolicy:
    type: Never
  volumes:
    - name: "test-volume"
      hostPath:
        path: "/tmp"
        type: Directory
  driver:
    cores: 1
    coreLimit: "1200m"
    memory: "512m"
    labels:
      version: 3.3.1
    annotations:
      yunikorn.apache.org/schedulingPolicyParameters: "placeholderTimeoutSeconds=30
 gangSchedulingStyle=Hard"
      yunikorn.apache.org/task-group-name: "spark-driver"
      yunikorn.apache.org/task-groups: |-
        [{
            "name": "spark-driver",
            "minMember": 1,
            "minResource": {
              "cpu": "1200m",
              "memory": "1Gi"
            },
            "nodeSelector": {
              "app": "spark"
            }
          },
          {
            "name": "spark-executor",
            "minMember": 1,
            "minResource": {
              "cpu": "1200m",
```

```
              "memory": "1Gi"
            },
            "nodeSelector": {
              "app": "spark"
            }
        }]
    serviceAccount: spark-sa
    volumeMounts:
      - name: "test-volume"
        mountPath: "/tmp"
  executor:
    cores: 1
    instances: 1
    memory: "512m"
    labels:
      version: 3.3.1
    annotations:
      yunikorn.apache.org/task-group-name: "spark-executor"
    volumeMounts:
      - name: "test-volume"
        mountPath: "/tmp"
```

3. 使用下列命令提交 Spark 應用程式。這也會建立 `SparkApplication` 物件，名為 `spark-pi`：

```
kubectl apply -f spark-pi.yaml
```

4. 使用下列命令檢查 `SparkApplication` 物件的事件：

```
kubectl describe sparkapplication spark-pi --namespace spark-job
```

第一個 Pod 事件顯示 YuniKorn 已排程 Pod：

```
Type    Reason            Age   From                          Message
----    ------            ----  ----                          -------
Normal Scheduling        3m12s yunikorn   spark-operator/org-apache-spark-examples-
sparkpi-2a777a88b98b8a95-driver is queued and waiting for allocation
Normal GangScheduling    3m12s yunikorn   Pod belongs to the taskGroup spark-
driver, it will be scheduled as a gang member
Normal Scheduled         3m10s yunikorn   Successfully assigned spark
Normal PodBindSuccessful 3m10s yunikorn   Pod spark-operator/
Normal TaskCompleted     2m3s  yunikorn   Task spark-operator/
Normal Pulling           3m10s kubelet    Pulling
```

## 使用 YuniKorn 排程器和 **spark-submit** 來執行 Spark 應用程式

1. 首先，請先完成 為 Amazon EMR on EKS 設定 spark-submit 一節中的步驟。
2. 設定以下環境變數的值：

```
export SPARK_HOME=spark-home
export MASTER_URL=k8s://Amazon-EKS-cluster-endpoint
```

3. 使用下列命令提交 Spark 應用程式：

   在下列範例中，註釋 `schedulingPolicyParameters` 會設定應用程式的群排程。然後，此範例會建立作業群組或「成群」作業，以指定在排程 Pod 開始作業執行之前必須具備的最小容量。最後，它會在任務群組定義中進行指定，以使用帶有 `"app": "spark"` 標籤的節點群組，如 建立您的叢集並設定 YuniKorn 區段中所定義。

```
$SPARK_HOME/bin/spark-submit \
 --class org.apache.spark.examples.SparkPi \
 --master $MASTER_URL \
 --conf spark.kubernetes.container.image=895885662937.dkr.ecr.us-
west-2.amazonaws.com/spark/emr-6.10.0:latest \
 --conf spark.kubernetes.authenticate.driver.serviceAccountName=spark-sa \
 --deploy-mode cluster \
 --conf spark.kubernetes.namespace=spark-job \
 --conf spark.kubernetes.scheduler.name=yunikorn \
 --conf spark.kubernetes.driver.annotation.yunikorn.apache.org/
schedulingPolicyParameters="placeholderTimeoutSeconds=30 gangSchedulingStyle=Hard"
 \
 --conf spark.kubernetes.driver.annotation.yunikorn.apache.org/task-group-
name="spark-driver" \
 --conf spark.kubernetes.executor.annotation.yunikorn.apache.org/task-group-
name="spark-executor" \
 --conf spark.kubernetes.driver.annotation.yunikorn.apache.org/task-groups='[{
            "name": "spark-driver",
            "minMember": 1,
            "minResource": {
              "cpu": "1200m",
              "memory": "1Gi"
            },
            "nodeSelector": {
              "app": "spark"
            }
          },
```

```
          {
            "name": "spark-executor",
            "minMember": 1,
            "minResource": {
              "cpu": "1200m",
              "memory": "1Gi"
            },
            "nodeSelector": {
              "app": "spark"
            }
        }]' \
 local:///usr/lib/spark/examples/jars/spark-examples.jar 20
```

4. 使用下列命令檢查 SparkApplication 物件的事件：

```
kubectl describe pod spark-driver-pod --namespace spark-job
```

第一個 Pod 事件顯示 YuniKorn 已排程 Pod：

```
Type    Reason           Age   From                          Message
----    ------           ----  ----                          -------
Normal Scheduling        3m12s yunikorn   spark-operator/org-apache-spark-examples-
sparkpi-2a777a88b98b8a95-driver is queued and waiting for allocation
Normal GangScheduling    3m12s yunikorn   Pod belongs to the taskGroup spark-
driver, it will be scheduled as a gang member
Normal Scheduled         3m10s yunikorn   Successfully assigned spark
Normal PodBindSuccessful 3m10s yunikorn   Pod spark-operator/
Normal TaskCompleted     2m3s  yunikorn   Task spark-operator/
Normal Pulling           3m10s kubelet    Pulling
```

# Amazon EMR on EKS 中的安全性

的雲端安全性 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，這些架構是為了符合最安全敏感組織的需求而建置。

安全性是 AWS 與您之間的共同責任。共同責任模型將其描述為雲端的安全性和雲端中的安全性：

- 雲端的安全性 – AWS 負責保護在 AWS Cloud 中執行 AWS 服務的基礎設施。 AWS 也為您提供可安全使用的服務。在AWS 合規計畫中，第三方稽核人員會定期測試和驗證我們的安全有效性。若要了解適用於 Amazon EMR 的合規計劃，請參閱AWS 合規計劃的 服務範圍。
- 雲端的安全性 – 您的責任取決於您使用 AWS 的服務。您也必須對其他因素負責，包括資料的機密性、您公司的要求和適用法律和法規。

本文件有助於您了解如何在使用 Amazon EMR on EKS 時套用共同責任模型。下列主題說明如何將 Amazon EMR on 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 EMR on EKS 資源。

主題


- Amazon EMR on EKS 安全最佳實務
- 資料保護
- 身分和存取權管理
- 使用 Amazon EMR on EKS 搭配 AWS Lake Formation 進行精細存取控制
- 日誌記錄和監控
- 將 Amazon S3 Access Grants 與 Amazon EMR on EKS 搭配使用
- Amazon EMR on EKS 的合規驗證
- Amazon EMR on EKS 的恢復能力
- Amazon EMR on EKS 中的基礎設施安全性
- 組態與漏洞分析
- 使用介面 VPC 端點連接至 Amazon EMR on EKS
- 設定 Amazon EMR on EKS 的跨帳戶存取權

# Amazon EMR on EKS 安全最佳實務

在您開發和實作自己的安全政策時，可考慮使用 Amazon EMR on EKS 提供的多種安全功能。以下最佳實務為一般準則，並不代表完整的安全解決方案。這些最佳實務可能不適用或無法滿足您的環境需求，因此請將其視為實用建議就好，而不要當作是指示。

> **Note**
>
> 如需安全最佳實務的詳細資訊，請參閱 Amazon EMR on EKS 安全最佳實務。

## 套用最低權限準則

Amazon EMR on EKS 為使用 IAM 角色 (例如執行角色) 的應用程式提供精細存取政策。這些執行角色會透過 IAM 角色的信任政策映射至 Kubernetes 服務帳戶。Amazon EMR on EKS 會在已註冊的 Amazon EKS 命名空間內建立 Pod，它們可執行使用者提供的應用程式碼。執行應用程式程式碼的任務 Pod 在連線至其他 AWS 服務時擔任執行角色。建議只授予作業所需的最低權限集，例如覆蓋應用程式和日誌目的地的存取權限。我們還建議定期以及在應用程式碼發生變更時審核作業許可。

## 端點的存取控制清單

僅可針對已設定為在 VPC 中至少使用一個私有子網路的 EKS 叢集建立受管端點。此組態會限制受管端點建立的負載平衡器的存取，以便只能從 VPC 存取它們。若要進一步增強安全性，建議使用這些負載平衡器設定安全群組，以便它們可以將傳入流量限制到選取的一組 IP 地址。

## 取得自訂映像的最新安全更新

若要搭配使用自訂映像與 Amazon EMR on EKS，您可以在映像上安裝任何二進位檔案和程式庫。您負責對新增至映像的二進位檔案進行安全修補。會定期使用最新的安全修補程式修補 Amazon EMR on EKS 映像。若要取得最新映像，只要有 Amazon EMR 版本的新基礎映像版本，就必須重建自訂映像。如需詳細資訊，請參閱 Amazon EMR on EKS 發行版本 和 選取基礎映像 URI 的詳細資訊。

## 限制 Pod 憑證存取

Kubernetes 支援數種將憑證指派給 Pod 的方法。佈建多個憑證提供者可能會增加安全模型的複雜性。Amazon EMR on EKS 已採用服務帳戶的 IAM 角色 (IRSA) 作為已註冊的 EKS 命名空間內的標準憑證提供者。不支援其他方法，包括 kube2iam、kiam 以及使用在叢集上執行之執行個體的 EC2 執行個體設定檔。

## 隔離不受信任的應用程式碼

Amazon EMR on EKS 不會檢查系統使用者提交的應用程式碼的完整性。如果您執行的是使用多個執行角色設定的多租戶虛擬叢集，而執行任意程式碼的不受信任的租用戶可使用這些角色提交作業，則可能存在惡意應用程式提升其權限的風險。在這種情況下，請考慮將具有類似權限的執行角色隔離到不同的虛擬叢集中。

## 角色型存取控制 (RBAC) 許可

管理員應嚴格控制 Amazon EMR on EKS 受管命名空間的角色型存取控制 (RBAC) 許可。至少，不應將下列許可授予給 Amazon EMR on EKS 受管命名空間中的作業提交者。

- Kubernetes RBAC 許可能夠修改 configmap，因為 Amazon EMR on EKS 使用 Kubernetes configmaps 來產生具有受管服務帳戶名稱的受管 Pod 範本。此屬性不應發生突變。
- Kubernetes RBAC 許可能夠執行至 Amazon EMR on EKS Pod 中，以避免對具有受管 SA 名稱的受管 Pod 範本授予存取權。此屬性不應發生突變。此許可准許存取掛載至 Pod 的 JWT 字符，然後可用來擷取執行角色憑證。
- 建立 Pod 的 Kubernetes RBAC 許可 - 防止使用者使用 Kubernetes ServiceAccount 建立 Pod，而 Kubernetes ServiceAccount 可能會對應到具有比使用者更多 AWS 權限的 IAM 角色。
- Kubernetes RBAC 許可能夠部署發生變化的 Webhook，以防止使用者使用發生變化的 Webhook 來變更由 Amazon EMR on EKS 建立的 Pod 的 Kubernetes ServiceAccount 名稱。
- Kubernetes RBAC 許可能夠讀取 Kubernetes 密碼，以防止使用者讀取儲存在這些密碼中的機密資料。

## 限制存取節點群組 IAM 角色或執行個體設定檔憑證

- 建議您將最低 AWS 許可指派給節點群組的 IAM 角色 (IAM)。這有助於避免使用 EKS 工作節點的執行個體設定檔憑證執行的程式碼進行權限提升。
- 若要完全阻止對在 Amazon EMR on EKS 受管命名空間中執行的所有 Pod 的執行個體設定檔憑證的存取，建議您在 EKS 節點上執行 `iptables` 命令。如需詳細資訊，請參閱限制存取 Amazon EC2 執行個體設定檔憑證。不過，請務必適當限制您的服務帳戶 IAM 角色範圍，以便 Pod 擁有所有必要的許可。例如，節點 IAM 角色會被指派從 Amazon ECR 中提取容器映像的許可。如果 Pod 沒有被指派這些許可，則該 Pod 無法從 Amazon ECR 中提取容器映像。VPC CNI 外掛程式也需要更新。如需詳細資訊，請參閱逐步解說：更新 VPC CNI 外掛程式以使用服務帳戶的 IAM 角色。

# 資料保護

AWS 共同責任模型適用於 Amazon EMR on EKS 中的資料保護。如此模型所述， AWS 負責保護執行所有雲端的 AWS 全域基礎設施。您必須負責維護在此基礎設施上託管之內容的控制權。此內容包括您所使用 AWS 服務的安全組態和管理任務。如需有關資料隱私權的詳細資訊，請參閱資料隱私權常見問答集。如需歐洲資料保護的相關資訊，請參閱 AWS 安全部落格上的AWS 共同責任模型和 GDPR 部落格文章。

基於資料保護目的，我們建議您保護 AWS 帳戶登入資料，並使用 AWS Identity and Access Management (IAM) 設定個別帳戶。如此一來，每個使用者都只會獲得授予完成其任務所必須的許可。我們也建議您採用下列方式保護資料：

- 每個帳戶均要使用多重要素驗證 (MFA)。
- 使用 SSL/TLS 與 AWS 資源通訊。建議使用 TLS 1.2 或更新版本。
- 使用 設定 API 和使用者活動記錄 AWS CloudTrail。
- 使用 AWS 加密解決方案，以及 AWS 服務中的所有預設安全控制。
- 使用進階的受管安全服務 (例如 Amazon Macie)，協助探索和保護儲存在 Simple Storage Service (Amazon Simple Storage Service (Amazon S3)) 的個人資料。
- 使用 Amazon EMR on EKS 加密選項來加密靜態和傳輸中的資料。
- 如果您在 AWS 透過命令列界面或 API 存取 時需要 FIPS 140-2 驗證的密碼編譯模組，請使用 FIPS 端點。如需有關 FIPS 和 FIPS 端點的更多相關資訊，請參閱聯邦資訊處理標準 (FIPS) 140-2 概觀。

我們強烈建議您絕對不要將客戶帳戶號碼等敏感的識別資訊，放在自由格式的欄位中，例如名稱欄位。這包括當您使用 Amazon EMR on EKS 或其他使用主控台 AWS CLI、API 或 AWS SDKs AWS 的服務時。在 Amazon EMR on EKS 或其他服務中輸入的任何資料都可能被選入診斷日誌中。當您提供外部伺服器的 URL 時，請勿在驗證您對該伺服器請求的 URL 中包含憑證資訊。

## 靜態加密

資料加密有助於防止未經授權的使用者讀取叢集上的資料和相關的資料儲存體系統。這包括儲存到持久性媒體的資料 (稱為靜態資料)，以及透過網路傳送時可能會被攔截的資料 (稱為傳輸中資料)。

資料加密需要金鑰和憑證。您可以從多個選項中選擇，包括 管理的金鑰 AWS Key Management Service、Amazon S3 管理的金鑰，以及來自您提供的自訂提供者的金鑰和憑證。使用 AWS KMS 做為金鑰提供者時，儲存和使用加密金鑰會收取費用。如需詳細資訊，請參閱 AWS KMS 定價。

在指定加密選項之前，請先決定要使用的金鑰和憑證管理系統。然後為您指定作為加密設定一部分的自訂提供者建立金鑰和憑證。

## Amazon S3 中 EMRFS 資料的靜態加密

Amazon S3 加密適用於讀取和寫入至 Amazon S3 的 EMR 檔案系統 (EMRFS) 物件。啟用靜態加密時，會指定 Amazon S3 伺服器端加密 (SSE) 或用戶端加密 (CSE) 作為預設加密模式。或者，您可以使用 Per bucket encryption overrides (每個儲存貯體加密覆寫) 為個別儲存貯體指定不同的加密方法。無論是否啟用了 Amazon S3 加密功能，Transport Layer Security (TLS) 都會將 EMR 叢集節點和 Amazon S3 之間傳送中的 EMRFS 物件加密。如需有關 Amazon S3 加密的深入資訊，請參閱《Amazon Simple Storage Service 開發人員指南》中的使用加密保護資料。

> **Note**
>
> 當您使用 時 AWS KMS，儲存和使用加密金鑰會收取費用。如需詳細資訊，請參閱 AWS KMS 定價。

## Amazon S3 伺服器端加密

當您設定 Amazon S3 伺服器端加密時，Amazon S3 會在將資料寫入磁碟時在物件層級加密資料，並在存取時解密資料。如需有關 SSE 的詳細資訊，請參閱《Amazon Simple Storage Service 開發人員指南》中的使用伺服器端加密保護資料。

當您在 Amazon EMR on EKS 中指定 SSE 時，可以在兩種不同的金鑰管理系統中選擇：

- SSE-S3 – Amazon S3 為您管理密鑰。
- SSE-KMS - 您可以使用 AWS KMS key 設定適用於 Amazon EMR on EKS 的政策。

具有客戶提供的金鑰之 SSE (SSE-C) 不適用於 Amazon EMR on EKS。

## Amazon S3 用戶端加密

使用 Amazon S3 用戶端加密，Amazon S3 加密及解密會在您 EMR 叢集上的 EMRFS 用戶端中進行。物件在上傳至 Amazon S3 之前會先加密，並在下載後解密。您指定的提供者會提供用戶端使用的加密金鑰。用戶端可以使用 AWS KMS (CSE-KMS) 提供的金鑰或提供用戶端根金鑰 (CSE-C) 的自訂 Java 類別。CSE-KMS 和 CSE-C 之間的加密細節略有不同，具體取決於指定的提供者和要解密或加密之物件的中繼資料。如需有關這些差異的詳細資訊，請參閱《Amazon Simple Storage Service 開發人員指南》中的使用用戶端加密保護資料。

> **Note**
>
> Amazon S3 CSE 只能確保與 Amazon S3 交換的 EMRFS 資料經過加密；而不會加密叢集執行個體磁碟區上的所有資料。此外，由於 Hue 不使用 EMRFS，因此 Hue S3 檔案瀏覽器寫入到 Amazon S3 的物件不會被加密。

### 本機磁碟加密

Apache Spark 支援加密寫入到本機磁碟的臨時資料。這涵蓋了隨機排序檔案、隨機排序溢出以及儲存在磁碟上用於快取和廣播變數的資料區塊。它不包括使用 API (例如 `saveAsHadoopFile` 或 `saveAsTable`) 對應用程式產生的輸出資料進行加密。它也可能不包括使用者明確建立的臨時檔案。如需詳細資訊，請參閱 Spark 文件中的本機儲存加密。Spark 不支援本機磁碟上的加密資料，例如當資料不適合記憶體時，由執行程式處理程序寫入本機磁碟的中繼資料。保留在磁碟的資料限定為作業執行期，而 Spark 會在每次執行作業時動態產生用來加密資料的金鑰。一旦 Spark 作業終止，沒有其他程序可以解密資料。

對於驅動程式和執行程式 Pod，您可以加密保留在已掛載磁碟區的靜態資料。您可以搭配 AWS Kubernetes 使用三種不同的原生儲存選項：EBS、EFS 和 FSx for Lustre。這三種選項全部都可使用服務受管金鑰或 AWS KMS key提供靜態加密。如需詳細資訊，請參閱 EKS 最佳實務指南。使用此方法，所有保留在已掛載磁碟區的資料都會加密。

### 金鑰管理

可以將 KMS 設定為自動輪換 KMS 金鑰。這將每年輪換一次金鑰，同時無限期儲存舊金鑰，以便您的資料仍然可以解密。如需詳細資訊，請參閱輪換 AWS KMS keys。

## 傳輸中加密

數種加密機制在使用傳輸中加密時啟用。這些是開放原始碼功能，僅適用特定應用程式，可能會隨 Amazon EMR on EKS 版本而異。下列應用程式特定的加密功能可透過 Amazon EMR on EKS 啟用：

- Spark
  - 會使用在 Amazon EMR 5.9.0 版本和更新版本中的 AES-256 密碼來加密 Spark 元件之間的內部 RPC 通訊 (例如區塊傳輸服務和外部混洗服務)。在較早版本中，內部 RPC 通訊使用 SASL 搭配 DIGEST-MD5 做為密碼進行加密。
  - HTTP 協定通訊具有例如 Spark 歷史記錄伺服器和已啟用 HTTPS 檔案伺服器的使用者介面，使用 Spark 的 SSL 組態進行加密。如需詳細資訊，請參閱 Spark 文件中的 SSL 組態。

如需詳細資訊，請參閱 Spark 安全設定。

- 應該使用 Amazon S3 儲存貯體 IAM 政策上的 aws:SecureTransport 條件，僅允許 HTTPS (TLS) 上的加密連線。
- 串流至 JDBC 或 ODBC 用戶端的查詢結果會使用 TLS 來加密。

# 身分和存取權管理

AWS Identity and Access Management (IAM) 是 AWS 服務 ，可協助管理員安全地控制對 AWS 資源的存取。IAM 管理員可以控制誰可以經過驗證 (已登入) 和授權 (具有許可) 來使用 Amazon EMR on EKS 資源。IAM 是 AWS 服務 您可以免費使用的 。

主題


- 目標對象
- 使用身分驗證
- 使用政策管理存取權
- Amazon EMR on EKS 如何搭配 IAM 運作
- 使用 Amazon EMR on EKS 的服務連結角色
- Amazon EMR on EKS 的受管政策
- 搭配使用作業執行角色與 Amazon EMR on EKS
- Amazon EMR on EKS 身分型政策範例
- 標籤型存取控制政策
- Amazon EMR on EKS 身分識別和存取疑難排解


## 目標對象

使用方式 AWS Identity and Access Management (IAM) 會有所不同，取決於您在 Amazon EMR on EKS 中執行的工作。

服務使用者 – 如果您使用 Amazon EMR on EKS 服務執行作業，管理員會為您提供所需的憑證和許可。隨著您為了執行作業而使用的 Amazon EMR on EKS 功能數量變多，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 EMR on EKS 中的功能，請參閱 Amazon EMR on EKS 身分識別和存取疑難排解。

服務管理員 – 若您在公司負責管理 Amazon EMR on EKS 資源，您應該可完全存取 Amazon EMR on EKS。您的作業是判斷服務使用者應存取哪些 Amazon EMR on 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如何搭配使用 IAM 與 Amazon EMR on EKS，請參閱 Amazon EMR on EKS 如何搭配 IAM 運作。

IAM 管理員 – 如果您是 IAM 管理員，建議您掌握如何撰寫政策以管理 Amazon EMR on EKS 存取權的詳細資訊。若要檢視您可以在 IAM 中使用的 Amazon EMR on EKS 身分型政策範例，請參閱 Amazon EMR on EKS 身分型政策範例。

## 使用身分驗證

身分驗證是您 AWS 使用身分憑證登入 的方式。您必須以 AWS 帳戶根使用者身分、IAM 使用者身分或擔任 IAM 角色來驗證 （登入 AWS)。

您可以使用透過身分來源提供的憑證，以聯合身分 AWS 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》中的如何登入您的 AWS 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。如需使用建議的方法自行簽署請求的詳細資訊，請參閱《IAM 使用者指南》中的適用於 API 請求的AWS Signature 第 4 版。

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 建議您使用多重驗證 (MFA) 來提高帳戶的安全性。如需更多資訊，請參閱《AWS IAM Identity Center 使用者指南》中的多重要素驗證和《IAM 使用者指南》中的 IAM 中的AWS 多重要素驗證。

### AWS 帳戶 根使用者

當您建立 時 AWS 帳戶，您會從一個登入身分開始，該身分可完整存取帳戶中的所有 AWS 服務 和資源。此身分稱為 AWS 帳戶 Theroot 使用者，可透過使用您用來建立帳戶的電子郵件地址和密碼登入來存取。強烈建議您不要以根使用者處理日常任務。保護您的根使用者憑證，並將其用來執行只能由根使用者執行的任務。如需這些任務的完整清單，了解需以根使用者登入的任務，請參閱 IAM 使用者指南中的需要根使用者憑證的任務。

## 聯合身分

最佳實務是， 要求人類使用者，包括需要管理員存取權的使用者，使用 聯合身分提供者 AWS 服務 來使用臨時憑證來存取 。

聯合身分是來自您的企業使用者目錄、Web 身分提供者、 AWS Directory Service、 Identity Center 目錄或任何使用透過身分來源提供的登入資料 AWS 服務 存取的使用者。當聯合身分存取時 AWS 帳戶，它們會擔任 角色，而角色會提供臨時登入資料。

對於集中式存取權管理，我們建議您使用 AWS IAM Identity Center。您可以在 IAM Identity Center 中建立使用者和群組，也可以連接並同步到您自己的身分來源中的一組使用者 AWS 帳戶 和群組，以便在所有 和應用程式中使用。如需 IAM Identity Center 的詳細資訊，請參閱 AWS IAM Identity Center 使用者指南中的什麼是 IAM Identity Center？。

## IAM 使用者和群組

IAM 使用者是 中的身分 AWS 帳戶 ，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。您無法以群組身分簽署。您可以使用群組來一次為多名使用者指定許可。群組可讓管理大量使用者許可的程序變得更為容易。例如，您可以擁有一個名為 IAMAdmins 的群組，並給予該群組管理 IAM 資源的許可。

使用者與角色不同。使用者只會與單一人員或應用程式建立關聯，但角色的目的是在由任何需要它的人員取得。使用者擁有永久的長期憑證，但角色僅提供臨時憑證。如需更多資訊，請參閱《IAM 使用者指南》中的 IAM 使用者的使用案例。

## IAM 角色

IAM 角色是 中具有特定許可 AWS 帳戶 的身分。它類似 IAM 使用者，但不與特定的人員相關聯。若要暫時在 中擔任 IAM 角色 AWS Management Console，您可以從使用者切換至 IAM 角色 （主控台）。您可以透過呼叫 AWS CLI 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》中的擔任角色的方法。

使用臨時憑證的 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 (聯合) 建立角色。如果您使用 IAM Identity Center，則需要設定許可集。為控制身分驗證後可以存取的內容，IAM Identity Center 將許可集與 IAM 中的角色相關聯。如需有關許可集的資訊，請參閱 AWS IAM Identity Center 使用者指南中的許可集。

- 暫時 IAM 使用者許可 – IAM 使用者或角色可以擔任 IAM 角色來暫時針對特定任務採用不同的許可。
- 跨帳戶存取權：您可以使用 IAM 角色，允許不同帳戶中的某人 (信任的主體) 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 中的功能 AWS 服務。例如，當您在服務中進行呼叫時，該服務通常會在 Amazon EC2 中執行應用程式或將物件儲存在 Amazon Simple Storage Service (Amazon S3) 中。服務可能會使用呼叫主體的許可、使用服務角色或使用服務連結角色來執行此作業。
  - 轉送存取工作階段 (FAS) – 當您使用 IAM 使用者或角色在其中執行動作時 AWS，您會被視為委託人。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 的委託人許可 AWS 服務，結合 AWS 服務 請求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》中的建立角色以委派許可權給 AWS 服務。
  - 服務連結角色 – 服務連結角色是一種連結至 的服務角色。 AWS 服務服務可以擔任代表您執行動作的角色。服務連結角色會出現在您的 中 AWS 帳戶 ，並由服務擁有。IAM 管理員可以檢視，但不能編輯服務連結角色的許可。
- 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臨時憑證。如需詳細資訊，請參閱《IAM 使用者指南》中的使用 IAM 角色來授予許可權給 Amazon EC2 執行個體上執行的應用程式。

## 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，當與身分或資源相關聯時， AWS 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 AWS CLI、 或 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。如需了解如何建立身分型政策，請參閱《IAM 使用者指南》中的透過客戶管理政策定義自訂 IAM 許可。

身分型政策可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 中的多個使用者、群組和角色 AWS 帳戶。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之間選擇，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、聯合身分使用者或 AWS 服務。

資源型政策是位於該服務中的內嵌政策。您無法在以資源為基礎的政策中使用來自 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。所產生的許可會是實體的身分型政策和其許可界限的交集。會在 `Principal` 欄位中指定使用者或角色的資源型政策則不會受到許可界限限制。所有這類政策中的明確拒絕都會覆寫該允許。如需許可界限的詳細資訊，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 中組織或組織單位 (OU) 的最大許可 AWS Organizations。 AWS Organizations 是一種用於分組和集中管理您企業擁有 AWS 帳戶 之多個 的服務。若您啟用組織中的所有功能，您可以將服務控制政策 (SCP) 套用到任何或所有帳戶。SCP 會限制成員帳戶中實體的許可，包括每個實體 AWS 帳戶根使用者。如需 Organizations 和 SCP 的詳細資訊，請參閱《AWS Organizations 使用者指南》中的服務控制政策。
- 資源控制政策 (RCP) - RCP 是 JSON 政策，可用來設定您帳戶中資源的可用許可上限，採取這種方式就不需要更新附加至您所擁有的每個資源的 IAM 政策。RCP 會限制成員帳戶中資源的許可，並可能影響身分的有效許可，包括 AWS 帳戶根使用者，無論它們是否屬於您的組織。如需 Organizations 和 RCPs的詳細資訊，包括 AWS 服務 支援 RCPs 清單，請參閱AWS Organizations《 使用者指南》中的資源控制政策 (RCPs)。
- 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 AWS 決定是否在涉及多個政策類型時允許請求，請參閱《IAM 使用者指南》中的政策評估邏輯。

## Amazon EMR on EKS 如何搭配 IAM 運作

在您使用 IAM 管理對 Amazon EMR on EKS 的存取權之前，請了解哪些 IAM 功能可以與 Amazon EMR on EKS 搭配使用。

您可以搭配 Amazon EMR on EKS 使用的 IAM 功能

| IAM 功能 | 支援 Amazon EMR on EKS |
| --- | --- |
| 身分型政策 | 是 |
| 資源型政策 | 否 |
| 政策動作 | 是 |
| 政策資源 | 是 |
| 政策條件索引鍵 | 是 |
| ACL | 否 |
| ABAC (政策中的標籤) | 是 |
| 臨時憑證 | 是 |
| 主體許可 | 是 |
| 服務角色 | 否 |
| 服務連結角色 | 是 |

若要全面了解 Amazon EMR on EKS 和其他 AWS 服務如何與大多數 IAM 功能搭配使用，請參閱《IAM 使用者指南》中的與 IAM AWS 搭配使用的 服務。

## Amazon EMR on EKS 身分型政策

支援身分型政策：是

身分型政策是可以附加到身分 (例如 IAM 使用者、使用者群組或角色) 的 JSON 許可政策文件。這些政策可控制身分在何種條件下能對哪些資源執行哪些動作。如需了解如何建立身分型政策，請參閱《IAM 使用者指南》中的透過客戶管理政策定義自訂 IAM 許可。

使用 IAM 身分型政策，您可以指定允許或拒絕的動作和資源，以及在何種條件下允許或拒絕動作。您無法在身分型政策中指定主體，因為這會套用至連接的使用者或角色。如要了解您在 JSON 政策中使用的所有元素，請參閱《IAM 使用者指南》中的 IAM JSON 政策元素參考。

Amazon EMR on EKS 身分型政策範例

若要檢視 Amazon EMR on EKS 身分型政策的範例，請參閱 Amazon EMR on EKS 身分型政策範例。

## Amazon EMR on EKS 中的資源型政策

支援資源型政策：否

資源型政策是附加到資源的 JSON 政策文件。資源型政策的最常見範例是 IAM 角色信任政策和 Amazon S3 儲存貯體政策。在支援資源型政策的服務中，服務管理員可以使用它們來控制對特定資源的存取權限。對於附加政策的資源，政策會定義指定的主體可以對該資源執行的動作以及在何種條件下執行的動作。您必須在資源型政策中指定主體。委託人可以包括帳戶、使用者、角色、聯合身分使用者或 AWS 服務。

如需啟用跨帳戶存取權，您可以指定在其他帳戶內的所有帳戶或 IAM 實體，做為資源型政策的主體。新增跨帳戶主體至資源型政策，只是建立信任關係的一半。當委託人和資源位於不同位置時 AWS 帳戶，信任帳戶中的 IAM 管理員也必須授予委託人實體（使用者或角色）存取資源的許可。其透過將身分型政策連接到實體來授與許可。不過，如果資源型政策會為相同帳戶中的主體授予存取，這時就不需要額外的身分型政策。如需詳細資訊，請參閱《IAM 使用者指南》中的 IAM 中的快帳戶資源存取。

## Amazon EMR on EKS 的政策動作

支援政策動作：是

管理員可以使用 AWS JSON 政策來指定誰可以存取內容。也就是說，哪個主體在什麼条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，例如沒有相符的 API 操作的僅限許可動作。也有一些作業需要政策中的多個動作。這些額外的動作稱為相依動作。

政策會使用動作來授予執行相關聯動作的許可。

如需 Amazon EMR on EKS 動作清單，請參閱服務授權參考中的適用於 Amazon EMR on EKS 的動作、資源及條件金鑰。

Amazon EMR on EKS 中的政策動作會在動作之前使用下列字首：

```
emr-containers
```

若要在單一陳述式中指定多個動作，請用逗號分隔。

```
"Action": [
      "emr-containers:action1",
      "emr-containers:action2"
         ]
```

若要檢視 Amazon EMR on EKS 身分型政策的範例，請參閱 Amazon EMR on EKS 身分型政策範例。

## Amazon EMR on EKS 的政策資源

支援政策資源：是

管理員可以使用 AWS JSON 政策來指定誰可以存取內容。也就是說，哪個主體在什麼條件下可以對什麼資源執行哪些動作。

`Resource` JSON 政策元素可指定要套用動作的物件。陳述式必須包含 `Resource` 或 `NotResource` 元素。最佳實務是使用其 Amazon Resource Name (ARN) 來指定資源。您可以針對支援特定資源類型的動作 (稱為資源層級許可) 來這麼做。

對於不支援資源層級許可的動作 (例如列出操作)，請使用萬用字元 (*) 來表示陳述式適用於所有資源。

```
"Resource": "*"
```

若要查看 Amazon EMR on EKS 資源類型及其 ARN 的清單，請參閱《服務授權參考》中的 Amazon EMR on EKS 定義的資源。若要了解可指定每個資源 ARN 的動作，請參閱適用於 Amazon EMR on EKS 的動作、資源及條件索金鑰。

若要檢視 Amazon EMR on EKS 身分型政策的範例，請參閱 Amazon EMR on EKS 身分型政策範例。

## Amazon EMR on EKS 的政策條件金鑰

支援服務特定政策條件金鑰：是

管理員可以使用 AWS JSON 政策來指定誰可以存取內容。也就是說，哪個主體在什麼條件下可以對什麼資源執行哪些動作。

`Condition` 元素 (或 `Condition` 區塊) 可讓您指定使陳述式生效的條件。`Condition` 元素是選用項目。您可以建立使用條件運算子的條件運算式 (例如等於或小於)，來比對政策中的條件和請求中的值。

若您在陳述式中指定多個 `Condition` 元素，或是在單一 `Condition` 元素中指定多個索引鍵， AWS 會使用邏輯 AND 操作評估他們。如果您為單一條件索引鍵指定多個值， 會使用邏輯OR操作 AWS 評估條件。必須符合所有條件，才會授與陳述式的許可。

您也可以在指定條件時使用預留位置變數。例如，您可以只在使用者使用其 IAM 使用者名稱標記時，將存取資源的許可授予該 IAM 使用者。如需更多資訊，請參閱 IAM 使用者指南中的 IAM 政策元素：變數和標籤。

AWS 支援全域條件金鑰和服務特定的條件金鑰。若要查看所有 AWS 全域條件索引鍵，請參閱《IAM 使用者指南》中的AWS 全域條件內容索引鍵。

如需 Amazon EMR on EKS 條件金鑰清單並要了解可使用條件金鑰的動作和資源，請參閱服務授權參考中的適用於 Amazon EMR on EKS 的動作、資源及條件金鑰。

若要檢視 Amazon EMR on EKS 身分型政策的範例，請參閱 Amazon EMR on EKS 身分型政策範例。

## Amazon EMR on EKS 中的存取控制清單 (ACL)

支援 ACL：否

存取控制清單 (ACL) 可控制哪些主體 (帳戶成員、使用者或角色) 擁有存取某資源的許可。ACL 類似於資源型政策，但它們不使用 JSON 政策文件格式。

## 屬性型存取控制 (ABAC) 與 Amazon EMR on EKS

| | |
|---|---|
| 支援 ABAC (政策中的標籤) | 是 |

屬性型存取控制 (ABAC) 是一種授權策略，可根據屬性來定義許可。在 中 AWS，這些屬性稱為標籤。您可以將標籤連接到 IAM 實體 （使用者或角色） 和許多 AWS 資源。為實體和資源加上標籤是 ABAC 的第一步。您接著要設計 ABAC 政策，允許在主體的標籤與其嘗試存取的資源標籤相符時操作。

ABAC 在成長快速的環境中相當有幫助，並能在政策管理變得繁瑣時提供協助。

如需根據標籤控制存取，請使用 `aws:ResourceTag/key-name`、`aws:RequestTag/key-name` 或 `aws:TagKeys` 條件索引鍵，在政策的條件元素中，提供標籤資訊。

如果服務支援每個資源類型的全部三個條件金鑰，則對該服務而言，值為 Yes。如果服務僅支援某些資源類型的全部三個條件金鑰，則值為 Partial。

如需 ABAC 的詳細資訊，請參閱《IAM 使用者指南》中的使用 ABAC 授權定義許可。如要查看含有設定 ABAC 步驟的教學課程，請參閱 IAM 使用者指南中的使用屬性型存取控制 (ABAC)。

## 將暫時憑證與 Amazon EMR on EKS 搭配使用

支援臨時憑證：是

當您使用臨時登入資料登入時，有些 AWS 服務 無法運作。如需詳細資訊，包括哪些 AWS 服務 使用臨時登入資料，請參閱《AWS 服務 IAM 使用者指南》中的使用 IAM 的 。

如果您 AWS Management Console 使用使用者名稱和密碼以外的任何方法登入 ，則會使用臨時登入資料。例如，當您 AWS 使用公司的單一登入 (SSO) 連結存取 時，該程序會自動建立臨時登入資料。當您以使用者身分登入主控台，然後切換角色時，也會自動建立臨時憑證。如需切換角色的詳細資訊，請參閱《IAM 使用者指南》中的從使用者切換至 IAM 角色 (主控台)。

您可以使用 AWS CLI 或 AWS API 手動建立臨時登入資料。然後，您可以使用這些臨時登入資料來存取 AWS。 AWS 建議您動態產生臨時登入資料，而不是使用長期存取金鑰。如需詳細資訊，請參閱 IAM 中的暫時性安全憑證。

## Amazon EMR on EKS 的跨服務主體許可

支援轉寄存取工作階段 (FAS)：是

當您使用 IAM 使用者或角色在 中執行動作時 AWS，您會被視為委託人。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 的委託人許可 AWS 服務，結合 AWS 服務 請求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。

## Amazon EMR on EKS 的服務角色

| | |
|---|---|
| 支援服務角色 | 否 |

## Amazon EMR on EKS 的服務連結角色

| | |
|---|---|
| 支援服務連結角色 | 是 |

如需建立或管理服務連結角色的詳細資訊，請參閱可搭配 IAM 運作的AWS 服務。在表格中尋找服務，其中包含服務連結角色欄中的 Yes。選擇是連結，以檢視該服務的服務連結角色文件。

# 使用 Amazon EMR on EKS 的服務連結角色

Amazon EMR on 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 EMR on EKS 的一種特殊 IAM 角色類型。服務連結角色由 Amazon EMR on 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。

服務連結角色可讓設定 Amazon EMR on EKS 更為簡單，因為您不必手動新增必要的許可。Amazon EMR on EKS 定義其服務連結角色的許可，除非另有定義，否則僅有 Amazon EMR on 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。如此可保護您的 Amazon EMR on EKS 資源，避免您不小心移除資源的存取許可。

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，並尋找 Service-Linked Role (服務連結角色) 欄顯示為 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 EMR on EKS 的服務連結角色許可

Amazon EMR on EKS 使用名稱為 AWSServiceRoleForAmazonEMRContainers 的服務連結角色。

`AWSServiceRoleForAmazonEMRContainers` 服務連結角色信任下列服務以擔任角色：

- `emr-containers.amazonaws.com`

此角色許可政策 `AmazonEMRContainersServiceRolePolicy` 允許 Amazon EMR on EKS 對指定資源完成一組動作，如以下政策聲明所示。

> **Note**
>
> 受管政策的內容會改變，因此此處顯示的政策有可能不是最新的。《 受管政策參考指南》中的 AmazonEMRContainersServiceRolePolicy 檢視up-to-date政策文件。 AWS

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
```

```
            "Effect": "Allow",
            "Action": [
                "eks:DescribeCluster",
                "eks:ListNodeGroups",
                "eks:DescribeNodeGroup",
                "ec2:DescribeRouteTables",
                "ec2:DescribeSubnets",
                "ec2:DescribeSecurityGroups",
                "elasticloadbalancing:DescribeInstanceHealth",
                "elasticloadbalancing:DescribeLoadBalancers",
                "elasticloadbalancing:DescribeTargetGroups",
                "elasticloadbalancing:DescribeTargetHealth",
                "eks:ListPodIdentityAssociations",
                "eks:DescribePodIdentityAssociation"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "acm:ImportCertificate",
                "acm:AddTagsToCertificate"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:RequestTag/emr-container:endpoint:managed-certificate":
 "true"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "acm:DeleteCertificate"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/emr-container:endpoint:managed-certificate":
 "true"
                }
            }
        }
```

```
    ]
}
```

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 EMR on EKS 的服務連結角色

您不需要手動建立一個服務連結角色。當您建立虛擬叢集時，Amazon EMR on 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。當您建立虛擬叢集時，Amazon EMR on EKS 會再次為您建立服務連結角色。

您也可以使用 IAM 主控台，透過 Amazon EMR on EKS 使用案例建立服務連結角色。在 AWS CLI 或 AWS API 中，使用服務名稱建立`emr-containers.amazonaws.com`服務連結角色。如需詳細資訊，請參閱 IAM 使用者指南中的建立服務連結角色。如果您刪除此服務連結角色，您可以使用此相同的程序以再次建立該角色。

## 編輯 Amazon EMR on EKS 的服務連結角色

Amazon EMR on EKS 不允許您編輯 `AWSServiceRoleForAmazonEMRContainers` 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。如需更多資訊，請參閱 IAM 使用者指南中的編輯服務連結角色。

## 刪除 Amazon EMR on 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，在手動刪除服務連結角色之前，您必須先清除資源。

> **Note**
>
> 若 Amazon EMR on 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。

若要刪除 **`AWSServiceRoleForAmazonEMRContainers`** 使用的 Amazon EMR on EKS 資源

1. 開啟 Amazon EMR 主控台。

2. 選擇虛擬叢集。
3. 在 `Virtual Cluster` 頁面中，選擇刪除。
4. 對帳戶中的任何其他虛擬叢集重複此程序。

使用 IAM 手動刪除服務連結角色

使用 IAM 主控台 AWS CLI、 或 AWS API 來刪除`AWSServiceRoleForAmazonEMRContainers`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 EMR on EKS 服務連結角色的支援區域

Amazon EMR on 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 EMR on EKS 服務端點和配額。

## Amazon EMR on EKS 的受管政策

檢視自 2021 年 3 月 1 日起 Amazon EMR on EKS AWS 受管政策更新的詳細資訊。

| 變更 | 描述 | 日期 |
| --- | --- | --- |
| `AmazonEMRContainersServiceRolePolicy` - 新增讀取許可以列出叢集中的 EKS Pod 身分關聯，以及另一個讀取許可，以傳回叢集中 Pod 身分關聯的描述性資訊。如需詳細資訊，請參閱 AmazonEMRContainersServiceRolePolicy。 | 下列許可會新增至政策：`eks:ListPodIdentityAssociations` 、`eks:DescribePodIdentityAssociation` 。 | 2023 年 2 月 3 日 |
| `AmazonEMRContainersServiceRolePolicy` – 新增了用於描述和列出 Amazon EKS 節點群組、描述負載平衡器目標群組以及 | 下列許可已新增至政策：`eks:ListNodeGroups` 、`eks:DescribeNodeGroup` 、`elasticloadbalancing:DescribeTargetGroups` 、`elasticloadbalancing:DescribeTargetHealth` 。 | 2023 年 3 月 13 日 |

| 變更 | 描述 | 日期 |
| --- | --- | --- |
| 描述負載平衡器目標運作狀態的許可。 | | |
| `AmazonEMRContainersServiceRolePolicy` - 新增了匯入和刪除憑證的許可 AWS Certificate Manager。 | 下列許可已新增至政策：`acm:ImportCertificate` 、`acm:AddTagsToCertificate` 、`acm:DeleteCertificate` 。 | 2021 年 12 月 3 日 |
| Amazon EMR on EKS 已開始追蹤變更 | Amazon EMR on EKS 開始追蹤其 AWS 受管政策的變更。 | 2021 年 3 月 1 日 |

## 搭配使用作業執行角色與 Amazon EMR on EKS

若要使用 `StartJobRun` 命令提交在 EKS 叢集上執行的作業，必須加入作業執行角色，才能與虛擬叢集搭配使用。如需詳細資訊，請參閱 設定 Amazon EMR on EKS 中的 建立作業執行角色。也可以按照 Amazon EMR on EKS Workshop 的建立 IAM 角色以執行作業一節中的指示進行操作。

作業執行角色的信任政策中必須包含下列許可。

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Federated": "arn:aws:iam::111122223333:oidc-
provider/OIDC_PROVIDER"
            },
            "Action": "sts:AssumeRoleWithWebIdentity",
            "Condition": {
                "StringLike": {
                    "OIDC_PROVIDER:sub": "system:serviceaccount:NAMESPACE:emr-
containers-sa-*-*-AWS_ACCOUNT_ID-BASE36_ENCODED_ROLE_NAME"
                }
            }
```

```
        }
    ]
}
```

上述範例中的信任政策只會向具有與 `emr-containers-sa-*-*-AWS_ACCOUNT_ID-BASE36_ENCODED_ROLE_NAME` 模式相符名稱的 Amazon EMR 受管 Kubernetes 服務帳戶授予許可。將在作業提交時自動建立具有此模式的服務帳戶，並將範圍設定為您提交作業所在的命名空間。此信任政策可讓這些服務帳戶擔任執行角色，並取得執行角色的暫時憑證。來自不同 Amazon EKS 叢集或相同 EKS 叢集內不同命名空間的服務帳戶受到限制，無法擔任執行角色。

可以執行下列命令，以上述格式自動更新信任政策。

```
aws emr-containers update-role-trust-policy \
       --cluster-name cluster \
       --namespace namespace \
       --role-name iam_role_name_for_job_execution
```

控制對執行角色的存取

Amazon EKS 叢集的管理員可以建立多租用戶 Amazon EMR on EKS 虛擬叢集，IAM 管理員可以為其新增多個執行角色。由於不受信任的租用戶可以使用這些執行角色來提交執行任意程式碼的作業，因此您可能想要限制這些租用戶，使其無法執行取得已指派給這些執行角色之許可的程式碼。若要限制附接至 IAM 身分的 IAM 政策，IAM 管理員可以使用選用的 Amazon Resource Name (ARN) 條件索引鍵 `emr-containers:ExecutionRoleArn`。此條件會接受具有虛擬叢集許可的執行角色 ARN 清單，如下列許可政策所示。

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "emr-containers:StartJobRun",
            "Resource": "arn:aws:emr-containers:us-east-1:AWS_ACCOUNT_ID:/
virtualclusters/VIRTUAL_CLUSTER_ID",
            "Condition": {
                "ArnEquals": {
```

```
                    "emr-containers:ExecutionRoleArn": [
                        "execution_role_arn_1",
                        "execution_role_arn_2"
                    ]
                }
            }
        }
    ]
}
```

如果想要允許所有以特定字首開頭的執行角色 (例如 `MyRole`)，可以將條件運算子 `ArnEquals` 取代為 `ArnLike` 運算子，並且可以將條件中的 `execution_role_arn` 值取代為萬用字元 `*`。例如 `arn:aws:iam::AWS_ACCOUNT_ID:role/MyRole*`。還支援所有其他 ARN 條件金鑰。

> **Note**
>
> 使用 Amazon EMR on EKS，您無法根據標籤或屬性將許可授予給執行角色。Amazon EMR on EKS 不支援執行角色的標籤型存取控制 (TBAC) 或屬性型存取控制 (ABAC)。

## Amazon EMR on EKS 身分型政策範例

根據預設，IAM 使用者和角色不具備建立或修改 Amazon EMR on EKS 資源的許可。他們也無法使用 AWS Management Console、 AWS Command Line Interface (AWS CLI) 或 AWS API 來執行任務。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。

如需了解如何使用這些範例 JSON 政策文件建立 IAM 身分型政策，請參閱《IAM 使用者指南》中的建立 IAM 政策 (主控台)。

如需 Amazon EMR on EKS 所定義之動作和資源類型的詳細資訊，包括每種資源類型的 ARN 格式，請參閱《服務授權參考》中的適用於 Amazon EMR on EKS 的動作、資源和條件金鑰。

主題


- 政策最佳實務
- 使用 Amazon EMR on EKS 主控台
- 允許使用者檢視他們自己的許可

## 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 EMR on EKS 資源。這些動作可能會讓您的 AWS 帳戶產生費用。當您建立或編輯身分型政策時，請遵循下列準則及建議事項：

- 開始使用 AWS 受管政策並邁向最低權限許可 – 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 中使用 AWS 帳戶。我們建議您定義特定於使用案例 AWS 的客戶受管政策，以進一步減少許可。如需更多資訊，請參閱 IAM 使用者指南中的 AWS 受管政策或任務職能的AWS 受管政策。
- 套用最低權限許可 – 設定 IAM 政策的許可時，請僅授予執行任務所需的許可。為實現此目的，您可以定義在特定條件下可以對特定資源採取的動作，這也稱為最低權限許可。如需使用 IAM 套用許可的更多相關資訊，請參閱 IAM 使用者指南中的 IAM 中的政策和許可。
- 使用 IAM 政策中的條件進一步限制存取權 – 您可以將條件新增至政策，以限制動作和資源的存取。例如，您可以撰寫政策條件，指定必須使用 SSL 傳送所有請求。如果透過特定 例如 使用服務動作 AWS 服務，您也可以使用條件來授予其存取權 AWS CloudFormation。如需詳細資訊，請參閱 IAM 使用者指南中的 IAM JSON 政策元素：條件。
- 使用 IAM Access Analyzer 驗證 IAM 政策，確保許可安全且可正常運作 – IAM Access Analyzer 驗證新政策和現有政策，確保這些政策遵從 IAM 政策語言 (JSON) 和 IAM 最佳實務。IAM Access Analyzer 提供 100 多項政策檢查及切實可行的建議，可協助您撰寫安全且實用的政策。如需詳細資訊，請參閱《IAM 使用者指南》中的使用 IAM Access Analyzer 驗證政策。
- 需要多重要素驗證 (MFA) – 如果您的案例需要 IAM 使用者或 中的根使用者 AWS 帳戶，請開啟 MFA 以提高安全性。如需在呼叫 API 操作時請求 MFA，請將 MFA 條件新增至您的政策。如需詳細資訊，請參閱《IAM 使用者指南》https://docs.aws.amazon.com/IAM/latest/UserGuide/id_credentials_mfa_configure-api-require.html中的透過 MFA 的安全 API 存取。

如需 IAM 中最佳實務的相關資訊，請參閱 IAM 使用者指南中的 IAM 安全最佳實務。

## 使用 Amazon EMR on EKS 主控台

若要存取 Amazon EMR on EKS 主控台，您必須擁有最基本的一組許可。這些許可必須允許您列出和檢視 AWS 帳戶中 Amazon EMR on EKS 資源的詳細資訊。如果您建立比最基本必要許可更嚴格的身分型政策，則對於具有該政策的實體 (使用者或角色) 而言，主控台就無法如預期運作。

對於僅呼叫 AWS CLI 或 AWS API 的使用者，您不需要允許最低主控台許可。反之，只需允許存取符合他們嘗試執行之 API 操作的動作就可以了。

為了確保使用者和角色仍然可以使用 Amazon EMR on EKS 主控台，也請將 Amazon EMR on EKS `ConsoleAccess`或`ReadOnly` AWS 受管政策連接到實體。如需詳細資訊，請參閱《IAM 使用者指南》中的新增許可到使用者。

## 允許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 使用者檢視附加到他們使用者身分的內嵌及受管政策。此政策包含在主控台或使用 或 AWS CLI AWS API 以程式設計方式完成此動作的許可。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
                "iam:GetUser"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
        {
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
            "Resource": "*"
        }
    ]
}
```

## 標籤型存取控制政策

可以使用身分型政策中的條件，根據標籤來控制對虛擬叢集和作業執行的存取。如需標記的相關資訊，請參閱為您的 Amazon EMR on EKS 資源加上標籤。

以下範例示範了搭配使用條件運算子與 Amazon EMR on EKS 條件金鑰的不同情況和方式。這些 IAM 政策陳述式僅作示範用途，不應用於生產環境。有多種方法可以結合政策陳述式，以根據您的需求授予和拒絕許可。如需規劃與測試 IAM 政策的詳細資訊，請參閱 IAM 使用者指南。

> **Important**
>
> 標記動作的明確拒絕許可是項重要的考量條件。這可防止使用者標記資源並將您無意授予的許可授予給他們。如果未拒絕資源的標記動作，使用者可以修改標籤並規避標籤型政策的意圖。如需有關可拒絕標記動作的政策範例，請參閱 拒絕新增和移除標籤的存取權。

以下範例示範的身分型許可政策用來控制允許 Amazon EMR on EKS 虛擬叢集採取的動作。

### 僅在具有特定標籤值的資源上允許動作

在以下政策範例中，StringEquals 條件運算子嘗試將 dev 與標籤部門的值進行比對。若標籤部門尚未新增到虛擬叢集，或不包含 dev 值，政策將無法套用，此政策也不允許動作。如果沒有其他政策陳述式允許動作，使用者只能使用具有此值標籤的虛擬叢集。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "emr-containers:DescribeVirtualCluster"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/department": "dev"
        }
      }
```

```
    }
  ]
}
```

您也可以使用條件運算子來指定多個標籤值。例如，若要在 department 標籤包含 dev 或 test 值的虛擬叢集上允許動作，可以用下列內容取代先前範例中的條件區塊。

```
"Condition": {
        "StringEquals": {
          "aws:ResourceTag/department": ["dev", "test"]
        }
      }
```

## 建立資源時需要進行標記

在以下範例中，需要在建立虛擬叢集時套用標籤。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "emr-containers:CreateVirtualCluster"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "aws:RequestTag/department": "dev"
        }
      }
    }
  ]
}
```

以下政策陳述式可讓使用者只在叢集具有可以包含任何值的 department 標籤時，才能建立虛擬叢集。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "emr-containers:CreateVirtualCluster"
      ],
      "Resource": "*",
      "Condition": {
        "Null": {
          "aws:RequestTag/department": "false"
        }
      }
    }
  ]
}
```

## 拒絕新增和移除標籤的存取權

此政策的效果是拒絕使用者在以包含 `department` 值的 `dev` 標籤所標記的虛擬叢集上新增或移除任何標籤的許可。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": [
        "emr-containers:TagResource",
        "emr-containers:UntagResource"
      ],
      "Resource": "*",
      "Condition": {
        "StringNotEquals": {
          "aws:ResourceTag/department": "dev"
```

```
            }
          }
        }
      ]
    }
```

## Amazon EMR on EKS 身分識別和存取疑難排解

請使用以下資訊來協助您診斷和修正使用 Amazon EMR on EKS 和 IAM 時可能遇到的常見問題。

主題

- 我未獲授權，不得在 Amazon EMR on EKS 中執行動作
- 我未獲得執行 iam:PassRole 的授權
- 我想要允許 AWS 帳戶外的人員存取我的 Amazon EMR on EKS 資源

### 我未獲授權，不得在 Amazon EMR on EKS 中執行動作

如果 AWS Management Console 告訴您無權執行 動作，則必須聯絡您的管理員尋求協助。您的管理員是提供您使用者名稱和密碼的人員。

下列範例錯誤會在 `mateojackson` 使用者嘗試使用主控台檢視一個虛構 *`my-example-widget`* 資源的詳細資訊，但卻無虛構 `emr-containers:`*`GetWidget`* 許可時發生。

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform: emr-
containers:GetWidget on resource: my-example-widget
```

在此情況下，Mateo 會請求管理員更新他的政策，允許他使用 *`my-example-widget`* 動作存取 `emr-containers:`*`GetWidget`* 資源。

### 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，允許您將角色傳遞給 Amazon EMR on 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 EMR on 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，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 EMR on 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。

如需進一步了解，請參閱以下內容：

- 若要了解 Amazon EMR on EKS 是否支援這些功能，請參閱 Amazon EMR on EKS 如何搭配 IAM 運作。
- 若要了解如何在您擁有 AWS 帳戶 的 資源之間提供存取權，請參閱《IAM 使用者指南》中的在您擁有 AWS 帳戶 的另一個 中為 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 AWS 帳戶 擁有。
- 如需了解如何透過聯合身分提供存取權，請參閱 IAM 使用者指南中的將存取權提供給在外部進行身分驗證的使用者 (聯合身分)。
- 如需了解使用角色和資源型政策進行跨帳戶存取之間的差異，請參閱《IAM 使用者指南》中的 IAM 中的跨帳戶資源存取。

## 使用 Amazon EMR on EKS 搭配 AWS Lake Formation 進行精細存取控制

透過 Amazon EMR 7.7 版和更新版本，您可以利用 AWS Lake Formation 對 Amazon S3 儲存貯體支援的 AWS Glue Data Catalog 資料表套用精細存取控制。此功能可讓您設定 Amazon EMR on EKS Spark 任務中讀取查詢的資料表、資料列、資料欄和儲存格層級存取控制。

主題

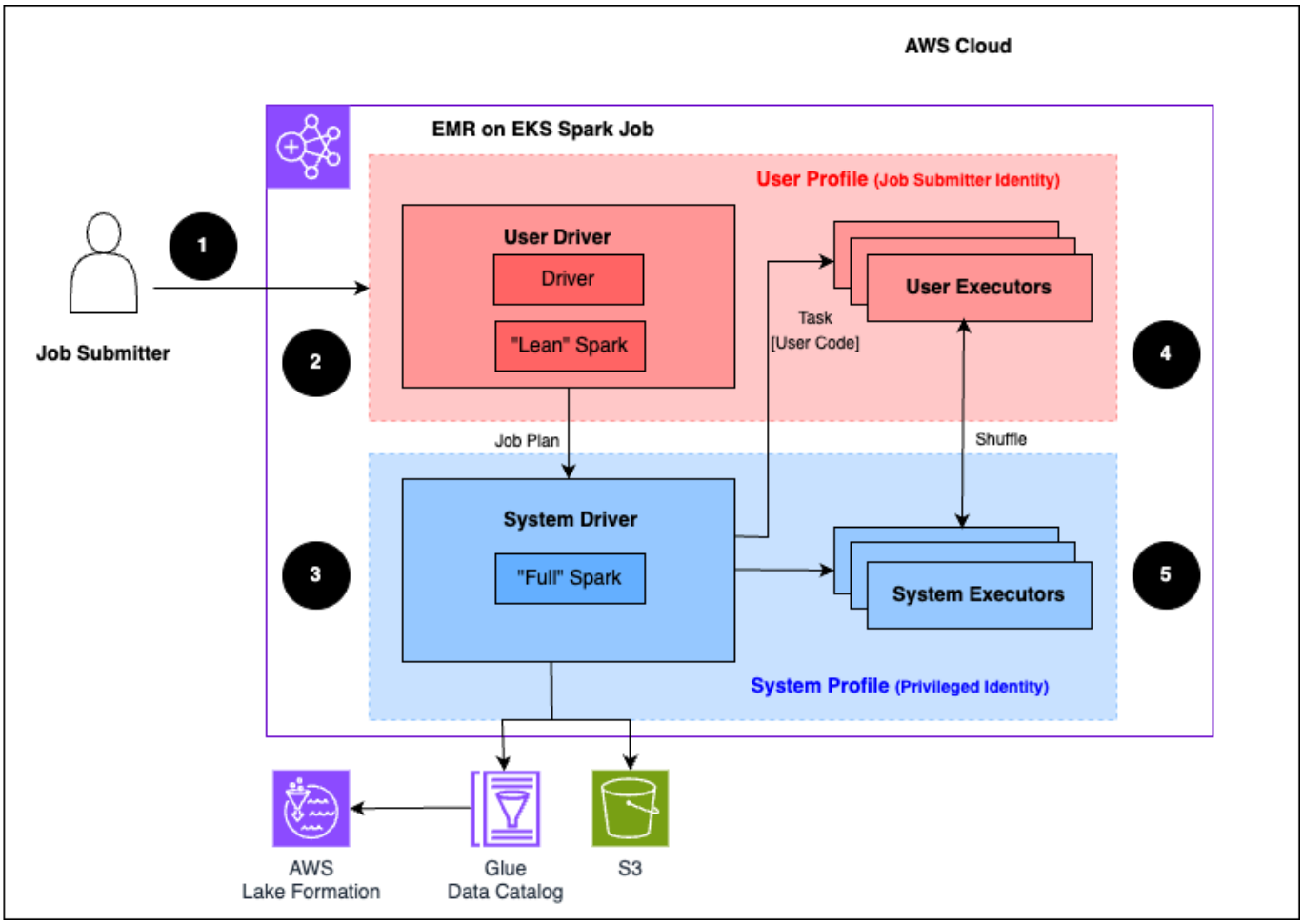

- Amazon EMR on EKS 如何與 AWS Lake Formation 搭配使用
- 使用 Amazon EMR on EKS 啟用 Lake Formation

- 考量與限制
- 疑難排解

## Amazon EMR on EKS 如何與 AWS Lake Formation 搭配使用

使用 Amazon EMR on EKS 搭配 Lake Formation 可讓您在每個 Spark 任務上強制執行一層許可，以在 Amazon EMR on EKS 執行任務時套用 Lake Formation 許可控制。Amazon EMR on EKS 使用 Spark 資源描述檔來建立兩個描述檔，以有效地執行任務。使用者設定檔會執行使用者提供的程式碼，而系統設定檔則會強制執行 Lake Formation 政策。每個已啟用 Lake Formation 的任務都會使用兩個 Spark 驅動程式，一個用於使用者設定檔，另一個用於系統設定檔。如需詳細資訊，請參閱什麼是 AWS Lake Formation。

以下是 Amazon EMR on EKS 如何存取 Lake Formation 安全政策所保護資料的高階概觀。




下列步驟說明此程序：

1. 使用者將 Spark 任務提交至已啟用 AWS Lake Formation 的 Amazon EMR on EKS 虛擬叢集。

2. Amazon EMR on EKS 服務會設定使用者驅動程式，並在使用者設定檔中執行任務。使用者驅動程式執行精簡版本的 Spark，無法啟動任務、請求執行器、存取 Amazon S3 或 Glue Data Catalog。它只會建置任務計劃。
3. Amazon EMR on EKS 服務會設定第二個稱為系統驅動程式的驅動程式，並在系統設定檔（具有特殊權限身分）中執行。Amazon EKS 會在兩個驅動程式之間設定加密的 TLS 頻道以進行通訊。使用者驅動程式使用 頻道將任務計劃傳送至系統驅動程式。系統驅動程式不會執行使用者提交的程式碼。它執行完整的 Spark，並與 Amazon S3 和 Data Catalog 通訊以進行資料存取。它請求執行器並將任務計畫編譯為一系列執行階段。
4. 然後，Amazon EMR on EKS 服務會在執行器上執行階段。任何階段中的使用者程式碼只會在使用者設定檔執行器上執行。
5. 從受 Lake Formation 保護的資料目錄資料表或套用安全篩選條件的資料表讀取資料的階段，會委派給系統執行器。

# 使用 Amazon EMR on EKS 啟用 Lake Formation

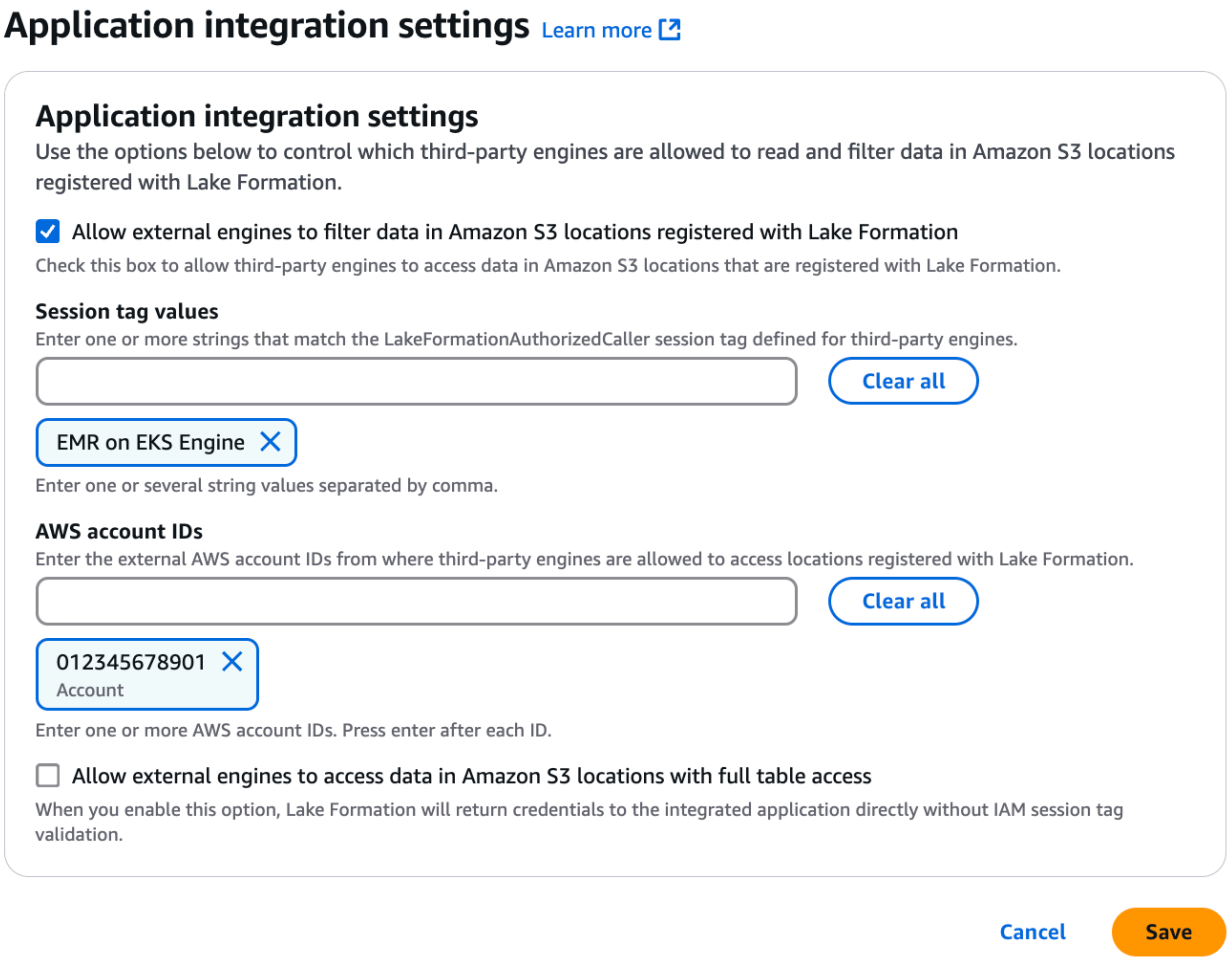
透過 Amazon EMR 7.7 版和更新版本，您可以利用 AWS Lake Formation 對 Amazon S3 支援的 Data Catalog 資料表套用精細存取控制。此功能可讓您設定 Amazon EMR on EKS Spark 任務中讀取查詢的資料表、資料列、資料欄和儲存格層級存取控制。

本節說明如何建立安全組態並設定 Lake Formation 以使用 Amazon EMR。它還說明如何使用您為 Lake Formation 建立的安全組態來建立虛擬叢集。這些區段應依序完成。

## 步驟 1：設定 Lake Formation 型資料欄、資料列或儲存格層級許可

首先，若要使用 Lake Formation 套用資料列和資料欄層級許可，Lake Formation 的資料湖管理員必須設定 LakeFormationAuthorizedCaller 工作階段標籤。Lake Formation 使用此工作階段標籤來授權呼叫者並提供對資料湖的存取權。

導覽至 AWS Lake Formation 主控台，然後從側邊列的管理區段中選取應用程式整合設定選項。然後，勾選允許外部引擎篩選向 Lake Formation 註冊的 Amazon S3 位置中的資料。新增執行 Spark 任務AWS 的帳戶 IDs，以及工作階段標籤值。




請注意，稍後當您設定 IAM 角色時，此處傳遞的 LakeFormationAuthorizedCaller 工作階段標籤會在 SecurityConfiguration 中傳遞，如第 3 節所述。

## 步驟 2：設定 EKS RBAC 許可

其次，您可以設定角色型存取控制的許可。

將 EKS 叢集許可提供給 Amazon EMR on EKS 服務

Amazon EMR on EKS Service 必須具有 EKS 叢集角色許可，才能建立跨命名空間許可，讓系統驅動程式在使用者命名空間中分割使用者執行器。

建立叢集角色

此範例定義資源集合的許可。

```
vim emr-containers-cluster-role.yaml
---
```

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: emr-containers
rules:
  - apiGroups: [""]
    resources: ["namespaces"]
    verbs: ["get"]
  - apiGroups: [""]
    resources: ["serviceaccounts", "services", "configmaps", "events", "pods", "pods/
log"]
    verbs: ["get", "list", "watch", "describe", "create", "edit", "delete",
 "deletecollection", "annotate", "patch", "label"]
  - apiGroups: [""]
    resources: ["secrets"]
    verbs: ["create", "patch", "delete", "watch"]
  - apiGroups: ["apps"]
    resources: ["statefulsets", "deployments"]
    verbs: ["get", "list", "watch", "describe", "create", "edit", "delete", "annotate",
 "patch", "label"]
  - apiGroups: ["batch"]
    resources: ["jobs"]
    verbs: ["get", "list", "watch", "describe", "create", "edit", "delete", "annotate",
 "patch", "label"]
  - apiGroups: ["extensions", "networking.k8s.io"]
    resources: ["ingresses"]
    verbs: ["get", "list", "watch", "describe", "create", "edit", "delete", "annotate",
 "patch", "label"]
  - apiGroups: ["rbac.authorization.k8s.io"]
    resources: ["clusterroles","clusterrolebindings","roles", "rolebindings"]
    verbs: ["get", "list", "watch", "describe", "create", "edit", "delete",
 "deletecollection", "annotate", "patch", "label"]
  - apiGroups: [""]
    resources: ["persistentvolumeclaims"]
    verbs: ["get", "list", "watch", "describe", "create", "edit", "delete",
 "deletecollection", "annotate", "patch", "label"]
---
```

```
kubectl apply -f emr-containers-cluster-role.yaml
```

## 建立叢集角色繫結

```
vim emr-containers-cluster-role-binding.yaml
```

```
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: emr-containers
subjects:
- kind: User
  name: emr-containers
  apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: ClusterRole
  name: emr-containers
  apiGroup: rbac.authorization.k8s.io
---
```

```
kubectl apply -f emr-containers-cluster-role-binding.yaml
```

提供 Amazon EMR on EKS 服務的命名空間存取權

建立兩個 Kubernetes 命名空間，一個用於使用者驅動程式和執行器，另一個用於系統驅動程式和執行器，並啟用 Amazon EMR on EKS 服務存取，以在使用者和系統命名空間中提交任務。請依照現有指南提供每個命名空間的存取權，該存取權可在使用 啟用叢集存取aws-auth中取得。

## 步驟 3：設定使用者和系統設定檔元件的 IAM 角色

第三，您可以為特定元件設定角色。啟用 Lake Formation 的 Spark 任務有兩個元件：使用者和系統。使用者驅動程式和執行器會在使用者命名空間中執行，並與 StartJobRun API 中傳遞的 JobExecutionRole 繫結。系統驅動程式和執行器會在系統命名空間中執行，並與 QueryEngine 角色繫結。

設定查詢引擎角色

QueryEngine 角色與系統空間元件繫結，並具有使用 LakeFormationAuthorizedCaller 工作階段標籤擔任 JobExecutionRole 的許可。查詢引擎角色的 IAM 許可政策如下：

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
```

```
        {
            "Sid": "AssumeJobRoleWithSessionTagAccessForSystemDriver",
            "Effect": "Allow",
            "Action": [
                "sts:AssumeRole",
                "sts:TagSession"
            ],
            "Resource": "arn:aws:iam::111122223333:role/JobExecutionRole",
            "Condition": {
                "StringLike": {
                    "aws:RequestTag/LakeFormationAuthorizedCaller": "EMR on EKS
 Engine"
                }
            }
        },
        {
            "Sid": "AssumeJobRoleWithSessionTagAccessForSystemExecutor",
            "Effect": "Allow",
            "Action": [
                "sts:AssumeRole"
            ],
            "Resource": "arn:aws:iam::111122223333:role/JobExecutionRole"
        },
        {
            "Sid": "CreateCertificateAccessForTLS",
            "Effect": "Allow",
            "Action": "emr-containers:CreateCertificate",
            "Resource": "*"
        }
    ]
}
```

設定查詢引擎角色的信任政策，以信任 Kubernetes 系統命名空間。

```
aws emr-containers update-role-trust-policy \
    --cluster-name eks cluster \
    --namespace eks system namespace \
    --role-name query_engine_iam_role_name
```

如需詳細資訊，請參閱更新角色信任政策。

設定任務執行角色

Lake Formation 許可控制對 AWS Glue Data Catalog 資源、Amazon S3 位置和這些位置基礎資料的存取。IAM 許可控制對 Lake Formation 和 AWS Glue APIs存取。雖然您可能擁有 Lake Formation 許可來存取 Data Catalog (SELECT) 中的資料表，但如果您沒有 `glue:Get*` API 操作的 IAM 許可，您的操作會失敗。

JobExecutionRole 的 IAM 許可政策：JobExecution 角色在其許可政策中應具有政策陳述式。

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "GlueCatalogAccess",
            "Effect": "Allow",
            "Action": [
                "glue:Get*",
                "glue:Create*",
                "glue:Update*"
            ],
            "Resource": ["*"]
        },
        {
            "Sid": "LakeFormationAccess",
            "Effect": "Allow",
            "Action": [
                "lakeformation:GetDataAccess"
            ],
            "Resource": ["*"]
        },
        {
            "Sid": "CreateCertificateAccessForTLS",
            "Effect": "Allow",
            "Action": "emr-containers:CreateCertificate",
            "Resource": "*"
        }
    ]
}
```

JobExecutionRole 的 IAM 信任政策：

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "TrustQueryEngineRoleForSystemDriver",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:role/QueryExecutionRole"
            },
            "Action": [
                "sts:AssumeRole",
                "sts:TagSession"
            ],
            "Condition": {
                "StringLike": {
                    "aws:RequestTag/LakeFormationAuthorizedCaller": "EMR on EKS
 Engine"
                }
            }
        },
        {
            "Sid": "TrustQueryEngineRoleForSystemExecutor",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:role/QueryEngineRole"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

設定任務執行角色的信任政策，以信任 Kubernetes 使用者命名空間：

```
aws emr-containers update-role-trust-policy \
    --cluster-name eks cluster \
    --namespace eks User namespace \
    --role-name job_execution_role_name
```

如需詳細資訊，請參閱更新任務執行角色的信任政策。

## 步驟 4：設定安全組態

若要執行已啟用 Lake Formation 的任務，您必須建立安全組態。

```
aws emr-containers create-security-configuration \
    --name 'security-configuration-name' \
    --security-configuration '{
        "authorizationConfiguration": {
            "lakeFormationConfiguration": {
                "authorizedSessionTagValue": "SessionTag configured in LakeFormation",
                "secureNamespaceInfo": {
                    "clusterId": "eks-cluster-name",
                    "namespace": "system-namespace-name"
                },
                "queryEngineRoleArn": "query-engine-IAM-role-ARN"
            }
        }
    }'
```

確保在 authorizedSessionTagValue 欄位中傳遞的工作階段標籤可以授權 Lake Formation。將 值設定為 中 Lake Formation 設定的值步驟 1：設定 Lake Formation 型資料欄、資料列或儲存格層級許可。

## 步驟 5：建立虛擬叢集

使用安全組態建立 Amazon EMR on EKS 虛擬叢集。

```
aws emr-containers create-virtual-cluster \
--name my-lf-enabled-vc \
--container-provider '{
    "id": "eks-cluster",
    "type": "EKS",
    "info": {
        "eksInfo": {
            "namespace": "user-namespace"
        }
    }
}' \
--security-configuration-id SecurityConfiguraionId
```

確保傳遞上一個步驟的 SecurityConfiguration ID，以便將 Lake Formation 授權組態套用至虛擬叢集上執行的所有任務。如需詳細資訊，請參閱向 Amazon EMR 註冊 Amazon EKS 叢集。

### 步驟 6：在 FGAC 啟用的 VirtualCluster 中提交任務

非 Lake Formation 和 Lake Formation 任務的任務提交程序都相同。如需詳細資訊，請參閱使用 提交任務執行`StartJobRun`。

系統驅動程式的 Spark 驅動程式、執行器和事件日誌存放在 AWS 服務帳戶的 S3 儲存貯體中以進行偵錯。我們建議在任務執行中設定客戶受管 KMS 金鑰，以加密存放在 AWS 服務儲存貯體中的所有日誌。如需啟用日誌加密的詳細資訊，請參閱加密 Amazon EMR on EKS 日誌。

## 考量與限制

將 Lake Formation 與 Amazon EMR on EKS 搭配使用時，請注意下列考量和限制：

- Amazon EMR on EKS 僅支援透過 Lake Formation 的精細存取控制，適用於 Apache Hive、Apache Iceberg、Apache Hudi 和 Delta 資料表格式。Apache Hive 格式包括 Parquet、ORC 和 xSV。
- `DynamicResourceAllocation` 預設為啟用，您無法關閉 Lake Formation `DynamicResourceAllocation` 任務。由於 DRA `spark.dynamicAllocation.maxExecutors`組態的預設值為無限，請根據您的工作負載設定適當的值。
- 啟用 Lake Formation 的任務不支援在系統驅動程式和系統執行器中使用自訂的 EMR on EKS 映像。
- 您只能將 Lake Formation 與 Spark 任務搭配使用。
- 使用 Lake Formation 的 EMR on EKS 僅支援整個任務的單一 Spark 工作階段。
- 使用 Lake Formation 的 EMR on EKS 僅支援透過資源連結共用的跨帳戶資料表查詢。
- 不支援下列項目：
  - 彈性分散式資料集 (RDD)
  - Spark 串流
  - 使用 Lake Formation 授予的許可寫入
  - 巢狀資料欄的存取控制
- EMR on EKS 會封鎖可能會破壞系統驅動程式完整隔離的功能，包括下列項目：
  - UDTs、HiveUDFs和任何涉及自訂類別的使用者定義函數
  - 自訂資料來源
  - 為 Spark 延伸模組、連接器或中繼存放區`ANALYZE TABLE`命令提供額外的 jar
- 為了強制執行存取控制，`EXPLAIN PLAN`以及 DDL 操作，例如 `DESCRIBE TABLE` 不會公開限制資訊。

- Amazon EMR on EKS 限制對已啟用 Lake Formation 任務之系統驅動程式 Spark 日誌的存取。由於系統驅動程式執行時具有更多存取權，因此系統驅動程式產生的事件和日誌可能包含敏感資訊。為了防止未經授權的使用者或程式碼存取此敏感資料，EMR on EKS 已停用對系統驅動程式日誌的存取。如需故障診斷，請聯絡 AWS 支援。
- 如果您已向 Lake Formation 註冊資料表位置，無論 EMR on EKS 任務執行角色的 IAM 許可為何，資料存取路徑都會經過 Lake Formation 儲存的登入資料。如果您錯誤設定已向資料表位置註冊的角色，提交的任務若使用具有資料表位置的 S3 IAM 許可的角色，將會失敗。
- 寫入 Lake Formation 資料表會使用 IAM 許可，而不是 Lake Formation 授予的許可。如果您的任務執行角色具有必要的 S3 許可，您可以使用它來執行寫入操作。

以下是使用 Apache Iceberg 時的考量和限制：

- 您只能搭配工作階段目錄使用 Apache Iceberg，不能任意命名目錄。
- 在 Lake Formation 中註冊的 Iceberg 資料表僅支援中繼資料表 `history`、`metadata_log_entries`、`snapshots`、`files`、 `manifests`和 `refs`。Amazon EMR 會隱藏可能具有敏感資料的資料欄，例如 `partitions`、 `path`和 `summaries`。此限制不適用於未在 Lake Formation 中註冊的 Iceberg 資料表。
- 未在 Lake Formation 中註冊的資料表支援所有 Iceberg 預存程序。任何資料表都不支援 `register_table`和 `migrate` 程序。
- 我們建議您使用 Iceberg DataFrameWriterV2 而非 V1。

如需詳細資訊，請參閱了解 Amazon EMR on EKS 概念和術語，以及啟用 Amazon EMR on EKS 的叢集存取。

## 資料管理員的免責聲明

> **Note**
>
> 當您將 Lake Formation 資源的存取權授予 EMR on EKS 的 IAM 角色時，您必須確保 EMR 叢集管理員或運算子是信任的管理員。這與跨多個組織和 AWS 帳戶共用的 Lake Formation 資源特別相關。

### EKS 管理員的責任

- System 命名空間應受到保護。不允許使用者、資源或實體或工具對System命名空間中的 Kubernetes 資源擁有任何 Kubernetes RBAC 許可。
- 除了 EMR on EKS 服務之外，任何使用者、資源或實體都無權CREATE存取 User 命名空間中的 POD、CONFIG_MAP 和 SECRET。
- System 驅動程式和System執行程式包含敏感資料。因此，命名System空間中的 Spark 事件、Spark 驅動程式日誌和 Spark 執行器日誌不應轉送至外部日誌儲存系統。

## 疑難排解

### 日誌

EMR on EKS 使用 Spark 資源設定檔來分割任務執行。Amazon EMR on EKS 使用使用者設定檔來執行您提供的程式碼，而系統設定檔則會強制執行 Lake Formation 政策。您可以使用 MonitoringConfiguration 設定 StartJobRun 請求，存取以使用者設定檔身分執行的容器日誌。

### Spark 歷史記錄伺服器

Spark 歷史記錄伺服器具有從使用者設定檔產生的所有 Spark 事件，以及從系統驅動程式產生的修訂事件。您可以在執行器索引標籤中查看來自使用者和系統驅動程式的所有容器。不過，日誌連結僅適用於使用者設定檔。

### Lake Formation 許可不足的任務失敗

請確定您的任務執行角色具有在您存取的資料表DESCRIBE上執行 SELECT和 的許可。

### RDD 執行的任務失敗

EMR on EKS 目前不支援已啟用 Lake Formation 任務的彈性分散式資料集 (RDD) 操作。

### 無法存取 Amazon S3 中的資料檔案

請確定您已在 Lake Formation 中註冊資料湖的位置。

### 安全驗證例外狀況

EMR on EKS 偵測到安全驗證錯誤。如需協助，請聯絡 AWS 支援。

### 跨帳戶共用 AWS Glue Data Catalog 和資料表

您可以跨帳戶共用資料庫和資料表，但仍使用 Lake Formation。如需詳細資訊，請參閱 Lake Formation 中的跨帳戶資料共用和如何使用 AWS Lake Formation 跨帳戶共用 AWS Glue Data Catalog 和資料表？。

### 未設定 AWS 區域的 Iceberg 任務擲回初始化錯誤

訊息如下：

```
25/02/25 13:33:19 ERROR SparkFGACExceptionSanitizer: Client received error with id =
 b921f9e6-f655-491f-b8bd-b2842cdc20c7,
reason = IllegalArgumentException, message = Cannot initialize
LakeFormationAwsClientFactory, please set client.region to a valid aws region
```

確定 Spark 組態`spark.sql.catalog.`*`catalog_name`*`.client.region`設定為有效的區域。

### Iceberg 任務擲回 SparkUnsupportedOperationException

訊息如下：

```
25/02/25 13:53:15 ERROR SparkFGACExceptionSanitizer: Client received error with id =
 921fef42-0800-448b-bef5-d283d1278ce0,
reason = SparkUnsupportedOperationException, message = Either glue.id or glue.account-
id is set with non-default account.
Cross account access with fine-grained access control is only supported with AWS
 Resource Access Manager.
```

確定 Spark 組態`spark.sql.catalog.`*`catalog_name`*`.glue.account-id`設定為有效的帳戶 ID。

## 日誌記錄和監控

若要偵測事件，在事件發生時收到提醒，以及回應它們，請搭配 Amazon EMR on EKS 使用這些選項：

- 使用 AWS CloudTrail - 監控 Amazon EMR on EKS AWS CloudTrail提供 Amazon EMR on EKS 中使用者、角色 AWS 或服務所採取動作的記錄。它會從 Amazon EMR 主控台擷取呼叫，並將 Amazon EMR on EKS API 操作的呼叫編碼為事件。這可讓您判斷對 Amazon EMR on EKS 提出的請求、提出請求的 IP 地址、提出請求的對象、提出請求的時間，以及其他詳細資訊。如需詳細資訊，請參閱使用 記錄 Amazon EMR on EKS API 呼叫 AWS CloudTrail。

- 搭配使用 CloudWatch Events 與 Amazon EMR on EKS - CloudWatch Events 可提供近乎即時的系統事件串流，說明 AWS 資源的變更。CloudWatch Events 會察覺這些操作變更的發生，回應它們並視需要採取更正動作，方法為傳送訊息來回應環境、啟用功能、執行變更和擷取狀態資訊。若要搭配 Amazon EMR on EKS 使用 CloudWatch Events，請透過 CloudTrail 建立 Amazon EMR on EKS API 呼叫觸發的規則。如需詳細資訊，請參閱使用 Amazon CloudWatch Events 監控作業。

## 使用受管儲存體加密 Amazon EMR on EKS 日誌

以下各節說明如何設定日誌的加密。

### Enable encryption (啟用加密)

若要使用您自己的 KMS 金鑰加密受管儲存中的日誌，請在提交任務執行時使用下列組態。

```
"monitoringConfiguration": {
            "managedLogs": {
                "allowAWSToRetainLogs":"ENABLED",
                "encryptionKeyArn":"KMS key arn"
            },
            "persistentAppUI": "ENABLED"
        }
```

`allowAWSToRetainLogs` 組態允許 AWS 在使用原生 FGAC 執行任務時保留系統命名空間日誌。`persistentAppUI` 組態允許 AWS 儲存用於產生 Spark UI 的事件日誌。`encryptionKeyArn` 用於指定您要用來加密所存放日誌的 KMS 金鑰 ARN AWS。

### 日誌加密的必要許可

提交任務或檢視 Spark UI 的使用者必須允許`kms:Decrypt`加密金鑰的動作 `kms:DescribeKey`、`kms:GenerateDataKey`和 。這些許可用於驗證金鑰的有效性，並檢查使用者是否具有讀取和寫入使用 KMS 金鑰加密之日誌的必要許可。如果提交任務的使用者缺少必要的金鑰許可，Amazon EMR on EKS 會拒絕任務執行提交。

用於呼叫 StartJobRun 的角色的 IAM 政策範例

JSON

```
{
    "Version": "2012-10-17",
```

```
    "Statement": [
        {
            "Action": "emr-containers:StartJobRun",
            "Resource": "*",
            "Effect": "Allow"
        },
        {
            "Action": [
                "kms:DescribeKey",
                "kms:Decrypt",
                "kms:GenerateDataKey"
            ],
            "Resource": "KMS key ARN",
            "Effect": "Allow"
        }
    ]
}
```

您也必須設定 KMS 金鑰，以允許 `persistentappui.elasticmapreduce.amazonaws.com`和 `elasticmapreduce.amazonaws.com` 服務主體使用 `kms:GenerateDataKey`和 `kms:Decrypt`。這可讓 EMR 將 KMS 金鑰加密的日誌讀取和寫入受管儲存。

範例 KMS 金鑰政策

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": "IAM role ARN used to call StartJobRun"
            },
            "Action": "kms:DescribeKey",
            "Resource": "*",
            "Condition": {
                "StringLike": {
                    "kms:viaService": "emr-containers.us-east-1.amazonaws.com"
                }
            }
```

```
        },
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": "IAM role ARN used to call StartJobRun"
            },
            "Action": [
                "kms:Decrypt",
                "kms:GenerateDataKey"
            ],
            "Resource": "*",
            "Condition": {
                "StringLike": {
                    "kms:viaService": "emr-containers.us-east-1.amazonaws.com",
                    "kms:EncryptionContext:aws:emr-containers:virtualClusterId":
 "virtual cluster id"
                }
            }
        },
        {
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "persistentappui.elasticmapreduce.amazonaws.com",
                    "elasticmapreduce.amazonaws.com"
                ]
            },
            "Action": [
                "kms:Decrypt",
                "kms:GenerateDataKey"
            ],
            "Resource": "*",
            "Condition": {
                "StringLike": {
                    "kms:EncryptionContext:aws:emr-containers:virtualClusterId":
 "virtual cluster id",
                    "aws:SourceArn": "virtual cluster ARN"
                }
            }
        }
    ]
}
```

作為安全最佳實務，我們建議您新增 `kms:viaService`、 `kms:EncryptionContext`和 `aws:SourceArn`條件。這些條件有助於確保金鑰僅供 Amazon EMR on EKS 使用，並僅用於從特定虛擬叢集中執行的任務所產生的日誌。

## 使用 記錄 Amazon EMR on EKS API 呼叫 AWS CloudTrail

Amazon EMR on EKS 已與 服務整合 AWS CloudTrail，此服務提供使用者、角色或 AWS 服務在 Amazon EMR on EKS 中採取之動作的記錄。CloudTrail 會將 Amazon EMR on EKS 的所有 API 呼叫擷取為事件。擷取的呼叫包括來自 Amazon EMR on EKS 主控台的呼叫，以及對 Amazon EMR on EKS API 操作發出的程式碼呼叫。如果您建立追蹤，就可以將 CloudTrail 事件持續交付到 Amazon S3 儲存貯體，包括 Amazon EMR on EKS 的事件。即使您未設定追蹤，依然可以透過 CloudTrail 主控台中的事件歷史記錄檢視最新事件。您可以利用 CloudTrail 所收集的資訊來判斷向 Amazon EMR on EKS 發出的請求，以及發出請求的 IP 地址、人員、時間和其他詳細資訊。

若要進一步了解 CloudTrail，請參閱「AWS CloudTrail 使用者指南」。

### CloudTrail 中的 Amazon EMR on EKS 資訊

當您建立 AWS 帳戶時，會在您的帳戶上啟用 CloudTrail。當活動在 Amazon EMR on EKS 中發生時，該活動會與事件歷史記錄中的其他 AWS 服務事件一起記錄在 CloudTrail 事件中。您可以在 AWS 帳戶中檢視、搜尋和下載最近的事件。如需詳細資訊，請參閱「使用 CloudTrail 事件歷史記錄檢視事件」。

若要持續記錄您 AWS 帳戶中的事件，包括 Amazon EMR on 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線索時，線索會套用至所有 AWS 區域。線索會記錄 AWS 分割區中所有區域的事件，並將日誌檔案傳送到您指定的 Amazon S3 儲存貯體。此外，您可以設定其他 AWS 服務，以進一步分析和處理 CloudTrail 日誌中所收集的事件資料。如需詳細資訊，請參閱下列內容：

- 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 EMR on EKS 動作，並記錄在 Amazon EMR on EKS API 文件中。例如，對 `CreateVirtualCluster`、`StartJobRun` 和 `ListJobRuns` 動作發出的呼叫會在 CloudTrail 記錄檔案中產生專案。

每一筆事件或日誌專案都會包含產生請求者的資訊。身分資訊可協助您判斷下列事項：

- 是否使用根或 AWS Identity and Access Management (IAM) 使用者登入資料提出請求。
- 提出該請求時，是否使用了特定角色或聯合身分使用者的暫時安全憑證。
- 請求是否由其他 AWS 服務提出。

如需詳細資訊，請參閱 CloudTrail userIdentity 元素。

## 了解 Amazon EMR on EKS 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。一個事件為任何來源提出的單一請求，並包含請求動作、請求的日期和時間、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 `ListJobRuns` 動作的 CloudTrail 日誌項目。

```
{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "AIDACKCEVSQ6C2EXAMPLE:admin",
    "arn": "arn:aws:sts::012345678910:assumed-role/Admin/admin",
    "accountId": "012345678910",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
    "sessionContext": {
      "sessionIssuer": {
        "type": "Role",
        "principalId": "AIDACKCEVSQ6C2EXAMPLE",
        "arn": "arn:aws:iam::012345678910:role/Admin",
        "accountId": "012345678910",
        "userName": "Admin"
      },
      "webIdFederationData": {},
      "attributes": {
        "mfaAuthenticated": "false",
        "creationDate": "2020-11-04T21:49:36Z"
      }
    }
  },
  "eventTime": "2020-11-04T21:52:58Z",
```

```
  "eventSource": "emr-containers.amazonaws.com",
  "eventName": "ListJobRuns",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "203.0.113.1",
  "userAgent": "aws-cli/1.11.167 Python/2.7.10 Darwin/16.7.0 botocore/1.7.25",
  "requestParameters": {
    "virtualClusterId": "1K48XXXXXXHCB"
  },
  "responseElements": null,
  "requestID": "890b8639-e51f-11e7-b038-EXAMPLE",
  "eventID": "874f89fa-70fc-4798-bc00-EXAMPLE",
  "readOnly": true,
  "eventType": "AwsApiCall",
  "recipientAccountId": "012345678910"
}
```

# 將 Amazon S3 Access Grants 與 Amazon EMR on EKS 搭配使用

## Amazon EMR on EKS 的 S3 Access Grants 概觀

在 Amazon EMR 6.15.0 及更高版本中，Amazon S3 Access Grants 提供可擴展的存取控制解決方案，您可以使用該解決方案來增強從 Amazon EMR on EKS 對 Amazon S3 資料的存取權限。如果您的 S3 資料有複雜或大型的許可組態，您可以使用 Access Grants 來擴展使用者、角色和應用程式的 S3 資料許可。

使用 S3 Access Grants 可增強對 Amazon S3 資料的存取權限，超越由執行期角色或連接至有權存取 Amazon EMR on EKS 叢集之身分的 IAM 角色所授予的許可。

如需詳細資訊，請參閱《Amazon EMR 管理指南》中的使用適用於 Amazon EMR 的 S3 Access Grants 管理存取權限以及《Amazon Simple Storage Service 使用者指南》中的使用 S3 Access Grants 管理存取權限。

本頁說明在具有 S3 Access Grants 整合的 Amazon EMR on EKS 中執行 Spark 作業的需求。使用 Amazon EMR on EKS 時，S3 Access Grants 需要在作業的執行角色中提供額外的 IAM 政策陳述式，以及 `StartJobRun` API 的額外覆寫組態。如需了解使用其他 Amazon EMR 部署設定 S3 Access Grants 的步驟，請參閱下列文件：

- 將 S3 Access Grants 與 Amazon EMR 搭配使用
- 將 S3 Access Grants 與 EMR Serverless 搭配使用

# 使用 S3 Access Grants 啟動 Amazon EMR on EKS 叢集以進行資料管理

您可以在 Amazon EMR on EKS 上啟用 S3 Access Grants，並啟動 Spark 作業。當您的應用程式提出 S3 資料請求時，Amazon S3 會提供僅限於特定儲存貯體、字首或物件的臨時憑證。

1. 為您的 Amazon EMR on EKS 叢集設定作業執行角色。包括執行 Spark 作業所需的必要 IAM 許可 `s3:GetDataAccess` 及 `s3:GetAccessGrantsInstanceForPrefix`：

```
{
    "Effect": "Allow",
    "Action": [
    "s3:GetDataAccess",
    "s3:GetAccessGrantsInstanceForPrefix"
    ],
    "Resource": [     //LIST ALL INSTANCE ARNS THAT THE ROLE IS ALLOWED TO QUERY
         "arn:aws_partition:s3:Region:account-id1:access-grants/default",
         "arn:aws_partition:s3:Region:account-id2:access-grants/default"
    ]
}
```

> **Note**
>
> 如果您指定用於作業執行的 IAM 角色具有直接存取 S3 的任何附加許可，則無論您在 S3 Access Grants 中定義的許可為何，使用者皆可能能夠存取資料

2. 將作業提交至 Amazon EMR on EKS 叢集 (Amazon EMR 發行標籤為 6.15 或更高版本以及 `emrfs-site` 分類)，如以下範例所示。使用適合您使用案例的值取代 *`red text`* 中的值。

```
{
  "name": "myjob",
  "virtualClusterId": "123456",
  "executionRoleArn": "iam_role_name_for_job_execution",
  "releaseLabel": "emr-7.8.0-latest",
  "jobDriver": {
    "sparkSubmitJobDriver": {
      "entryPoint": "entryPoint_location",
      "entryPointArguments": ["argument1", "argument2"],
       "sparkSubmitParameters": "--class main_class"
    }
  },
  "configurationOverrides": {
```

```
    "applicationConfiguration": [
      {
        "classification": "emrfs-site",
        "properties": {
          "fs.s3.s3AccessGrants.enabled": "true",
          "fs.s3.s3AccessGrants.fallbackToIAM": "false"
         }
      }
    ],
  }
}
```

## S3 Access Grants 與 Amazon EMR on EKS 搭配的考量事項

如需在將 Amazon S3 Access Grants 與 Amazon EMR on EKS 搭配使用時的重要支援、相容性和行為資訊，請參閱《Amazon EMR 管理指南》中的 S3 Access Grants 與 Amazon EMR on EKS 搭配的考量事項。

# Amazon EMR on EKS 的合規驗證

第三方稽核人員會在多個合規計畫中評估 Amazon EMR on EKS 的安全性和 AWS 合規性。這些計劃包括 SOC、PCI、FedRAMP、HIPAA 等等。

# Amazon EMR on EKS 的恢復能力

AWS 全球基礎設施是以 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、高輸送量和高冗餘聯網連接。透過可用區域，您可以設計與操作的應用程式和資料庫，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、容錯能力和擴展能力，均較單一或多個資料中心的傳統基礎設施還高。

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。

除了 AWS 全球基礎設施之外，Amazon EMR on EKS 透過 EMRFS 提供與 Amazon S3 的整合，以協助支援您的資料彈性和備份需求。

# Amazon EMR on EKS 中的基礎設施安全性

Amazon EMR 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設計您的 AWS 環境，請參閱 Security Pillar 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 EMR。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) 以產生暫時安全憑證以簽署請求。

## 組態與漏洞分析

AWS 處理基本安全任務，例如訪客作業系統 (OS) 和資料庫修補、防火牆組態和災難復原。這些程序已由適當的第三方進行檢閱並認證。如需詳細資訊，請參閱以下 資源：

- Amazon EMR on EKS 的合規驗證
- 共同的責任模型
- Amazon Web Services：安全程序概觀 (白皮書)

# 使用介面 VPC 端點連接至 Amazon EMR on EKS

您可以使用虛擬私有雲端 (VPC) 中的介面 VPC 端點 (AWS PrivateLink) 直接連接至 Amazon EMR on EKS，而非透過網際網路進行連接。當您使用界面 VPC 端點時，VPC 與 Amazon EMR on EKS 之間的通訊完全在 AWS 網路中執行。每個 VPC 端點皆會由一個或多個具私有 IP 地址彈性網路介面 (ENI) 來表示，而該介面位於 VPC 子網路中。

界面 VPC 端點會將您的 VPC 直接連接到 Amazon EMR on EKS，無需網際網路閘道、NAT 裝置、VPN 連線或 AWS Direct Connect 連線。VPC 中的執行個體不需要公有 IP 地址，就能與 Amazon EMR on EKS API 進行通訊。

您可以使用 AWS Management Console 或 AWS Command Line Interface (AWS CLI) 命令，建立介面 VPC 端點以連線至 Amazon EMR on EKS。如需詳細資訊，請參閱建立界面端點。

在建立介面 VPC 端點之後，如果您啟用了此端點的私有 DNS 主機名稱，則預設的 Amazon EMR on EKS 端點會解析為您的 VPC 端點。預設的 Amazon EMR on EKS 服務名稱端點會採用下列格式。

```
emr-containers.Region.amazonaws.com
```

如果您尚未啟用私有 DNS 主機名稱，Amazon VPC 會透過以下格式提供一個 DNS 端點名稱供您使用。

```
VPC_Endpoint_ID.emr-containers.Region.vpce.amazonaws.com
```

如需詳細資訊，請參閱《Amazon VPC 使用者指南》中的界面 VPC 端點 (AWS PrivateLink)。Amazon EMR on EKS 支援在您的 VPC 內呼叫其所有 API 動作。

可以將 VPC 端點政策附接至某個 VPC 端點，以控制 IAM 主體的存取權。您也可以將安全群組與 VPC 端點建立關聯，藉以根據網路流量的來源和目的地 (例如 IP 位址範圍) 來控制輸入和輸出存取。如需詳細資訊，請參閱使用 VPC 端點控制服務的存取。

## 為 Amazon EMR on EKS 建立 VPC 端點政策

可以為 Amazon EMR on EKS 的 Amazon VPC 端點建立政策，以指定下列各項：

- 可執行或不可執行動作的委託人
- 可執行的動作
- 可在其中執行動作的資源

如需詳細資訊，請參閱《Amazon VPC 使用者指南》中的使用 VPC 端點控制服務的存取。

Example VPC 端點政策拒絕來自指定 AWS 帳戶的所有存取

下列 VPC 端點政策拒絕 AWS 帳戶 *123456789012* 使用端點存取資源。

```
{
    "Statement": [
        {
            "Action": "*",
            "Effect": "Allow",
            "Resource": "*",
            "Principal": "*"
        },
        {
```

```
            "Action": "*",
            "Effect": "Deny",
            "Resource": "*",
            "Principal": {
                "AWS": [
                    "123456789012"
                ]
            }
        }
    ]
}
```

Example 可用來僅允許來自指定 IAM 主體 (使用者) 之 VPC 存取的 VPC 端點政策

下列 VPC 端點政策僅允許對 AWS 帳戶 *123456789012* 中的 IAM 使用者 *lijuan* 進行完整存取。所有其他 IAM 主體均無法存取該端點。

```
{
    "Statement": [
        {
            "Action": "*",
            "Effect": "Allow",
            "Resource": "*",
            "Principal": {
                "AWS": [
                    "arn:aws:iam::123456789012:user/lijuan"
                ]
            }
        }
    ]
}
```

Example 可用來允許唯讀 Amazon EMR on EKS 操作的 VPC 端點政策

下列 VPC 端點政策僅允許 AWS 帳戶 *123456789012* 執行指定的 Amazon EMR on EKS 動作。

指定的動作為 Amazon EMR on EKS 提供等效的唯讀存取權。拒絕指定的帳戶存取在該 VPC 上的所有其他動作。拒絕所有其他帳戶的任何存取。如需 Amazon EMR on EKS 動作清單，請參閱適用於 Amazon EMR on EKS 的動作、資源及條件金鑰。

```
{
    "Statement": [
        {
```

```
            "Action": [
                "emr-containers:DescribeJobRun",
                "emr-containers:DescribeVirtualCluster",
                "emr-containers:ListJobRuns",
                "emr-containers:ListTagsForResource",
                "emr-containers:ListVirtualClusters"
            ],
            "Effect": "Allow",
            "Resource": "*",
            "Principal": {
                "AWS": [
                    "123456789012"
                ]
            }
        }
    ]
}
```

Example 拒絕存取指定虛擬叢集的 VPC 端點政策

下列 VPC 端點政策允許所有帳戶和主體的完整存取，但拒絕對具有叢集 ID *A1B2CD34EF5G* 的虛擬叢集上執行的動作存取 AWS 帳戶 *123456789012*。仍然允許其他不支援虛擬叢集資源層級許可的 Amazon EMR on EKS 動作。如需 Amazon EMR on EKS 動作及其對應資源類型的清單，請參閱《AWS Identity and Access Management 使用者指南》中的適用於 Amazon EMR on EKS 的動作、資源及條件金鑰。

```
{
    "Statement": [
        {
            "Action": "*",
            "Effect": "Allow",
            "Resource": "*",
            "Principal": "*"
        },
        {
            "Action": "*",
            "Effect": "Deny",
            "Resource": "arn:aws:emr-containers:us-west-2:123456789012:/
virtualclusters/A1B2CD34EF5G",
            "Principal": {
                "AWS": [
                    "123456789012"
                ]
```

```
            }
        }
    ]
}
```

# 設定 Amazon EMR on EKS 的跨帳戶存取權

您可設定 Amazon EMR on EKS 的跨帳戶存取權。跨帳戶存取可讓某個 AWS 帳戶的使用者執行 Amazon EMR on EKS 任務，並存取屬於另一個 AWS 帳戶的基礎資料。

## 先決條件

若要設定 Amazon EMR on EKS 的跨帳戶存取權，您將在登入下列 AWS 帳戶時完成任務：

- `AccountA` - 您已透過在 EKS 叢集上註冊 Amazon EMR 命名空間來建立 Amazon EMR on EKS 虛擬叢集 AWS 的帳戶。
- `AccountB` - 包含您希望 Amazon EMR on EKS 任務存取的 Amazon S3 儲存貯體或 DynamoDB 資料表 AWS 的帳戶。

您必須先在 AWS 帳戶中備妥下列項目，才能設定跨帳戶存取：

- 您想要在其中執行作業的 `AccountA` 中的 Amazon EMR on EKS 虛擬叢集。
- `AccountA` 中的作業執行角色，它具有在虛擬叢集中執行作業所需的許可。如需詳細資訊，請參閱建立作業執行角色及搭配使用作業執行角色與 Amazon EMR on EKS。

## 如何存取跨帳戶 Amazon S3 儲存貯體或 DynamoDB 資料表

若要設定 Amazon EMR on EKS 的跨帳戶存取權，請完成以下步驟。

1. 在 `AccountB` 中建立 Amazon S3 儲存貯體 `cross-account-bucket`。如需詳細資訊，請參閱建立儲存貯體。如果想要擁有對 DynamoDB 的跨帳戶存取權，也可以在 `AccountB` 中建立 DynamoDB 資料表。如需詳細資訊，請參閱建立 DynamoDB 資料表。
2. 在 `AccountB` 中建立可存取 `cross-account-bucket` 的 `Cross-Account-Role-B` IAM 角色。

    1. 登入 IAM 主控台。

2. 選擇角色，並建立一個新角色 `Cross-Account-Role-B`。如需有關如何建立 IAM 角色的詳細資訊，請參閱《IAM 使用者指南》中的建立 IAM 角色。
3. 建立 IAM 政策，為 `Cross-Account-Role-B` 指定存取 `cross-account-bucket` S3 儲存貯體的許可，如下列政策陳述式所示。然後，將 IAM 政策附接至 `Cross-Account-Role-B`。如需詳細資訊，請參閱《IAM 使用者指南》中的建立新政策。

   JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "s3:*",
            "Resource": [
                "arn:aws:s3:::cross-account-bucket",
                "arn:aws:s3:::cross-account-bucket/*"
            ]
        }
    ]
}
```

   如果需要 DynamoDB 存取權，則請建立 IAM 政策，以指定存取跨帳戶 DynamoDB 表的許可。然後，將 IAM 政策附接至 `Cross-Account-Role-B`。如需詳細資訊，請參閱《IAM 使用者指南》中的建立 DynamoDB 資料表。

   以下是存取 DynamoDB 資料表 `CrossAccountTable` 的政策。

   JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "dynamodb:*",
            "Resource": "arn:aws:dynamodb:us-east-1:AccountB:table/
CrossAccountTable"
        }
    ]
}
```

3. 編輯 Cross-Account-Role-B 角色的信任關係。

   1. 若要設定角色的信任關係，請在 IAM 主控台中為「步驟 2：Cross-Account-Role-B」中建立的角色選擇信任關係索引標籤。
   2. 選取編輯信任關係。
   3. 新增下列政策文件，它允許 AccountA 中的 Job-Execution-Role-A 擔任此 Cross-Account-Role-B 角色。

      JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:role/Job-Execution-Role-
A"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

4. 在 AccountA 中對 Job-Execution-Role-A 授予 STS Assume role 許可以擔任 Cross-Account-Role-B。

   1. 在 AWS 帳戶 的 IAM 主控台中AccountA，選取 Job-Execution-Role-A。
   2. 將以下政策陳述式新增至 Job-Execution-Role-A 以允許 Cross-Account-Role-B 角色的 AssumeRole 動作。

      JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "sts:AssumeRole",
            "Resource": "arn:aws:iam::111122223333:role/Cross-Account-Role-
B"
        }
```

```
    ]
}
```

5. 對於 Amazon S3 存取，請在將作業提交至 Amazon EMR on EKS 時，設定下列 `spark-submit` 參數 (`spark conf`)。

> **Note**
>
> 依預設，EMRFS 會使用作業執行角色從作業中存取 S3 儲存貯體。但是，當 `customAWSCredentialsProvider` 設定為 `AssumeRoleAWSCredentialsProvider` 時，EMRFS 會使用您透過 `ASSUME_ROLE_CREDENTIALS_ROLE_ARN` 指定的對應角色而非 `Job-Execution-Role-A` 來進行 Amazon S3 存取。

- `--conf spark.hadoop.fs.s3.customAWSCredentialsProvider=com.amazonaws.emr.AssumeRol`
- `--conf spark.kubernetes.driverEnv.ASSUME_ROLE_CREDENTIALS_ROLE_ARN=arn:aws:iam::Ac` `Cross-Account-Role-B \`
- `--conf spark.executorEnv.ASSUME_ROLE_CREDENTIALS_ROLE_ARN=arn:aws:iam::AccountB:ro` `Cross-Account-Role-B \`

> **Note**
>
> 您必須在作業 Spark 組態中為執行程式和驅動程式 env 設定 `ASSUME_ROLE_CREDENTIALS_ROLE_ARN`。

對於 DynamoDB 跨帳戶存取權，必須設定 `--conf spark.dynamodb.customAWSCredentialsProvider=com.amazonaws.emr.AssumeRoleAWSCr`

6. 如下列範例所示，透過跨帳戶存取權執行 Amazon EMR on EKS 作業。

```
aws emr-containers start-job-run \
--virtual-cluster-id 123456 \
--name myjob \
```

```
--execution-role-arn execution-role-arn \
--release-label emr-6.2.0-latest \
--job-driver '{"sparkSubmitJobDriver": {"entryPoint": "entryPoint_location",
 "entryPointArguments": ["arguments_list"], "sparkSubmitParameters": "--class
 <main_class> --conf spark.executor.instances=2 --conf spark.executor.memory=2G
 --conf spark.executor.cores=2 --conf spark.driver.cores=1 --conf
 spark.hadoop.fs.s3.customAWSCredentialsProvider=com.amazonaws.emr.AssumeRoleAWSCredentials
 --conf
 spark.kubernetes.driverEnv.ASSUME_ROLE_CREDENTIALS_ROLE_ARN=arn:aws:iam::AccountB:role/
Cross-Account-Role-B --conf
 spark.executorEnv.ASSUME_ROLE_CREDENTIALS_ROLE_ARN=arn:aws:iam::AccountB:role/
Cross-Account-Role-B"}} ' \
--configuration-overrides '{"applicationConfiguration": [{"classification":
 "spark-defaults", "properties": {"spark.driver.memory": "2G"}}],
 "monitoringConfiguration": {"cloudWatchMonitoringConfiguration":
 {"logGroupName": "log_group_name", "logStreamNamePrefix": "log_stream_prefix"},
 "persistentAppUI":"ENABLED",  "s3MonitoringConfiguration": {"logUri": "s3://
my_s3_log_location" }}}'
```

# 為您的 Amazon EMR on EKS 資源加上標籤

為協助您管理 Amazon EMR on EKS 資源，可以使用標籤將您自己的中繼資料指派給每個資源。本主題提供標籤功能的概觀，並展示如何建立標籤。

主題


- 標籤基本概念
- 標記您的 資源
- 標籤限制
- 使用 AWS CLI 和 Amazon EMR on EKS API 處理標籤


## 標籤基本概念

標籤是您指派給 AWS 資源的標籤。每個標籤皆包含由您定義的一個金鑰與一個選用值。

標籤可讓您依用途、擁有者或環境等屬性來分類 AWS 資源。當您有許多相同類型的資源時，您可以依據先前指派的標籤，快速識別特定的資源。例如，您可以為 Amazon EMR on EKS 叢集定義一組標籤，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。

標籤不會自動指派給您的資源。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 EMR on EKS 來說不具有任何語意意義，並會嚴格解譯為字元字串。

標籤值可以為空白字串，但不得是 null。標籤金鑰不得為空白字串。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。

如需標籤型存取控制政策範例，請參閱 標籤型存取控制政策。

## 標記您的 資源

可以為新的或現有的虛擬叢集以及處於使用中狀態的作業執行加上標籤。作業執行的作用中狀態包括：`PENDING`、`SUBMITTED`、`RUNNING` 和 `CANCEL_PENDING`。虛擬叢集的作用中狀態包

括：RUNNING、TERMINATING 和 ARRESTED。如需詳細資訊，請參閱 作業執行狀態 和 虛擬叢集狀態。

當虛擬叢集終止時，會清除標籤且無法再存取。

如果您使用的是 Amazon EMR on EKS API AWS CLI、 或 AWS SDK，您可以使用相關 API 動作上的標籤參數，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。

在建立資源時，可以使用一些資源建立動作來指定資源的標籤。在這種情況下，如果在建立資源時無法套用標籤，則無法建立資源。該機制可確保您要在建立時標記的資源是以指定的標籤建立，不然就根本不會建立。如果您在建立時標記資源，則不需要在建立資源之後執行自訂標記指令碼。

下表描述了可加上標籤的 Amazon EMR on EKS 資源。

| 資源 | 支援標籤 | 支援標籤傳播 | 支援建立時標記 (Amazon EMR on EKS API AWS CLI和 AWS SDK) | 用於建立的 API (可以在建立過程中新增標籤) |
| --- | --- | --- | --- | --- |
| 虛擬叢集 | 是 | 否。與虛擬叢集關聯的標籤不會傳播到提交至該虛擬叢集的作業執行。 | 是 | CreateVirtualCluster |
| 任務執行 | 是 | 否 | 是 | StartJobRun |

# 標籤限制

以下基本限制適用於標籤：

- 每一資源最多標籤數 – 50
- 對於每一個資源，每個標籤金鑰必須是唯一的，且每個標籤金鑰只能有一個值。
- 索引鍵長度上限 - 128 個 UTF-8 Unicode 字元
- 值的長度上限 - 256 個 UTF-8 Unicode 字元
- 如果您的標記結構描述用於多個 AWS 服務和資源，請記住，其他 服務可能限制允許的字元。通常允許的字元包括：可用 UTF-8 表示的英文字母、數字和空格，還有以下字元：+ - = . _ : / @。

- 標籤鍵與值皆區分大小寫。
- 標籤值可以為空白字串，但不得是 null。標籤金鑰不得為空白字串。
- 請勿使用 `aws:`、`AWS:` 或任何大小寫組合作為索引鍵或值的字首。這些僅供保留 AWS 使用。

# 使用 AWS CLI 和 Amazon EMR on EKS API 處理標籤

使用下列 AWS CLI 命令或 Amazon EMR on EKS API 操作來新增、更新、列出和刪除資源的標籤。

| 任務 | AWS CLI | API 動作 |
| --- | --- | --- |
| 新增或覆寫一或多個標籤 | tag-resource | TagResource |
| 列出資源的標籤 | list-tags-for-resource | ListTagsForResource |
| 刪除一或多個標籤 | untag-resource | UntagResource |

下列範例示範如何使用 AWS CLI來標記或取消標記資源。

範例 1：為現有虛擬叢集加上標籤

以下命令會為現有叢集加上標籤。

```
aws emr-containers tag-resource --resource-arn resource_ARN --tags team=devs
```

範例 2：取消現有虛擬叢集的標籤

以下命令從現有虛擬叢集中刪除標籤。

```
aws emr-containers 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 emr-containers list-tags-for-resource --resource-arn resource_ARN
```

# Amazon EMR on EKS 疑難排解

本章節描述了如何疑難排解 Amazon EMR on EKS 的問題。如需有關如何疑難排解 Amazon EMR 一般問題的資訊，請參閱 Amazon EMR 管理指南中的對叢集進行疑難排解。

主題


- 使用 PersistentVolumeClaims (PVC) 對作業進行疑難排解
- 對 Amazon EMR on EKS 垂直自動擴展進行疑難排解
- 對 Amazon EMR on EKS Spark Operator 進行疑難排解


## 使用 PersistentVolumeClaims (PVC) 對作業進行疑難排解

如果需要建立、列出或刪除作業的 PersistentVolumeClaims (PVC)，但未將 PVC 許可新增至預設 Kubernetes 角色 emr-containers，則當您提交作業時，作業會失敗。如果沒有這些許可，emr-containers 角色就無法為 Spark 驅動程式或 Spark 用戶端建立必要的角色。如錯誤訊息所建議，將許可新增至 Spark 驅動程式或用戶端角色是不夠的。emr-containers 主要角色也必須包含必要的許可。本章節說明如何將必要的許可新增至 emr-containers 主要角色。

### 驗證

若要確認您的 emr-containers 角色是否擁有必要的許可，請使用自己的值來設定 NAMESPACE 變數，然後執行下列命令：

```
export NAMESPACE=YOUR_VALUE
kubectl describe role emr-containers -n ${NAMESPACE}
```

此外，若要驗證 Spark 和用戶端角色是否具有必要的許可，請執行下列命令：

```
kubectl describe role emr-containers-role-spark-driver -n ${NAMESPACE}
kubectl describe role emr-containers-role-spark-client -n ${NAMESPACE}
```

如果許可不存在，請繼續執行修補程式，如下所示。

### 修補程式

1. 如果沒有許可的作業目前正在執行中，則請停止這些作業。
2. 建立一個名為 RBAC_Patch.py 的檔案，如下所示：

```
import os
import subprocess as sp
import tempfile as temp
import json
import argparse
import uuid

def delete_if_exists(dictionary: dict, key: str):
    if dictionary.get(key, None) is not None:
        del dictionary[key]

def doTerminalCmd(cmd):
    with temp.TemporaryFile() as f:
        process = sp.Popen(cmd, stdout=f, stderr=f)
        process.wait()
        f.seek(0)
        msg = f.read().decode()
    return msg

def patchRole(roleName, namespace, extraRules, skipConfirmation=False):
    cmd = f"kubectl get role {roleName} -n {namespace} --output json".split(" ")
    msg = doTerminalCmd(cmd)
    if "(NotFound)" in msg and "Error" in msg:
        print(msg)
        return False
    role = json.loads(msg)
    rules = role["rules"]
    rulesToAssign = extraRules[::]
    passedRules = []
    for rule in rules:
        apiGroups = set(rule["apiGroups"])
        resources = set(rule["resources"])
        verbs = set(rule["verbs"])
        for extraRule in extraRules:
            passes = 0
            apiGroupsExtra = set(extraRule["apiGroups"])
            resourcesExtra = set(extraRule["resources"])
            verbsExtra = set(extraRule["verbs"])
            passes += len(apiGroupsExtra.intersection(apiGroups)) >=
 len(apiGroupsExtra)
            passes += len(resourcesExtra.intersection(resources)) >=
 len(resourcesExtra)
            passes += len(verbsExtra.intersection(verbs)) >= len(verbsExtra)
```

```
            if passes >= 3:
                if extraRule not in passedRules:
                    passedRules.append(extraRule)
                    if extraRule in rulesToAssign:
                        rulesToAssign.remove(extraRule)
                break
    prompt_text = "Apply Changes?"
    if len(rulesToAssign) == 0:
        print(f"The role {roleName} seems to already have the necessary
 permissions!")
        prompt_text = "Proceed anyways?"
    for ruleToAssign in rulesToAssign:
        role["rules"].append(ruleToAssign)
    delete_if_exists(role, "creationTimestamp")
    delete_if_exists(role, "resourceVersion")
    delete_if_exists(role, "uid")
    new_role = json.dumps(role, indent=3)
    uid = uuid.uuid4()
    filename = f"Role-{roleName}-New_Permissions-{uid}-TemporaryFile.json"
    try:
        with open(filename, "w+") as f:
            f.write(new_role)
            f.flush()
        prompt = "y"
        if not skipConfirmation:
            prompt = input(
                doTerminalCmd(f"kubectl diff -f {filename}".split(" ")) +
 f"\n{prompt_text} y/n: "
            ).lower().strip()
            while prompt != "y" and prompt != "n":
                prompt = input("Please make a valid selection. y/n:
 ").lower().strip()
        if prompt == "y":
            print(doTerminalCmd(f"kubectl apply -f {filename}".split(" ")))
    except Exception as e:
        print(e)
    os.remove(f"./{filename}")

if __name__ == '__main__':
    parser = argparse.ArgumentParser()
    parser.add_argument("-n", "--namespace",
                        help="Namespace of the Role. By default its the
 VirtualCluster's namespace",
                        required=True,
```

```
                        dest="namespace"
                        )

    parser.add_argument("-p", "--no-prompt",
                        help="Applies the patches without asking first",
                        dest="no_prompt",
                        default=False,
                        action="store_true"
                        )
    args = parser.parse_args()

    emrRoleRules = [
        {
            "apiGroups": [""],
            "resources": ["persistentvolumeclaims"],
            "verbs": ["list", "create", "delete", "patch"]
         }

    ]

    driverRoleRules = [
        {
            "apiGroups": [""],
            "resources": ["persistentvolumeclaims"],
            "verbs": ["list", "create", "delete", "patch", "deletecollection"]
        },
        {
            "apiGroups": [""],
            "resources": ["services"],
            "verbs": ["get", "list", "describe", "create", "delete", "watch",
 "deletecollection"]
        },
        {
            "apiGroups": [""],
            "resources": ["configmaps", "pods"],
            "verbs": ["deletecollection"]
        }
    ]

    clientRoleRules = [
        {
            "apiGroups": [""],
            "resources": ["persistentvolumeclaims"],
            "verbs": ["list", "create", "delete", "patch"]
```

```
        }
    ]

    patchRole("emr-containers", args.namespace, emrRoleRules, args.no_prompt)
    patchRole("emr-containers-role-spark-driver", args.namespace, driverRoleRules,
 args.no_prompt)
    patchRole("emr-containers-role-spark-client", args.namespace, clientRoleRules,
 args.no_prompt)
```

3. 執行 Python 指令碼：

```
python3 RBAC_Patch.py -n ${NAMESPACE}
```

4. 新許可與舊許可之間的 kubectl 差異會出現。按 y 來修補角色。
5. 驗證具有其他許可的三個角色，如下所示：

```
kubectl describe role -n ${NAMESPACE}
```

6. 執行 Python 指令碼：

```
python3 RBAC_Patch.py -n ${NAMESPACE}
```

7. 執行命令後，它將顯示新許可和舊許可之間的 kubectl 差異。按 y 來修補角色。
8. 驗證具有其他許可的三個角色：

```
kubectl describe role -n ${NAMESPACE}
```

9. 再次提交作業。

## 手動修補

如果應用程式所需的許可適用於 PVC 規則以外的項目，可以視需要為 Amazon EMR 虛擬叢集手動新增 Kubernetes 許可。

> **Note**
>
> emr-containers 角色是主要角色。這意味著它必須提供所有必要的許可，然後才能變更基礎驅動程式或用戶端角色。

1. 透過執行以下命令將當前許可下載到 yaml 檔案中：

```
kubectl get role -n ${NAMESPACE} emr-containers -o yaml >> emr-containers-role-
patch.yaml
kubectl get role -n ${NAMESPACE} emr-containers-role-spark-driver -o yaml >> driver-
role-patch.yaml
kubectl get role -n ${NAMESPACE} emr-containers-role-spark-client -o yaml >> client-
role-patch.yaml
```

2. 根據應用程式所需的許可，編輯每個檔案並新增其他規則，例如：

   - emr-containers-role-patch.yaml

```
- apiGroups:
  - ""
  resources:
  - persistentvolumeclaims
  verbs:
  - list
  - create
  - delete
  - patch
```

   - driver-role-patch.yaml

```
- apiGroups:
  - ""
  resources:
  - persistentvolumeclaims
  verbs:
  - list
  - create
  - delete
  - patch
  - deletecollection
- apiGroups:
  - ""
  resources:
  - services
  verbs:
  - get
  - list
  - describe
  - create
```

```
  - delete
  - watch
  - deletecollection
- apiGroups:
  - ""
  resources:
  - configmaps
  - pods
  verbs:
  - deletecollection
```

- client-role-patch.yaml

```
- apiGroups:
  - ""
  resources:
  - persistentvolumeclaims
  verbs:
  - list
  - create
  - delete
  - patch
```

3. 移除下列屬性及其值。這對於套用更新是必要的。
   - creationTimestamp
   - resourceVersion
   - uid
4. 最後，執行修補程式：

```
kubectl apply -f emr-containers-role-patch.yaml
kubectl apply -f driver-role-patch.yaml
kubectl apply -f client-role-patch.yaml
```

# 對 Amazon EMR on EKS 垂直自動擴展進行疑難排解

如果在具有 Operator Lifecycle Manager 的 Amazon EKS 叢集上設定 Amazon EMR on EKS 垂直自動擴展運算子時遇到問題，請參閱以下各章節。如需詳細資訊，包括完成安裝的步驟，請參閱 搭配使用垂直自動擴展與 Amazon EMR Spark 作業。

## 「403 禁止」錯誤

如果遵循 在 Amazon EKS 叢集上安裝 Operator Lifecycle Manager (OLM) 中的步驟，執行 `olm status` 命令，並且它傳回如下所示的 `403 Forbidden` 錯誤，則可能尚未為運算子取得 Amazon ECR 儲存庫的驗證字符。

若要解決此問題，請重複 安裝 Amazon EMR on EKS 垂直自動擴展運算子 中的步驟以取得字符。然後，請再次嘗試安裝。

```
Error: FATA[0002] Failed to run bundle: pull bundle image: error pulling image IMAGE.
error resolving name : unexpected status code [manifests latest]: 403 Forbidden
```

## 找不到 Kubernetes 命名空間

當您在 Amazon EKS 叢集上設定 Amazon EMR on EKS 垂直自動擴展運算子時，可能會收到如下所示的 `namespaces not found` 錯誤：

```
FATA[0020] Failed to run bundle: create catalog: error creating catalog source:
 namespaces "NAME" not found.
```

如果指定的命名空間不存在，則 OLM 將不會安裝自動垂直擴展運算子。若要解決此問題，請使用以下命令來建立命名空間。然後，請再次嘗試安裝。

```
kubectl create namespace NAME
```

## 儲存 Docker 憑證時發生錯誤

若要設定垂直自動擴展，必須驗證並擷取 Amazon EMR on EKS 垂直自動擴展相關 Docker 映像檔。執行此操作時，如果 Docker 未執行，則可能會收到類似以下錯誤：

```
aws ecr get-login-password \
 --region $REGION | docker login \
 --username AWS \
 --password-stdin $ACCOUNT_ID.dkr.ecr.$REGION.amazonaws.com

Error saving credentials: error storing credentials - err: exit status 1
out: 'Post "http://ipc/registry/credstore-updated": dial unix backend.sock: connect: no
 such file or directory'
```

要解決此問題，請確認 Docker 正在執行或開啟 Docker Desktop。然後，嘗試再次儲存您的憑證。

# 對 Amazon EMR on EKS Spark Operator 進行疑難排解

如果遇到 Amazon EMR on EKS Spark Operator 的問題，則請參閱以下各章節。如需詳細資訊，包括完成安裝的步驟，請參閱 使用 Spark Operator 執行 Spark 作業。

## Helm Chart 安裝錯誤

如果遵循 安裝 Spark Operator 中的步驟，當您嘗試安裝或確認 Helm Chart 時，它傳回如下所示的 `INSTALLATION FAILED` 錯誤，則可能尚未為運算子取得 Amazon ECR 儲存庫的驗證字符。

要解決此問題，請重複 安裝 Spark Operator 中的步驟，向 Amazon ECR 登錄檔驗證 Helm 用戶端。然後，請再次嘗試安裝步驟。

```
Error: INSTALLATION FAILED: Kubernetes cluster unreachable: the server has asked for
 the client to provide credentials
```

## UnsupportedFileSystemException: No FileSystem for scheme "s3"

您可能會在執行緒「main」中遇到以下例外狀況：

```
org.apache.hadoop.fs.UnsupportedFileSystemException: No FileSystem for scheme "s3"
```

如果發生這種情況，請將下列例外狀況新增至 `SparkApplication` 規格：

```
 hadoopConf:
   # EMRFS filesystem
   fs.s3.customAWSCredentialsProvider:
 com.amazonaws.auth.WebIdentityTokenCredentialsProvider
   fs.s3.impl: com.amazon.ws.emr.hadoop.fs.EmrFileSystem
   fs.AbstractFileSystem.s3.impl: org.apache.hadoop.fs.s3.EMRFSDelegate
   fs.s3.buffer.dir: /mnt/s3
   fs.s3.getObject.initialSocketTimeoutMilliseconds: "2000"
   mapreduce.fileoutputcommitter.algorithm.version.emr_internal_use_only.EmrFileSystem:
 "2"
   mapreduce.fileoutputcommitter.cleanup-
failures.ignored.emr_internal_use_only.EmrFileSystem: "true"
 sparkConf:
   # Required for EMR Runtime
```

```
   spark.driver.extraClassPath: /usr/lib/hadoop-lzo/lib/*:/usr/lib/hadoop/hadoop-
aws.jar:/usr/share/aws/aws-java-sdk/*:/usr/share/aws/emr/emrfs/conf:/usr/share/aws/
emr/emrfs/lib/*:/usr/share/aws/emr/emrfs/auxlib/*:/usr/share/aws/emr/security/conf:/
usr/share/aws/emr/security/lib/*:/usr/share/aws/hmclient/lib/aws-glue-datacatalog-
spark-client.jar:/usr/share/java/Hive-JSON-Serde/hive-openx-serde.jar:/usr/share/aws/
sagemaker-spark-sdk/lib/sagemaker-spark-sdk.jar:/home/hadoop/extrajars/*
   spark.driver.extraLibraryPath: /usr/lib/hadoop/lib/native:/usr/lib/hadoop-lzo/lib/
native:/docker/usr/lib/hadoop/lib/native:/docker/usr/lib/hadoop-lzo/lib/native
   spark.executor.extraClassPath: /usr/lib/hadoop-lzo/lib/*:/usr/lib/hadoop/hadoop-
aws.jar:/usr/share/aws/aws-java-sdk/*:/usr/share/aws/emr/emrfs/conf:/usr/share/aws/
emr/emrfs/lib/*:/usr/share/aws/emr/emrfs/auxlib/*:/usr/share/aws/emr/security/conf:/
usr/share/aws/emr/security/lib/*:/usr/share/aws/hmclient/lib/aws-glue-datacatalog-
spark-client.jar:/usr/share/java/Hive-JSON-Serde/hive-openx-serde.jar:/usr/share/aws/
sagemaker-spark-sdk/lib/sagemaker-spark-sdk.jar:/home/hadoop/extrajars/*
   spark.executor.extraLibraryPath: /usr/lib/hadoop/lib/native:/usr/lib/hadoop-lzo/lib/
native:/docker/usr/lib/hadoop/lib/native:/docker/usr/lib/hadoop-lzo/lib/native
```

# Amazon EMR on EKS 服務端點和配額

以下是 Amazon EMR on EKS 的服務端點和服務配額。若要以程式設計方式連線至 AWS 服務，您可以使用 端點。除了標準 AWS 端點之外，某些 AWS 服務還在所選區域中提供 FIPS 端點。如需詳細資訊，請參閱 AWS 服務端點。服務配額 (也稱為限制) 是 AWS 帳戶的服務資源或操作的最大數量。如需更多相關資訊，請參閱 AWS Service Quotas。

## 服務端點

| AWS 區域 名稱 | 代碼 | 端點 | 通訊協定 |
| --- | --- | --- | --- |
| 美國東部 (維吉尼亞北部) | us-east-1 | `emr-containers.us-east-1.amazonaws.com` | HTTPS |
| 美國東部 (俄亥俄) | us-east-2 | `emr-containers.us-east-2.amazonaws.com` | HTTPS |
| 美國西部 (加利佛尼亞北部) | us-west-1 | `emr-containers.us-west-1.amazonaws.com` | HTTPS |
| 美國西部 (奧勒岡) | us-west-2 | `emr-containers.us-west-2.amazonaws.com` | HTTPS |
| 亞太區域 (東京) | ap-northeast-1 | `emr-containers.ap-northeast-1.amazonaws.com` | HTTPS |
| 亞太區域 (首爾) | ap-northeast-2 | `emr-containers.ap-northeast-2.amazonaws.com` | HTTPS |
| 亞太區域 (大阪) | ap-northeast-3 | `emr-containers.ap-northeast-3.amazonaws.com` | HTTPS |
| 亞太區域 (孟買) | ap-south-1 | `emr-containers.ap-south-1.amazonaws.com` | HTTPS |
| 亞太區域 (海德拉巴) | ap-south-2 | `emr-containers.ap-south-2.amazonaws.com` | HTTPS |

| AWS 區域 名稱 | 代碼 | 端點 | 通訊協定 |
| --- | --- | --- | --- |
| 亞太區域 (新加坡) | ap-southeast-1 | emr-containers.ap-southeast-1.amazonaws.com | HTTPS |
| 亞太區域 (悉尼) | ap-southeast-2 | emr-containers.ap-southeast-2.amazonaws.com | HTTPS |
| 亞太區域 (雅加達) | ap-southeast-3 | emr-containers.ap-southeast-3.amazonaws.com | HTTPS |
| 亞太區域 (香港) | ap-east-1 | emr-containers.ap-east-1.amazonaws.com | HTTPS |
| 非洲 (開普敦) | af-south-1 | emr-containers.af-south-1.amazonaws.com | HTTPS |
| 加拿大 (中部) | ca-central-1 | emr-containers.ca-central-1.amazonaws.com | HTTPS |
| 中國 (寧夏) | cn-northwest-1 | emr-containers.cn-northwest-1.amazonaws.com.cn | HTTPS |
| 中國 (北京) | cn-north-1 | emr-containers.cn-north-1.amazonaws.com.cn | HTTPS |
| 歐洲 (法蘭克福) | eu-central-1 | emr-containers.eu-central-1.amazonaws.com | HTTPS |
| 歐洲 (蘇黎世) | eu-central-2 | emr-containers.eu-central-2.amazonaws.com | HTTPS |
| 歐洲 (愛爾蘭) | eu-west-1 | emr-containers.eu-west-1.amazonaws.com | HTTPS |
| 歐洲 (倫敦) | eu-west-2 | emr-containers.eu-west-2.amazonaws.com | HTTPS |

| AWS 區域 名稱 | 代碼 | 端點 | 通訊協定 |
| --- | --- | --- | --- |
| 歐洲 (巴黎) | eu-west-3 | `emr-containers.eu-west-3.amazonaws.com` | HTTPS |
| 歐洲 (斯德哥爾摩) | eu-north-1 | `emr-containers.eu-north-1.amazonaws.com` | HTTPS |
| 歐洲 (米蘭) | eu-south-1 | `emr-containers.eu-south-1.amazonaws.com` | HTTPS |
| 歐洲 (西班牙) | eu-south-2 | `emr-containers.eu-south-2.amazonaws.com` | HTTPS |
| 以色列 (特拉維夫) | il-central-1 | `emr-containers.il-central-1.amazonaws.com` | HTTPS |
| 南美洲 (聖保羅) | sa-east-1 | `emr-containers.sa-east-1.amazonaws.com` | HTTPS |
| 中東 (阿拉伯聯合大公國) | me-central-1 | `emr-containers.me-central-1.amazonaws.com` | HTTPS |
| Middle East (Bahrain) | me-south-1 | `emr-containers.me-south-1.amazonaws.com` | HTTPS |
| AWS GovCloud （美國東部） | us-gov-east-1 | `emr-containers.us-gov-east-1.amazonaws.com` | HTTPS |
| AWS GovCloud （美國西部） | us-gov-west-1 | `emr-containers.us-gov-west-1.amazonaws.com` | HTTPS |

# Service Quotas

Amazon EMR on EKS 會針對每個區域的每個 AWS 帳戶調節下列 API 請求。如需有關如何套用限流的詳細資訊，請參閱 Amazon EC2 API 參考中的 API 請求限流。您可以依照下列指南，請求提高 AWS 帳戶的 API 限流配額。

| API 動作 | 儲存貯體容量上限 | 儲存貯體重新填滿速率 (每秒) |
| --- | --- | --- |
| CancelJobRun | 25 | 1 |
| CreateJobTemplate | 25 | 1 |
| CreateManagedEndpoint | 25 | 1 |
| CreateSecurityConfiguration | 25 | 1 |
| CreateVirtualCluster | 25 | 1 |
| DeleteJobTemplate | 25 | 1 |
| DeleteManagedEndpoint | 25 | 1 |
| DeleteVirtualCluster | 25 | 1 |
| DescribeJobRun | 100 | 20 |
| DescribeJobTemplate | 25 | 1 |
| DescribeManagedEndpoint | 100 | 5 |
| DescribeSecurityConfiguration | 25 | 1 |
| DescribeVirtualCluster | 100 | 5 |
| GetManagedEndpoint SessionCredentials | 25 | 1 |
| ListJobRuns | 100 | 5 |
| ListJobTemplates | 25 | 1 |
| ListManagedEndpoints | 25 | 1 |
| ListSecurityConfigurations | 25 | 1 |
| ListVirtualClusters | 100 | 5 |
| StartJobRun | 25 | 1 |

| API 動作 | 儲存貯體容量上限 | 儲存貯體重新填滿速率 (每秒) |
| --- | --- | --- |
| Throttle quota for all EMR on EKS API requests | 200 | 20 |

當您建立 AWS 帳戶時，我們會 AWS 針對每個區域的資源設定預設配額（也稱為限制）。如果您嘗試超過資源的配額，請求會失敗。如果發生這種情況，您可以減少資源用量或請求增加配額。

Service Quotas 主控台是一個集中位置，您可以在其中檢視和管理 AWS 服務的配額，並針對您使用的許多資源請求增加配額。使用此處提供的配額資訊來管理您的 AWS 基礎設施。計劃在您需要的時候，先行請求提高配額。

## 檢視配額和請求增加配額

您可以使用 Service Service Quotas配額。如需詳細說明，請參閱《Service Quotas 使用者指南》中的檢視服務配額。 Service Quotas

您可以從 AWS 管理主控台或使用 CLI AWS 請求增加配額。步驟詳述於請求提高配額。

# Amazon EMR on EKS 發行版本

Amazon EMR 版本是一組來自大數據生態系統的開源應用程式。每個版本都含有不同的大數據應用程式、組件和功能，您可以在執行作業時選擇要讓 Amazon EMR on EKS 部署和設定哪些項目。

從 Amazon EMR 5.32.0 和 6.2.0 版開始，您可部署 Amazon EMR on EKS。舊版 Amazon EMR 發行版本無法使用此部署選項。提交作業時，必須指定支援的發行版本。

Amazon EMR on EKS 使用以下形式的發行標籤：`emr-x.x.x-latest` 或具有特定發行日期的 `emr-x.x.x-yyyymmdd`。例如 `emr-7.8.0-latest` 或 `emr-7.8.0-20210129`。使用 `-latest` 尾碼時，請確保 Amazon EMR 版本始終包含最新的安全更新。

> **Note**
>
> 如需 Amazon EMR on EKS 和在 EC2 上執行的 Amazon EMR 之間的比較，請參閱 AWS 網站上的 Amazon EMR FAQs。

主題


- Amazon EMR on EKS 7.9.0 版
- Amazon EMR on EKS 7.8.0 版本
- Amazon EMR on EKS 7.7.0 版本
- Amazon EMR on EKS 7.6.0 版
- Amazon EMR on EKS 7.5.0 版
- Amazon EMR on EKS 7.4.0 版本
- Amazon EMR on EKS 7.3.0 版本
- Amazon EMR on EKS 7.2.0 版本
- Amazon EMR on EKS 7.1.0 版
- Amazon EMR on EKS 7.0.0 版
- Amazon EMR on EKS 6.15.0 版
- Amazon EMR on EKS 6.14.0 版
- Amazon EMR on EKS 6.13.0 版
- Amazon EMR on EKS 6.12.0 版

- Amazon EMR on EKS 6.11.0 版
- Amazon EMR on EKS 6.10.0 版
- Amazon EMR on EKS 6.9.0 版
- Amazon EMR on EKS 6.8.0 版
- Amazon EMR on EKS 6.7.0 版
- Amazon EMR on EKS 6.6.0 版
- Amazon EMR on EKS 6.5.0 版
- Amazon EMR on EKS 6.4.0 版
- Amazon EMR on EKS 6.3.0 版
- Amazon EMR on EKS 6.2.0 版
- Amazon EMR on EKS 5.36.0 版
- Amazon EMR on EKS 5.35.0 版
- Amazon EMR on EKS 5.34.0 版
- Amazon EMR on EKS 5.33.0 版
- Amazon EMR on EKS 5.32.0 版

# Amazon EMR on EKS 7.9.0 版

本頁面描述了針對 Amazon EMR on EKS 部署的 Amazon EMR 新功能和更新功能。如需在 Amazon EC2 上執行的 Amazon EMR 和一般 Amazon EMR 7.9.0 版本的詳細資訊，請參閱《Amazon EMR 版本指南》中的 Amazon EMR 7.9.0。

## Amazon EMR on EKS 7.9 版

下列 Amazon EMR 7.9.0 版本適用於 Amazon EMR on EKS。選取特定的 emr-7.9.0-XXXX 版本，以檢視更多詳細資訊，例如相關的容器映像標籤。

Flink releases

當您執行 Flink 應用程式時，Amazon EMR on EKS 可使用下列 Amazon EMR 7.9.0 版本。

- emr-7.9.0-flink-latest
- emr-7.9.0-flink-20250425

Spark releases

當您執行 Spark 應用程式時，Amazon EMR on EKS 可使用下列 Amazon EMR 7.9.0 版本。

- emr-7.9.0-latest
- emr-7.9.0-20250425
- emr-7.9.0-spark-rapids-latest
- emr-7.9.0-spark-rapids-20250425
- emr-7.9.0-java11-latest
- emr-7.9.0-java11-20250425
- emr-7.9.0-java8-latest
- emr-7.9.0-java8-20250425
- emr-7.9.0-spark-rapids-java8-latest
- emr-7.9.0-spark-rapids-java8-20250425
- notebook-spark/emr-7.9.0-latest
- notebook-spark/emr-7.9.0-20250425
- notebook-spark/emr-7.9.0-spark-rapids-latest
- notebook-spark/emr-7.9.0-spark-rapids-20250425
- notebook-spark/emr-7.9.0-java11-latest
- notebook-spark/emr-7.9.0-java11-20250425
- notebook-spark/emr-7.9.0-java8-latest
- notebook-spark/emr-7.9.0-java8-20250425
- notebook-spark/emr-7.9.0-spark-rapids-java8-latest
- notebook-spark/emr-7.9.0-spark-rapids-java8-20250425
- notebook-python/emr-7.9.0-latest
- notebook-python/emr-7.9.0-20250425
- notebook-python/emr-7.9.0-spark-rapids-latest
- notebook-python/emr-7.9.0-spark-rapids-20250425
- notebook-python/emr-7.9.0-java11-latest
- notebook-python/emr-7.9.0-java11-20250425
- notebook-python/emr-7.9.0-java8-latest
- notebook-python/emr-7.9.0-java8-20250425

- notebook-python/emr-7.9.0-spark-rapids-java8-latest
- notebook-python/emr-7.9.0-spark-rapids-java8-20250425
- livy/emr-7.9.0-latest
- livy/emr-7.9.0-20250425
- livy/emr-7.9.0-java11-latest
- livy/emr-7.9.0-java11-20250425
- livy/emr-7.9.0-java8-latest
- livy/emr-7.9.0-java8-20250425

## 版本備註

Amazon EMR on EKS 7.9.0 的版本備註

- 支援的應用程式：適用於 Java 的 AWS SDK 2.31.16 and 1.12.782, Apache Spark 3.5.5, Apache Hudi 0.15.0-amzn-6, Apache Iceberg 1.7.1-amzn-2, Delta 3.3.0-amzn-1, Apache Spark RAPIDS 25.02.1-amzn-0, Jupyter Enterprise Gateway 2.6.0, Apache Flink 1.20.0-amzn-3, Flink Operator 1.10.0-amzn-3
- 支援的元件 - `emr-ddb`、`emr-goodies`、`emr-s3-select`、`emrfs`、`hadoop-client`、`hudihudi-spark`、`iceberg`、、`spark-kubernetes`。
- 支援的組態分類

  若要與 StartJobRun 和 CreateManagedEndpoint API 搭配使用：

| 分類 | 描述 |
| --- | --- |
| `core-site` | 變更 `core-site.xml` Hadoop 檔案中的值。 |
| `emrfs-site` | 變更 EMRFS 設定。 |
| `spark-metrics` | 變更 `metrics.properties` Spark 檔案中的值。 |
| `spark-defaults` | 變更 `spark-defaults.conf` Spark 檔案中的值。 |

| 分類 | 描述 |
| --- | --- |
| spark-env | 變更 Spark 環境中的值。 |
| spark-hive-site | 變更 hive-site.xml Spark 檔案中的值。 |
| spark-log4j2 | 變更 log4j2.properties Spark 檔案中的值。 |
| emr-job-submitter | 作業提交者 Pod 的組態。 |

專門搭配 CreateManagedEndpoint API 使用：

| 分類 | 描述 |
| --- | --- |
| jeg-config | 變更 Jupyter Enterprise Gateway jupyter_enterprise_gateway_config.py 檔案中的值。 |
| jupyter-kernel-overrides | 在 Jupyter 核心規格檔案中變更核心映像的值。 |

組態分類可讓您自訂應用程式。這些檔案通常對應於應用程式的組態 XML 檔案，例如 spark-hive-site.xml。如需詳細資訊，請參閱設定應用程式。

## 變更

Amazon EMR on EKS 7.9.0 版包含下列變更：

- 版本沒有變更。

## emr-7.9.0-latest

版本說明：emr-7.9.0-latest 目前指向 emr-7.9.0-20250425。

區域：`emr-7.9.0-latest` 適用於 Amazon EMR on EKS 支援的所有區域。如需詳細資訊，請參閱 Amazon EMR on EKS 服務端點。

容器映像標籤：`emr-7.9.0:latest`

### emr-7.9.0-20250425

版本備註：`emr-7.9.0-20250425`已於 2025 年 2 月發行。這是 Amazon EMR 7.9.0 (Spark) 的初始版本。

區域：`emr-emr-7.9.0-20250425` 適用於 Amazon EMR on EKS 支援的所有區域。如需詳細資訊，請參閱 Amazon EMR on EKS 服務端點。

容器映像標籤：`emr-7.9.0-20250425`

### emr-7.9.0-flink-latest

版本備註：`emr-7.9.0-flink-latest`目前指向 `emr-7.9.0-flink-20250425`

區域：`emr-7.9.0-flink-latest` 適用於 Amazon EMR on EKS 支援的所有區域。如需詳細資訊，請參閱 Amazon EMR on EKS 服務端點。

容器映像標籤：`emr-7.9.0-flink:latest`

### emr-7.9.0-flink-20250425

版本備註：`7.9.0-flink-20250425`已於 2025 年 2 月發行。這是 Amazon EMR 7.9.0 (Flink) 的初始版本。

區域：`emr-7.9.0-flink-20250425` 適用於 Amazon EMR on EKS 支援的所有區域。如需詳細資訊，請參閱 Amazon EMR on EKS 服務端點。

容器映像標籤：`emr-7.9.0-flink:20250425`

## Amazon EMR on EKS 7.8.0 版本

本頁面描述了針對 Amazon EMR on EKS 部署的 Amazon EMR 新功能和更新功能。如需在 Amazon EC2 上執行的 Amazon EMR 和一般 Amazon EMR 7.8.0 版本的詳細資訊，請參閱《Amazon EMR 版本指南》中的 Amazon EMR 7.8.0。

# Amazon EMR on EKS 7.8 版

下列 Amazon EMR 7.8.0 版本適用於 Amazon EMR on EKS。選取特定的 emr-7.8.0-XXXX 版本，以檢視更多詳細資訊，例如相關的容器映像標籤。

Flink releases

當您執行 Flink 應用程式時，下列 Amazon EMR 7.8.0 版本可供 Amazon EMR on EKS 使用。

- emr-7.8.0-flink-latest
- emr-7.8.0-flink-20250228

Spark releases

當您執行 Spark 應用程式時，Amazon EMR on EKS 可使用下列 Amazon EMR 7.8.0 版本。

- emr-7.8.0-latest
- emr-7.8.0-20250228
- emr-7.8.0-spark-rapids-latest
- emr-7.8.0-spark-rapids-20250228
- emr-7.8.0-java11-latest
- emr-7.8.0-java11-20250228
- emr-7.8.0-java8-latest
- emr-7.8.0-java8-20250228
- emr-7.8.0-spark-rapids-java8-latest
- emr-7.8.0-spark-rapids-java8-20250228
- notebook-spark/emr-7.8.0-latest
- notebook-spark/emr-7.8.0-20250228
- notebook-spark/emr-7.8.0-spark-rapids-latest
- notebook-spark/emr-7.8.0-spark-rapids-20250228
- notebook-spark/emr-7.8.0-java11-latest
- notebook-spark/emr-7.8.0-java11-20250228
- notebook-spark/emr-7.8.0-java8-latest
- notebook-spark/emr-7.8.0-java8-20250228

- notebook-spark/emr-7.8.0-spark-rapids-java8-latest
- notebook-spark/emr-7.8.0-spark-rapids-java8-20250228
- notebook-python/emr-7.8.0-latest
- notebook-python/emr-7.8.0-20250228
- notebook-python/emr-7.8.0-spark-rapids-latest
- notebook-python/emr-7.8.0-spark-rapids-20250228
- notebook-python/emr-7.8.0-java11-latest
- notebook-python/emr-7.8.0-java11-20250228
- notebook-python/emr-7.8.0-java8-latest
- notebook-python/emr-7.8.0-java8-20250228
- notebook-python/emr-7.8.0-spark-rapids-java8-latest
- notebook-python/emr-7.8.0-spark-rapids-java8-20250228
- livy/emr-7.8.0-latest
- livy/emr-7.8.0-20250228
- livy/emr-7.8.0-java11-latest
- livy/emr-7.8.0-java11-20250228
- livy/emr-7.8.0-java8-latest
- livy/emr-7.8.0-java8-20250228

## 版本備註

Amazon EMR on EKS 7.8.0 的版本備註

- 支援的應用程式：適用於 Java 的 AWS SDK 2.29.52 and 1.12.780, Apache Spark 3.5.4, Apache Hudi 0.15.0-amzn-5, Apache Iceberg 1.7.1-amzn-1, Delta 3.3.0-amzn-0, Apache Spark RAPIDS 24.12.0-amzn-0, Jupyter Enterprise Gateway 2.6.0, Apache Flink 1.20.0-amzn-2, Flink Operator 1.10.0-amzn-2
- 支援的元件 - `emr-ddb`、`emr-goodies`、`emr-s3-select`、`emrfs`、`hadoop-clienthudi`、`hudi-spark`、`iceberg`、、`spark-kubernetes`。
- 支援的組態分類

  若要與 StartJobRun 和 CreateManagedEndpoint API 搭配使用：

| 分類 | 描述 |
| --- | --- |
| core-site | 變更 core-site.xml Hadoop 檔案中的值。 |
| emrfs-site | 變更 EMRFS 設定。 |
| spark-metrics | 變更 metrics.properties Spark 檔案中的值。 |
| spark-defaults | 變更 spark-defaults.conf Spark 檔案中的值。 |
| spark-env | 變更 Spark 環境中的值。 |
| spark-hive-site | 變更 hive-site.xml Spark 檔案中的值。 |
| spark-log4j2 | 變更 log4j2.properties Spark 檔案中的值。 |
| emr-job-submitter | 作業提交者 Pod 的組態。 |

專門搭配 CreateManagedEndpoint API 使用：

| 分類 | 描述 |
| --- | --- |
| jeg-config | 變更 Jupyter Enterprise Gateway jupyter_enterprise_gateway_config.py 檔案中的值。 |
| jupyter-kernel-overrides | 在 Jupyter 核心規格檔案中變更核心映像的值。 |

組態分類可讓您自訂應用程式。這些檔案通常對應於應用程式的組態 XML 檔案，例如 spark-hive-site.xml。如需詳細資訊，請參閱設定應用程式。

# 變更

Amazon EMR on EKS 7.8.0 版包含下列變更：

- 原生 FGAC 功能，包括：
  - Iceberg 支援在精細存取控制 (FGAC) 虛擬叢集中的非ake Formation Tables 上執行動作的任務。(IAM 有後援。)
  - S3 資料表支援
- Spark 連線

## emr-7.8.0-latest

版本說明：`emr-7.8.0-latest` 目前指向 `emr-7.8.0-20250228`。

區域：`emr-7.8.0-latest` 適用於 Amazon EMR on EKS 支援的所有區域。如需詳細資訊，請參閱 Amazon EMR on EKS 服務端點。

容器映像標籤：`emr-7.8.0:latest`

## emr-7.8.0-20250228

版本備註：`emr-7.8.0-20250228`已於 2025 年 2 月發行。這是 Amazon EMR 7.8.0 (Spark) 的初始版本。

區域：`emr-emr-7.8.0-20250228` 適用於 Amazon EMR on EKS 支援的所有區域。如需詳細資訊，請參閱 Amazon EMR on EKS 服務端點。

容器映像標籤：`emr-7.8.0-20250228`

## emr-7.8.0-flink-latest

版本備註：`emr-7.8.0-flink-latest`目前指向 `emr-7.8.0-flink-20250228`

區域：`emr-7.8.0-flink-latest` 適用於 Amazon EMR on EKS 支援的所有區域。如需詳細資訊，請參閱 Amazon EMR on EKS 服務端點。

容器映像標籤：`emr-7.8.0-flink:latest`

## emr-7.8.0-flink-20250228

版本備註：`7.8.0-flink-20250228`已於 2025 年 2 月發行。這是 Amazon EMR 7.8.0 (Flink) 的初始版本。

區域：`emr-7.8.0-flink-20250228` 適用於 Amazon EMR on EKS 支援的所有區域。如需詳細資訊，請參閱 Amazon EMR on EKS 服務端點。

容器映像標籤：`emr-7.8.0-flink:20250228`

# Amazon EMR on EKS 7.7.0 版本

本頁面描述了針對 Amazon EMR on EKS 部署的 Amazon EMR 新功能和更新功能。如需在 Amazon EC2 上執行的 Amazon EMR 和一般 Amazon EMR 7.7.0 版本的詳細資訊，請參閱《Amazon EMR 版本指南》中的 Amazon EMR 7.7.0。

## Amazon EMR on EKS 7.7 版

下列 Amazon EMR 7.7.0 版本適用於 Amazon EMR on EKS。選取特定的 emr-7.7.0-XXXX 版本，以檢視更多詳細資訊，例如相關的容器映像標籤。

Flink releases

當您執行 Flink 應用程式時，Amazon EMR on EKS 可使用下列 Amazon EMR 7.7.0 版本。

- emr-7.7.0-flink-latest
- emr-7.7.0-flink-20250131

Spark releases

當您執行 Spark 應用程式時，Amazon EMR on EKS 可使用下列 Amazon EMR 7.7.0 版本。

- emr-7.7.0-latest
- emr-7.7.0-20250131
- emr-7.7.0-spark-rapids-latest
- emr-7.7.0-spark-rapids-20250131
- emr-7.7.0-java11-latest

- emr-7.7.0-java11-20250131
- emr-7.7.0-java8-latest
- emr-7.7.0-java8-20250131
- emr-7.7.0-spark-rapids-java8-latest
- emr-7.7.0-spark-rapids-java8-20250131
- notebook-spark/emr-7.7.0-latest
- notebook-spark/emr-7.7.0-20250131
- notebook-spark/emr-7.7.0-spark-rapids-latest
- notebook-spark/emr-7.7.0-spark-rapids-20250131
- notebook-spark/emr-7.7.0-java11-latest
- notebook-spark/emr-7.7.0-java11-20250131
- notebook-spark/emr-7.7.0-java8-latest
- notebook-spark/emr-7.7.0-java8-20250131
- notebook-spark/emr-7.7.0-spark-rapids-java8-latest
- notebook-spark/emr-7.7.0-spark-rapids-java8-20250131
- notebook-python/emr-7.7.0-latest
- notebook-python/emr-7.7.0-20250131
- notebook-python/emr-7.7.0-spark-rapids-latest
- notebook-python/emr-7.7.0-spark-rapids-20250131
- notebook-python/emr-7.7.0-java11-latest
- notebook-python/emr-7.7.0-java11-20250131
- notebook-python/emr-7.7.0-java8-latest
- notebook-python/emr-7.7.0-java8-20250131
- notebook-python/emr-7.7.0-spark-rapids-java8-latest
- notebook-python/emr-7.7.0-spark-rapids-java8-20250131
- livy/emr-7.7.0-latest
- livy/emr-7.7.0-20250131
- livy/emr-7.7.0-java11-latest
- livy/emr-7.7.0-java11-20250131

- livy/emr-7.7.0-java8-latest
- livy/emr-7.7.0-java8-20250131

## 版本備註

Amazon EMR on EKS 7.7.0 的版本備註

- 支援的應用程式：適用於 Java 的 AWS SDK 2.29.25 and 1.12.779, Apache Spark 3.5.3-amzn-0, Apache Hudi 0.15.0-amzn-3, Apache Iceberg 1.6.1-amzn-2, Delta 3.2.1-amzn-1, Apache Spark RAPIDS 24.10.1-amzn-0, Jupyter Enterprise Gateway 2.6.0, Apache Flink 1.20.0-amzn-0, Flink Operator 1.10.0-amzn-0
- 支援的元件 - `emr-ddb`、`emr-goodies`、`emr-s3-select`、`emrfs`、`hadoop-clienthudi`、`hudi-spark`、`iceberg`、、`spark-kubernetes`。
- 支援的組態分類

  若要與 StartJobRun 和 CreateManagedEndpoint API 搭配使用：

| 分類 | 描述 |
| --- | --- |
| `core-site` | 變更 `core-site.xml` Hadoop 檔案中的值。 |
| `emrfs-site` | 變更 EMRFS 設定。 |
| `spark-metrics` | 變更 `metrics.properties` Spark 檔案中的值。 |
| `spark-defaults` | 變更 `spark-defaults.conf` Spark 檔案中的值。 |
| `spark-env` | 變更 Spark 環境中的值。 |
| `spark-hive-site` | 變更 `hive-site.xml` Spark 檔案中的值。 |
| `spark-log4j2` | 變更 `log4j2.properties` Spark 檔案中的值。 |
| `emr-job-submitter` | 作業提交者 Pod 的組態。 |

專門搭配 CreateManagedEndpoint API 使用：

| 分類 | 描述 |
| --- | --- |
| jeg-config | 變更 Jupyter Enterprise Gateway jupyter_enterprise_gateway_config.py 檔案中的值。 |
| jupyter-kernel-overrides | 在 Jupyter 核心規格檔案中變更核心映像的值。 |

組態分類可讓您自訂應用程式。這些檔案通常對應於應用程式的組態 XML 檔案，例如 spark-hive-site.xml。如需詳細資訊，請參閱設定應用程式。

## 變更

Amazon EMR on EKS 7.7.0 版包含下列變更：

- 截至 EMR 7.7.0 使用的 Iceberg 版本不再支援 Java 8。此外，Iceberg 會從下列 Java 8 映像中排除： emr-7.7.0-java8-latest和 emr-7.7.0-spark-rapids-java8-latest。

## emr-7.7.0-latest

版本說明：emr-7.7.0-latest 目前指向 emr-7.7.0-20250131。

區域：emr-7.7.0-latest 適用於 Amazon EMR on EKS 支援的所有區域。如需詳細資訊，請參閱 Amazon EMR on EKS 服務端點。

容器映像標籤：emr-7.7.0:latest

## emr-7.7.0-20250131

版本備註：emr-7.7.0-20250131已於 2025 年 2 月發行。這是 Amazon EMR 7.7.0 (Spark) 的初始版本。

區域：emr-emr-7.7.0-20250131 適用於 Amazon EMR on EKS 支援的所有區域。如需詳細資訊，請參閱 Amazon EMR on EKS 服務端點。

容器映像標籤：`emr-7.7.0-20250131`

### emr-7.7.0-flink-latest

版本備註：`emr-7.7.0-flink-latest`目前指向 `emr-7.7.0-flink-20250131`

區域：`emr-7.7.0-flink-latest` 適用於 Amazon EMR on EKS 支援的所有區域。如需詳細資訊，請參閱 Amazon EMR on EKS 服務端點。

容器映像標籤：`emr-7.7.0-flink:latest`

### emr-7.7.0-flink-20250131

版本備註：`7.7.0-flink-20250131`已於 2025 年 2 月發行。這是 Amazon EMR 7.7.0 (Flink) 的初始版本。

區域：`emr-7.7.0-flink-20250131` 適用於 Amazon EMR on EKS 支援的所有區域。如需詳細資訊，請參閱 Amazon EMR on EKS 服務端點。

容器映像標籤：`emr-7.7.0-flink:20250131`

## Amazon EMR on EKS 7.6.0 版

本頁面描述了針對 Amazon EMR on EKS 部署的 Amazon EMR 新功能和更新功能。如需在 Amazon EC2 上執行的 Amazon EMR 和一般 Amazon EMR 7.6.0 版本的詳細資訊，請參閱《Amazon EMR 版本指南》中的 Amazon EMR 7.6.0。

### Amazon EMR on EKS 7.6 版

下列 Amazon EMR 7.6.0 版本適用於 Amazon EMR on EKS。選取特定的 emr-7.6.0-XXXX 版本，以檢視更多詳細資訊，例如相關的容器映像標籤。

Flink releases

當您執行 Flink 應用程式時，下列 Amazon EMR 7.6.0 版本可供 Amazon EMR on EKS 使用。

- emr-7.6.0-flink-latest
- emr-7.6.0-flink-20241213

Spark releases

當您執行 Spark 應用程式時，Amazon EMR on EKS 可使用下列 Amazon EMR 7.6.0 版本。

- emr-7.6.0-latest
- emr-7.6.0-20241213
- emr-7.6.0-spark-rapids-latest
- emr-7.6.0-spark-rapids-20241213
- emr-7.6.0-java11-latest
- emr-7.6.0-java11-20241213
- emr-7.6.0-java8-latest
- emr-7.6.0-java8-20241213
- emr-7.6.0-spark-rapids-java8-latest
- emr-7.6.0-spark-rapids-java8-20241213
- notebook-spark/emr-7.6.0-latest
- notebook-spark/emr-7.6.0-20241213
- notebook-spark/emr-7.6.0-spark-rapids-latest
- notebook-spark/emr-7.6.0-spark-rapids-20241213
- notebook-spark/emr-7.6.0-java11-latest
- notebook-spark/emr-7.6.0-java11-20241213
- notebook-spark/emr-7.6.0-java8-latest
- notebook-spark/emr-7.6.0-java8-20241213
- notebook-spark/emr-7.6.0-spark-rapids-java8-latest
- notebook-spark/emr-7.6.0-spark-rapids-java8-20241213
- notebook-python/emr-7.6.0-latest
- notebook-python/emr-7.6.0-20241213
- notebook-python/emr-7.6.0-spark-rapids-latest
- notebook-python/emr-7.6.0-spark-rapids-20241213
- notebook-python/emr-7.6.0-java11-latest
- notebook-python/emr-7.6.0-java11-20241213

- notebook-python/emr-7.6.0-java8-latest
- notebook-python/emr-7.6.0-java8-20241213
- notebook-python/emr-7.6.0-spark-rapids-java8-latest
- notebook-python/emr-7.6.0-spark-rapids-java8-20241213
- livy/emr-7.6.0-latest
- livy/emr-7.6.0-20241213
- livy/emr-7.6.0-java11-latest
- livy/emr-7.6.0-java11-20241213
- livy/emr-7.6.0-java8-latest
- livy/emr-7.6.0-java8-20241213

## 版本備註

Amazon EMR on EKS 7.6.0 的版本備註

- 支援的應用程式：適用於 Java 的 AWS SDK 2.29.25 and 1.12.779, Apache Spark 3.5.3-amzn-0, Apache Hudi 0.15.0-amzn-3, Apache Iceberg 1.6.1-amzn-2, Delta 3.2.1-amzn-1, Apache Spark RAPIDS 24.10.1-amzn-0, Jupyter Enterprise Gateway 2.6.0, Apache Flink 1.20.0-amzn-0, Flink Operator 1.10.0-amzn-0
- 支援的元件 - `aws-sagemaker-spark-sdk`、`emr-ddb`、`emr-goodies`、`emr-s3-select`、`emrfs`、`hadoop-client`、`hudi`、`hudi-spark`、`iceberg`、`spark-kubernetes`。
- 支援的組態分類

  若要與 StartJobRun 和 CreateManagedEndpoint API 搭配使用：

| 分類 | 描述 |
| --- | --- |
| `core-site` | 變更 `core-site.xml` Hadoop 檔案中的值。 |
| `emrfs-site` | 變更 EMRFS 設定。 |
| `spark-metrics` | 變更 `metrics.properties` Spark 檔案中的值。 |

| 分類 | 描述 |
| --- | --- |
| spark-defaults | 變更 spark-defaults.conf Spark 檔案中的值。 |
| spark-env | 變更 Spark 環境中的值。 |
| spark-hive-site | 變更 hive-site.xml Spark 檔案中的值。 |
| spark-log4j2 | 變更 log4j2.properties Spark 檔案中的值。 |
| emr-job-submitter | 作業提交者 Pod 的組態。 |

專門搭配 CreateManagedEndpoint API 使用：

| 分類 | 描述 |
| --- | --- |
| jeg-config | 變更 Jupyter Enterprise Gateway jupyter_enterprise_gateway_config.py 檔案中的值。 |
| jupyter-kernel-overrides | 在 Jupyter 核心規格檔案中變更核心映像的值。 |

組態分類可讓您自訂應用程式。這些檔案通常對應於應用程式的組態 XML 檔案，例如 spark-hive-site.xml。如需詳細資訊，請參閱設定應用程式。

## 值得注意的功能

Amazon EMR on EKS 7.6.0 版包含下列功能：

- 監控 Apache Spark Operator 的組態支援 – 監控組態可讓您輕鬆地將 Spark 應用程式和運算子日誌的日誌封存設定到 Amazon S3 或 Amazon CloudWatch。您可以選擇其中一個或兩者。這樣做會將日誌代理程式附屬項目新增至 Spark 運算子 Pod、驅動程式和執行器 Pod，然後將這些元件的日誌轉送到您設定的接收器。如需詳細資訊，請參閱使用監控組態來監控 Spark Kubernetes 運算子和 Spark 任務。

# 變更

Amazon EMR on EKS 7.6.0 版包含下列變更：

- 版本沒有變更。

## emr-7.6.0-latest

版本說明：`emr-7.6.0-latest` 目前指向 `emr-7.6.0-20241213`。

區域：`emr-7.6.0-latest` 適用於 Amazon EMR on EKS 支援的所有區域。如需詳細資訊，請參閱 Amazon EMR on EKS 服務端點。

容器映像標籤：`emr-7.6.0:latest`

## emr-7.6.0-20241213

版本備註：`7.6.0-20241213`已於 2024 年 1 月發行。這是 Amazon EMR 7.6.0 (Spark) 的初始版本。

區域：`emr-7.6.0-20241213` 適用於 Amazon EMR on EKS 支援的所有區域。如需詳細資訊，請參閱 Amazon EMR on EKS 服務端點。

容器映像標籤：`emr-7.6.0:20241213`

## emr-7.6.0-flink-latest

版本備註：`emr-7.6.0-flink-latest`目前指向 `emr-7.6.0-flink-20241213`

區域：`emr-7.6.0-flink-latest` 適用於 Amazon EMR on EKS 支援的所有區域。如需詳細資訊，請參閱 Amazon EMR on EKS 服務端點。

容器映像標籤：`emr-7.6.0-flink:latest`

## emr-7.6.0-flink-20241213

版本備註：`7.6.0-flink-20241213`已於 2024 年 1 月發行。這是 Amazon EMR 7.6.0 (Flink) 的初始版本。

區域：`emr-7.6.0-flink-20241213` 適用於 Amazon EMR on EKS 支援的所有區域。如需詳細資訊，請參閱 Amazon EMR on EKS 服務端點。

容器映像標籤：`emr-7.6.0-flink:20241213`

# Amazon EMR on EKS 7.5.0 版

本頁面描述了針對 Amazon EMR on EKS 部署的 Amazon EMR 新功能和更新功能。如需在 Amazon EC2 上執行的 Amazon EMR 和一般 Amazon EMR 7.5.0 版本的詳細資訊，請參閱《Amazon EMR 版本指南》中的 Amazon EMR 7.5.0。

## Amazon EMR on EKS 7.5 版

下列 Amazon EMR 7.5.0 版本適用於 Amazon EMR on EKS。選取特定的 emr-7.5.0-XXXX 版本，以檢視更多詳細資訊，例如相關的容器映像標籤。

### 版本備註

Amazon EMR on EKS 7.5.0 的版本備註

- 支援的應用程式：適用於 Java 的 AWS SDK 2.28.8 and 1.12.772, Apache Spark 3.5.2-amzn-1, Apache Hudi 0.15.0-amzn-1, Apache Iceberg 1.6.1-amzn-0, Delta 3.2.0-amzn-1, Apache Spark RAPIDS 24.08.1-amzn-1, Jupyter Enterprise Gateway 2.6.0, Apache Flink 1.19.1-amzn-1, Flink Operator 1.9.0-amzn-0
- 支援的元件 - `aws-sagemaker-spark-sdk`、`emr-ddb`、`emr-goodies`、`emr-s3-select`、`emrfs`、`hadoop-client`、`hudi`、`hudi-spark`、`iceberg`、`spark-kubernetes`。

# Amazon EMR on EKS 7.4.0 版本

本頁面描述了針對 Amazon EMR on EKS 部署的 Amazon EMR 新功能和更新功能。如需在 Amazon EC2 上執行的 Amazon EMR 和一般 Amazon EMR 7.4.0 版本的詳細資訊，請參閱《Amazon EMR 版本指南》中的 Amazon EMR 7.4.0。

## Amazon EMR on EKS 7.4 版

下列 Amazon EMR 7.4.0 版本適用於 Amazon EMR on EKS。選取特定的 emr-7.4.0-XXXX 版本，以檢視更多詳細資訊，例如相關的容器映像標籤。

### 版本備註

Amazon EMR on EKS 7.4.0 的版本備註

- 支援的應用程式：適用於 Java 的 AWS SDK 2.25.70 and 1.12.772, Apache Spark 3.5.2-amzn-0, Apache Hudi 0.15.0-amzn-1, Apache Iceberg 1.6.1-amzn-0, Delta 3.2.0-amzn-1, Apache Spark RAPIDS 24.08.1-amzn-0, Jupyter Enterprise Gateway 2.6.0, Apache Flink 1.19.1-amzn-0, Flink Operator 1.9.0-amzn-1
- 支援的元件 - `aws-sagemaker-spark-sdk`、`emr-ddb`、`emr-goodies`、`emr-s3-select`、`emrfs`、`hadoop-client`、`hudi`、`hudi-spark`、`iceberg`、`spark-kubernetes`。

# Amazon EMR on EKS 7.3.0 版本

本頁面描述了針對 Amazon EMR on EKS 部署的 Amazon EMR 新功能和更新功能。如需在 Amazon EC2 上執行的 Amazon EMR 和一般 Amazon EMR 7.3.0 版本的詳細資訊，請參閱《Amazon EMR 版本指南》中的 Amazon EMR 7.3.0。

## Amazon EMR on EKS 7.3 版

下列 Amazon EMR 7.3.0 版本適用於 Amazon EMR on EKS。選取特定的 emr-7.3.0-XXXX 版本，以檢視更多詳細資訊，例如相關的容器映像標籤。

Flink releases

當您執行 Flink 應用程式時，Amazon EMR on EKS 可使用下列 Amazon EMR 7.3.0 版本。

- emr-7.3.0-flink-latest
- emr-7.3.0-flink-29240920

Spark releases

當您執行 Spark 應用程式時，Amazon EMR on EKS 可使用下列 Amazon EMR 7.3.0 版本。

- emr-7.3.0-latest
- emr-7.3.0-29240920
- emr-7.3.0-spark-rapids-latest
- emr-7.3.0-spark-rapids-29240920
- emr-7.3.0-java11-latest
- emr-7.3.0-java11-29240920
- emr-7.3.0-java8-latest

- emr-7.3.0-java8-29240920
- emr-7.3.0-spark-rapids-java8-latest
- emr-7.3.0-spark-rapids-java8-29240920
- notebook-spark/emr-7.3.0-latest
- notebook-spark/emr-7.3.0-29240920
- notebook-spark/emr-7.3.0-spark-rapids-latest
- notebook-spark/emr-7.3.0-spark-rapids-29240920
- notebook-spark/emr-7.3.0-java11-latest
- notebook-spark/emr-7.3.0-java11-29240920
- notebook-spark/emr-7.3.0-java8-latest
- notebook-spark/emr-7.3.0-java8-29240920
- notebook-spark/emr-7.3.0-spark-rapids-java8-latest
- notebook-spark/emr-7.3.0-spark-rapids-java8-29240920
- notebook-python/emr-7.3.0-latest
- notebook-python/emr-7.3.0-29240920
- notebook-python/emr-7.3.0-spark-rapids-latest
- notebook-python/emr-7.3.0-spark-rapids-29240920
- notebook-python/emr-7.3.0-java11-latest
- notebook-python/emr-7.3.0-java11-29240920
- notebook-python/emr-7.3.0-java8-latest
- notebook-python/emr-7.3.0-java8-29240920
- notebook-python/emr-7.3.0-spark-rapids-java8-latest
- notebook-python/emr-7.3.0-spark-rapids-java8-29240920
- livy/emr-7.3.0-latest
- livy/emr-7.3.0-29240920
- livy/emr-7.3.0-java11-latest
- livy/emr-7.3.0-java11-29240920
- livy/emr-7.3.0-java8-latest
- livy/emr-7.3.0-java8-29240920

# 版本備註

Amazon EMR on EKS 7.3.0 的版本備註

- 支援的應用程式：適用於 Java 的 AWS SDK 2.25.70 and 1.12.747, Apache Spark 3.5.1-amzn-1, Apache Hudi 0.15.0-amzn-0, Apache Iceberg 1.5.2-amzn-0, Delta 3.2.0-amzn-0, Apache Spark RAPIDS 24.06.1-amzn-0, Jupyter Enterprise Gateway 2.6.0, Apache Flink 1.18.1-amzn-2, Flink Operator 1.9.0-amzn-0
- 支援的元件 - `aws-sagemaker-spark-sdk`、`emr-ddb`、`emr-goodies`、`emr-s3-select`、`emrfs`、`hadoop-client`、`hudi`、`hudi-spark`、`iceberg`、`spark-kubernetes`。
- 支援的組態分類

  若要與 StartJobRun 和 CreateManagedEndpoint API 搭配使用：

| 分類 | 描述 |
| --- | --- |
| `core-site` | 變更 `core-site.xml` Hadoop 檔案中的值。 |
| `emrfs-site` | 變更 EMRFS 設定。 |
| `spark-metrics` | 變更 `metrics.properties` Spark 檔案中的值。 |
| `spark-defaults` | 變更 `spark-defaults.conf` Spark 檔案中的值。 |
| `spark-env` | 變更 Spark 環境中的值。 |
| `spark-hive-site` | 變更 `hive-site.xml` Spark 檔案中的值。 |
| `spark-log4j2` | 變更 `log4j2.properties` Spark 檔案中的值。 |
| `emr-job-submitter` | 作業提交者 Pod 的組態。 |

  專門搭配 CreateManagedEndpoint API 使用：

| 分類 | 描述 |
| --- | --- |
| `jeg-config` | 變更 Jupyter Enterprise Gateway `jupyter_enterprise_gateway_config.py` 檔案中的值。 |
| `jupyter-kernel-overrides` | 在 Jupyter 核心規格檔案中變更核心映像的值。 |

組態分類可讓您自訂應用程式。這些檔案通常對應於應用程式的組態 XML 檔案，例如 `spark-hive-site.xml`。如需詳細資訊，請參閱設定應用程式。

## 值得注意的功能

Amazon EMR on EKS 7.3.0 版包含下列功能。

- 應用程式升級 – Amazon EMR on EKS 現在包含 Flink Operator 1.9.0。除了其他功能之外，Flink Kubernetes 現在可讓您設定自動擴展器的 CPU 和記憶體配額。
- Apache Iceberg 支援 Apache Flink – Apache Iceberg 是一種開放原始碼的高效能格式巨型分析資料表。從 Amazon EMR 7.3.0 開始，您可以在 Amazon EMR on EKS 上執行 Apache Flink 時使用 Apache Iceberg 資料表。如需詳細資訊，請參閱使用 Apache Iceberg 搭配 Amazon EMR on EKS 的 Amazon EMR on EKS。
- Delta Lake 對 Apache Flink 的支援 – Delta Lake 是通常建置在 Amazon S3 上的湖房架構的儲存層架構。透過 Amazon EMR 7.3.0 及更高版本，您可以在 Amazon EMR on EKS 上執行 Apache Flink 時使用 Delta 資料表。如需詳細資訊，請參閱搭配使用 Delta Lake 與 Amazon EMR on EKS。

## 變更

Amazon EMR on EKS 7.3.0 版包含下列變更。

- 使用 Amazon EMR on EKS 7.3.0 及更高版本時，Apache Flink 現在預設會使用 Java 17 執行時間。

## emr-7.3.0-latest

版本說明：`emr-7.3.0-latest` 目前指向 `emr-7.3.0-29240920`。

區域：`emr-7.3.0-latest` 適用於 Amazon EMR on EKS 支援的所有區域。如需詳細資訊，請參閱 Amazon EMR on EKS 服務端點。

容器映像標籤：`emr-7.3.0:latest`

### emr-7.3.0-29240920

版本備註：已於 2023 年 12 月發行 `7.3.0-29240920`。這是 Amazon EMR 7.3.0 (Spark) 的初始版本。

區域：`emr-7.3.0-29240920` 適用於 Amazon EMR on EKS 支援的所有區域。如需詳細資訊，請參閱 Amazon EMR on EKS 服務端點。

容器映像標籤：`emr-7.3.0:29240920`

### emr-7.3.0-flink-latest

版本說明：`emr-7.3.0-flink-latest` 目前指向 `emr-7.3.0-flink-29240920`。

區域：`emr-7.3.0-flink-latest` 適用於 Amazon EMR on EKS 支援的所有區域。如需詳細資訊，請參閱 Amazon EMR on EKS 服務端點。

容器映像標籤：`emr-7.3.0-flink:latest`

### emr-7.3.0-flink-29240920

版本備註：已於 2023 年 12 月發行 `7.3.0-flink-29240920`。這是 Amazon EMR 7.3.0 (Flink) 的初始版本。

區域：`emr-7.3.0-flink-29240920` 適用於 Amazon EMR on EKS 支援的所有區域。如需詳細資訊，請參閱 Amazon EMR on EKS 服務端點。

容器映像標籤：`emr-7.3.0-flink:29240920`

## Amazon EMR on EKS 7.2.0 版本

本頁面描述了針對 Amazon EMR on EKS 部署的 Amazon EMR 新功能和更新功能。如需在 Amazon EC2 上執行的 Amazon EMR 和一般 Amazon EMR 7.2.0 版本的詳細資訊，請參閱《Amazon EMR 版本指南》中的 Amazon EMR 7.2.0。

# Amazon EMR on EKS 7.2 版

下列 Amazon EMR 7.2.0 版本適用於 Amazon EMR on EKS。選取特定的 emr-7.2.0-XXXX 版本，以檢視更多詳細資訊，例如相關的容器映像標籤。

Flink releases

當您執行 Flink 應用程式時，下列 Amazon EMR 7.2.0 版本可供 Amazon EMR on EKS 使用。

- emr-7.2.0-flink-latest
- emr-7.2.0-flink-20240610

Spark releases

當您執行 Spark 應用程式時，Amazon EMR on EKS 可使用下列 Amazon EMR 7.2.0 版本。

- emr-7.2.0-latest
- emr-7.2.0-20240610
- emr-7.2.0-spark-rapids-latest
- emr-7.2.0-spark-rapids-20240610
- emr-7.2.0-java11-latest
- emr-7.2.0-java11-20240610
- emr-7.2.0-java8-latest
- emr-7.2.0-java8-20240610
- emr-7.2.0-spark-rapids-java8-latest
- emr-7.2.0-spark-rapids-java8-20240610
- notebook-spark/emr-7.2.0-latest
- notebook-spark/emr-7.2.0-20240610
- notebook-spark/emr-7.2.0-spark-rapids-latest
- notebook-spark/emr-7.2.0-spark-rapids-20240610
- notebook-spark/emr-7.2.0-java11-latest
- notebook-spark/emr-7.2.0-java11-20240610
- notebook-spark/emr-7.2.0-java8-latest
- notebook-spark/emr-7.2.0-java8-20240610

- notebook-spark/emr-7.2.0-spark-rapids-java8-latest
- notebook-spark/emr-7.2.0-spark-rapids-java8-20240610
- notebook-python/emr-7.2.0-latest
- notebook-python/emr-7.2.0-20240610
- notebook-python/emr-7.2.0-spark-rapids-latest
- notebook-python/emr-7.2.0-spark-rapids-20240610
- notebook-python/emr-7.2.0-java11-latest
- notebook-python/emr-7.2.0-java11-20240610
- notebook-python/emr-7.2.0-java8-latest
- notebook-python/emr-7.2.0-java8-20240610
- notebook-python/emr-7.2.0-spark-rapids-java8-latest
- notebook-python/emr-7.2.0-spark-rapids-java8-20240610
- livy/emr-7.2.0-latest
- livy/emr-7.2.0-20240610
- livy/emr-7.2.0-java11-latest
- livy/emr-7.2.0-java11-20240610
- livy/emr-7.2.0-java8-latest
- livy/emr-7.2.0-java8-20240610

## 版本備註

Amazon EMR on EKS 7.2.0 的版本備註

- 支援的應用程式：適用於 Java 的 AWS SDK 2.23.18 and 1.12.705, Apache Spark 3.5.1-amzn-1, Apache Hudi 0.14.1-amzn-0, Apache Iceberg 1.5.0-amzn-0, Delta 3.1.0, Apache Spark RAPIDS 24.02.0-amzn-1, Jupyter Enterprise Gateway 2.6.0, Apache Flink 1.18.1-amzn-0, Flink Operator 1.8.0-amzn-1
- 支援的元件 - `aws-sagemaker-spark-sdk`、`emr-ddb`、`emr-goodies`、`emr-s3-select`、`emrfs`、`hadoop-client`、`hudi`、`hudi-spark`、`iceberg`、`spark-kubernetes`。
- 支援的組態分類

  若要與 StartJobRun 和 CreateManagedEndpoint API 搭配使用：

| 分類 | 描述 |
| --- | --- |
| core-site | 變更 core-site.xml Hadoop 檔案中的值。 |
| emrfs-site | 變更 EMRFS 設定。 |
| spark-metrics | 變更 metrics.properties Spark 檔案中的值。 |
| spark-defaults | 變更 spark-defaults.conf Spark 檔案中的值。 |
| spark-env | 變更 Spark 環境中的值。 |
| spark-hive-site | 變更 hive-site.xml Spark 檔案中的值。 |
| spark-log4j2 | 變更 log4j2.properties Spark 檔案中的值。 |
| emr-job-submitter | 作業提交者 Pod 的組態。 |

專門搭配 CreateManagedEndpoint API 使用：

| 分類 | 描述 |
| --- | --- |
| jeg-config | 變更 Jupyter Enterprise Gateway jupyter_enterprise_gateway_config.py 檔案中的值。 |
| jupyter-kernel-overrides | 在 Jupyter 核心規格檔案中變更核心映像的值。 |

組態分類可讓您自訂應用程式。這些檔案通常對應於應用程式的組態 XML 檔案，例如 spark-hive-site.xml。如需詳細資訊，請參閱設定應用程式。

## 值得注意的功能

Amazon EMR on EKS 7.2.0 版包含下列功能。

- 應用程式升級 – Amazon EMR on EKS 7.2.0 應用程式升級包括 Spark 3.5.1、Flink 1.18.1 和 Flink Operator 1.8.0。
- Flink 更新的自動擴展器 – 7.2.0 版本使用開放原始碼組態`job.autoscaler.restart.time-tracking.enabled`來啟用重新擴展時間估算，因此您不再需要手動指派經驗值來重新啟動時間。如果您執行 7.1.0 或更低版本，您仍然可以使用 Amazon EMR Autoscaling。
- Apache Hudi 整合 Apache Flink on Amazon EMR on EKS – 此版本新增了 Apache Hudi 和 Apache Flink 之間的整合，因此您可以使用 Flink Kubernetes Operator 來執行 Hudi 任務。Hudi 可讓您使用記錄層級操作，以簡化資料管理和資料管道開發。
- Amazon S3 Express One Zone 與 Amazon EMR on EKS 整合 – 透過 7.2.0 及更高版本，您可以使用 Amazon EMR on EKS 將資料上傳至 S3 Express One Zone。S3 Express One Zone 是一種高效能的單一區域 Amazon S3 儲存類別，可為對大多數延遲敏感的應用程式提供一致的單一位數毫秒資料存取。在發布時，S3 Express One Zone 提供 Amazon S3 中最低延遲和最高效能的雲端物件儲存。
- Spark 運算子中預設組態的支援 – Amazon EKS 上的 Spark 運算子現在支援與 Amazon EMR on EKS 7.2.0 及更高版本的啟動任務執行模型相同的預設組態。這表示 Amazon S3 和 EMRFS 等功能不再需要 yaml 檔案中的手動組態。

## emr-7.2.0-latest

版本說明：`emr-7.2.0-latest` 目前指向 `emr-7.2.0-20240610`。

區域：`emr-7.2.0-latest` 適用於 Amazon EMR on EKS 支援的所有區域。如需詳細資訊，請參閱 Amazon EMR on EKS 服務端點。

容器映像標籤：`emr-7.2.0:latest`

## emr-7.2.0-20240610

版本備註：已於 2023 年 12 月發行 `7.2.0-20240610`。這是 Amazon EMR 7.2.0 (Spark) 的初始版本。

區域：`emr-7.2.0-20240610` 適用於 Amazon EMR on EKS 支援的所有區域。如需詳細資訊，請參閱 Amazon EMR on EKS 服務端點。

容器映像標籤：`emr-7.2.0:20240610`

### emr-7.2.0-flink-latest

版本說明：`emr-7.2.0-flink-latest` 目前指向 `emr-7.2.0-flink-20240610`。

區域：`emr-7.2.0-flink-latest` 適用於 Amazon EMR on EKS 支援的所有區域。如需詳細資訊，請參閱 Amazon EMR on EKS 服務端點。

容器映像標籤：`emr-7.2.0-flink:latest`

### emr-7.2.0-flink-20240610

版本備註：已於 2023 年 12 月發行 `7.2.0-flink-20240610`。這是 Amazon EMR 7.2.0 (Flink) 的初始版本。

區域：`emr-7.2.0-flink-20240610` 適用於 Amazon EMR on EKS 支援的所有區域。如需詳細資訊，請參閱 Amazon EMR on EKS 服務端點。

容器映像標籤：`emr-7.2.0-flink:20240610`

## Amazon EMR on EKS 7.1.0 版

本頁面描述了針對 Amazon EMR on EKS 部署的 Amazon EMR 新功能和更新功能。如需在 Amazon EC2 上執行的 Amazon EMR 和一般 Amazon EMR 7.1.0 版本的詳細資訊，請參閱《Amazon EMR 版本指南》中的 Amazon EMR 7.1.0。

### Amazon EMR on EKS 7.1 版

下列 Amazon EMR 7.1.0 版本適用於 Amazon EMR on EKS。選取特定的 emr-7.1.0-XXXX 版本，以檢視更多詳細資訊，例如相關的容器映像標籤。

Flink releases

當您執行 Flink 應用程式時，Amazon EMR on EKS 可使用下列 Amazon EMR 7.1.0 版本。

- emr-7.1.0-flink-latest
- emr-7.1.0-flink-20240321

Spark releases

當您執行 Spark 應用程式時，Amazon EMR on EKS 可使用下列 Amazon EMR 7.1.0 版本。

- emr-7.1.0-latest
- emr-7.1.0-20240321
- emr-7.1.0-spark-rapids-latest
- emr-7.1.0-spark-rapids-20240321
- emr-7.1.0-java11-latest
- emr-7.1.0-java11-20240321
- emr-7.1.0-java8-latest
- emr-7.1.0-java8-20240321
- emr-7.1.0-spark-rapids-java8-latest
- emr-7.1.0-spark-rapids-java8-20240321
- notebook-spark/emr-7.1.0-latest
- notebook-spark/emr-7.1.0-20240321
- notebook-spark/emr-7.1.0-spark-rapids-latest
- notebook-spark/emr-7.1.0-spark-rapids-20240321
- notebook-spark/emr-7.1.0-java11-latest
- notebook-spark/emr-7.1.0-java11-20240321
- notebook-spark/emr-7.1.0-java8-latest
- notebook-spark/emr-7.1.0-java8-20240321
- notebook-spark/emr-7.1.0-spark-rapids-java8-latest
- notebook-spark/emr-7.1.0-spark-rapids-java8-20240321
- notebook-python/emr-7.1.0-latest
- notebook-python/emr-7.1.0-20240321
- notebook-python/emr-7.1.0-spark-rapids-latest
- notebook-python/emr-7.1.0-spark-rapids-20240321
- notebook-python/emr-7.1.0-java11-latest
- notebook-python/emr-7.1.0-java11-20240321
- notebook-python/emr-7.1.0-java8-latest
- notebook-python/emr-7.1.0-java8-20240321

- notebook-python/emr-7.1.0-spark-rapids-java8-latest
- notebook-python/emr-7.1.0-spark-rapids-java8-20240321
- livy/emr-7.1.0-latest
- livy/emr-7.1.0-20240321
- livy/emr-7.1.0-java11-latest
- livy/emr-7.1.0-java11-20240321
- livy/emr-7.1.0-java8-latest
- livy/emr-7.1.0-java8-20240321

## 版本備註

Amazon EMR on EKS 7.1.0 的版本備註

- 支援的應用程式：適用於 Java 的 AWS SDK 2.23.18 and 1.12.656, Apache Spark 3.5.0-amzn-1, Apache Hudi 0.14.1-amzn-0, Apache Iceberg 1.4.3-amzn-0, Delta 3.0.0, Apache Spark RAPIDS 23.10.0-amzn-1, Jupyter Enterprise Gateway 2.6.0, Apache Flink 1.18.1-amzn-0, Flink Operator 1.6.1-amzn-1
- 支援的元件 - `aws-sagemaker-spark-sdk`、`emr-ddb`、`emr-goodies`、`emr-s3-select`、`emrfs`、`hadoop-client`、`hudi`、`hudi-spark`、`iceberg`、`spark-kubernetes`。
- 支援的組態分類

  若要與 StartJobRun 和 CreateManagedEndpoint API 搭配使用：

| 分類 | 描述 |
| --- | --- |
| `core-site` | 變更 `core-site.xml` Hadoop 檔案中的值。 |
| `emrfs-site` | 變更 EMRFS 設定。 |
| `spark-metrics` | 變更 `metrics.properties` Spark 檔案中的值。 |
| `spark-defaults` | 變更 `spark-defaults.conf` Spark 檔案中的值。 |

| 分類 | 描述 |
| --- | --- |
| spark-env | 變更 Spark 環境中的值。 |
| spark-hive-site | 變更 hive-site.xml Spark 檔案中的值。 |
| spark-log4j2 | 變更 log4j2.properties Spark 檔案中的值。 |
| emr-job-submitter | 作業提交者 Pod 的組態。 |

專門搭配 CreateManagedEndpoint API 使用：

| 分類 | 描述 |
| --- | --- |
| jeg-config | 變更 Jupyter Enterprise Gateway jupyter_enterprise_gateway_config.py 檔案中的值。 |
| jupyter-kernel-overrides | 在 Jupyter 核心規格檔案中變更核心映像的值。 |

組態分類可讓您自訂應用程式。這些檔案通常對應於應用程式的組態 XML 檔案，例如 spark-hive-site.xml。如需詳細資訊，請參閱設定應用程式。

## 值得注意的功能

Amazon EMR on EKS 7.1.0 版包含下列功能。

- Amazon EMR on EKS 的 Apache Livy 支援 – 透過 Amazon EMR on EKS 7.1.0 版及更高版本，您可以在 Amazon EKS 叢集上使用 Apache Livy 來建立 Apache Livy REST 介面，以提交 Spark 任務或 Spark 程式碼片段。這樣做可讓您同步和非同步擷取結果，同時仍然利用 Amazon EMR on EKS 優點，例如 Amazon EMR 最佳化 Spark 執行期、啟用 SSL 的 Livy 端點，以及程式設計設定體驗。

## emr-7.1.0-latest

版本說明：`emr-7.1.0-latest` 目前指向 `emr-7.1.0-20240321`。

區域：`emr-7.1.0-latest` 適用於 Amazon EMR on EKS 支援的所有區域。如需詳細資訊，請參閱 Amazon EMR on EKS 服務端點。

容器映像標籤：`emr-7.1.0:latest`

## emr-7.1.0-20240321

版本備註：已於 2023 年 12 月發行 `7.1.0-20240321`。這是 Amazon EMR 7.1.0 (Spark) 的初始版本。

區域：`emr-7.1.0-20240321` 適用於 Amazon EMR on EKS 支援的所有區域。如需詳細資訊，請參閱 Amazon EMR on EKS 服務端點。

容器映像標籤：`emr-7.1.0:20240321`

## emr-7.1.0-flink-latest

版本說明：`emr-7.1.0-flink-latest` 目前指向 `emr-7.1.0-flink-20240321`。

區域：`emr-7.1.0-flink-latest` 適用於 Amazon EMR on EKS 支援的所有區域。如需詳細資訊，請參閱 Amazon EMR on EKS 服務端點。

容器映像標籤：`emr-7.1.0-flink:latest`

## emr-7.1.0-flink-20240321

版本備註：已於 2023 年 12 月發行 `7.1.0-flink-20240321`。這是 Amazon EMR 7.1.0 (Flink) 的初始版本。

區域：`emr-7.1.0-flink-20240321` 適用於 Amazon EMR on EKS 支援的所有區域。如需詳細資訊，請參閱 Amazon EMR on EKS 服務端點。

容器映像標籤：`emr-7.1.0-flink:20240321`

# Amazon EMR on EKS 7.0.0 版

本頁面描述了針對 Amazon EMR on EKS 部署的 Amazon EMR 新功能和更新功能。如需有關在 Amazon EC2 上執行的 Amazon EMR 以及一般 Amazon EMR 7.0.0 版的詳細資訊，請參閱《Amazon EMR 版本指南》中的 Amazon EMR 7.0.0。

## Amazon EMR on EKS 7.0 版

以下 Amazon EMR 7.0.0 版可用於 Amazon EMR on EKS。選取特定的 emr-7.0.0-XXXX 版本，以檢視更多詳細資訊，例如相關的容器映像標籤。

Flink releases

當您執行 Flink 應用程式時，以下 Amazon EMR 7.0.0 版可用於 Amazon EMR on EKS。

- emr-7.0.0-flink-latest
- emr-7.0.0-flink-2024321
- emr-7.0.0-flink-20231211

Spark releases

當您執行 Spark 應用程式時，以下 Amazon EMR 7.0.0 版可用於 Amazon EMR on EKS。

- emr-7.0.0-latest
- emr-7.0.0-20231211
- emr-7.0.0-spark-rapids-latest
- emr-7.0.0-spark-rapids-20231211
- emr-7.0.0-java11-latest
- emr-7.0.0-java11-20231211
- emr-7.0.0-java8-latest
- emr-7.0.0-java8-20231211
- emr-7.0.0-spark-rapids-java8-latest
- emr-7.0.0-spark-rapids-java8-20231211
- notebook-spark/emr-7.0.0-latest
- notebook-spark/emr-7.0.0-20231211

- notebook-spark/emr-7.0.0-spark-rapids-latest
- notebook-spark/emr-7.0.0-spark-rapids-20231211
- notebook-spark/emr-7.0.0-java11-latest
- notebook-spark/emr-7.0.0-java11-20231211
- notebook-spark/emr-7.0.0-java8-latest
- notebook-spark/emr-7.0.0-java8-20231211
- notebook-spark/emr-7.0.0-spark-rapids-java8-latest
- notebook-spark/emr-7.0.0-spark-rapids-java8-20231211
- notebook-python/emr-7.0.0-latest
- notebook-python/emr-7.0.0-20231211
- notebook-python/emr-7.0.0-spark-rapids-latest
- notebook-python/emr-7.0.0-spark-rapids-20231211
- notebook-python/emr-7.0.0-java11-latest
- notebook-python/emr-7.0.0-java11-20231211
- notebook-python/emr-7.0.0-java8-latest
- notebook-python/emr-7.0.0-java8-20231211
- notebook-python/emr-7.0.0-spark-rapids-java8-latest
- notebook-python/emr-7.0.0-spark-rapids-java8-20231211

## 版本備註

Amazon EMR on EKS 7.0.0 的版本備註

- 支援的應用程式：適用於 Java 的 AWS SDK 2.20.160-amzn-0 and 1.12.595, Apache Spark 3.5.0-amzn-0, Apache Flink 1.18.0-amzn-0, Flink Operator 1.6.1, Apache Hudi 0.14.0-amzn-1, Apache Iceberg 1.4.2-amzn-0, Delta 3.0.0, Apache Spark RAPIDS 23.10.0-amzn-0, Jupyter Enterprise Gateway 2.6.0
- 支援的元件 - `aws-sagemaker-spark-sdk`、`emr-ddb`、`emr-goodies`、`emr-s3-select`、`emrfs`、`hadoop-client`、`hudi`、`hudi-spark`、`iceberg`、`spark-kubernetes`。
- 支援的組態分類

  若要與 StartJobRun 和 CreateManagedEndpoint API 搭配使用：

| 分類 | 描述 |
| --- | --- |
| core-site | 變更 core-site.xml Hadoop 檔案中的值。 |
| emrfs-site | 變更 EMRFS 設定。 |
| spark-metrics | 變更 metrics.properties Spark 檔案中的值。 |
| spark-defaults | 變更 spark-defaults.conf Spark 檔案中的值。 |
| spark-env | 變更 Spark 環境中的值。 |
| spark-hive-site | 變更 hive-site.xml Spark 檔案中的值。 |
| spark-log4j | 變更 log4j2.properties Spark 檔案中的值。 |
| emr-job-submitter | 作業提交者 Pod 的組態。 |

專門搭配 CreateManagedEndpoint API 使用：

| 分類 | 描述 |
| --- | --- |
| jeg-config | 變更 Jupyter Enterprise Gateway jupyter_enterprise_gateway_config.py 檔案中的值。 |
| jupyter-kernel-overrides | 在 Jupyter 核心規格檔案中變更核心映像的值。 |

組態分類可讓您自訂應用程式。這些檔案通常對應於應用程式的組態 XML 檔案，例如 spark-hive-site.xml。如需詳細資訊，請參閱設定應用程式。

## 值得注意的功能

Amazon EMR on EKS 7.0 版包含以下功能。

- 應用程式升級：Amazon EMR on EKS 7.0.0 應用程式升級包括 Spark 3.5、Flink 1.18 和 Flink Operator 1.6.1。
- Flink Autoscaler 參數自動調整：Flink Autoscaler 用於擴展計算的預設參數可能不是給定作業的最佳值。Amazon EMR on EKS 7.0.0 使用特定擷取指標的歷史趨勢來計算為作業量身打造的最佳參數。

## 變更

Amazon EMR on EKS 7.0 版包含以下變更。

- Amazon Linux 2023：使用 Amazon EMR on EKS 7.0.0 及更高版本時，所有容器映像均以 Amazon Linux 2023 為基礎。
- Spark 使用 Java 17 作為預設執行期：Amazon EMR on EKS 7.0.0 Spark 使用 Java 17 作為預設執行期。如有需要，您可以透過 Amazon EMR on EKS 7.0 版 清單中提供的對應發行標籤，切換為使用 Java 8 或 Java 11。

## emr-7.0.0-latest

版本說明：`emr-7.0.0-latest` 目前指向 `emr-7.0.0-2024321`。

區域：`emr-7.0.0-latest` 適用於 Amazon EMR on EKS 支援的所有區域。如需詳細資訊，請參閱 Amazon EMR on EKS 服務端點。

容器映像標籤：`emr-7.0.0:latest`

## emr-7.0.0-2024321

版本備註：`7.0.0-2024321`已於 2024 年 3 月 11 日發行。相較於先前的版本，此版本已使用最近更新的 Amazon Linux 套件和重要修正重新整理。

區域：`emr-7.0.0-2024321` 適用於 Amazon EMR on EKS 支援的所有區域。如需詳細資訊，請參閱 Amazon EMR on EKS 服務端點。

容器映像標籤：`emr-7.0.0:2024321`

## emr-7.0.0-20231211

版本備註：已於 2023 年 12 月發行 `7.0.0-20231211`。這是 Amazon EMR 7.0.0 (Spark) 的初始版本。

區域：`emr-7.0.0-20231211` 適用於 Amazon EMR on EKS 支援的所有區域。如需詳細資訊，請參閱 Amazon EMR on EKS 服務端點。

容器映像標籤：`emr-7.0.0:20231211`

## emr-7.0.0-flink-latest

版本說明：`emr-7.0.0-flink-latest` 目前指向 `emr-7.0.0-flink-2024321`。

區域：`emr-7.0.0-flink-latest` 適用於 Amazon EMR on EKS 支援的所有區域。如需詳細資訊，請參閱 Amazon EMR on EKS 服務端點。

容器映像標籤：`emr-7.0.0-flink:latest`

## emr-7.0.0-flink-2024321

版本備註：`7.0.0-flink-2024321`已於 2024 年 3 月 11 日發行。相較於先前的版本，此版本已使用最近更新的 Amazon Linux 套件和重要修正重新整理。

區域：`emr-7.0.0-flink-2024321` 適用於 Amazon EMR on EKS 支援的所有區域。如需詳細資訊，請參閱 Amazon EMR on EKS 服務端點。

容器映像標籤：`emr-7.0.0-flink:2024321`

## emr-7.0.0-flink-20231211

版本備註：已於 2023 年 12 月發行 `7.0.0-flink-20231211`。這是 Amazon EMR 7.0.0 (Flink) 的初始版本。

區域：`emr-7.0.0-flink-20231211` 適用於 Amazon EMR on EKS 支援的所有區域。如需詳細資訊，請參閱 Amazon EMR on EKS 服務端點。

容器映像標籤：`emr-7.0.0-flink:20231211`

# Amazon EMR on EKS 6.15.0 版

本頁面描述了針對 Amazon EMR on EKS 部署的 Amazon EMR 新功能和更新功能。如需有關在 Amazon EC2 上執行的 Amazon EMR 以及一般 Amazon EMR 6.15.0 版的詳細資訊，請參閱《Amazon EMR 版本指南》中的 Amazon EMR 6.15.0。

## Amazon EMR on EKS 6.15 版

以下 Amazon EMR 6.15.0 版可用於 Amazon EMR on EKS。選取特定的 emr-6.15.0-XXXX 版本，以檢視更多詳細資訊，例如相關的容器映像標籤。

Flink releases

當您執行 Flink 應用程式時，以下 Amazon EMR 6.15.0 版可用於 Amazon EMR on EKS。

- emr-6.15.0-flink-latest
- emr-6.15.0-flink-20240105
- emr-6.15.0-flink-20231109

Spark releases

當您執行 Spark 應用程式時，以下 Amazon EMR 6.15.0 版可用於 Amazon EMR on EKS。

- emr-6.15.0-latest
- emr-6.15.0-20231109
- emr-6.15.0-spark-rapids-latest
- emr-6.15.0-spark-rapids-20231109
- emr-6.15.0-java11-latest
- emr-6.15.0-java11-20231109
- emr-6.15.0-java17-latest
- emr-6.15.0-java17-20231109
- emr-6.15.0-java17-al2023-latest
- emr-6.15.0-java17-al2023-20231109
- emr-6.15.0-spark-rapids-java17-latest
- emr-6.15.0-spark-rapids-java17-20231109
- emr-6.15.0-spark-rapids-java17-al2023-latest

- emr-6.15.0-spark-rapids-java17-al2023-20231109
- notebook-spark/emr-6.15.0-latest
- notebook-spark/emr-6.15.0-20231109
- notebook-spark/emr-6.15.0-spark-rapids-latest
- notebook-spark/emr-6.15.0-spark-rapids-20231109
- notebook-spark/emr-6.15.0-java11-latest
- notebook-spark/emr-6.15.0-java11-20231109
- notebook-spark/emr-6.15.0-java17-latest
- notebook-spark/emr-6.15.0-java17-20231109
- notebook-spark/emr-6.15.0-java17-al2023-latest
- notebook-spark/emr-6.15.0-java17-al2023-20231109
- notebook-python/emr-6.15.0-latest
- notebook-python/emr-6.15.0-20231109
- notebook-python/emr-6.15.0-spark-rapids-latest
- notebook-python/emr-6.15.0-spark-rapids-20231109
- notebook-python/emr-6.15.0-java11-latest
- notebook-python/emr-6.15.0-java11-20231109
- notebook-python/emr-6.15.0-java17-latest
- notebook-python/emr-6.15.0-java17-20231109
- notebook-python/emr-6.15.0-java17-al2023-latest
- notebook-python/emr-6.15.0-java17-al2023-20231109

## 版本備註

Amazon EMR on EKS 6.15.0 的版本備註

- 支援的應用程式：適用於 Java 的 AWS SDK 1.12.569, Apache Spark 3.4.1-amzn-2, Apache Flink 1.17.1-amzn-1, Apache Hudi 0.14.0-amzn-0, Apache Iceberg 1.4.0-amzn-0, Delta 2.4.0, Apache Spark RAPIDS 23.08.01-amzn-0, Jupyter Enterprise Gateway 2.6.0
- 支援的元件 - `aws-sagemaker-spark-sdk`、`emr-ddb`、`emr-goodies`、`emr-s3-select`、`emrfs`、`hadoop-client`、`hudi`、`hudi-spark`、`iceberg`、`spark-kubernetes`。
- 支援的組態分類

若要與 StartJobRun 和 CreateManagedEndpoint API 搭配使用：

| 分類 | 描述 |
| --- | --- |
| core-site | 變更 core-site.xml Hadoop 檔案中的值。 |
| emrfs-site | 變更 EMRFS 設定。 |
| spark-metrics | 變更 metrics.properties Spark 檔案中的值。 |
| spark-defaults | 變更 spark-defaults.conf Spark 檔案中的值。 |
| spark-env | 變更 Spark 環境中的值。 |
| spark-hive-site | 變更 hive-site.xml Spark 檔案中的值。 |
| spark-log4j | 變更 log4j2.properties Spark 檔案中的值。 |
| emr-job-submitter | 作業提交者 Pod 的組態。 |

專門搭配 CreateManagedEndpoint API 使用：

| 分類 | 描述 |
| --- | --- |
| jeg-config | 變更 Jupyter Enterprise Gateway jupyter_enterprise_gateway_config.py 檔案中的值。 |
| jupyter-kernel-overrides | 在 Jupyter 核心規格檔案中變更核心映像的值。 |

組態分類可讓您自訂應用程式。這些檔案通常對應於應用程式的組態 XML 檔案，例如 spark-hive-site.xml。如需詳細資訊，請參閱設定應用程式。

## 值得注意的功能

Amazon EMR on EKS 6.15 版包含以下功能。

- Amazon EMR on EKS 搭配 Apache Flink：使用 Amazon EMR on EKS 6.15.0 時，您可以在同一個 Amazon EKS 叢集上執行基於 Apache Flink 的應用程式以及其他類型的應用程式。這有助於提高資源使用率並簡化基礎設施管理。您可以在 Flink 應用程式中利用 Spot 執行個體與正常解除委任，並透過 Amazon EBS 的精細復原和任務本機復原縮短重新啟動時間。可存取性和監控功能包括使用存放在 Amazon S3 中的 jar 啟動 Flink 應用程式、存取 AWS Glue Data Catalog、監控與 Amazon S3 和 Amazon CloudWatch 的整合，以及容器日誌輪換。

## emr-6.15.0-latest

版本說明：`emr-6.15.0-latest` 目前指向 `emr-6.15.0-20240105`。

區域：`emr-6.15.0-latest` 適用於 Amazon EMR on EKS 支援的所有區域。如需詳細資訊，請參閱 Amazon EMR on EKS 服務端點。

容器映像標籤：`emr-6.15.0:latest`

## emr-6.15.0-20240105

版本備註：`6.15.0-20240105`已於 2024 年 1 月 17 日發行。相較於先前的版本，此版本已使用最近更新的 Amazon Linux 套件和重要修正重新整理。

區域：`emr-6.15.0-20240105` 適用於 Amazon EMR on EKS 支援的所有區域。如需詳細資訊，請參閱 Amazon EMR on EKS 服務端點。

容器映像標籤：`emr-6.15.0:20240105`

## emr-6.15.0-20231109

版本備註：已於 2023 年 11 月 17 日發行 `6.15.0-20231109`。這是 Amazon EMR 6.15.0 的初始版本。

區域：`emr-6.15.0-20231109` 適用於 Amazon EMR on EKS 支援的所有區域。如需詳細資訊，請參閱 Amazon EMR on EKS 服務端點。

容器映像標籤：`emr-6.15.0:20231109`

## emr-6.15.0-flink-latest

版本說明：`emr-6.15.0-flink-latest` 目前指向 `emr-6.15.0-flink-20240105`。

區域：`emr-6.15.0-flink-latest` 適用於 Amazon EMR on EKS 支援的所有區域。如需詳細資訊，請參閱 Amazon EMR on EKS 服務端點。

容器映像標籤：`emr-6.15.0-flink:latest`

## emr-6.15.0-flink-20240105

版本備註：`6.15.0-flink-20240105`已於 2024 年 1 月 17 日發行。相較於先前的版本，此版本已使用最近更新的 Amazon Linux 套件和重要修正重新整理。

區域：`emr-6.15.0-flink-20240105` 適用於 Amazon EMR on EKS 支援的所有區域。如需詳細資訊，請參閱 Amazon EMR on EKS 服務端點。

容器映像標籤：`emr-6.15.0-flink:20240105`

## emr-6.15.0-flink-20231109

版本備註：已於 2023 年 11 月 17 日發行 `6.15.0-flink-20231109`。這是 Amazon EMR 6.15.0 的初始版本。

區域：`emr-6.15.0-flink-20231109` 適用於 Amazon EMR on EKS 支援的所有區域。如需詳細資訊，請參閱 Amazon EMR on EKS 服務端點。

容器映像標籤：`emr-6.15.0-flink:20231109`

# Amazon EMR on EKS 6.14.0 版

本頁面描述了針對 Amazon EMR on EKS 部署的 Amazon EMR 新功能和更新功能。如需有關在 Amazon EC2 上執行的 Amazon EMR 以及一般 Amazon EMR 6.14.0 版的詳細資訊，請參閱 Amazon EMR 版本指南中的 Amazon EMR 6.14.0。

## Amazon EMR on EKS 6.14 版

以下 Amazon EMR 6.14.0 版可用於 Amazon EMR on EKS。選取特定的 emr-6.14.0-XXXX 版本，以檢視更多詳細資訊，例如相關的容器映像標籤。

- emr-6.14.0-latest

- emr-6.14.0-20231005
- emr-6.14.0-spark-rapids-latest
- emr-6.14.0-spark-rapids-20231005
- emr-6.14.0-java11-latest
- emr-6.14.0-java11-20231005
- emr-6.14.0-java17-latest
- emr-6.14.0-java17-20231005
- emr-6.14.0-java17-al2023-latest
- emr-6.14.0-java17-al2023-20231005
- emr-6.14.0-spark-rapids-java17-latest
- emr-6.14.0-spark-rapids-java17-20231005
- emr-6.14.0-spark-rapids-java17-al2023-latest
- emr-6.14.0-spark-rapids-java17-al2023-20231005
- notebook-spark/emr-6.14.0-latest
- notebook-spark/emr-6.14.0-20231005
- notebook-spark/emr-6.14.0-spark-rapids-latest
- notebook-spark/emr-6.14.0-spark-rapids-20231005
- notebook-spark/emr-6.14.0-java11-latest
- notebook-spark/emr-6.14.0-java11-20231005
- notebook-spark/emr-6.14.0-java17-latest
- notebook-spark/emr-6.14.0-java17-20231005
- notebook-spark/emr-6.14.0-java17-al2023-latest
- notebook-spark/emr-6.14.0-java17-al2023-20231005
- notebook-python/emr-6.14.0-latest
- notebook-python/emr-6.14.0-20231005
- notebook-python/emr-6.14.0-spark-rapids-latest
- notebook-python/emr-6.14.0-spark-rapids-20231005
- notebook-python/emr-6.14.0-java11-latest
- notebook-python/emr-6.14.0-java11-20231005
- notebook-python/emr-6.14.0-java17-latest

- notebook-python/emr-6.14.0-java17-20231005
- notebook-python/emr-6.14.0-java17-al2023-latest
- notebook-python/emr-6.14.0-java17-al2023-20231005

## 版本備註

Amazon EMR on EKS 6.14.0 的版本資訊

- 支援的應用程式 - 適用於 Java 的 AWS SDK 1.12.543、Apache Spark 3.4.1-amzn-1、Apache Hudi 0.13.1-amzn-2、Apache Iceberg 1.3.0-amzn-0、Delta 2.4.0、Apache Spark RAPIDS 23.06.0-amzn-2、Jupyter Enterprise Gateway 2.7.0
- 支援的元件 - `aws-sagemaker-spark-sdk`、`emr-ddb`、`emr-goodies`、`emr-s3-select`、`emrfs`、`hadoop-client`、`hudi`、`hudi-spark`、`iceberg`、`spark-kubernetes`。
- 支援的組態分類

  若要與 StartJobRun 和 CreateManagedEndpoint API 搭配使用：

| 分類 | 描述 |
| --- | --- |
| `core-site` | 變更 `core-site.xml` Hadoop 檔案中的值。 |
| `emrfs-site` | 變更 EMRFS 設定。 |
| `spark-metrics` | 變更 `metrics.properties` Spark 檔案中的值。 |
| `spark-defaults` | 變更 `spark-defaults.conf` Spark 檔案中的值。 |
| `spark-env` | 變更 Spark 環境中的值。 |
| `spark-hive-site` | 變更 `hive-site.xml` Spark 檔案中的值。 |
| `spark-log4j` | 變更 `log4j2.properties` Spark 檔案中的值。 |
| `emr-job-submitter` | 作業提交者 Pod 的組態。 |

專門搭配 CreateManagedEndpoint API 使用：

| 分類 | 描述 |
| --- | --- |
| jeg-config | 變更 Jupyter Enterprise Gateway jupyter_enterprise_gateway_config.py 檔案中的值。 |
| jupyter-kernel-overrides | 在 Jupyter 核心規格檔案中變更核心映像的值。 |

組態分類可讓您自訂應用程式。這些檔案通常對應於應用程式的組態 XML 檔案，例如 spark-hive-site.xml。如需詳細資訊，請參閱設定應用程式。

## 值得注意的功能

Amazon EMR on EKS 6.14 版包含以下功能。

- Apache Livy 支援 - Amazon EMR on EKS 現在支援具有 spark-submit 的 Apache Livy。

## emr-6.14.0-latest

版本說明：emr-6.14.0-latest 目前指向 emr-6.14.0-20231005。

區域：emr-6.14.0-latest 適用於 Amazon EMR on EKS 支援的所有區域。如需詳細資訊，請參閱 Amazon EMR on EKS 服務端點。

容器映像標籤：emr-6.14.0:latest

## emr-6.14.0-20231005

版本注釋：已於 2023 年 10 月 17 日發行 6.14.0-20231005。這是 Amazon EMR 6.14.0 的初始發行。

區域：emr-6.14.0-20231005 適用於 Amazon EMR on EKS 支援的所有區域。如需詳細資訊，請參閱 Amazon EMR on EKS 服務端點。

容器映像標籤：`emr-6.14.0:20231005`

# Amazon EMR on EKS 6.13.0 版

本頁面描述了針對 Amazon EMR on EKS 部署的 Amazon EMR 新功能和更新功能。如需有關在 Amazon EC2 上執行的 Amazon EMR 以及一般 Amazon EMR 6.13.0 版本的詳細資訊，請參閱 Amazon EMR 版本指南中的 Amazon EMR 6.13.0。

## Amazon EMR on EKS 6.13 版

以下 Amazon EMR 6.13.0 版本可用於 Amazon EMR on EKS。選取特定的 emr-6.13.0-XXXX 版本，以檢視更多詳細資訊，例如相關的容器映像標籤。

- emr-6.13.0-latest
- emr-6.13.0-20230814
- emr-6.13.0-spark-rapids-latest
- emr-6.13.0-spark-rapids-20230814
- emr-6.13.0-java11-latest
- emr-6.13.0-java11-20230814
- emr-6.13.0-java17-latest
- emr-6.13.0-java17-20230814
- emr-6.13.0-java17-al2023-latest
- emr-6.13.0-java17-al2023-20230814
- emr-6.13.0-spark-rapids-java17-latest
- emr-6.13.0-spark-rapids-java17-20230814
- emr-6.13.0-spark-rapids-java17-al2023-latest
- emr-6.13.0-spark-rapids-java17-al2023-20230814
- notebook-spark/emr-6.13.0-latest
- notebook-spark/emr-6.13.0-20230814
- notebook-spark/emr-6.13.0-spark-rapids-latest
- notebook-spark/emr-6.13.0-spark-rapids-20230814
- notebook-spark/emr-6.13.0-java11-latest
- notebook-spark/emr-6.13.0-java11-20230814

- notebook-spark/emr-6.13.0-java17-latest
- notebook-spark/emr-6.13.0-java17-20230814
- notebook-spark/emr-6.13.0-java17-al2023-latest
- notebook-spark/emr-6.13.0-java17-al2023-20230814
- notebook-python/emr-6.13.0-latest
- notebook-python/emr-6.13.0-20230814
- notebook-python/emr-6.13.0-spark-rapids-latest
- notebook-python/emr-6.13.0-spark-rapids-20230814
- notebook-python/emr-6.13.0-java11-latest
- notebook-python/emr-6.13.0-java11-20230814
- notebook-python/emr-6.13.0-java17-latest
- notebook-python/emr-6.13.0-java17-20230814
- notebook-python/emr-6.13.0-java17-al2023-latest
- notebook-python/emr-6.13.0-java17-al2023-20230814

## 版本備註

Amazon EMR on EKS 6.13.0 的版本資訊

- 支援的應用程式 - 適用於 Java 的 AWS SDK 1.12.513、Apache Spark 3.4.1-amzn-0、Apache Hudi 0.13.1-amzn-0、Apache Iceberg 1.3.0-amzn-0、Delta 2.4.0、Apache Spark RAPIDS 23.06.0-amzn-1、Jupyter Enterprise Gateway 2.6.0.amzn
- 支援的元件 - `aws-sagemaker-spark-sdk`、`emr-ddb`、`emr-goodies`、`emr-s3-select`、`emrfs`、`hadoop-client`、`hudi`、`hudi-spark`、`iceberg`、`spark-kubernetes`。
- 支援的組態分類

  若要與 StartJobRun 和 CreateManagedEndpoint API 搭配使用：

| 分類 | 描述 |
| --- | --- |
| `core-site` | 變更 `core-site.xml` Hadoop 檔案中的值。 |
| `emrfs-site` | 變更 EMRFS 設定。 |

| 分類 | 描述 |
| --- | --- |
| spark-metrics | 變更 metrics.properties Spark 檔案中的值。 |
| spark-defaults | 變更 spark-defaults.conf Spark 檔案中的值。 |
| spark-env | 變更 Spark 環境中的值。 |
| spark-hive-site | 變更 hive-site.xml Spark 檔案中的值。 |
| spark-log4j | 變更 log4j2.properties Spark 檔案中的值。 |
| emr-job-submitter | 作業提交者 Pod 的組態。 |

專門搭配 CreateManagedEndpoint API 使用：

| 分類 | 描述 |
| --- | --- |
| jeg-config | 變更 Jupyter Enterprise Gateway jupyter_enterprise_gateway_config.py 檔案中的值。 |
| jupyter-kernel-overrides | 在 Jupyter 核心規格檔案中變更核心映像的值。 |

組態分類可讓您自訂應用程式。這些檔案通常對應於應用程式的組態 XML 檔案，例如 spark-hive-site.xml。如需詳細資訊，請參閱設定應用程式。

## 值得注意的功能

Amazon EMR on EKS 6.13 版包含以下功能。

- Amazon Linux 2023 - 透過 Amazon EMR on EKS 6.13 及更高版本，可以使用 AL2023 作為作業系統與 Java 17 執行期一起啟動 Spark。若要執行此操作，請在其名稱中搭配使用發行標籤與

al2023。例如：emr-6.13.0-java17-al2023-latest。建議您先驗證並執行效能測試，然後再將生產工作負載移至 AL2023 和 Java 17。

- 帶有 Apache Flink 的 Amazon EMR on EKS (公開預覽) - Amazon EMR on EKS 6.13 版和更高版本支援 Apache Flink，以公開預覽模式提供。此次啟動後，可以在同一個 Amazon EKS 叢集上執行 Apache Flink 型應用程式以及其他類型的應用程式。這有助於提高資源使用率並簡化基礎設施管理。如果已經在 Amazon EKS 上執行大數據框架，現在就可以讓 Amazon EMR 自動化您的佈建和管理。

## emr-6.13.0-latest

版本說明：emr-6.13.0-latest 目前指向 emr-6.13.0-20230814。

區域：emr-6.13.0-latest 適用於 Amazon EMR on EKS 支援的所有區域。如需詳細資訊，請參閱 Amazon EMR on EKS 服務端點。

容器映像標籤：emr-6.13.0:latest

## emr-6.13.0-20230814

版本注釋：已於 2023 年 9 月 7 日發行 6.13.0-20230814。這是 Amazon EMR 6.13.0 的初始發行。

區域：emr-6.13.0-20230814 適用於 Amazon EMR on EKS 支援的所有區域。如需詳細資訊，請參閱 Amazon EMR on EKS 服務端點。

容器映像標籤：emr-6.13.0:20230814

# Amazon EMR on EKS 6.12.0 版

本頁面描述了針對 Amazon EMR on EKS 部署的 Amazon EMR 新功能和更新功能。如需有關在 Amazon EC2 上執行的 Amazon EMR 以及一般 Amazon EMR 6.12.0 版本的詳細資訊，請參閱 Amazon EMR 版本指南中的 Amazon EMR 6.12.0。

## Amazon EMR on EKS 6.12 版

以下 Amazon EMR 6.12.0 版本可用於 Amazon EMR on EKS。選取特定的 emr-6.12.0-XXXX 版本，以檢視更多詳細資訊，例如相關的容器映像標籤。

- emr-6.12.0-latest
- emr-6.12.0-20240321
- emr-6.12.0-20230701
- emr-6.12.0-spark-rapids-latest
- emr-6.12.0-spark-rapids-20230701
- emr-6.12.0-java11-latest
- emr-6.12.0-java11-20230701
- emr-6.12.0-java17-latest
- emr-6.12.0-java17-20230701
- emr-6.12.0-spark-rapids-java17-latest
- emr-6.12.0-spark-rapids-java17-20230701
- notebook-spark/emr-6.12.0-latest
- notebook-spark/emr-6.12.0-20230701
- notebook-spark/emr-6.12.0-spark-rapids-latest
- notebook-spark/emr-6.12.0-spark-rapids-20230701
- notebook-python/emr-6.12.0-latest
- notebook-python/emr-6.12.0-20230701
- notebook-python/emr-6.12.0-spark-rapids-latest
- notebook-python/emr-6.12.0-spark-rapids-20230701

## 版本備註

Amazon EMR on EKS 6.12.0 的版本資訊

- 支援的應用程式 - 適用於 Java 的 AWS SDK 1.12.490、Apache Spark 3.4.0-amzn-0、Apache Hudi 0.13.1-amzn-0、Apache Iceberg 1.3.0-amzn-0、Delta 2.4.0、Apache Spark RAPIDS 23.06.0-amzn-0、Jupyter Enterprise Gateway 2.6.0
- 支援的元件 - `aws-sagemaker-spark-sdk`、`emr-ddb`、`emr-goodies`、`emr-s3-select`、`emrfs`、`hadoop-client`、`hudi`、`hudi-spark`、`iceberg`、`spark-kubernetes`。
- 支援的組態分類

  若要與 StartJobRun 和 CreateManagedEndpoint API 搭配使用：

| 分類 | 描述 |
| --- | --- |
| core-site | 變更 core-site.xml Hadoop 檔案中的值。 |
| emrfs-site | 變更 EMRFS 設定。 |
| spark-metrics | 變更 metrics.properties Spark 檔案中的值。 |
| spark-defaults | 變更 spark-defaults.conf Spark 檔案中的值。 |
| spark-env | 變更 Spark 環境中的值。 |
| spark-hive-site | 變更 hive-site.xml Spark 檔案中的值。 |
| spark-log4j | 變更 log4j2.properties Spark 檔案中的值。 |
| emr-job-submitter | 作業提交者 Pod 的組態。 |

專門搭配 CreateManagedEndpoint API 使用：

| 分類 | 描述 |
| --- | --- |
| jeg-config | 變更 Jupyter Enterprise Gateway jupyter_enterprise_gateway_config.py 檔案中的值。 |
| jupyter-kernel-overrides | 在 Jupyter 核心規格檔案中變更核心映像的值。 |

組態分類可讓您自訂應用程式。這些檔案通常對應於應用程式的組態 XML 檔案，例如 spark-hive-site.xml。如需詳細資訊，請參閱設定應用程式。

## 值得注意的功能

Amazon EMR on EKS 6.12 版包含以下功能。

- Java 17 - 透過 Amazon EMR on EKS 6.12 及更高版本，可使用 Java 17 執行期啟動 Spark。為此，將 `emr-6.12.0-java17-latest` 作為發行標籤進行傳遞。建議您先驗證並執行效能測試，然後再將生產工作負載從 Java 映像的早期版本移至 Java 17 映像。

## emr-6.12.0-latest

版本說明：`emr-6.12.0-latest` 目前指向 `emr-6.12.0-20240321`。

區域：`emr-6.12.0-latest` 適用於 Amazon EMR on EKS 支援的所有區域。如需詳細資訊，請參閱 Amazon EMR on EKS 服務端點。

容器映像標籤：`emr-6.12.0:latest`

## emr-6.12.0-20240321

版本備註：`6.12.0-20240321`已於 2024 年 3 月 11 日發行。相較於先前的版本，此版本已使用最近更新的 Amazon Linux 套件和重要修正重新整理。

區域：`emr-6.12.0-20240321` 適用於 Amazon EMR on EKS 支援的所有區域。如需詳細資訊，請參閱 Amazon EMR on EKS 服務端點。

容器映像標籤：`emr-6.12.0:20240321`

## emr-6.12.0-20230701

版本注釋：已於 2023 年 7 月 1 日發行 `6.12.0-20230701`。這是 Amazon EMR 6.12.0 的初始發行。

區域：`emr-6.12.0-20230701` 適用於 Amazon EMR on EKS 支援的所有區域。如需詳細資訊，請參閱 Amazon EMR on EKS 服務端點。

容器映像標籤：`emr-6.12.0:20230701`

# Amazon EMR on EKS 6.11.0 版

本頁面描述了針對 Amazon EMR on EKS 部署的 Amazon EMR 新功能和更新功能。如需有關在 Amazon EC2 上執行的 Amazon EMR 以及一般 Amazon EMR 6.11.0 版本的詳細資訊，請參閱 Amazon EMR 版本指南中的 Amazon EMR 6.11.0。

## Amazon EMR on EKS 6.11 版

以下 Amazon EMR 6.11.0 版本可用於 Amazon EMR on EKS。選取特定的 emr-6.11.0-XXXX 版本，以檢視更多詳細資訊，例如相關的容器映像標籤。

- emr-6.11.0-latest
- emr-6.11.0-20230905
- emr-6.11.0-20230509

- emr-6.11.0-spark-rapids-latest
- emr-6.11.0-spark-rapids-20230509
- emr-6.11.0-java11-latest
- emr-6.11.0-java11-20230509
- notebook-spark/emr-6.11.0-latest
- notebook-spark/emr-6.11.0-20230509
- notebook-python/emr-6.11.0-latest
- notebook-python/emr-6.11.0-20230509

## 版本備註

Amazon EMR on EKS 6.11.0 的版本資訊

- 支援的應用程式 - 適用於 Java 的 AWS SDK 1.12.446、Apache Spark 3.3.2-amzn-0、Apache Hudi 0.13.0-amzn-0、Apache Iceberg 1.2.0-amzn-0、Delta 2.2.0、Apache Spark RAPIDS 23.02.0-amzn-0、Jupyter Enterprise Gateway 2.6.0
- 支援的元件 - `aws-sagemaker-spark-sdk`、`emr-ddb`、`emr-goodies`、`emr-s3-select`、`emrfs`、`hadoop-client`、`hudi`、`hudi-spark`、`iceberg`、`spark-kubernetes`。
- 支援的組態分類

若要與 StartJobRun 和 CreateManagedEndpoint API 搭配使用：

| 分類 | 描述 |
| --- | --- |
| core-site | 變更 core-site.xml Hadoop 檔案中的值。 |
| emrfs-site | 變更 EMRFS 設定。 |
| spark-metrics | 變更 metrics.properties Spark 檔案中的值。 |
| spark-defaults | 變更 spark-defaults.conf Spark 檔案中的值。 |
| spark-env | 變更 Spark 環境中的值。 |
| spark-hive-site | 變更 hive-site.xml Spark 檔案中的值。 |
| spark-log4j | 變更 log4j.properties Spark 檔案中的值。 |

專門搭配 CreateManagedEndpoint API 使用：

| 分類 | 描述 |
| --- | --- |
| jeg-config | 變更 Jupyter Enterprise Gateway jupyter_enterprise_gateway_config.py 檔案中的值。 |
| jupyter-kernel-overrides | 在 Jupyter 核心規格檔案中變更核心映像的值。 |

組態分類可讓您自訂應用程式。這些檔案通常對應於應用程式的組態 XML 檔案，例如 spark-hive-site.xml。如需詳細資訊，請參閱設定應用程式。

## 值得注意的功能

Amazon EMR on EKS 6.11 版包含以下功能。

- Amazon ECR 公共映像庫中的 Amazon EMR on EKS 基礎映像 - 如果使用自訂映像功能，我們的基礎映像會提供必要的 jar、組態和程式庫，以便與 Amazon EMR on EKS 互動。現在可以在 Amazon ECR 公共映像庫中尋找基礎映像。
- Spark 容器日誌輪換 - Amazon EMR on EKS 6.11 支援 Spark 容器日誌輪換。可以在 `StartJobRun` API 的 `MonitoringConfiguration` 操作中使用 `containerLogRotationConfiguration` 啟用該功能。可以設定 `rotationSize` 和 `maxFilestoKeep`，以指定您希望 Amazon EMR on EKS 在 Spark 驅動程式和執行程式 Pod 中保留的日誌檔案數量和大小。如需詳細資訊，請參閱使用 Spark 容器日誌輪換。
- Spark operator 和 spark-submit 中支援 Volcano - Amazon EMR on EKS 6.11 支援在 Spark operator 和 spark-submit 中使用 Volcano 作為 Kubernetes 自訂排程器來執行 Spark 作業。可以使用群排程、佇列管理、先佔和公平共用排程等功能，以達到高排程輸送量和優化容量。如需詳細資訊，請參閱使用 Volcano 作為 Amazon EMR on EKS 上 Apache Spark 的自訂排程器。

## emr-6.11.0-latest

版本說明：`emr-6.11.0-latest` 目前指向 `emr-20230905`。

區域：`emr-6.11.0-latest` 適用於 Amazon EMR on EKS 支援的所有區域。如需詳細資訊，請參閱 Amazon EMR on EKS 服務端點。

容器映像標籤：`emr-6.11.0:latest`

## emr-6.11.0-20230905

版本備註：`6.11.0-20230905`已於 2023 年 9 月 29 日發行。相較於先前的版本，此版本已使用最近更新的 Amazon Linux 套件和重要修正重新整理。

區域：`emr-6.11.0-20230509` 適用於 Amazon EMR on EKS 支援的所有區域。如需詳細資訊，請參閱 Amazon EMR on EKS 服務端點。

容器映像標籤：`emr-6.11.0:20230509`

## emr-6.11.0-20230509

版本注釋：已於 2023 年 5 月 9 日發行 `6.11.0-20230509`。這是 Amazon EMR 6.11.0 的初始發行。

區域：`emr-6.11.0-20230509` 適用於 Amazon EMR on EKS 支援的所有區域。如需詳細資訊，請參閱 Amazon EMR on EKS 服務端點。

容器映像標籤：`emr-6.11.0:20230509`

# Amazon EMR on EKS 6.10.0 版

以下 Amazon EMR 6.10.0 版本可用於 Amazon EMR on EKS。選取特定的 emr-6.10.0-XXXX 版本，以檢視更多詳細資訊，例如相關的容器映像標籤。

- emr-6.10.0-latest
- emr-6.10.0-20230905
- emr-6.10.0-20230624
- emr-6.10.0-20230421
- emr-6.10.0-20230403
- emr-6.10.0-20230220
- emr-6.10.0-spark-rapids-latest
- emr-6.10.0-spark-rapids-20230624
- emr-6.10.0-spark-rapids-20230220
- emr-6.10.0-java11-latest
- emr-6.10.0-java11-20230624
- emr-6.10.0-java11-20230220
- notebook-spark/emr-6.10.0-latest
- notebook-spark/emr-6.10.0-20230624
- notebook-spark/emr-6.10.0-20230220
- notebook-python/emr-6.10.0-latest
- notebook-python/emr-6.10.0-20230624
- notebook-python/emr-6.10.0-20230220

Amazon EMR on EKS 6.10.0 的版本資訊

- 支援的應用程式 - 適用於 Java 的 AWS SDK 1.12.397、Spark 3.3.1-amzn-0、Hudi 0.12.2-amzn-0、Iceberg 1.1.0-amzn-0、Delta 2.2.0。
- 支援的元件 - `aws-sagemaker-spark-sdk`、`emr-ddb`、`emr-goodies`、`emr-s3-select`、`emrfs`、`hadoop-client`、`hudi`、`hudi-spark`、`iceberg`、`spark-kubernetes`。
- 支援的組態分類：

  若要與 StartJobRun 和 CreateManagedEndpoint API 搭配使用：

| 分類 | 描述 |
| --- | --- |
| `core-site` | 變更 Hadoop `core-site.xml` 檔案中的值。 |
| `emrfs-site` | 變更 EMRFS 設定。 |
| `spark-metrics` | 變更 Spark `metrics.properties` 檔案中的值。 |
| `spark-defaults` | 變更 Spark `spark-defaults.conf` 檔案中的值。 |
| `spark-env` | 變更 Spark 環境中的值。 |
| `spark-hive-site` | 變更 Spark `hive-site.xml` 檔案中的值。 |
| `spark-log4j` | 變更 Spark `log4j.properties` 檔案中的值。 |

  專門搭配 CreateManagedEndpoint API 使用：

| 分類 | 描述 |
| --- | --- |
| `jeg-config` | 變更 Jupyter Enterprise Gateway `jupyter_enterprise_gateway_config.py` 檔案中的值。 |

| 分類 | 描述 |
| --- | --- |
| jupyter-kernel-overrides | 在 Jupyter 核心規格檔案中變更核心映像的值。 |

組態分類可讓您自訂應用程式。這些檔案通常對應於應用程式的組態 XML 檔案，例如 spark-hive-site.xml。如需詳細資訊，請參閱設定應用程式。

值得注意的功能

- Spark Operator - 透過 Amazon EMR on EKS 6.10.0 及更高版本，可以使用 Kubernetes Operator for Apache Spark 或 Spark operator，在您自己的 Amazon EKS 叢集上利用 Amazon EMR 發行執行期部署和管理 Spark 應用程式。如需詳細資訊，請參閱使用 Spark Operator 執行 Spark 作業。
- Java 11 - 透過 Amazon EMR on EKS 6.10 及更高版本，可使用 Java 11 執行期啟動 Spark。為此，將 emr-6.10.0-java11-latest 作為版本標籤進行傳遞。建議您先驗證並執行效能測試，然後再將生產工作負載從 Java 8 映像移至 Java 11 映像。
- 對於 Apache Spark 的 Amazon Redshift 整合，Amazon EMR on EKS 6.10.0 會刪除 minimal-json.jar 的相依性，並自動將所需的 spark-redshift 相關 jar 新增到 Spark 的執行程式類別路徑：spark-redshift.jar、spark-avro.jar 和 RedshiftJDBC.jar。

變更

- 現在預設為 parquet、ORC 和基於文字的格式 (包括 CSV 和 JSON) 啟用 EMRFS S3 優化提交程式。

## emr-6.10.0-latest

版本說明：emr-6.10.0-latest 目前指向 emr-6.10.0-20230905。

區域：emr-6.10.0-latest 適用於 Amazon EMR on EKS 支援的所有區域。如需詳細資訊，請參閱 Amazon EMR on EKS 服務端點。

容器映像標籤：emr-6.10.0:latest

## emr-6.10.0-20230905

版本備註：6.10.0-20230905已於 2023 年 9 月 29 日發行。與舊版相比，此版本已進行重新整理，具有最近更新的 Amazon Linux 套件和重要修正。

區域：emr-6.10.0-20230905 適用於 Amazon EMR on EKS 支援的所有區域。如需詳細資訊，請參閱 Amazon EMR on EKS 服務端點。

容器映像標籤：emr-6.10.0:20230905

## emr-6.10.0-20230624

版本注釋：已於 2023 年 7 月 7 日發行 6.10.0-20230624。與舊版相比，此版本已進行重新整理，具有最近更新的 Amazon Linux 套件和重要修正。

區域：emr-6.10.0-20230624 適用於 Amazon EMR on EKS 支援的所有區域。如需詳細資訊，請參閱 Amazon EMR on EKS 服務端點。

容器映像標籤：emr-6.10.0:20230624

## emr-6.10.0-20230421

版本注釋：已於 2023 年 4 月 28 日發行 6.10.0-20230421。與舊版相比，此版本已進行重新整理，具有最近更新的 Amazon Linux 套件和重要修正。

區域：emr-6.10.0-20230421 適用於 Amazon EMR on EKS 支援的所有區域。如需詳細資訊，請參閱 Amazon EMR on EKS 服務端點。

容器映像標籤：emr-6.10.0:20230421

## emr-6.10.0-20230403

版本注釋：已於 2023 年 4 月 12 日發行 6.10.0-20230403。與舊版相比，此版本已進行重新整理，具有最近更新的 Amazon Linux 套件和重要修正。

區域：emr-6.10.0-20230403 適用於 Amazon EMR on EKS 支援的所有區域。如需詳細資訊，請參閱 Amazon EMR on EKS 服務端點。

容器映像標籤：emr-6.10.0:20230403

## emr-6.10.0-20230220

版本注釋：已於 2023 年 2 月 20 日發行 `emr-6.10.0-20230220`。這是 Amazon EMR 6.10.0 的初始發行。

區域：`emr-6.10.0-20230220` 適用於 Amazon EMR on EKS 支援的所有區域。如需詳細資訊，請參閱 Amazon EMR on EKS 服務端點。

容器映像標籤：`emr-6.10.0:20230220`

# Amazon EMR on EKS 6.9.0 版

以下 Amazon EMR 6.9.0 版本可用於 Amazon EMR on EKS。選取特定的 emr-6.9.0-XXXX 版本，以檢視更多詳細資訊，例如相關的容器映像標籤。

- emr-6.9.0-latest
- emr-6.9.0-20230905
- emr-6.9.0-20230624
- emr-6.9.0-20221108
- emr-6.9.0-spark-rapids-latest
- emr-6.9.0-spark-rapids-20230624
- emr-6.9.0-spark-rapids-20221108
- notebook-spark/emr-6.9.0-latest
- notebook-spark/emr-6.9.0-20230624
- notebook-spark/emr-6.9.0-20221108
- notebook-python/emr-6.9.0-latest
- notebook-python/emr-6.9.0-20230624
- notebook-python/emr-6.9.0-20221108

Amazon EMR on EKS 6.9.0 的版本資訊

- 支援的應用程式 - 適用於 Java 的 AWS SDK 1.12.331、Spark 3.3.0-amzn-1、Hudi 0.12.1-amzn-0、Iceberg 0.14.1-amzn-0、Delta 2.1.0。
- 支援的元件 - `aws-sagemaker-spark-sdk`、`emr-ddb`、`emr-goodies`、`emr-s3-select`、`emrfs`、`hadoop-client`、`hudi`、`hudi-spark`、`iceberg`、`spark-kubernetes`。

- 支援的組態分類：

  若要與 StartJobRun 和 CreateManagedEndpoint API 搭配使用：

| 分類 | 描述 |
| --- | --- |
| `core-site` | 變更 Hadoop 的 core-site.xml 檔案中的值。 |
| `emrfs-site` | 變更 EMRFS 設定。 |
| `spark-metrics` | 變更 Spark 中 metrics.properties 檔案的值。 |
| `spark-defaults` | 變更 Spark 的 spark-defaults.conf 檔案中的值。 |
| `spark-env` | 變更 Spark 環境中的值。 |
| `spark-hive-site` | 變更 Spark 的 hive-site.xml 檔案中的值 |
| `spark-log4j` | 變更 Spark 中 log4j.properties 檔案的值。 |

  專門搭配 CreateManagedEndpoint API 使用：

| 分類 | 描述 |
| --- | --- |
| `jeg-config` | 變更 Jupyter Enterprise Gateway `jupyter_enterprise_gateway_config.py` 檔案中的值。 |
| `jupyter-kernel-overrides` | 在 Jupyter 核心規格檔案中變更核心映像的值。 |

  組態分類可讓您自訂應用程式。這些檔案通常對應於應用程式的組態 XML 檔案，例如 `spark-hive-site.xml`。如需詳細資訊，請參閱設定應用程式。

值得注意的功能

- Nvidia RAPIDS Accelerator for Apache Spark - Amazon EMR on EKS 可使用 EC2 圖形處理單元 (GPU) 執行個體類型加速 Spark。若要搭配使用 Spark 影像與 RAPIDS Accelerator，請將發行標籤指定為 emr-6.9.0-spark-rapids-latest。如需進一步了解，請造訪文件頁面。
- Spark-Redshift 連接器 - Apache Spark 的 Amazon Redshift 整合包含在 Amazon EMR 6.9.0 及更高版本中。以前是一個開放原始碼工具，本機整合是一個 Spark 連接器，可用於建置在 Amazon Redshift 和 Amazon Redshift Serverless 中讀取和寫入資料的 Apache Spark 應用程式。如需詳細資訊，請參閱針對 Apache Spark on Amazon EMR on EKS 使用 Amazon Redshift 整合。
- Delta Lake-Delta Lake 是一種開放原始碼儲存格式，可啟用資料湖的建置，並具有交易一致性、一致的資料集定義、結構描述演進變化以及資料變動支援。如需進一步了解，請造訪使用 Delta Lake。
- 修改 PySpark 參數 - 互動端點現在支援修改與 EMR Studio Jupyter 筆記本中的 PySpark 工作階段相關聯的 Spark 參數。如需進一步了解，請造訪修改 PySpark 工作階段參數。

已解決的問題

- 搭配使用 DynamoDB 連接器與 Amazon EMR 6.6.0、6.7.0 和 6.8.0 版本上的 Spark 時，即使輸入分割參照非空白資料，從資料表中進行的所有讀取都會傳回空白結果。Amazon EMR 6.9.0 版解決了此問題。
- Amazon EMR on EKS 6.8.0 在使用 Apache Spark 產生的 Parquet 檔案中繼資料中錯誤地填充了建置雜湊。此問題可能會導致從 Amazon EMR on EKS 6.8.0 產生的 Parquet 檔案中剖析中繼資料版本字串的工具失敗。

已知問題

- 如果針對 Apache Spark 使用 Amazon Redshift 整合，並且具有 Parquet 格式的精確度為微秒的 time、timetz、timestamp 或 timestamptz，則連接器會將時間值四捨五入為最接近的微秒值。請使用文字卸載格式 `unload_s3_format` 參數作為一種解決方法。

## emr-6.9.0-latest

版本說明：`emr-6.9.0-latest` 目前指向 `emr-6.9.0-20230905`。

區域：`emr-6.9.0-latest` 適用於 Amazon EMR on EKS 支援的所有區域。如需詳細資訊，請參閱 Amazon EMR on EKS 服務端點。

容器映像標籤：`emr-6.9.0:latest`

## emr-6.9.0-20230905

版本備註：emr-6.9.0-20230905。相較於先前的版本，此版本已使用最近更新的 Amazon Linux 套件和重要修正重新整理。

區域：emr-6.9.0-20230905 適用於 Amazon EMR on EKS 支援的所有區域。如需詳細資訊，請參閱 Amazon EMR on EKS 服務端點。

容器映像標籤：emr-6.9.0:20230905

## emr-6.9.0-20230624

版本注釋：已於 2023 年 7 月 7 日發行 emr-6.9.0-20230624。

區域：emr-6.9.0-20230624 適用於 Amazon EMR on EKS 支援的所有區域。如需詳細資訊，請參閱 Amazon EMR on EKS 服務端點。

容器映像標籤：emr-6.9.0:20230624

## emr-6.9.0-20221108

版本注釋：已於 2022 年 12 月 8 日發行 emr-6.9.0-20221108。這是 Amazon EMR 6.9.0 的初始發行。

區域：emr-6.9.0-20221108 適用於 Amazon EMR on EKS 支援的所有區域。如需詳細資訊，請參閱 Amazon EMR on EKS 服務端點。

容器映像標籤：emr-6.9.0:20221108

# Amazon EMR on EKS 6.8.0 版

以下 Amazon EMR 6.8.0 版本可用於 Amazon EMR on EKS。選取特定的 emr-6.8.0-XXXX 版本，以檢視更多詳細資訊，例如相關的容器映像標籤。

- emr-6.8.0-latest
- emr-6.8.0-20230905
- emr-6.8.0-20230624
- emr-6.8.0-20221219
- emr-6.8.0-20220802

Amazon EMR on EKS 6.8.0 的版本資訊

- 支援的應用程式 - 適用於 Java 的 AWS SDK 1.12.170、Spark 3.3.0-amzn-0、Hudi 0.11.1-amzn-0、Iceberg 0.14.0-amzn-0。
- 支援的元件 - `aws-sagemaker-spark-sdk`、`emr-ddb`、`emr-goodies`、`emr-s3-select`、`emrfs`、`hadoop-client`、`hudi`、`hudi-spark`、`iceberg`、`spark-kubernetes`。
- 支援的組態分類：

| 分類 | 描述 |
| --- | --- |
| `core-site` | 變更 Hadoop 的 core-site.xml 檔案中的值。 |
| `emrfs-site` | 變更 EMRFS 設定。 |
| `spark-metrics` | 變更 Spark 中 metrics.properties 檔案的值。 |
| `spark-defaults` | 變更 Spark 的 spark-defaults.conf 檔案中的值。 |
| `spark-env` | 變更 Spark 環境中的值。 |
| `spark-hive-site` | 變更 Spark 的 hive-site.xml 檔案中的值 |
| `spark-log4j` | 變更 Spark 中 log4j.properties 檔案的值。 |

組態分類可讓您自訂應用程式。這些檔案通常對應於應用程式的組態 XML 檔案，例如 `spark-hive-site.xml`。如需詳細資訊，請參閱設定應用程式。

值得注意的功能

- Spark3.3.0 - Amazon EMR on EKS 6.8 包含 Spark 3.3.0，它支援為 Spark 驅動程式執行程式 Pod 使用不同的節點選取器標籤。這些新標籤可讓您在 StartJobRun API 中單獨定義驅動程式和執行程式 Pod 的節點類型，而無需使用 Pod 範本。
  - 驅動程式節點選取器屬性：spark.kubernetes.driver.node.selector.[labelKey]
  - 執行程式節點選取器屬性：spark.kubernetes.executor.node.selector.[labelKey]
- 增強的作業失敗訊息 - 此版本引入了 `spark.stage.extraDetailsOnFetchFailures.enabled` 和

spark.stage.extraDetailsOnFetchFailures.maxFailuresToInclude 設定，可追蹤因使用者程式碼造成的作業失敗。當因隨機擷取失敗而中止某個階段時，這些詳細資訊將用於增強驅動程式日誌中顯示的失敗訊息。

| 屬性名稱 | 預設值 | 意義 | 自版本 |
| --- | --- | --- | --- |
| spark.stage.extraDetailsOnFetchFailures.enabled | false | 如果設定為 true，此屬性用於在因隨機擷取失敗而中止某個階段時，增強驅動程式日誌中顯示的作業失敗訊息。依預設，會追蹤使用者程式碼造成的最後 5 個任務失敗，並在驅動程式日誌中附加失敗錯誤訊息。<br><br>若需增加要追蹤的使用者例外狀況的任務失敗次數，請參閱 spark.stage.extraDetailsOnFetchFailures.maxFailuresToInclude 組態。 | emr-6.8 |

| 屬性名稱 | 預設值 | 意義 | 自版本 |
| --- | --- | --- | --- |
| `spark.stage.extraDetailsOnFetchFailures.maxFailuresToInclude` | 5 | 每個階段和每次嘗試要追蹤的任務失敗次數。此屬性用於在因隨機擷取失敗而中止某個階段時，增強驅動程式日誌中顯示的使用者例外狀況的作業失敗訊息。<br><br>只有在 spark.stage.extraDetailsOnFetchFailures.enabled 組態設為 True 時，此屬性才有效。 | emr-6.8 |

如需詳細資訊，請參閱 Apache Spark 組態文件。

已知問題

- Amazon EMR on EKS 6.8.0 在使用 Apache Spark 產生的 Parquet 檔案中繼資料中錯誤地填充了建置雜湊。此問題可能會導致從 Amazon EMR on EKS 6.8.0 產生的 Parquet 檔案中剖析中繼資料版本字串的工具失敗。從 Parquet 中繼資料中剖析版本字串並依賴於建置雜湊的客戶應切換到不同的 Amazon EMR 版本並重寫檔案。

已解決的問題

- PySpark 核心的 Interrupt Kernel 功能 - 透過使用 `Interrupt Kernel` 功能，可以停止由筆記本中的執行儲存格觸發的正在進行的互動式工作負載。引入了一個修復程式，以便此功能適用於 PySpark 核心。這也可以在開放原始碼中找到，位於處理 PySpark Kubernetes Kernel #1115 中斷的變更。

## emr-6.8.0-latest

版本說明：`emr-6.8.0-latest` 目前指向 `emr-6.8.0-20230624`。

區域：`emr-6.8.0-latest` 適用於 Amazon EMR on EKS 支援的所有區域。如需詳細資訊，請參閱 Amazon EMR on EKS 服務端點。

容器映像標籤：`emr-6.8.0:latest`

## emr-6.8.0-20230905

版本備註：`emr-6.8.0-20230905`已於 2023 年 9 月 29 日發行。相較於先前的版本，此版本已使用最近更新的 Amazon Linux 套件和重要修正重新整理。

區域：`emr-6.8.0-20230905` 適用於 Amazon EMR on EKS 支援的所有區域。如需詳細資訊，請參閱 Amazon EMR on EKS 服務端點。

容器映像標籤：`emr-6.8.0:20230905`

## emr-6.8.0-20230624

版本注釋：已於 2023 年 7 月 7 日發行 `emr-6.8.0-20230624`。相較於先前的版本，此版本已使用最近更新的 Amazon Linux 套件和重要修正重新整理。

區域：`emr-6.8.0-20230624` 適用於 Amazon EMR on EKS 支援的所有區域。如需詳細資訊，請參閱 Amazon EMR on EKS 服務端點。

容器映像標籤：`emr-6.8.0:20230624`

## emr-6.8.0-20221219

版本注釋：已於 2023 年 1 月 19 日發行 `emr-6.8.0-20221219`。相較於先前的版本，此版本已使用最近更新的 Amazon Linux 套件和重要修正重新整理。

區域：`emr-6.8.0-20221219` 適用於 Amazon EMR on EKS 支援的所有區域。如需詳細資訊，請參閱 Amazon EMR on EKS 服務端點。

容器映像標籤：`emr-6.8.0:20221219`

## emr-6.8.0-20220802

版本注釋：已於 2022 年 9 月 27 日發行 `emr-6.8.0-20220802`。這是 Amazon EMR 6.8.0 的初始發行。

區域：`emr-6.8.0-20220802` 適用於 Amazon EMR on EKS 支援的所有區域。如需詳細資訊，請參閱 Amazon EMR on EKS 服務端點。

容器映像標籤：`emr-6.8.0:20220802`

# Amazon EMR on EKS 6.7.0 版

以下 Amazon EMR 6.7.0 版本可用於 Amazon EMR on EKS。選取特定的 emr-6.7.0-XXXX 版本，以檢視更多詳細資訊，例如相關的容器映像標籤。

- emr-6.7.0-latest
- emr-6.7.0-20240321
- emr-6.7.0-20230624
- emr-6.7.0-20221219
- emr-6.7.0-20220630

Amazon EMR on EKS 6.7.0 的版本資訊

- 支援的應用程式 - Spark 3.2.1-amzn-0、Jupyter Enterprise Gateway 2.6、Hudi 0.11-amzn-0、Iceberg 0.13.1。
- 支援的元件-`aws-hm-client` (Glue 連接器)、`aws-sagemaker-spark-sdk`、`emr-s3-select`、`emrfs`、`emr-ddb`、`hudi-spark`。
- 隨著升級到 JEG 2.6，核心管理現在是非同步的，這意味著 JEG 不會在核心啟動過程中阻止交易。透過提供下列能力，可大幅改善使用者體驗：
  - 當其他核心啟動正在進行時，能夠在目前正執行的筆記本中執行命令的能力
  - 能夠同時啟動多個核心且不會影響正在執行核心的能力
- 支援的組態分類：

| 分類 | 描述 |
| --- | --- |
| `core-site` | 變更 Hadoop `core-site.xml` 檔案中的值。 |
| `emrfs-site` | 變更 EMRFS 設定。 |

| 分類 | 描述 |
| --- | --- |
| spark-metrics | 變更 Spark metrics.properties 檔案中的值。 |
| spark-defaults | 變更 Spark spark-defaults.conf 檔案中的值。 |
| spark-env | 變更 Spark 環境中的值。 |
| spark-hive-site | 變更 Spark hive-site.xml 檔案中的值。 |
| spark-log4j | 變更 Spark log4j.properties 檔案中的值。 |

組態分類可讓您自訂應用程式。這些檔案通常對應於應用程式的組態 XML 檔案，例如 spark-hive-site.xml。如需詳細資訊，請參閱設定應用程式。

已解決的問題

- Amazon EMR on EKS 6.7 修正了在 6.6 版本中搭配使用 Apache Spark 的 Pod 範本功能與互動端點時遇到的一個問題。Amazon EMR on EKS 6.4、6.5 和 6.6 中出現了該問題。現在可以使用 Pod 範本來定義在使用互動端點來執行互動分析時 Spark 驅動程式和執行程式 Pod 的開始方式。
- 在 Amazon EMR on EKS 的舊版中，Jupyter Enterprise Gateway 會在核心啟動進行時阻止交易，而這會阻礙目前執行中的筆記本工作階段的執行。當其他核心啟動正在進行時，您可在目前正執行的筆記本中執行命令。您也可以同時啟動多個核心，而不會失去與已執行之核心的連線。

## emr-6.7.0-latest

版本說明：emr-6.7.0-latest 目前指向 emr-6.7.0-20240321。

區域：emr-6.7.0-latest 適用於 Amazon EMR on EKS 支援的所有區域。如需詳細資訊，請參閱 Amazon EMR on EKS 服務端點。

容器映像標籤：emr-6.7.0:latest

## emr-6.7.0-20240321

版本備註：emr-6.7.0-20240321已於 2024 年 3 月 11 日發行。相較於先前的版本，此版本已使用最近更新的 Amazon Linux 套件和重要修正重新整理。

區域：emr-6.7.0-20240321 適用於 Amazon EMR on EKS 支援的所有區域。如需詳細資訊，請參閱 Amazon EMR on EKS 服務端點。

容器映像標籤：emr-6.7.0:20240321

## emr-6.7.0-20230624

版本注釋：已於 2023 年 7 月 7 日發行 emr-6.7.0-20230624。相較於舊版，此版本已使用最近更新的 Amazon Linux 套件和重要修正重新整理。

區域：emr-6.7.0-20230624 適用於 Amazon EMR on EKS 支援的所有區域。如需詳細資訊，請參閱 Amazon EMR on EKS 服務端點。

容器映像標籤：emr-6.7.0:20230624

## emr-6.7.0-20221219

版本注釋：已於 2023 年 1 月 19 日發行 emr-6.7.0-20221219。相較於舊版，此版本已使用最近更新的 Amazon Linux 套件和重要修正重新整理。

區域：emr-6.7.0-20221219 適用於 Amazon EMR on EKS 支援的所有區域。如需詳細資訊，請參閱 Amazon EMR on EKS 服務端點。

容器映像標籤：emr-6.7.0:20221219

## emr-6.7.0-20220630

版本注釋：已於 2022 年 7 月 12 日發行 emr-6.7.0-20220630。這是 Amazon EMR 6.7.0 的初始發行。

區域：emr-6.7.0-20220630 適用於 Amazon EMR on EKS 支援的所有區域。如需詳細資訊，請參閱 Amazon EMR on EKS 服務端點。

容器映像標籤：emr-6.7.0:20220630

# Amazon EMR on EKS 6.6.0 版

以下 Amazon EMR 6.6.0 版本可用於 Amazon EMR on EKS。選取特定的 emr-6.6.0-XXXX 版本，以檢視更多詳細資訊，例如相關的容器映像標籤。

- emr-6.6.0-latest
- emr-6.6.0-20240321
- emr-6.6.0-20230624
- emr-6.6.0-20221219
- emr-6.6.0-20220411

Amazon EMR on EKS 6.6.0 的版本資訊

- 支援的應用程式 - Spark 3.2.0-amzn-0、Jupyter Enterprise Gateway (端點、公開預覽)、Hudi 0.10.1-amzn-0、Iceberg 0.13.1。
- 支援的元件-`aws-hm-client` (Glue 連接器)、`aws-sagemaker-spark-sdk`、`emr-s3-select`、`emrfs`、`emr-ddb`、`hudi-spark`。
- 支援的組態分類：

| 分類 | 描述 |
| --- | --- |
| `core-site` | 變更 Hadoop 的 core-site.xml 檔案中的值。 |
| `emrfs-site` | 變更 EMRFS 設定。 |
| `spark-metrics` | 變更 Spark 中 metrics.properties 檔案的值。 |
| `spark-defaults` | 變更 Spark 的 spark-defaults.conf 檔案中的值。 |
| `spark-env` | 變更 Spark 環境中的值。 |
| `spark-hive-site` | 變更 Spark 的 hive-site.xml 檔案中的值 |
| `spark-log4j` | 變更 Spark 中 log4j.properties 檔案的值。 |

組態分類可讓您自訂應用程式。這些檔案通常對應於應用程式的組態 XML 檔案，例如 spark-hive-site.xml。如需詳細資訊，請參閱設定應用程式。

已知問題

- 具有互動端點的 Spark Pod 範本功能無法在 Amazon EMR on EKS 6.4、6.5 和 6.6 中運作。

已解決的問題

- 互動端點日誌會上傳到 Cloudwatch 和 S3。

## emr-6.6.0-latest

版本說明：`emr-6.6.0-latest` 目前指向 `emr-6.6.0-20240321`。

區域：`emr-6.6.0-latest` 適用於 Amazon EMR on EKS 支援的所有區域。如需詳細資訊，請參閱 Amazon EMR on EKS 服務端點。

容器映像標籤：`emr-6.6.0:latest`

## emr-6.6.0-20240321

版本備註：`emr-6.6.0-20240321`已於 2024 年 3 月 11 日發行。相較於先前的版本，此版本已使用最近更新的 Amazon Linux 套件和重要修正重新整理。

區域：`emr-6.6.0-20240321` 適用於 Amazon EMR on EKS 支援的所有區域。如需詳細資訊，請參閱 Amazon EMR on EKS 服務端點。

容器映像標籤：`emr-6.6.0:20240321`

## emr-6.6.0-20230624

版本注釋：已於 2023 年 1 月 27 日發行 `emr-6.6.0-20230624`。相較於舊版，此版本已使用最近更新的 Amazon Linux 套件和重要修正重新整理。

區域：`emr-6.6.0-20230624` 適用於 Amazon EMR on EKS 支援的所有區域。如需詳細資訊，請參閱 Amazon EMR on EKS 服務端點。

容器映像標籤：`emr-6.6.0:20230624`

## emr-6.6.0-20221219

版本注釋：已於 2023 年 1 月 27 日發行 `emr-6.6.0-20221219`。相較於舊版，此版本已使用最近更新的 Amazon Linux 套件和重要修正重新整理。

區域：`emr-6.6.0-20221219` 適用於 Amazon EMR on EKS 支援的所有區域。如需詳細資訊，請參閱 Amazon EMR on EKS 服務端點。

容器映像標籤：`emr-6.6.0:20221219`

## emr-6.6.0-20220411

版本注釋：已於 2022 年 5 月 20 日發行 `emr-6.6.0-20220411`。這是 Amazon EMR 6.6.0 的初始發行。

區域：`emr-6.6.0-20220411` 適用於 Amazon EMR on EKS 支援的所有區域。如需詳細資訊，請參閱 Amazon EMR on EKS 服務端點。

容器映像標籤：`emr-6.6.0:20220411`

# Amazon EMR on EKS 6.5.0 版

以下 Amazon EMR 6.5.0 版本可用於 Amazon EMR on EKS。選取特定的 emr-6.5.0-XXXX 版本，以檢視更多詳細資訊，例如相關的容器映像標籤。

- emr-6.5.0-latest
- emr-6.5.0-20240321
- emr-6.5.0-20221219
- emr-6.5.0-20220802
- emr-6.5.0-20211119

Amazon EMR on EKS 6.5.0 的版本資訊

- 支援的應用程式 - Spark 3.1.2-amzn-1、Jupyter Enterprise Gateway (端點、公開預覽)。
- 支援的元件-`aws-hm-client` (Glue 連接器)、`aws-sagemaker-spark-sdk`、`emr-s3-select`、`emrfs`、`emr-ddb`、`hudi-spark`。

- 支援的組態分類：

| 分類 | 描述 |
| --- | --- |
| core-site | 變更 Hadoop 的 core-site.xml 檔案中的值。 |
| emrfs-site | 變更 EMRFS 設定。 |
| spark-metrics | 變更 Spark 中 metrics.properties 檔案的值。 |
| spark-defaults | 變更 Spark 的 spark-defaults.conf 檔案中的值。 |
| spark-env | 變更 Spark 環境中的值。 |
| spark-hive-site | 變更 Spark 的 hive-site.xml 檔案中的值 |
| spark-log4j | 變更 Spark 中 log4j.properties 檔案的值。 |

組態分類可讓您自訂應用程式。這些檔案通常對應於應用程式的組態 XML 檔案，例如 spark-hive-site.xml。如需詳細資訊，請參閱設定應用程式。

已知問題

- 具有互動端點的 Spark Pod 範本功能無法在 Amazon EMR on EKS 6.4 和 6.5 中運作。

## emr-6.5.0-latest

版本說明：emr-6.5.0-latest 目前指向 emr-6.5.0-20240321。

區域：emr-6.5.0-latest 適用於 Amazon EMR on EKS 支援的所有區域。如需詳細資訊，請參閱 Amazon EMR on EKS 服務端點。

容器映像標籤：emr-6.5.0:latest

## emr-6.5.0-20240321

版本備註：emr-6.5.0-20240321已於 2024 年 3 月 11 日發行。相較於舊版，此版本已使用最近更新的 Amazon Linux 套件和重要修正重新整理。

區域：emr-6.5.0-20240321 適用於 Amazon EMR on EKS 支援的所有區域。如需詳細資訊，請參閱 Amazon EMR on EKS 服務端點。

容器映像標籤：emr-6.5.0:20240321

## emr-6.5.0-20221219

版本注釋：已於 2023 年 1 月 19 日發行 emr-6.5.0-20221219。相較於舊版，此版本已使用最近更新的 Amazon Linux 套件和重要修正重新整理。

區域：emr-6.5.0-20221219 適用於 Amazon EMR on EKS 支援的所有區域。如需詳細資訊，請參閱 Amazon EMR on EKS 服務端點。

容器映像標籤：emr-6.5.0:20221219

## emr-6.5.0-20220802

版本注釋：已於 2022 年 8 月 24 日發行 emr-6.5.0-20220802。相較於舊版，此版本已使用最近更新的 Amazon Linux 套件重新整理。

區域：emr-6.5.0-20220802 適用於 Amazon EMR on EKS 支援的所有區域。如需詳細資訊，請參閱 Amazon EMR on EKS 服務端點。

容器映像標籤：emr-6.5.0:20220802

## emr-6.5.0-20211119

版本注釋：已於 2022 年 1 月 20 日發行 emr-6.5.0-20211119。這是 Amazon EMR 6.5.0 的初始發行。

區域：emr-6.5.0-20211119 適用於 Amazon EMR on EKS 支援的所有區域。如需詳細資訊，請參閱 Amazon EMR on EKS 服務端點。

容器映像標籤：emr-6.5.0:20211119

# Amazon EMR on EKS 6.4.0 版

以下 Amazon EMR 6.4.0 版本可用於 Amazon EMR on EKS。選取特定的 emr-6.4.0-XXXX 版本，以檢視更多詳細資訊，例如相關的容器映像標籤。

- emr-6.4.0-latest
- emr-6.4.0-20240321

- emr-6.4.0-20221219
- emr-6.4.0-20210830

Amazon EMR on EKS 6.4.0 的版本資訊

- 支援的應用程式 - Spark 3.1.2-amzn-0、Jupyter Enterprise Gateway (端點、公開預覽)。
- 支援的元件-`aws-hm-client` (Glue 連接器)、`aws-sagemaker-spark-sdk`、`emr-s3-select`、`emrfs`、`emr-ddb`、`hudi-spark`。
- 支援的組態分類：

| 分類 | 描述 |
| --- | --- |
| `core-site` | 變更 Hadoop 的 core-site.xml 檔案中的值。 |
| `emrfs-site` | 變更 EMRFS 設定。 |
| `spark-metrics` | 變更 Spark 中 metrics.properties 檔案的值。 |
| `spark-defaults` | 變更 Spark 的 spark-defaults.conf 檔案中的值。 |
| `spark-env` | 變更 Spark 環境中的值。 |
| `spark-hive-site` | 變更 Spark 的 hive-site.xml 檔案中的值 |
| `spark-log4j` | 變更 Spark 中 log4j.properties 檔案的值。 |

組態分類可讓您自訂應用程式。這些檔案通常對應於應用程式的組態 XML 檔案，例如 spark-hive-site.xml。如需詳細資訊，請參閱設定應用程式。

已知問題

- 具有互動端點的 Spark Pod 範本功能無法在 Amazon EMR on EKS 6.4 中運作。

## emr-6.4.0-latest

版本說明：`emr-6.4.0-latest` 目前指向 `emr-6.4.0-20240321`。

區域：`emr-6.4.0-latest` 適用於 Amazon EMR on EKS 支援的所有區域。如需詳細資訊，請參閱 Amazon EMR on EKS 服務端點。

容器映像標籤：`emr-6.4.0:latest`

## emr-6.4.0-20240321

版本備註：`emr-6.4.0-20240321`已於 2024 年 3 月 11 日發行。相較於先前的版本，此版本已使用最近更新的 Amazon Linux 套件和重要修正重新整理。

區域：`emr-6.4.0-20240321` 適用於 Amazon EMR on EKS 支援的所有區域。如需詳細資訊，請參閱 Amazon EMR on EKS 服務端點。

容器映像標籤：`emr-6.4.0:20240321`

## emr-6.4.0-20221219

版本注釋：已於 2023 年 1 月 27 日發行 `emr-6.4.0-20221219`。相較於先前的版本，此版本已使用最近新增的 Amazon Linux 套件重新整理。

區域：`emr-6.4.0-20221219` 適用於 Amazon EMR on EKS 支援的所有區域。如需詳細資訊，請參閱 Amazon EMR on EKS 服務端點。

容器映像標籤：`emr-6.4.0:20221219`

## emr-6.4.0-20210830

版本注釋：已於 2021 年 12 月 9 日發行 `emr-6.4.0-20210830`。這是 Amazon EMR 6.4.0 的初始發行。

區域：`emr-6.4.0-20210830` 適用於 Amazon EMR on EKS 支援的所有區域。如需詳細資訊，請參閱 Amazon EMR on EKS 服務端點。

容器映像標籤：`emr-6.4.0:20210830`

# Amazon EMR on EKS 6.3.0 版

以下 Amazon EMR 6.3.0 版本可用於 Amazon EMR on EKS。選取特定的 emr-6.3.0-XXXX 版本，以檢視更多詳細資訊，例如相關的容器映像標籤。

- emr-6.3.0-latest
- emr-6.3.0-20240321
- emr-6.3.0-20220802
- emr-6.3.0-20211008
- emr-6.3.0-20210802
- emr-6.3.0-20210429

Amazon EMR on EKS 6.3.0 的版本資訊

- 新功能-從 6.x 版本系列中的 Amazon EMR 6.3.0 開始，Amazon EMR on EKS 支援 Spark 的 Pod 範本功能。您也可以為 Amazon EMR on EKS 啟動 Spark 事件日誌輪換功能。如需詳細資訊，請參閱使用 Pod 範本及使用 Spark 事件日誌輪換。
- 支援的應用程式 - Spark 3.1.1-amzn-0、Jupyter Enterprise Gateway (端點、公開預覽)。
- 支援的元件-`aws-hm-client` (Glue 連接器)、`aws-sagemaker-spark-sdk`、`emr-s3-select`、`emrfs`、`emr-ddb`、`hudi-spark`。
- 支援的組態分類：

| 分類 | 描述 |
| --- | --- |
| `core-site` | 變更 Hadoop 的 core-site.xml 檔案中的值。 |
| `emrfs-site` | 變更 EMRFS 設定。 |
| `spark-metrics` | 變更 Spark 中 metrics.properties 檔案的值。 |
| `spark-defaults` | 變更 Spark 的 spark-defaults.conf 檔案中的值。 |
| `spark-env` | 變更 Spark 環境中的值。 |
| `spark-hive-site` | 變更 Spark 的 hive-site.xml 檔案中的值 |
| `spark-log4j` | 變更 Spark 中 log4j.properties 檔案的值。 |

組態分類可讓您自訂應用程式。這些檔案通常對應於應用程式的組態 XML 檔案，例如 spark-hive-site.xml。如需詳細資訊，請參閱設定應用程式。

## emr-6.3.0-latest

版本說明：`emr-6.3.0-latest` 目前指向 `emr-6.3.0-20240321`。

區域：`emr-6.3.0-latest` 適用於 Amazon EMR on EKS 支援的所有區域。如需詳細資訊，請參閱 Amazon EMR on EKS 服務端點。

容器映像標籤：`emr-6.3.0:latest`

## emr-6.3.0-20240321

版本備註：`emr-6.3.0-20240321`已於 2024 年 3 月 11 日發行。相較於先前的版本，此版本已使用最近更新的 Amazon Linux 套件和重要修正重新整理。

區域：`emr-6.3.0-20240321` 適用於 Amazon EMR on EKS 支援的所有區域。如需詳細資訊，請參閱 Amazon EMR on EKS 服務端點。

容器映像標籤：`emr-6.3.0:20240321`

## emr-6.3.0-20220802

版本注釋：已於 2022 年 9 月 27 日發行 `emr-6.3.0-20220802`。相較於舊版，此版本已使用最近更新的 Amazon Linux 套件重新整理。

區域：`emr-6.3.0-20220802` 適用於 Amazon EMR on EKS 支援的所有區域。如需詳細資訊，請參閱 Amazon EMR on EKS 服務端點。

容器映像標籤：`emr-6.3.0:20220802`

## emr-6.3.0-20211008

版本注釋：已於 2021 年 12 月 9 日發行 `emr-6.3.0-20211008`。與以前的版本相比，此版本包含問題修復和安全更新。

區域：`emr-6.3.0-20211008` 適用於 Amazon EMR on EKS 支援的所有區域。如需詳細資訊，請參閱 Amazon EMR on EKS 服務端點。

容器映像標籤：`emr-6.3.0:20211008`

## emr-6.3.0-20210802

版本注釋：已於 2021 年 8 月 2 日發行 `emr-6.3.0-20210802`。與以前的版本相比，此版本包含問題修復和安全更新。

區域：`emr-6.3.0-20210802` 適用於 Amazon EMR on EKS 支援的所有區域。如需詳細資訊，請參閱 Amazon EMR on EKS 服務端點。

容器映像標籤：`emr-6.3.0:20210802`

### emr-6.3.0-20210429

版本注釋：已於 2021 年 4 月 29 日發行 `emr-6.3.0-20210429`。這是 Amazon EMR 6.3.0 的初始發行。

區域：`emr-6.3.0-20210429` 適用於 Amazon EMR on EKS 支援的所有區域。如需詳細資訊，請參閱 Amazon EMR on EKS 服務端點。

容器映像標籤：`emr-6.3.0:20210429`

## Amazon EMR on EKS 6.2.0 版

以下 Amazon EMR 6.2.0 版本可用於 Amazon EMR on EKS。選取特定的 emr-6.2.0-XXXX 版本，以檢視更多詳細資訊，例如相關的容器映像標籤。

- emr-6.2.0-latest
- emr-6.2.0-20240321
- emr-6.2.0-20220802
- emr-6.2.0-20211008
- emr-6.2.0-20210802
- emr-6.2.0-20210615
- emr-6.2.0-20210129
- emr-6.2.0-20201218
- emr-6.2.0-20201201

Amazon EMR on EKS 6.2.0 的版本資訊

- 支援的應用程式 - Spark 3.0.1-amzn-0、Jupyter Enterprise Gateway (端點、公開預覽)。
- 支援的元件`aws-hm-client` (Glue 連接器)、`aws-sagemaker-spark-sdk`、`emr-s3-select`、`emrfs`、`emr-ddb`、`hudi-spark`。
- 支援的組態分類：

| 分類 | 描述 |
| --- | --- |
| core-site | 變更 Hadoop 的 core-site.xml 檔案中的值。 |
| emrfs-site | 變更 EMRFS 設定。 |
| spark-metrics | 變更 Spark 中 metrics.properties 檔案的值。 |
| spark-defaults | 變更 Spark 的 spark-defaults.conf 檔案中的值。 |
| spark-env | 變更 Spark 環境中的值。 |
| spark-hive-site | 變更 Spark 的 hive-site.xml 檔案中的值 |
| spark-log4j | 變更 Spark 中 log4j.properties 檔案的值。 |

組態分類可讓您自訂應用程式。這些檔案通常對應於應用程式的組態 XML 檔案，例如 spark-hive-site.xml。如需詳細資訊，請參閱設定應用程式。

## emr-6.2.0-latest

版本說明：emr-6.2.0-latest 目前指向 emr-6.2.0-20240321。

區域：emr-6.2.0-latest 適用於 Amazon EMR on EKS 支援的所有區域。如需詳細資訊，請參閱 Amazon EMR on EKS 服務端點。

容器映像標籤：emr-6.2.0:20240321

## emr-6.2.0-20240321

版本備註：emr-6.2.0-20240321已於 2024 年 3 月 11 日發行。相較於先前的版本，此版本已使用最近更新的 Amazon Linux 套件和重要修正重新整理。

區域：emr-6.2.0-20240321 適用於 Amazon EMR on EKS 支援的所有區域。如需詳細資訊，請參閱 Amazon EMR on EKS 服務端點。

容器映像標籤：emr-6.2.0:20240321

## emr-6.2.0-20220802

版本注釋：已於 2022 年 9 月 27 日發行 emr-6.2.0-20220802。相較於舊版，此版本已使用最近更新的 Amazon Linux 套件重新整理。

區域：emr-6.2.0-20220802 適用於 Amazon EMR on EKS 支援的所有區域。如需詳細資訊，請參閱 Amazon EMR on EKS 服務端點。

容器映像標籤：emr-6.2.0:20220802

## emr-6.2.0-20211008

版本注釋：已於 2021 年 12 月 9 日發行 emr-6.2.0-20211008。與以前的版本相比，此版本包含問題修復和安全更新。

區域：emr-6.2.0-20211008 可在以下區域使用：美國東部 (維吉尼亞北部)、美國西部 (奧勒岡)、亞太區域 (東京)、歐洲 (愛爾蘭)、南美洲 (聖保羅)。

容器映像標籤：emr-6.2.0:20211008

## emr-6.2.0-20210802

版本注釋：已於 2021 年 8 月 2 日發行 emr-6.2.0-20210802。與以前的版本相比，此版本包含問題修復和安全更新。

區域：emr-6.2.0-20210802 可在以下區域使用：美國東部 (維吉尼亞北部)、美國西部 (奧勒岡)、亞太區域 (東京)、歐洲 (愛爾蘭)、南美洲 (聖保羅)。

容器映像標籤：emr-6.2.0:20210802

## emr-6.2.0-20210615

版本注釋：已於 2021 年 6 月 15 日發行 emr-6.2.0-20210615。與以前的版本相比，此版本包含問題修復和安全更新。

區域：emr-6.2.0-20210615 可在以下區域使用：美國東部 (維吉尼亞北部)、美國西部 (奧勒岡)、亞太區域 (東京)、歐洲 (愛爾蘭)、南美洲 (聖保羅)。

容器映像標籤：emr-6.2.0:20210615

## emr-6.2.0-20210129

版本注釋：已於 2021 年 1 月 29 日發行 emr-6.2.0-20210129。與 emr-6.2.0-20201218 相比，此版本包含問題修復和安全更新。

區域：emr-6.2.0-20210129 可在以下區域使用：美國東部 (維吉尼亞北部)、美國西部 (奧勒岡)、亞太區域 (東京)、歐洲 (愛爾蘭)、南美洲 (聖保羅)。

容器映像標籤：emr-6.2.0-20210129

## emr-6.2.0-20201218

版本注釋：已於 2020 年 12 月 18 日發行 emr-6.2.0-20201218。與 emr-6.2.0-20201201 相比，此版本包含問題修復和安全更新。

區域：emr-6.2.0-20201218 可在以下區域使用：美國東部 (維吉尼亞北部)、美國西部 (奧勒岡)、亞太區域 (東京)、歐洲 (愛爾蘭)、南美洲 (聖保羅)。

容器映像標籤：emr-6.2.0-20201218

## emr-6.2.0-20201201

版本注釋：已於 2020 年 12 月 1 日發行 emr-6.2.0-20201201。這是 Amazon EMR 6.2.0 的初始發行。

區域：emr-6.2.0-20201201 可在以下區域使用：美國東部 (維吉尼亞北部)、美國西部 (奧勒岡)、亞太區域 (東京)、歐洲 (愛爾蘭)、南美洲 (聖保羅)。

容器映像標籤：emr-6.2.0-20201201

# Amazon EMR on EKS 5.36.0 版

以下 Amazon EMR 5.36.0 版本可用於 Amazon EMR on EKS。選取特定的 emr-5.36.0-XXXX 版本，以檢視更多詳細資訊，例如相關的容器映像標籤。

- emr-5.36.0-latest
- emr-5.36.0-20240321
- emr-5.36.0-20221219
- emr-5.36.0-20220620

- emr-5.36.0-20220525

Amazon EMR on EKS 5.36.0 的版本資訊

- 已修正 log4j2 安全問題。
- 支援的應用程式 - Spark 2.4.8-amzn-2、Jupyter Enterprise Gateway (端點、公開預覽；不支援 Scala 核心)、livy-0.7.1、fluentd-4.0.0。
- 支援的元件 - aws-hm-client、aws-sagemaker-spark-sdk、emr-ddb、emr-goodies、emr-kinesis、kerberos-server。
- 支援的組態分類：

| 分類 | 描述 |
| --- | --- |
| `core-site` | 變更 Hadoop 的 core-site.xml 檔案中的值。 |
| `emrfs-site` | 變更 EMRFS 設定。 |
| `spark-metrics` | 變更 Spark 中 metrics.properties 檔案的值。 |
| `spark-defaults` | 變更 Spark 的 spark-defaults.conf 檔案中的值。 |
| `spark-env` | 變更 Spark 環境中的值。 |
| `spark-hive-site` | 變更 Spark 的 hive-site.xml 檔案中的值 |
| `spark-log4j` | 變更 Spark 中 log4j.properties 檔案的值。 |

組態分類可讓您自訂應用程式。這些檔案通常對應於應用程式的組態 XML 檔案，例如 spark-hive-site.xml。如需詳細資訊，請參閱設定應用程式。

## emr-5.36.0-latest

版本說明：`emr-5.36.0-latest` 目前指向 `emr-5.36.0-20240321`。

區域：`emr-5.36.0-latest` 適用於 Amazon EMR on EKS 支援的所有區域。如需詳細資訊，請參閱 Amazon EMR on EKS 服務端點。

容器映像標籤：emr-5.36.0:latest

## emr-5.36.0-20240321

版本備註：emr-5.36.0-20240321已於 2024 年 3 月 11 日發行。相較於先前的版本，此版本已使用最近更新的 Amazon Linux 套件和重要修正重新整理。

區域：emr-5.36.0-20240321 適用於 Amazon EMR on EKS 支援的所有區域。如需詳細資訊，請參閱 Amazon EMR on EKS 服務端點。

容器映像標籤：emr-5.36.0:20240321

## emr-5.36.0-20221219

版本注釋：已於 2023 年 1 月 27 日發行 emr-5.36.0-20221219。相較於舊版，此版本已使用最近更新的 Amazon Linux 套件重新整理。

區域：emr-5.36.0-20221219 適用於 Amazon EMR on EKS 支援的所有區域。如需詳細資訊，請參閱 Amazon EMR on EKS 服務端點。

容器映像標籤：emr-5.36.0:20221219

## emr-5.36.0-20220620

版本注釋：已於 2022 年 7 月 27 日發行 emr-5.36.0-20220620。相較於舊版，此版本已使用最近更新的 Amazon Linux 套件重新整理。

區域：emr-5.36.0-20220620 適用於 Amazon EMR on EKS 支援的所有區域。如需詳細資訊，請參閱 Amazon EMR on EKS 服務端點。

容器映像標籤：emr-5.36.0:20220620

## emr-5.36.0-20220525

版本注釋：已於 2022 年 6 月 16 日發行 emr-5.36.0-20220525。這是 Amazon EMR 5.36.0 的初始發行。

區域：emr-5.36.0-20220525 適用於 Amazon EMR on EKS 支援的所有區域。如需詳細資訊，請參閱 Amazon EMR on EKS 服務端點。

容器映像標籤：`emr-5.36.0:20220525`

# Amazon EMR on EKS 5.35.0 版

以下 Amazon EMR 5.35.0 版本可用於 Amazon EMR on EKS。選取特定的 emr-5.35.0-XXXX 版本，以檢視更多詳細資訊，例如相關的容器映像標籤。

- emr-5.35.0-latest
- emr-5.35.0-20240321
- emr-5.35.0-20221219
- emr-5.35.0-20220802
- emr-5.35.0-20220307

Amazon EMR on EKS 5.35.0 的版本資訊

- 已修正 log4j2 安全問題。
- 支援的應用程式 - Spark 2.4.8-amzn-1、Hudi 0.9.0-amzn-2、Jupyter Enterprise Gateway (端點、公開預覽；不支援 Scala 核心)。
- 支援的元件 - aws-hm-client (Glue 連接器)、aws-sagemaker-spark-sdk、emr-s3-select、emrfs、emr-ddb、hudi-spark。
- 支援的組態分類：

| 分類 | 描述 |
| --- | --- |
| `core-site` | 變更 Hadoop 的 core-site.xml 檔案中的值。 |
| `emrfs-site` | 變更 EMRFS 設定。 |
| `spark-metrics` | 變更 Spark 中 metrics.properties 檔案的值。 |
| `spark-defaults` | 變更 Spark 的 spark-defaults.conf 檔案中的值。 |
| `spark-env` | 變更 Spark 環境中的值。 |
| `spark-hive-site` | 變更 Spark 的 hive-site.xml 檔案中的值 |

| 分類 | 描述 |
| --- | --- |
| spark-log4j | 變更 Spark 中 log4j.properties 檔案的值。 |

組態分類可讓您自訂應用程式。這些檔案通常對應於應用程式的組態 XML 檔案，例如 spark-hive-site.xml。如需詳細資訊，請參閱設定應用程式。

## emr-5.35.0-latest

版本說明：emr-5.35.0-latest 目前指向 emr-5.35.0-20240321。

區域：emr-5.35.0-latest 適用於 Amazon EMR on EKS 支援的所有區域。如需詳細資訊，請參閱 Amazon EMR on EKS 服務端點。

容器映像標籤：emr-5.35.0:latest

## emr-5.35.0-20240321

版本備註：emr-5.35.0-20240321已於 2024 年 3 月 11 日發行。相較於先前的版本，此版本已使用最近更新的 Amazon Linux 套件和重要修正重新整理。

區域：emr-5.35.0-20240321 適用於 Amazon EMR on EKS 支援的所有區域。如需詳細資訊，請參閱 Amazon EMR on EKS 服務端點。

容器映像標籤：emr-5.35.0:20240321

## emr-5.35.0-20221219

版本注釋：已於 2023 年 1 月 27 日發行 emr-5.35.0-20221219。相較於舊版，此版本已使用最近更新的 Amazon Linux 套件重新整理。

區域：emr-5.35.0-20221219 適用於 Amazon EMR on EKS 支援的所有區域。如需詳細資訊，請參閱 Amazon EMR on EKS 服務端點。

容器映像標籤：emr-5.35.0:20221219

## emr-5.35.0-20220802

版本注釋：已於 2022 年 9 月 27 日發行 emr-5.35.0-20220802。相較於舊版，此版本已使用最近更新的 Amazon Linux 套件重新整理。

區域：`emr-5.35.0-20220802` 適用於 Amazon EMR on EKS 支援的所有區域。如需詳細資訊，請參閱 Amazon EMR on EKS 服務端點。

容器映像標籤：`emr-5.35.0:20220802`

## emr-5.35.0-20220307

版本注釋：已於 2022 年 3 月 30 日發行 `emr-5.35.0-20220307`。相較於舊版，此版本已使用最近更新的 Amazon Linux 套件重新整理。

區域：`emr-5.35.0-20220307` 適用於 Amazon EMR on EKS 支援的所有區域。如需詳細資訊，請參閱 Amazon EMR on EKS 服務端點。

容器映像標籤：`emr-5.35.0:20220307`

# Amazon EMR on EKS 5.34.0 版

以下 Amazon EMR 5.34.0 版本可用於 Amazon EMR on EKS。選取特定的 emr-5.34.0-XXXX 版本，以檢視更多詳細資訊，例如相關的容器映像標籤。

- emr-5.34.0-latest
- emr-5.34.0-20240321
- emr-5.34.0-20220802

Amazon EMR on EKS 5.34.0 的版本資訊

- 支援的應用程式 - Spark 2.4.8-amzn-0、Jupyter Enterprise Gateway (端點、公開預覽；不支援 Scala 核心)。
- 支援的元件-`aws-hm-client` (Glue 連接器)、`aws-sagemaker-spark-sdk`、`emr-s3-select`、`emrfs`、`emr-ddb`、`hudi-spark`。
- 支援的組態分類：

| 分類 | 描述 |
| --- | --- |
| `core-site` | 變更 Hadoop 的 core-site.xml 檔案中的值。 |
| `emrfs-site` | 變更 EMRFS 設定。 |

| 分類 | 描述 |
| --- | --- |
| spark-metrics | 變更 Spark 中 metrics.properties 檔案的值。 |
| spark-defaults | 變更 Spark 的 spark-defaults.conf 檔案中的值。 |
| spark-env | 變更 Spark 環境中的值。 |
| spark-hive-site | 變更 Spark 的 hive-site.xml 檔案中的值 |
| spark-log4j | 變更 Spark 中 log4j.properties 檔案的值。 |

組態分類可讓您自訂應用程式。這些檔案通常對應於應用程式的組態 XML 檔案，例如 spark-hive-site.xml。如需詳細資訊，請參閱設定應用程式。

## emr-5.34.0-latest

版本說明：emr-5.34.0-latest 目前指向 emr-5.34.0-20220802。

區域：emr-5.34.0-latest 適用於 Amazon EMR on EKS 支援的所有區域。如需詳細資訊，請參閱 Amazon EMR on EKS 服務端點。

容器映像標籤：emr-5.34.0:latest

## emr-5.34.0-20240321

版本備註：emr-5.34.0-20240321已於 2024 年 3 月 11 日發行。相較於先前的版本，此版本已使用最近更新的 Amazon Linux 套件和重要修正重新整理。

區域：emr-5.34.0-20240321 適用於 Amazon EMR on EKS 支援的所有區域。如需詳細資訊，請參閱 Amazon EMR on EKS 服務端點。

容器映像標籤：emr-5.34.0:20240321

## emr-5.34.0-20220802

版本注釋：已於 2022 年 8 月 24 日發行 emr-5.34.0-20220802。相較於舊版，此版本已使用最近更新的 Amazon Linux 套件重新整理。

區域：`emr-5.34.0-20220802` 適用於 Amazon EMR on EKS 支援的所有區域。如需詳細資訊，請參閱 Amazon EMR on EKS 服務端點。

容器映像標籤：`emr-5.34.0:20220802`

### emr-5.34.0-20211208

版本注釋：已於 2022 年 1 月 20 日發行 `emr-5.34.0-20211208`。相較於舊版，此版本已使用最近更新的 Amazon Linux 套件重新整理。

區域：`emr-5.34.0-20211208` 適用於 Amazon EMR on EKS 支援的所有區域。如需詳細資訊，請參閱 Amazon EMR on EKS 服務端點。

容器映像標籤：`emr-5.34.0:20211208`

## Amazon EMR on EKS 5.33.0 版

以下 Amazon EMR 5.33.0 版本可用於 Amazon EMR on EKS。選取特定的 emr-5.33.0-XXXX 版本，以檢視更多詳細資訊，例如相關的容器映像標籤。

- emr-5.33.0-latest
- emr-5.33.0-20240321
- emr-5.33.0-20221219
- emr-5.33.0-20220802
- emr-5.33.0-20211008
- emr-5.33.0-20210802
- emr-5.33.0-20210615
- emr-5.33.0-20210323

Amazon EMR on EKS 5.33.0 的版本資訊

- 新功能-從 5.x 版本系列中的 Amazon EMR 5.33.0 開始，Amazon EMR on EKS 支援 Spark 的 Pod 範本功能。如需詳細資訊，請參閱使用 Pod 範本。
- 支援的應用程式 - Spark 2.4.7-amzn-1、Jupyter Enterprise Gateway (端點、公開預覽；不支援 Scala 核心)。
- 支援的元件-`aws-hm-client` (Glue 連接器)、`aws-sagemaker-spark-sdk`、`emr-s3-select`、`emrfs`、`emr-ddb`、`hudi-spark`。

- 支援的組態分類：

| 分類 | 描述 |
| --- | --- |
| core-site | 變更 Hadoop 的 core-site.xml 檔案中的值。 |
| emrfs-site | 變更 EMRFS 設定。 |
| spark-metrics | 變更 Spark 中 metrics.properties 檔案的值。 |
| spark-defaults | 變更 Spark 的 spark-defaults.conf 檔案中的值。 |
| spark-env | 變更 Spark 環境中的值。 |
| spark-hive-site | 變更 Spark 的 hive-site.xml 檔案中的值 |
| spark-log4j | 變更 Spark 中 log4j.properties 檔案的值。 |

組態分類可讓您自訂應用程式。這些檔案通常對應於應用程式的組態 XML 檔案，例如 spark-hive-site.xml。如需詳細資訊，請參閱設定應用程式。

## emr-5.33.0-latest

版本說明：emr-5.33.0-latest 目前指向 emr-5.33.0-20240321。

區域：emr-5.33.0-latest 適用於 Amazon EMR on EKS 支援的所有區域。如需詳細資訊，請參閱 Amazon EMR on EKS 服務端點。

容器映像標籤：emr-5.33.0:latest

## emr-5.33.0-20240321

版本備註：emr-5.33.0-20240321已於 2024 年 3 月 11 日發行。相較於先前的版本，此版本已使用最近更新的 Amazon Linux 套件和重要修正重新整理。

區域：emr-5.33.0-20240321 適用於 Amazon EMR on EKS 支援的所有區域。如需詳細資訊，請參閱 Amazon EMR on EKS 服務端點。

容器映像標籤：emr-5.33.0:20240321

## emr-5.33.0-20221219

版本注釋：已於 2023 年 1 月 19 日發行 emr-5.33.0-20221219。相較於舊版，此版本已使用最近更新的 Amazon Linux 套件和重要修正重新整理。

區域：emr-5.33.0-20221219 適用於 Amazon EMR on EKS 支援的所有區域。如需詳細資訊，請參閱 Amazon EMR on EKS 服務端點。

容器映像標籤：emr-5.33.0:20221219

## emr-5.33.0-20220802

版本注釋：已於 2022 年 8 月 24 日發行 emr-5.33.0-20220802。相較於舊版，此版本已使用最近更新的 Amazon Linux 套件重新整理。

區域：emr-5.33.0-20220802 適用於 Amazon EMR on EKS 支援的所有區域。如需詳細資訊，請參閱 Amazon EMR on EKS 服務端點。

容器映像標籤：emr-5.33.0:20220802

## emr-5.33.0-20211008

版本注釋：已於 2021 年 12 月 9 日發行 emr-5.33.0-20211008。與以前的版本相比，此版本包含問題修復和安全更新。

區域：emr-5.33.0-20211008 適用於 Amazon EMR on EKS 支援的所有區域。如需詳細資訊，請參閱 Amazon EMR on EKS 服務端點。

容器映像標籤：emr-5.33.0:20211008

## emr-5.33.0-20210802

版本注釋：已於 2021 年 8 月 2 日發行 emr-5.33.0-20210802。與以前的版本相比，此版本包含問題修復和安全更新。

區域：emr-5.33.0-20210802 適用於 Amazon EMR on EKS 支援的所有區域。如需詳細資訊，請參閱 Amazon EMR on EKS 服務端點。

容器映像標籤：emr-5.33.0:20210802

## emr-5.33.0-20210615

版本注釋：已於 2021 年 6 月 15 日發行 `emr-5.33.0-20210615`。與以前的版本相比，此版本包含問題修復和安全更新。

區域：`emr-5.33.0-20210615` 適用於 Amazon EMR on EKS 支援的所有區域。如需詳細資訊，請參閱 Amazon EMR on EKS 服務端點。

容器映像標籤：`emr-5.33.0:20210615`

## emr-5.33.0-20210323

版本注釋：已於 2021 年 3 月 23 日發行 `emr-5.33.0-20210323`。這是 Amazon EMR 5.33.0 的初始發行。

區域：`emr-5.33.0-20210323` 適用於 Amazon EMR on EKS 支援的所有區域。如需詳細資訊，請參閱 Amazon EMR on EKS 服務端點。

容器映像標籤：`emr-5.33.0-20210323`

# Amazon EMR on EKS 5.32.0 版

以下 Amazon EMR 5.32.0 版本可用於 Amazon EMR on EKS。選取特定的 emr-5.32.0-XXXX 版本，以檢視更多詳細資訊，例如相關的容器映像標籤。

- emr-5.32.0-latest
- emr-5.32.0-20240321
- emr-5.32.0-20220802
- emr-5.32.0-20211008
- emr-5.32.0-20210802
- emr-5.32.0-20210615
- emr-5.32.0-20210129
- emr-5.32.0-20201218
- emr-5.32.0-20201201

Amazon EMR on EKS 5.32.0 的版本資訊

- 支援的應用程式 - Spark 2.4.7-amzn-0、Jupyter Enterprise Gateway (端點、公開預覽；不支援 Scala 核心)。
- 支援的元件-`aws-hm-client` (Glue 連接器)、`aws-sagemaker-spark-sdk`、`emr-s3-select`、`emrfs`、`emr-ddb`、`hudi-spark`。
- 支援的組態分類：

| 分類 | 描述 |
| --- | --- |
| `core-site` | 變更 Hadoop 的 core-site.xml 檔案中的值。 |
| `emrfs-site` | 變更 EMRFS 設定。 |
| `spark-metrics` | 變更 Spark 中 metrics.properties 檔案的值。 |
| `spark-defaults` | 變更 Spark 的 spark-defaults.conf 檔案中的值。 |
| `spark-env` | 變更 Spark 環境中的值。 |
| `spark-hive-site` | 變更 Spark 的 hive-site.xml 檔案中的值 |
| `spark-log4j` | 變更 Spark 中 log4j.properties 檔案的值。 |

組態分類可讓您自訂應用程式。這些檔案通常對應於應用程式的組態 XML 檔案，例如 spark-hive-site.xml。如需詳細資訊，請參閱設定應用程式。

## emr-5.32.0-latest

版本說明：`emr-5.32.0-latest` 目前指向 `emr-5.32.0-20240321`。

區域：`emr-5.32.0-latest` 適用於 Amazon EMR on EKS 支援的所有區域。如需詳細資訊，請參閱 Amazon EMR on EKS 服務端點。

容器映像標籤：`emr-5.32.0:latest`

## emr-5.32.0-20240321

版本備註：emr-5.32.0-20240321已於 2024 年 3 月 11 日發行。相較於先前的版本，此版本已使用最近更新的 Amazon Linux 套件和重要修正重新整理。

區域：emr-5.32.0-20240321 適用於 Amazon EMR on EKS 支援的所有區域。如需詳細資訊，請參閱 Amazon EMR on EKS 服務端點。

容器映像標籤：emr-5.32.0:20240321

## emr-5.32.0-20220802

版本注釋：已於 2022 年 8 月 24 日發行 emr-5.32.0-20220802。相較於舊版，此版本已使用最近更新的 Amazon Linux 套件重新整理。

區域：emr-5.32.0-20220802 適用於 Amazon EMR on EKS 支援的所有區域。如需詳細資訊，請參閱 Amazon EMR on EKS 服務端點。

容器映像標籤：emr-5.32.0:20220802

## emr-5.32.0-20211008

版本注釋：已於 2021 年 12 月 9 日發行 emr-5.32.0-20211008。與以前的版本相比，此版本包含問題修復和安全更新。

區域：emr-5.32.0-20211008 可在以下區域使用：美國東部 (維吉尼亞北部)、美國西部 (奧勒岡)、亞太區域 (東京)、歐洲 (愛爾蘭)、南美洲 (聖保羅)。

容器映像標籤：emr-5.32.0:20211008

## emr-5.32.0-20210802

版本注釋：已於 2021 年 8 月 2 日發行 emr-5.32.0-20210802。與以前的版本相比，此版本包含問題修復和安全更新。

區域：emr-5.32.0-20210802 可在以下區域使用：美國東部 (維吉尼亞北部)、美國西部 (奧勒岡)、亞太區域 (東京)、歐洲 (愛爾蘭)、南美洲 (聖保羅)。

容器映像標籤：emr-5.32.0:20210802

## emr-5.32.0-20210615

版本注釋：已於 2021 年 6 月 15 日發行 emr-5.32.0-20210615。與以前的版本相比，此版本包含問題修復和安全更新。

區域：emr-5.32.0-20210615 可在以下區域使用：美國東部 (維吉尼亞北部)、美國西部 (奧勒岡)、亞太區域 (東京)、歐洲 (愛爾蘭)、南美洲 (聖保羅)。

容器映像標籤：emr-5.32.0:20210615

## emr-5.32.0-20210129

版本注釋：已於 2021 年 1 月 29 日發行 emr-5.32.0-20210129。與 emr-5.32.0-20201218 相比，此版本包含問題修復和安全更新。

區域：emr-5.32.0-20210129 可在以下區域使用：美國東部 (維吉尼亞北部)、美國西部 (奧勒岡)、亞太區域 (東京)、歐洲 (愛爾蘭)、南美洲 (聖保羅)。

容器映像標籤：emr-5.32.0-20210129

## emr-5.32.0-20201218

版本注釋：已於 2020 年 12 月 18 日發行 5.32.0-20201218。與 5.32.0-20201201 相比，此版本包含問題修復和安全更新。

區域：emr-5.32.0-20201218 可在以下區域使用：美國東部 (維吉尼亞北部)、美國西部 (奧勒岡)、亞太區域 (東京)、歐洲 (愛爾蘭)、南美洲 (聖保羅)。

容器映像標籤：emr-5.32.0-20201218

## emr-5.32.0-20201201

版本注釋：已於 2020 年 12 月 1 日發行 5.32.0-20201201。這是 Amazon EMR 5.32.0 的初始發行。

區域：5.32.0-20201201 可在以下區域使用：美國東部 (維吉尼亞北部)、美國西部 (奧勒岡)、亞太區域 (東京)、歐洲 (愛爾蘭)、南美洲 (聖保羅)。

容器映像標籤：emr-5.32.0-20201201

# 文件歷史記錄

下表描述了自上次發行 Amazon EMR on EKS 後，對文件的重要變更。如需有關此文件更新的詳細資訊，您可以訂閱 RSS 摘要。

| 變更 | 描述 | 日期 |
| --- | --- | --- |
| 更新內容 | Amazon EMR on EKS 的受管政策 – 的其他許可`AmazonEMRContainersServiceRolePolicy` 。 | 2025 年 2 月 3 日 |
| 新版本 | Amazon EMR on EKS 7.6.0 版 | 2025 年 1 月 10 日 |
| 新版本 | Amazon EMR on EKS 7.5.0 版 | 2024 年 11 月 21 日 |
| 新版本 | Amazon EMR on EKS 7.4.0 版本 | 2024 年 11 月 13 日 |
| 新版本 | Amazon EMR on EKS 7.3.0 版本 | 2024 年 10 月 16 日 |
| 新版本 | Amazon EMR on EKS 7.2.0 版本 | 2024 年 7 月 25 日 |
| 新版本 | Amazon EMR on EKS 7.1.0 版 | 2024 年 4 月 17 日 |
| 新版本 | Amazon EMR on EKS 7.0.0 版 | 2023 年 12 月 22 日 |
| 新版本 | Amazon EMR on EKS 6.15.0 版 | 2023 年 11 月 17 日 |
| 新版本 | Amazon EMR on EKS 6.14.0 版 | 2023 年 10 月 17 日 |
| 更新內容 | 將「受管端點」重新命名為互動端點；互動端點通用性 | 2023 年 9 月 29 日 |
| 新版本 | Amazon EMR on EKS 6.13.0 版，以及使用 Amazon EMR on EKS 執行 Flink 作業 的公開預覽文件 | 2023 年 9 月 12 日 |
| 新版本 | Amazon EMR on EKS 6.12.0 版 | 2023 年 7 月 21 日 |
| 新內容 | 已新增 使用 Volcano 作為 Amazon EMR on EKS 上 Apache Spark 的自訂排程器 | 2023 年 6 月 13 日 |

| 變更 | 描述 | 日期 |
| --- | --- | --- |
| 新內容 | 已新增 使用 Volcano 作為 Amazon EMR on EKS 上 Apache Spark 的自訂排程器 | 2023 年 6 月 13 日 |
| 新內容 | 已新增 使用 Spark 容器日誌輪換 | 2023 年 6 月 12 日 |
| 更新內容 | 更新了自訂映像文件，以尋找 Amazon ECR 公共映像庫中的基礎映像資訊。 | 2023 年 6 月 8 日 |
| 新版本 | Amazon EMR on EKS 6.11.0 版 | 2023 年 6 月 8 日 |
| 新內容 | 在 使用 Amazon EMR on EKS 執行 Spark 任務 下新增了 使用 Spark Operator 執行 Spark 作業 並重新組織「作業執行」區段。 | 2023 年 6 月 5 日 |
| 新內容 | 已新增兩個區段：搭配使用垂直自動擴展與 Amazon EMR Spark 作業 和 使用自助託管的 Jupyter 筆記本 | 2023 年 5 月 4 日 |
| 文件歷史紀錄頁面 | 已建立 Amazon EMR on EKS 的文件歷史記錄頁面。 | 2023 年 3 月 13 日 |
| 受管政策頁面 | 已建立 Amazon EMR on EKS 的受管政策頁面。 | 2023 年 3 月 13 日 |

本文為英文版的機器翻譯版本，如內容有任何歧義或不一致之處，概以英文版為準。