aws

Amazon EKS 開EMR發指南

Amazon EMR

Amazon EMR: Amazon EKS 開EMR發指南

Table of Contents

什麼是 Amazon EMR on EKS？ 1
架構 2
概念 3
Kubernetes 命名空間 3
虛擬叢集 3
作業執行 3
Amazon EMR 容器 4
元件如何一起工作 4
入門 6
執行 Spark 應用程式 6
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
必要條件 15
步驟 1：從 Amazon 彈性容器註冊表（AmazonECR）檢索基本映像 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 16
步驟 4：發布自訂映像 18
步驟 5：EMR使用自訂映像在 Amazon 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 22
如何選擇基本映像 URI 24
Amazon ECR 註冊賬戶 25
考量事項 26

執行 Flink 作業 27
Flink Kubernetes Operator 27
設定 27
開始使用 29
執行 Flink 應用程式 30
安全 34
解除安裝 Operator 36
Native Kubernetes 37
設定 37
開始使用 37
安全要求 40
Docker 影像 41
自訂 Flink 和 FluentD 的泊塢視窗影像 41
監控 44
使用 Amazon Managed Service for Prometheus 44
使用 Flink UI 46
使用監控組態 47
作業彈性 51
使用高可用性 52
優化重新啟動時間 57
正常解除委任 63
使用 Autoscaler 65
自動配置器參數自動調諧 67
維護和疑難排解 75
遷移 75
疑難排解 76
支援的版本 78
執行 Spark 作業 79
StartJobRun 79
設定 80
入門 104
Spark Operator 106
設定 106
開始使用 107
垂直自動擴展 111
解除安裝 115

安全 116
spark-submit 125
設定 125
開始使用 126
安全 127
Apache Livy 132
設定 133
開始使用 133
運行一個星火應用 138
解除安裝 140
安全 140
安裝屬性 149
故障診斷 153
管理作業執行 154
使用 CLI 進行管理 154
執行 Spark SQL 指令碼 159
作業執行狀態 162
在主控台中檢視作業 162
常見作業執行錯誤 163
使用作業提交器分類 168
概要 168
範例 169
使用作業範本 172
建立並使用作業範本來啟動作業執行 172
定義作業範本參數 174
控制對作業範本的存取 176
使用 Pod 範本 177
常用案例 178
使用 Amazon EMR on EKS 啟用 Pod 範本 179
Pod 範本欄位 181
附屬容器的考量 184
使用重試政策 186
設定重試政策 186
擷取政策狀態 188
監控作業 189
尋找驅動程式日誌 189

使用 Spark 事件日誌輪換 190
使用 Spark 容器日誌輪換 191
使用垂直自動擴展 192
設定 193
開始使用 195
組態 197
監控建議 202
解除安裝 203
執行互動式工作負載 204
互動端點概觀 204
互動端點先決條件 206
AWS CLI 206
eksctl 206
Amazon EKS 集群 206
授予叢集存取權 207
啟用服務帳戶的IAM角色 207
建立IAM工作執行角色 207
授予使用者存取權 207
註冊 Amazon EKS 群集與 Amazon EMR 208
Load Balancer 控制器 208
建立互動端點 208
建立互動端點 208
指定自訂參數 209
..... 210
互動端點參數 211
配置互動端點的設定 212
監控 Spark 任務 212
自訂 Pod 範本 213
將JEG網繭部署到節點群組 214
JEG組態選項 217
修改 PySpark參數 218
自訂核心映像 218
監控互動端點 220
範例 222
使用自助託管的 Jupyter 筆記本 222
建立安全群組 223

建立互動端點 223
取得閘道伺服器 URL 223
取得驗證字符 224
部署筆記本 225
清除 230
其他操作 231
..... 231
列出互動端點 232
刪除互動端點 233
上傳資料 235
必要條件 235
開始使用 235
監控任務 237
使用 Amazon CloudWatch 活動監控任務 237
透過活動在 EKS 上自動執行 Amazon EMR CloudWatch 238
範例：設定調用 Lambda 的規則 239
使用 Amazon CloudWatch 事件以重試政策監控任務的驅動程式網繭 239
管理虛擬叢集 240
建立虛擬叢集 240
列出虛擬叢集 241
描述虛擬叢集 242
刪除虛擬叢集 242
虛擬叢集狀態 242
教學課程 243
使用 Delta Lake 243
使用 Iceberg 244
使用 PyFlink 245
使用 AWS Glue 與熔接 246
使用阿帕奇胡迪 248
提交阿帕奇胡迪工作 249
使用 Spark RAPIDS 252
使用 Spark on Redshift 256
啟動 Spark 應用程式 257
向 Amazon Redshift 進行身分驗證 258
讀取和寫入 Amazon Redshift 260
考量事項 262

使用 Volcano 262
概要 263
安裝 263
提交：Spark Operator 264
提交：spark-submit 266
使用 YuniKorn 267
概要 267
建立 叢集 267
安裝 YuniKorn 269
提交：Spark Operator 270
提交：spark-submit 272
安全 12
最佳實務 276
套用最低權限準則 276
端點的存取控制清單 276
取得自訂映像的最新安全更新 276
限制 Pod 憑證存取 276
隔離不受信任的應用程式碼 277
角色型存取控制 (RBAC) 許可 277
限制存取節點群組 IAM 角色或執行個體設定檔憑證 277
資料保護 278
靜態加密 278
傳輸中加密 280
身分和存取權管理 281
物件 281
使用身分驗證 282
使用政策管理存取權 285
Amazon EMR 如EKS何與 IAM 286
使用服務連結角色 292
Amazon EMR 的受管政策 EKS 295
搭配使用作業執行角色與 Amazon EMR on EKS 296
身分型政策範例 298
標籤型存取控制政策 300
故障診斷 303
記錄和監控 305
CloudTrail日誌 305

S3 Access Grants 308
概觀 308
啟動叢集 308
考量事項 309
合規驗證 310
復原能力 310
基礎設施安全性 310
組態與漏洞分析 310
界面 VPC 端點 311
為 Amazon EMR on EKS 建立 VPC 端點政策 311
跨帳戶存取 314
先決條件 314
如何存取跨帳戶 Amazon S3 儲存貯體或 DynamoDB 資料表 315
標記 資源 319
標籤基本概念 319
標記您的 資源 319
標籤限制 320
使用 AWS CLI 和 Amazon EMR on EKS API 來處理標籤 321
疑難排解 13
PVC 作業失敗 322
驗證 322
修補程式 322
手動修補 326
垂直自動擴展失敗 328
「403 禁止」錯誤 328
找不到命名空間 328
Docker 憑證錯誤 329
Spark Operator 失敗 329
Helm Chart 安裝失敗 329
Unsupported filesystem exception 330
服務端點和配額 331
服務端點 331
Service Quotas 332
發行版本 334
7.2.0 版本發布 335
推出 335

版本備註 337
功能 338
電腦 -7.2.0-最新 339
電子郵件 -7.2.0-20240610 339
emr-7.2.0-打火石最新 339
火石火石 339
7.1.0 版本發布 340
推出 340
版本備註 341
功能 343
電子報 -7.1.0-最新 343
電腦 -7.1.0-20240321 343
emr-7.1.0-打火石最新 343
火石火石 344
7.0.0 版 344
推出 344
版本備註 345
功能 347
變更 347
emr-7.0.0-latest 347
電腦 -7.0.0-2024321 348
emr-7.0.0-20231211 348
emr-7.0.0-flink-latest 348
火石火石 348
emr-7.0.0-flink-20231211 348
6.15.0 版 349
推出 349
版本備註 350
功能 352
emr-6.15.0-latest 352
電子商品 -6.15.0-20240105 352
emr-6.15.0-20231109 353
emr-6.15.0-flink-latest 353
火石火石 353
emr-6.15.0-flink-20231109 353
6.14.0 版 354

推出 354
版本備註 355
功能 356
emr-6.14.0-latest 357
emr-6.14.0-20231005 357
6.13.0 版 357
推出 357
版本備註 358
功能 360
emr-6.13.0-latest 360
emr-6.13.0-20230814 360
6.12.0 版 361
推出 361
版本備註 361
功能 363
emr-6.12.0-latest 363
電腦 -6.12.0-20240321 363
emr-6.12.0-20230701 363
6.11.0 版 364
推出 364
版本備註 364
功能 366
emr-6.11.0-latest 366
電腦 -6.11.0-20230905 366
emr-6.11.0-20230509 367
6.10.0 版 367
emr-6.10.0-latest 369
電腦 -6.10.0-20230905 370
emr-6.10.0-20230624 370
emr-6.10.0-20230421 370
emr-6.10.0-20230403 370
emr-6.10.0-20230220 371
6.9.0 版 371
emr-6.9.0-latest 373
電腦 -6.9.0-20230905 374
emr-6.9.0-20230624 374

emr-6.9.0-20221108 374
6.8.0 版 374
emr-6.8.0-latest 377
電腦 -6.8.0-20230905 378
emr-6.8.0-20230624 378
emr-6.8.0-20221219 378
emr-6.8.0-20220802 378
6.7.0 版 379
emr-6.7.0-latest 380
電腦 -6.7.0-20240321 381
emr-6.7.0-20230624 381
emr-6.7.0-20221219 381
emr-6.7.0-20220630 381
6.6.0 版 382
emr-6.6.0-latest 383
電腦 -6.0-20240321 383
emr-6.6.0-20230624 383
emr-6.6.0-20221219 384
emr-6.6.0-20220411 384
6.5.0 版 384
emr-6.5.0-latest 385
電腦 -6.5.0-20240321 385
emr-6.5.0-20221219 386
emr-6.5.0-20220802 386
emr-6.5.0-20211119 386
6.4.0 版 386
emr-6.4.0-latest 387
電腦 -6.4.0-20240321 388
emr-6.4.0-20221219 388
emr-6.4.0-20210830 388
6.3.0 版 388
emr-6.3.0-latest 390
電腦 -6.3.0-20240321 390
emr-6.3.0-20220802 390
emr-6.3.0-20211008 390
emr-6.3.0-20210802 390

emr-6.3.0-20210429 391
6.2.0 版 391
emr-6.2.0-latest 392
電腦 -6.2.0-20240321 392
emr-6.2.0-20220802 393
emr-6.2.0-20211008 393
emr-6.2.0-20210802 393
emr-6.2.0-20210615 393
emr-6.2.0-20210129 394
emr-6.2.0-20201218 394
emr-6.2.0-20201201 394
5.36.0 版 394
emr-5.36.0-latest 395
埃姆尔 -5.36.0-20240321 396
emr-5.36.0-20221219 396
emr-5.36.0-20220620 396
emr-5.36.0-20220525 396
5.35.0 版 397
emr-5.35.0-latest 398
埃姆尔 -5.35.0-20240321 398
emr-5.35.0-20221219 398
emr-5.35.0-20220802 398
emr-5.35.0-20220307 399
5.34 版 399
emr-5.34.0-latest 400
埃姆尔 -5.34.0-20240321 400
emr-5.34.0-20220802 400
emr-5.34.0-20211208 401
5.33.0 版 401
emr-5.33.0-latest 402
埃姆尔 -5.33.0-20240321 402
emr-5.33.0-20221219 403
emr-5.33.0-20220802 403
emr-5.33.0-20211008 403
emr-5.33.0-20210802 403
emr-5.33.0-20210615 404

emr-5.33.0-20210323 404
5.32.0 版 404
emr-5.32.0-latest 405
埃姆尔 -5.32.0-20240321 406
emr-5.32.0-20220802 406
emr-5.32.0-20211008 406
emr-5.32.0-20210802 406
emr-5.32.0-20210615 407
emr-5.32.0-20210129 407
emr-5.32.0-20201218 407
emr-5.32.0-20201201 407
文件歷史紀錄 408
..... cdx

什麼是 Amazon EMR on EKS？

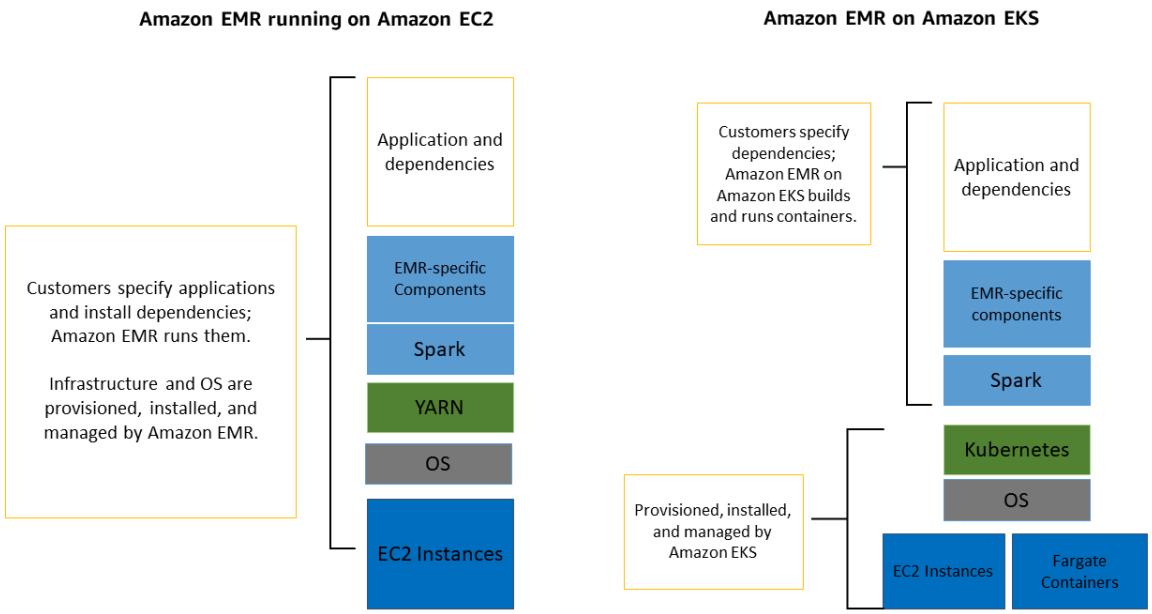
Amazon EMR on EKS 為 Amazon EMR 提供了一個部署選項，可讓您在 Amazon Elastic Kubernetes Service (Amazon EKS) 上執行開放原始碼大數據框架。使用此部署選項，您可以專注於執行分析工作負載，同時 Amazon EMR on EKS 可為開放原始碼應用程式建置、設定和管理容器。

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。可以使用 Amazon EMR Studio 和 AWS SDK 或者 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。使用 Amazon EC2 和 AWS Fargate，可以啟用多個運算資源，以滿足效能、營運或財務需求。

下圖表示 Amazon EMR 的兩種不同部署模型。



主題

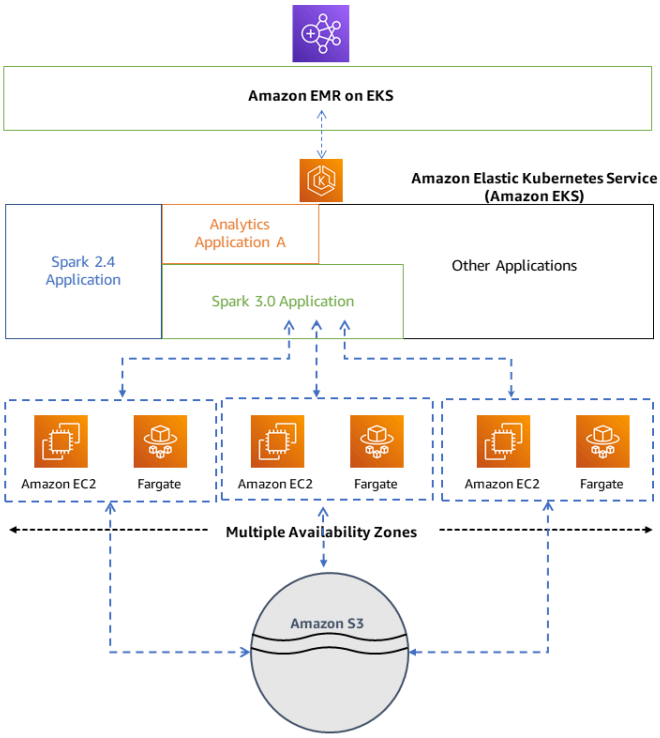
- 架構
- 概念
- 元件如何一起工作

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 將執行作業所需的 Amazon EC2 和 AWS Fargate 的運算資源帶到線上。

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。



概念

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 命名空間可以將 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

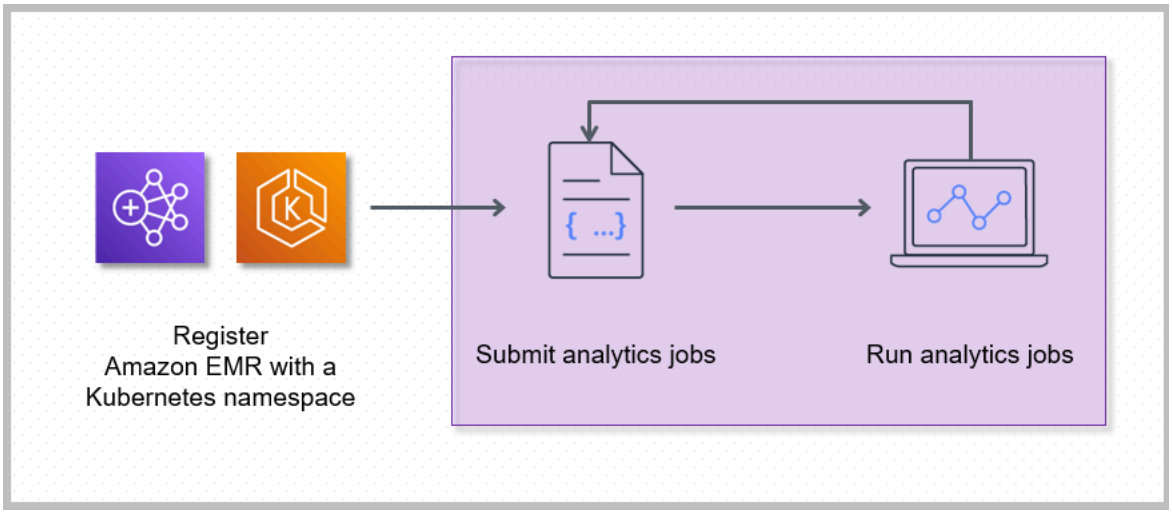
Amazon EMR 容器是 Amazon EMR on EKS 的 API 名稱。`emr-containers` 字首可用於下列情況：

- 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。

元件如何一起工作

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 將作業提交至虛擬叢集。



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。

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。

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。開始之前，請確定您已完成設置 Amazon EMR EKS 所述的步驟。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。

```
{
    "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:::DOC-EXAMPLE-BUCKET-OUTPUT",
                "arn:aws:s3:::DOC-EXAMPLE-BUCKET-OUTPUT/*",
                "arn:aws:s3:::DOC-EXAMPLE-BUCKET-LOGGING",
                "arn:aws:s3:::DOC-EXAMPLE-BUCKET-LOGGING/*"
            ]
        },
        {
            "Sid": "WriteToLoggingAndOutputDataBuckets",
            "Effect": "Allow",
            "Action": [
                "s3:PutObject",
                "s3:DeleteObject"
            ],
            "Resource": [
                "arn:aws:s3:::DOC-EXAMPLE-BUCKET-OUTPUT/*",
                "arn:aws:s3:::DOC-EXAMPLE-BUCKET-LOGGING/*"
            ]
        },
        {
            "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。
  - *DOC-EXAMPLE-BUCKET-OUTPUT* - 您為輸出資料定義的儲存貯體。
  - *DOC-EXAMPLE-BUCKET-LOGGING* -您為日誌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://DOC-EXAMPLE-BUCKET-OUTPUT/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://DOC-EXAMPLE-BUCKET-LOGGING"
    }
  }
}'
```

將在 s3://*DOC-EXAMPLE-BUCKET-OUTPUT*/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。如需更多詳細資訊，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://DOC-EXAMPLE-BUCKET-LOGGING"
    }
  }
}'
```

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。

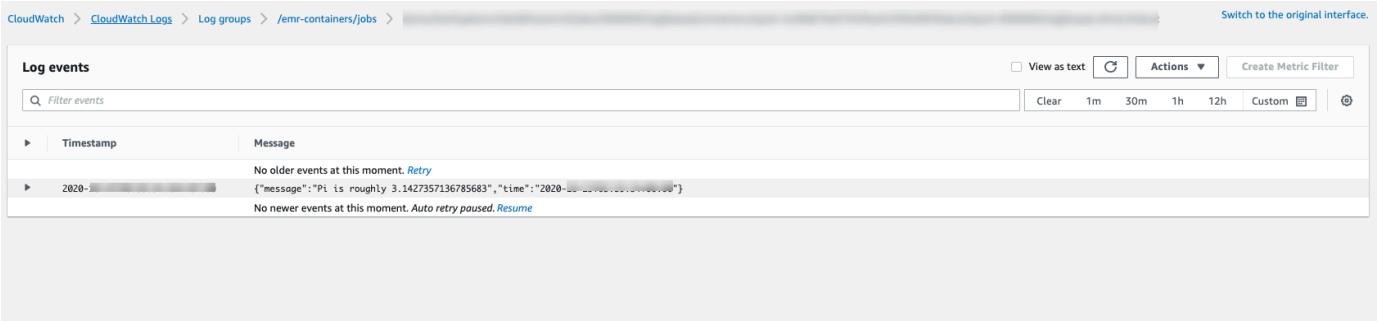
```
CREATE DATABASE demo;
CREATE EXTERNAL TABLE IF NOT EXISTS demo.amazonreview( marketplace string,
 customer_id string, review_id  string, product_id  string, product_parent  string,
 product_title  string, star_rating  integer, helpful_votes  integer, total_votes
  integer, vine  string, verified_purchase  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。

鏈接到有關 EKS 最佳實踐指南的 Amazon EMR GitHub

我們已使用開放原始碼社群協同合作來建置 Amazon EMR on EKS 最佳實務指南，因此我們可以快速迭代並針對各種使用案例提供建議。建議您針對這些章節使用 Amazon EMR on EKS 最佳實務指南。選擇每個部分中的鏈接以轉到該 GitHub站點。

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 Secrets Manager 來儲存機密。

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：提供不同方法來設定 AWS Glue 型錄。

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 AWS Outposts 執行 Amazon EMR on EKS

為 Amazon EMR 定制碼頭圖像 EKS

您可以EMR在 Amazon 上EKS使用自定義 Docker 圖像。EMR在EKS執行階段中自訂 Amazon 映像檔具有以下優點：

- 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 執行階段的公開散佈中廣泛使用。
- 將 Amazon EMR 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
- 如何選擇基本映像 URI
- 考量事項

如何自訂 Docker 映像檔

採取以下步驟為 Amazon EMR 定制碼頭圖像。EKS

- 必要條件
- 步驟 1：從 Amazon 彈性容器註冊表（AmazonECR）檢索基本映像
- 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：EMR使用自訂映像在 Amazon 中提交 Spark 工作負載

以下是在自訂 Docker 映像檔時可能需要考慮的其他選項：

- 為互動端點自訂 Docker 映像檔
- 使用多架構映像

必要條件

- 完成 Amazon EMR 的設置 Amazon EMR EKS步驟EKS。
- 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 彈性容器註冊表（AmazonECR）檢索基本映像

基本映像檔包含 Amazon EMR 執行階段和用於存取其他 AWS 服務的連接器。對於 Amazon EMR 6.9.0 及更高版本，您可以從 Amazon ECR 公共畫廊獲取基本圖像。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 7.2.0 版本，下列`docker pull`命令會取得最新的標準基本映像。您可以替換`emr-7.2.0:latest``emr-7.2.0-spark-rapids:latest`為檢索具有 Nvidia RAPIDS 加速器的圖像。也可以將 `emr-7.2.0:latest` 取代為 `emr-7.2.0-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-7.2.0:latest
```

如果您想要擷取 Amazon EMR 6.9.0 或其他版本的基本映像，或者想要從每個區域的 Amazon ECR 登錄帳戶擷取，請使用下列步驟：

1. 選擇基本影像URI。影像會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 如何選擇基本映像 URI。

2. 登入存放基本映像的 Amazon ECR 儲存庫。Replace (取代) *895885662937* 以及 *us-west-2* 使用 Amazon ECR 註冊表帳戶和您選擇的 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。Replace (取代) *emr-6.6.0:latest* 使用您選擇的容器圖像標籤。

   ```
   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

請按照以下步驟自定義從 Amazon 提取的基本圖像ECR。

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EMR在EKS自訂映像上使用 Amazon CLI 來檢查映像檔是否具有必要的檔案結構以及在 Amazon EMR 上執行的正確組態EKS。

> **Note**
>
> EKS自定義圖像EMR上的 Amazon CLI 無法確認您的圖像沒有錯誤。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. EMR在EKS自定義映像上下載並安裝 Amazon CLI。如需詳細資訊，請參閱 Amazon EMR 上的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指定需要驗證的本地圖像URI。這可以是影像URI、您為影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。EKS自定義圖像CLI版本EMR上的當前 Amazon 僅支持 Spark 運行時映像。

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 映像發佈到您的 Amazon ECR 註冊表。

1. 執行下列命令以建立用於ECR存放 Docker 映像的 Amazon 儲存庫。提供儲存庫的名稱，例如：*emr6.6_custom_repo*。 取代 *us-west-2* 與您的區域。

   ```
   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 使用者指南中的對預設登錄進行驗證。

3. 標記映像並將其發佈到您建立的 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
```

有關更多信息，請參閱 Amazon ECR 用戶指南ECR中的將圖像推送到 Amazon。

步驟 5：EMR使用自訂映像在 Amazon 中提交 Spark 工作負載

建立並發佈自訂映像後，您可以使用自訂映像提交任務 Amazon EMR。EKS

首先，建立 start-job-run-request .json 檔案並指定`spark.kubernetes.container.image`參數來參考自訂影像，如下列範例JSON檔案所示。

> **Note**
>
> 您可以使用`local://`方案來引用自定義圖像中可用的文件，如下面的JSON代碼片段中的參`entryPoint`數。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例子中，替換 *emr-6.6.0-latest* 與您的 Amazon EMR 發布版本。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 Amazon EMR 發行版本及其影像標籤的詳細資訊，請參閱如何選擇基本映像 URI。

> **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。這有助於確保您在從 EMR Studio 執行互動式工作負載時具有所需的相依性。

1. 請依照上述步驟 1-4 自訂 Docker 映像檔。對於 Amazon EMR 6.9.0 版本及更高版本，您可以URI從 Amazon ECR 公共畫廊獲取基本圖像。對於 Amazon EMR 6.9.0 之前的版本，您可以在每個 Amazon ECR 註冊表帳戶中獲取圖像 AWS 區域，唯一的區別是您的 Docerfile URI 中的基本映像。基本圖像的格式URI如下：

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

   您需要在基本映像`notebook-spark`中使用URI，而不是`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 如何選擇基本映像 URI。

   > **Note**
   >
   > 目前僅支援基本映像檔的覆寫，並且不支援引入除基礎映像檔 AWS 以外的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，創建一個JSON名為`custom-image-managed-endpoint.json`以下內容的文件。

   ```
   {
       "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式端點，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 Connect 到互動式端點。如需詳細資訊，請參閱從 Studio 連接。

使用多架構映像

Amazon EMR 上EKS支持 Amazon 彈性容器註冊表（AmazonECR）的多架構容器映像。如需詳細資訊，請參閱為 Amazon ECR 簡介多架構容器映像。

EKS自訂映像檔EMR上的 Amazon 同時支援以 AWS 重力為基礎的EC2執行個體和非重力型執行個體。EC2ECR以重力為基礎的映像檔會與非重力式映像儲存在 Amazon 相同的映像儲存庫中。

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 介紹多架構容器映像中的說明ECR建立多架構映像。

> **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，請參閱運用多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。在具有 AWS 重力型和非重力型執行個體的異質叢集中，執行個體會根據擷取映像檔的EC2執行個體架構判斷正確的架構映像檔。

如何選擇基本映像 URI

> **Note**
>
> 對於 Amazon EMR 6.9.0 和更高版本，您可以從 Amazon ECR 公共圖庫擷取基本映像，因此您不需要直接按照本頁面上URI的說明建構基本映像。若要尋找基本映像檔的容器映像標籤，請參閱版本說明頁面，瞭解 Amazon EMR on 的對應版本EKS。

您可以選擇的基本 Docker 映像存儲在 Amazon 彈性容器註冊表（AmazonECR）中。影像會URI遵循以下格式：*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-7.2.0:latest
```

互動式端點URI的影像會遵循下列格式：*ECR-registry-account*.dkr.ecr.*Region*.amazonaws.com/notebook-spark/*container-image-tag*，如下列範例所示。您需要在基本映像notebook-spark中使用URI，而不是spark。

```
895885662937.dkr.ecr.us-west-2.amazonaws.com/notebook-spark/emr-7.2.0:latest
```

同樣地，對於互動式端點的非 SPARK 映python3像，映像URI為*ECR-registry-account*.dkr.ecr.*Region*.amazonaws.com/notebook-python/*container-image-tag*。下列範例URI格式正確：

```
895885662937.dkr.ecr.us-west-2.amazonaws.com/notebook-python/emr-7.2.0:latest
```

若要尋找基本映像檔的容器映像標籤，請參閱版本說明頁面，瞭解 Amazon EMR on 的對應版本 EKS。

Amazon ECR 註冊表帳戶按區域

若要避免高度網路延遲，請從最接近的位置擷取基本映像檔 AWS 區域。根據下表，選取與您從中提取映像的區域對應的 Amazon ECR 登錄帳戶。

區域	Amazon ECR 註冊賬戶
ap-northeast-1	059004520145
ap-northeast-2	996579266876
ap-south-1	235914868574
ap-southeast-1	671219180197
ap-southeast-2	038297999601
ca-central-1	351826393999
eu-central-1	107292555468
eu-north-1	830386416364
eu-west-1	483788554619
eu-west-2	118780647275
eu-west-3	307523725174
sa-east-1	052806832358
us-east-1	755674844232

區域	Amazon ECR 註冊賬戶	
us-east-2	711395599931	
us-west-1	608033475327	
us-west-2	895885662937	

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，請遵循這些最佳實務：

- 安全是 AWS 與您之間共同承擔的責任。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 在EKS安裝在圖像的配置頂部的文件, 如`spark-defaults.conf`, 在運行時. 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 在運行時EKS安裝某些文件夾。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。如需如何使用選取的 Docker 儲存庫設定 Amazon EKS 叢集身份驗證的詳細資訊，請參閱從私有登錄提取映像。

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

- Flink Kubernetes Operator
- Native Kubernetes
- EKS使用阿帕奇 Flink EMR 在 Amazon 上定制碼頭圖像
- 監控 Flink Kubernetes Operator 和 Flink 作業
- 作業彈性
- 針對 Flink 應用程式使用 Autoscaler
- 維護和疑難排解
- Amazon EMR 支持的版本EKS與阿帕奇 Flink

Flink Kubernetes Operator

以下幾頁說明如何設定和使用 Flink Kubernetes 運算子在 Amazon 上執行 Flink 任務。EMR EKS

主題

- 在 Amazon 上設置 Flink 庫伯內特斯運營商 EMR EKS
- 在 Amazon 上開始使用 Flink Kubernetes 運營商 EMR EKS
- 執行 Flink 應用程式
- 安全
- 在 Amazon 上卸載 Flink 庫伯內特斯運營商 EMR EKS

在 Amazon 上設置 Flink 庫伯內特斯運營商 EMR EKS

在 Amazon 上安裝 Flink Kubernetes 操作員之前，請先完成以下任務以進行設置。EKS如果您已經註冊了 Amazon Web Services（AWS）並使用 AmazonEKS，那麼您幾乎可以EMR在上使用 Amazon

EKS。完成以下任務，即可在 Amazon EKS 上為 Flink 操作員進行設定。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 AWS CLI— 如果您已經安裝了 AWS CLI，請確認您擁有最新版本。
- 安裝 eksctl — eksctl 是您用來與 Amazon 通信的命令行工具。EKS
- 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。
- 設定 Amazon EKS 叢集 — 按照以下步驟建立具有 Amazon 節點的新 Kubernetes 叢集。EKS
- 選擇 Amazon EMR 版本標籤 (6.13.0 版或更高版本) — Amazon 6.13.0 及更高版本支援 Flink Kubernetes 營運商。EMR
- 在 Amazon EKS 叢集上啟用服務帳戶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。如果您想要為運算子設定不同的IAM角色，您可以建立單獨的角色。
- 更新操作員執行角色的信任政策。您必須針對想要用於 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：

```
{
   "Version": "2012-10-17",
   "Statement": [
       {
           "Effect": "Allow",
           "Principal": {
               "Federated": "arn:aws:iam::ACCOUNT_ID:oidc-provider/OIDC_PROVIDER"
           },
           "Action": "sts:AssumeRoleWithWebIdentity",
           "Condition": {
               "StringLike": {
                   "OIDC_PROVIDER:sub": "system:serviceaccount:NAMESPACE:emr-
containers-sa-flink-operator"
               }
           }
       }
   ]
}
```

在 Amazon 上開始使用 Flink Kubernetes 運營商 EMR EKS

本主題可協助您EKS透過部署 Flink 部署，開始在 Amazon 上使用 Flink Kubernetes 運算子。

安裝該 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 群集一次）以啟用添加 webhook 組件。

```
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=7.2.0 # 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`將與 Amazon EMR EKS 發行版本的 Flink 運算子版本相對應。如需詳細資訊，請參閱Amazon EMR 支持的版本EKS與阿帕奇 Flink。

```
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-7.2.0    x.y.z-amzn-n
```

執行 Flink 應用程式

使用 Amazon EMR 6.13.0 及更高版本，您可以在 Amazon 上的應用程序模式下使用 Flink Kubernetes 操作員運行 Flink 應用程序。EMR EKS使用 Amazon EMR 6.15.0 及更高版本，您也可以在工作階段模式下執行 Flink 應用程式。本頁說明EMR在 Amazon 上EKS執行 Flink 應用程式時可使用的兩種方法。

> **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 “安裝該 Operator” 中的步驟。

Application mode

使用 Amazon EMR 6.13.0 及更高版本，您可以在 Amazon 上的應用程序模式下使用 Flink Kubernetes 操作員運行 Flink 應用程序。EMR EKS

1. 使用以下範例內容，建立 `FlinkDeployment` 定義檔案 `basic-example-app-cluster.yaml`：

```
apiVersion: flink.apache.org/v1beta1
kind: FlinkDeployment
metadata:
```

```
  name: basic-example-app-cluster
spec:
  flinkConfiguration:
    taskmanager.numberOfTaskSlots: "2"
    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。請記得清除為此工作建立的 S3 成品，例如檢查點、高可用性、儲存指向中繼資料和日誌。 CloudWatch

如需有關透過 Flink Kubernetes 運算子將應用程式提交至 Flink 的詳細資訊，請參閱上資料夾中的 Flink Kuber netes 運算子範例。apache/flink-kubernetes-operator GitHub

Session mode

使用 Amazon EMR 6.15.0 及更高版本，您可以使用 Flink Kubernetes 運算子在 Amazon 上的工作階段模式下執行 Flink 應用程式。EMR EKS

1. 使用以下範例內容，建立 FlinkDeployment 定義檔案 basic-example-session-cluster.yaml：

```
apiVersion: flink.apache.org/v1beta1
kind: FlinkDeployment
metadata:
  name: basic-example-session-cluster
spec:
  flinkConfiguration:
    taskmanager.numberOfTaskSlots: "2"
    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。請記得清除為此工作建立的 S3 成品，例如檢查點、高可用性、儲存指向中繼資料和日誌。 CloudWatch

安全

RBAC

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 會在您安裝操作員時預設建立兩個角色。

Operator 角色

我們使用操作員角色`flinkdeployments`來管理創建和管理每 JobManager 個 Flink 任務和其他資源（如服務）。

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

會 JobManager 使用工作角色來建立 TaskManagers 和管理 ConfigMaps 每個工作。

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

在 Amazon 上卸載 Flink 庫伯內特斯運營商 EMR EKS

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
   ```

Native 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
- 原生 Kubernetes 的 Flink JobManager 服務帳戶安全性需求

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 AWS CLI — 如果已經安裝 AWS CLI，請確認擁有最新版本。
- 設定 Amazon EKS 叢集 — 按照以下步驟在 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看 TaskManager 網繭的適當權限。如需詳細資訊，請參閱原生 Kubernetes 的 Flink JobManager 服務帳戶安全性需求。
- 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

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
```

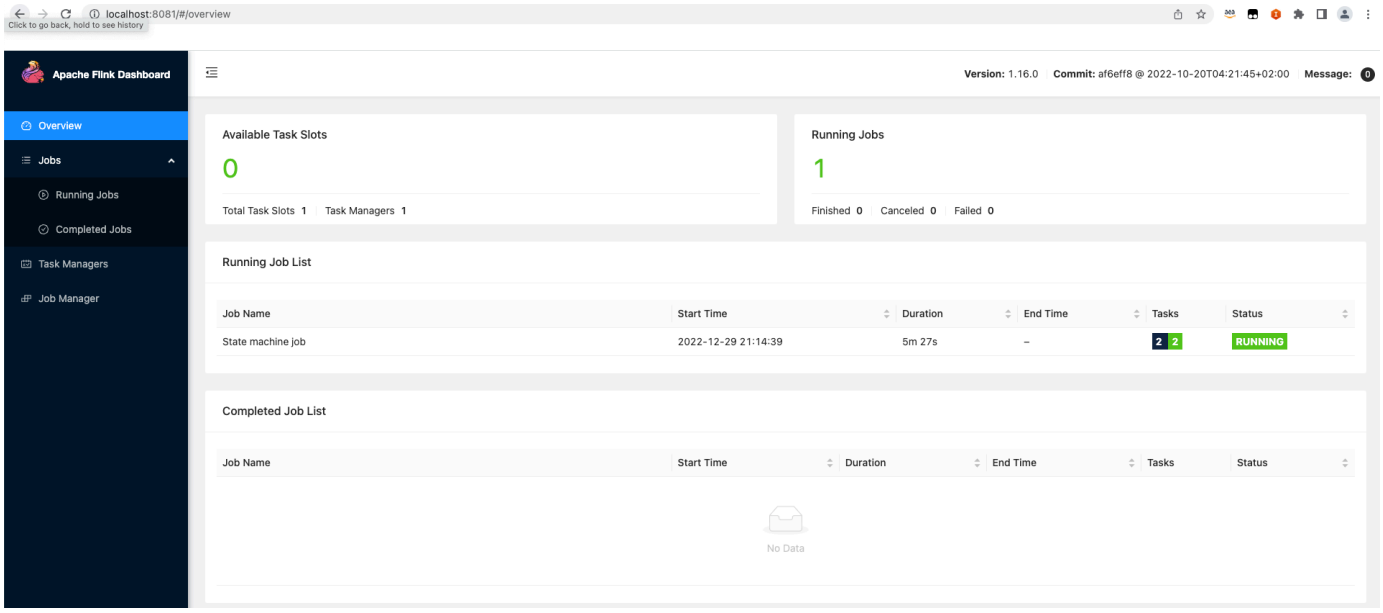
6. 檢查已建立的 Kubernetes 資源。

```
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。

原生 Kubernetes 的 Flink JobManager 服務帳戶安全性需求

Flink JobManager 網繭會使用 Kubernetes 服務帳戶來存取 Kubernetes API 伺服器，以建立和監視網繭。 TaskManager JobManager 服務帳戶必須具有適當的權限才能建立/刪除 TaskManager 網繭，並允許監視領導者 ConfigMaps 擷取叢集 ResourceManager 中的 JobManager 和位址。 TaskManager

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:
  - "*"
```

EKS使用阿帕奇 Flink EMR 在 Amazon 上定制碼頭圖像

以下各節說明如何EMR在 Amazon 上EKS自訂 Docker 映像檔。

主題

- 自訂 Flink 和 FluentD 的泊塢視窗影像

自訂 Flink 和 FluentD 的泊塢視窗影像

採取以下步驟，EKS使用 Apache Flink 或 FluentD 圖像為 Amazon EMR 自定義碼頭圖像。

主題

- 必要條件
- 步驟 1：從 Amazon 彈性容器註冊表中檢索基本映像
- 步驟 2：自訂基礎映像
- 步驟 3：發佈您的自訂影像
- 步驟 4：EMR使用自訂映像在 Amazon 中提交 Flink 工作負載

必要條件

在您自訂 Docker 映像檔之前，請確定您已完成下列先決條件：

- 按步驟完成 Amazon Flink Kubernetes 運營商的設置。EMR EKS
- 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 彈性容器註冊表中檢索基本映像

基本映像檔包含 Amazon EMR 執行階段和您需要存取其他連接器的連接器 AWS 服務。如果您在 Flink 6.14.0 或更高版本EMR上EKS使用 Amazon，則可以從 Amazon ECR 公共圖庫獲取基本圖像。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 發行版本，下列`docker pull`命令會傳回最新的標準基本映像。以您想要的發行版本取代`emr-6.14.0:latest`。

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

以下是 Flink 畫廊圖片和 Fluentd 畫廊圖片的連結：

- emr-on-eks/燧石/埃姆爾 -6.14.0-氟鏈
- emr-on-eks/流感/emr-6.14.0 (

步驟 2：自訂基礎映像

以下步驟說明如何自訂您從 Amazon 提取的基本映像檔ECR。

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Dockerfile使用您從中提取的容器映像public.ecr.aws/emr-on-eks/flink/emr-7.2.0-flink:latest。

```
FROM public.ecr.aws/emr-on-eks/flink/emr-7.2.0-flink:latest
USER root
### Add customization commands here ####
USER hadoop:hadoop
```

   如果您正在使用，請使用以下配置Fluentd。

```
FROM public.ecr.aws/emr-on-eks/fluentd/emr-7.2.0: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。下面的命令演示了如何安裝 Python 庫。

```
FROM public.ecr.aws/emr-on-eks/flink/emr-7.2.0-flink:latest
USER root
RUN pip3 install --upgrade boto3 pandas numpy // For python 3
USER hadoop:hadoop
```

4. 在建立位置的相同目錄中DockerFile，執行下列命令以建立 Docker 映像檔。您在-t旗標後面提供的欄位即為映像檔的自訂名稱。

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

步驟 3：發佈您的自訂影像

現在，您可以將新的 Docker 映像發佈到您的 Amazon ECR 註冊表。

1. 運行以下命令以創建一個 Amazon ECR 存儲庫來存儲您的 Docker 映像。提供儲存庫的名稱，例如如emr_custom_repo.需詳細資訊，請參閱 Amazon 彈性容器登錄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 彈性容器登錄使用者指南中的對預設登錄進行驗證。

```
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 彈性容器登錄使用者指南ECR中的將映像推送至Amazon。

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

步驟 4：EMR使用自訂映像在 Amazon 中提交 Flink 工作負載

對您的FlinkDeployment規格進行下列變更以使用自訂映像檔。若要這麼做，請在部署規格的spec.image行中輸入您自己的映像檔。

```
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"
```

若要在 Fluentd 工作中使用自訂映像檔，請在部署規格的monitoringConfiguration.image行中輸入您自己的映像檔。

```
  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。

主題

- 使用 Amazon Managed Service for Prometheus 來監控 Flink 作業
- 使用 Flink UI 監控 Flink 作業
- 使用監控組態來監控 Flink Kubernetes Operator 和 Flink 作業

使用 Amazon Managed Service for Prometheus 來監控 Flink 作業

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覆寫 Prometheus 設定的 Amazon 受管服務。確保使用自己的角色。

```
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
```

此命令會自動安裝 Prometheus 記者在端口 9999 運營商。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 作業

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。此儀表板提供有關工作狀態、數量 TaskManagers以及工作指標和記錄的資訊。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. 使用此`kubectl port-forward`命令將本機連接埠轉寄至 Flink Web 儀表板在 Flink 應用程式網繭中執行的 TaskManager 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：

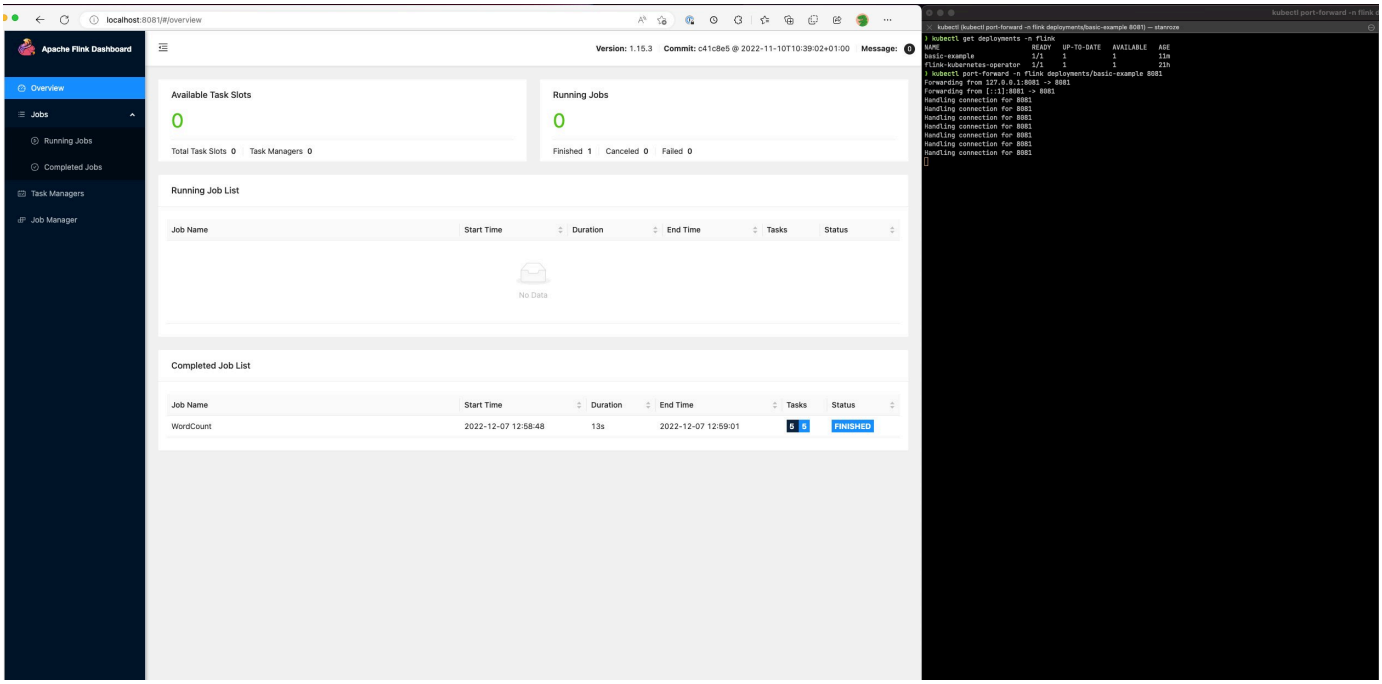
   ```
   kubectl get deployments -n flink-namespace
   NAME                        READY   UP-TO-DATE   AVAILABLE   AGE
   basic-example               1/1       1            1           11m
   flink-kubernetes-operator   1/1       1            1           21h
   ```

   ```
   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。此儀表板會顯示執行中 Flink 應用程式的相關資訊，例如工作狀態 TaskManagers、數量以及工作的指標和記錄。



使用監控組態來監控 Flink Kubernetes Operator 和 Flink 作業

監控配置可讓您輕鬆地將 Flink 應用程式和操作員日誌的日誌存檔設置到 S3 和/或 CloudWatch （您可以選擇一個或兩個）。這樣做會將 FluentD 附屬程式新增至您的 JobManager 和 TaskManager Pod，然後將這些元件的記錄檔轉送至已設定的接收器。

> **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 在 Amazon 上設置 Flink 庫伯內特斯運營商 EMR EKS 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`— 設置轉發到的配置密鑰 CloudWatch。
  - `logGroupName`(必要) — 您要傳送記錄檔的 CloudWatch 目標記錄群組名稱 (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。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。您可以啟用 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`— 設置轉發到的配置密鑰 CloudWatch。
  - `logGroupName`(必要) — 您要傳送防護記錄檔的記錄群組名稱。 CloudWatch 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。中的格 CloudWatch 式如下：

```
${logStreamNamePrefix}/${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。

作業彈性

以下各節概述了如何讓您的 Flink 作業更加可靠且高度可用。

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)

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。您只需在 FlinkDeployment 規範中提供 Jaruri 的 S3 位置即可。

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

下列範例會示範如何將此功能用於 PyFlink 工作。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 特定部分。

FlinkDeployment 範例規格：

```
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)。
- 默認情況下，我們fs.s3a.aws.credentials.provider在flink-conf.yaml文件中設置，這是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) 允許工作繼續進行，即使遇到暫時性錯誤並發生 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。若未啟用 HA，Kubernetes 將會重新啟動您的工作 JobManager，但您的工作將會以全新工作的形式開始，而且會失去進度。設定 HA 之後，我們可以告知 Kubernetes 將 HA 中繼資料儲存在持續性儲存體中，以便在中發生暫時性故障時進行參考，JobManager 然後從上次成功的檢查點繼續我們的工作。

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`(選擇性，預設值為 2)：將此數字設定為大於 1 會建立其他待命狀態， JobManagers 並可加快工作的復原速度。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。

本頁說明 Amazon EMR Flink 在任務復原或擴展操作期間縮短作業重新啟動時間的一些方法。

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：

- J ust-in-time 檢查點 — Flink 串流作業可回應 Spot 執行個體中斷、執行 just-in-time (JIT) 正在執行的工作的檢查點，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。

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。

必要條件

若要使用正常的解析功能，請在執行 Apache Flink 的 EKS 叢集上的 Amazon EMR 上建立並執行串流任務。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
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"
  ...
```

此組態使用 Amazon 最新版本的預設值EMR。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中包含前置詞。如果使用 7.1.0 或更低版本，則必須在每個模型組態之前使用前置詞。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`— 指定是否啟用自動配置器執行頂點縮放。預設值為 `true`。如果停用此組態，自動配置器只會收集測量結果並評估每個頂點的建議平行處理原則，但不會升級工作。
- `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。預設值為 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`，表示在觸發擴展動作之前，允許與目標使用率有 30% 的偏差。
- `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。

自動配置器參數自動調諧

> **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`來啟用重新調整時間估算。重新調整時間估計具有與 Amazon EMR 自動調整相同的功能，因此您不必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 自動調諧。

7.2.0 and higher

Amazon EMR 7.2.0 及更高版本會衡量實際所需的重新啟動時間，以套用自動調度資源決策。在 7.1.0 及更低版本中，您必須使用組態`job.autoscaler.restart.time`來手動設定預估的重新啟動時間上限。通過使用配置`job.autoscaler.restart.time-tracking.enabled`，您只需要輸入第一個擴展的重新啟動時間。之後，操作員會記錄實際的重新啟動時間，並將其用於後續的縮放。

若要啟用此追蹤，請使用下列命令：

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

以下是重新調整時間估計的相關組態。

組態	必要	預設	描述
重新啟用時間跟踪	否	False	指出 Flink 自動配置器是否應隨時間自動調整配置，以最佳化縮放分析。請注意，自動配置器只能自動調整自動配置器參數。`restart.time`
工作自動清理器. 重新啟動.	否	5m	Amazon EMR EKS 使用的預期重新啟動時間，直到操作員可以判斷先前擴展的實際重新啟動時間為止。
重新啟動. 時間跟踪. 限制	否	15m	設定為時，觀察到的重新啟動時`job.autoscaler.restart.time-tracking.enabled` 間上限`true`。

以下是您可以用來嘗試重新調整時間估計的範例部署規格：

```
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-7.2.0-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 指令碼上傳至您的 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()
```

若要確認重新調整時間估計是否有效，請確定已啟用 Flink 運算子的DEBUG層級記錄。下面的例子演示了如何更新舵圖文件values.yaml。然後重新安裝更新後的掌舵圖，並再次執行 Flink 工作。

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

獲取您的領導者網繭的名稱。

```
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量度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"
```

您應該會看到類似下列的記錄檔。請注意，只有第一個縮放使用 `job.autoscaler.restart.time`。隨後的縮放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

開源內置 Flink 自動配置器使用許多指標來做出最佳擴展決策。但是，它用於計算的預設值應適用於大多數工作負載，並且可能不適用於特定工作。Flink 操作員EKS版本中添加到 Amazon 的自動調整功能會查看在特定捕獲指標EMR上觀察到的歷史趨勢，然後相應地嘗試計算針對給定任務量身定制的最佳值。

組態	必要	預設	描述
自動調整。操作員。自動調整。	否	False	指出 Flink 自動配置器是否應隨時間自動調整配置，以最佳化自動配置器縮放分析。目前，自動配置器只能

組態	必要	預設	描述
			自動調整自動配置器參數。`restart.time`
自動調整器. 度量. 歷史記錄.	否	3	指示自動配置器在指EKS標配置地圖EMR上在 Amazon 中保留的指EKS標EMR上的歷史記錄數量。
作業自動調整器. 量度. 重新計數	否	3	指出自動配置器在開始計算指定工作的平均重新啟動時間之前，執行的重新啟動次數。

若要啟用自動調整，您必須完成下列步驟：

- 設定`kubernetes.operator.job.autoscaler.autotune.enable:`為 `true`
- 設定`metrics.job.status.enable:`為 `TOTAL_TIME`
- 遵循針對 Flink 應用程式使用自動配置器的設定，以啟用自動調度資源。

以下是您可以用來嘗試自動調整的範例部署規格。

```
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 指令碼上傳至您的 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()
```

若要驗證自動調諧器是否正常運作，請使用下列指令。請注意，您必須針對 Flink 操作員使用自己的引線網繭資訊。

首先得到你的領導者的名字。

```
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"
```

一旦你有了你的領導網繭的名稱，你可以運行以下命令。

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

您應該會看到類似下列的記錄檔。

```
[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
```

維護和疑難排解

以下各節將概述如何維護長時間執行的 Flink 作業，並提供如何對一些常見問題進行疑難排解的指引。

遷移 Flink 應用程式

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。如果工作處於不健康狀態，則會使用最後一個檢查點 (除非 kubernetes.operator.job.upgrade)。 last-state-fallback啟用被設置為假）。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
   ```

Amazon EMR 支持的版本EKS與阿帕奇 Flink

阿帕奇 Flink 可與以下 Amazon EKS 發EMR布。如需所有可用版本的資訊，請參閱 Amazon EMR 上 EKS發布。

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 執行作業

任務執行是您在 EKS 上提交給 Amazon EMR 的一個工作單位，例如星火罐、 PySpark 指令碼或 SparkSQL 查詢。本主題提供使用管理任務執行 AWS CLI、使用 Amazon EMR 主控台檢視任務執行，以及疑難排解常見任務執行錯誤的概觀。

請注意，您無法在 EKS 上的 Amazon EMR 上運行 IPv6 星火任務

> **Note**
>
> 在使用 Amazon EMR on EKS 提交作業執行之前，必須完成 設置 Amazon EMR 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
- 在 Amazon EMR 上使用阿帕奇利維 EKS
- 管理 Amazon EMR on EKS 作業執行
- 使用作業提交器分類
- 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

主題

- 設置 Amazon EMR EKS
- 使用 StartJobRun 提交作業執行

設置 Amazon EMR EKS

完成下列任務，即可EMR在 Amazon 上進行設定EKS。如果您已經註冊了 Amazon Web Services（AWS）並且一直在使用 AmazonEKS，那麼您幾乎可以EMR在上使用 Amazon EKS。略過您已完成的任何任務。

> **Note**
>
> 您還可以按照 Amazon 在EKS研討會EMR上設置所有必要的資源，以在 Amazon EMR 上運行 Spark 任務EKS。該研討會還通過使用 CloudFormation 模板來創建入門所需的資源來提供自動化。如需其他範本和最佳做法，請參閱上的《EMR容器最佳做法指南》 GitHub。

1. 安裝 AWS CLI
2. 安裝 `eksctl`
3. 設置 Amazon EKS 群集
4. EMR在上啟用 Amazon 的叢集存取 EKS
5. 在EKS叢集上啟用服務帳戶的IAM角色 (IRSA)
6. 建立作業執行角色
7. 更新作業執行角色的信任政策
8. 授予使用者EMR對 Amazon 的存取權 EKS
9. 註冊 Amazon EKS 群集與 Amazon EMR

安裝 AWS CLI

您可以安裝 AWS CLI 適用於 macOS、Linux 或視窗的最新版本。

> **Important**
>
> 要EMR在上設置 AmazonEKS，您必須 AWS CLI 安裝最新版本的。

若要安裝或更新適 AWS CLI 用於 macOS

1. 如果您目前已 AWS CLI 安裝，請確定您已安裝的版本。

```
aws --version
```

2. 如果您有較早版本的 AWS CLI，請使用下列命令來安裝最新 AWS CLI 版本 2。如需其他安裝選項，或是要升級目前安裝的版本 2，請參閱在 macOS 上升級 AWS CLI 版本 2。

```
curl "https://awscli.amazonaws.com/AWSCLIV2.pkg" -o "AWSCLIV2.pkg"
sudo installer -pkg AWSCLIV2.pkg -target /
```

如果您無法使用 AWS CLI 版本 2，請確保您使用以下命令安裝了最新AWS CLI 版本 1 的版本。

```
pip3 install awscli --upgrade --user
```

若要安裝或更新 Linux 版 AWS CLI

1. 如果您目前已 AWS CLI 安裝，請確定您已安裝的版本。

```
aws --version
```

2. 如果您有較早版本的 AWS CLI，請使用下列命令來安裝最新 AWS CLI 版本 2。如需其他安裝選項或升級目前安裝的版本 2，請參閱在 Linux 上升級 AWS CLI 版本 2。

```
curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"
unzip awscliv2.zip
sudo ./aws/install
```

如果您無法使用 AWS CLI 版本 2，請確保您使用以下命令安裝了最新AWS CLI 版本 1 的版本。

```
pip3 install --upgrade --user awscli
```

若要安裝或更新視窗 AWS CLI

1. 如果您目前已 AWS CLI 安裝，請確定您已安裝的版本。

```
aws --version
```

2. 如果您有較早版本的 AWS CLI，請使用下列命令來安裝最新 AWS CLI 版本 2。如需其他安裝選項，或升級目前安裝的版本 2，請參閱在 Windows 上升級 AWS CLI 版本 2。

1. 下 AWS CLI MSI載適用於視窗 (64 位元) 的安裝AWSCLIV2程式 https://awscli.amazonaws.com/
2. 執行下載的MSI安裝程式，並依照螢幕上的指示操作 默認情況下， AWS CLI 安裝到C:\Program Files\Amazon\AWSCLIV2.

如果您無法使用 AWS CLI 版本 2，請確保您使用以下命令安裝了最新AWS CLI 版本 1 的版本。

```
pip3 install --user --upgrade awscli
```

設定您的 AWS CLI 認證

eksctl 和 AWS CLI 要求您在環境中配置 AWS 憑據。`aws configure` 命令是設定 AWS CLI 安裝以用於一般用途的最快方法。

```
$ aws configure
AWS Access Key ID [None]: <AKIAIOSFODNN7EXAMPLE>
AWS Secret Access Key [None]: <wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY>
Default region name [None]: <region-code>
Default output format [None]: <json>
```

當您輸入此命令時，會 AWS CLI 提示您輸入四個資訊：存取金鑰、秘密存取金鑰、 AWS 區域和輸出格式。此資訊會存放在名為 `default` 的設定檔 (設定集合)。除非您指定另一個命令，否則在執行命令時會使用此設定檔。如需詳細資訊，請參閱《使用指南》 AWS CLI中的 AWS Command Line Interface 〈配置〉。

安裝 `eksctl`

在 macOS、Linux 或 Windows 中安裝最新版本的 `eksctl` 命令列公用程式。有關更多資訊，請參閱 https://eksctl.io/。

> **Important**
>
> 我們建議您下載最新的 eksctl，因為 Amazon EMR 上的某些功能EKS需要更新的版本。如需詳細資訊，請參閱安裝 `eksctl`。

若要使用 Homebrew 在 macOS 上安裝或升級 eksctl

開始使用 Amazon EKS 和 macOS 的最簡單方法是使用自製軟件安裝 eksctl。eksctl 自製軟件配方安裝 eksctl 和 Amazon EKS 所需的任何其他依賴關係，例如 kubectl。配方也會安裝 aws-iam-authenticator，如果您沒有安裝 1.16.156 或更新 AWS CLI 版本，這是必要的。

1. 如果您還沒有在 macOS 上安裝 Homebrew，請使用下列命令進行安裝。

```
/bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/
master/install.sh)"
```

2. 安裝 Weaveworks Homebrew tap。

```
brew tap weaveworks/tap
```

3. 1. 安裝或升級 eksctl。
   - 使用下列命令安裝 `eksctl`。

     ```
     brew install weaveworks/tap/eksctl
     ```

   - 如果已安裝 `eksctl`，請執行下列命令進行升級。

     ```
     brew upgrade eksctl & brew link --overwrite eksctl
     ```

   2. 使用下列命令，測試您的安裝是否成功。必須擁有 `eksctl` 0.34.0 版或更高版本。

      ```
      eksctl version
      ```

使用 **`curl`** 在 Linux 上安裝或升級 **`eksctl`**

1. 使用下列命令下載並擷取最新版的 eksctl。

```
curl --silent --location "https://github.com/weaveworks/eksctl/releases/latest/
download/eksctl_$(uname -s)_amd64.tar.gz" | tar xz -C /tmp
```

2. 將擷取的二進位檔案移動到 /usr/local/bin。

```
sudo mv /tmp/eksctl /usr/local/bin
```

3. 使用下列命令，測試您的安裝是否成功。必須擁有 `eksctl` 0.34.0 版或更高版本。

```
eksctl version
```

使用 Chocolatey 在 Windows 上安裝或升級 **eksctl**

1. 如果您還沒有在 Windows 系統上安裝 Chocolatey，請參閱安裝 Chocolatey。
2. 安裝或升級 eksctl。
   - 使用下列命令，安裝二進位檔案。

     ```
     choco install -y eksctl
     ```
   - 如果已安裝，請執行下列命令來進行升級：

     ```
     choco upgrade -y eksctl
     ```
3. 使用下列命令，測試您的安裝是否成功。必須擁有 eksctl 0.34.0 版或更高版本。

   ```
   eksctl version
   ```

設置 Amazon EKS 群集

Amazon EKS 是一種受管服務，可讓您輕鬆地在上執行 Kubernetes， AWS 而無需安裝、操作和維護自己的 Kubernetes 控制平面或節點。請按照以下概述的步驟建立具有 Amazon 節點的新 Kubernetes 叢集。EKS

必要條件

> **Important**
>
> 在建立 Amazon EKS 叢集之前，請先完成 Amazon EKS VPC 和子網路要求和 Amazon EKS 使用者指南中的考量事項，以確保 Amazon EKS 叢集能夠按預期運作和擴展。

您必須安裝並設定下列建立和管理 Amazon EKS 叢集所需的工具和資源：

- 最新版本的 AWS CLI.
- kubectl 1.20 或更新版本。

- `eksctl` 的最新版本。

如需詳細資訊，請參閱 安裝 AWS CLI、安裝 `kubectl` 以及 安裝 `eksctl`。

使用建立 Amazon EKS 叢集 **`eksctl`**

請執行以下步驟使用建立 Amazon EKS 叢集`eksctl`。

> **Important**
>
> 若要快速開始使用，您可以使用預設設定建立EKS叢集和節點。但對於生產使用，建議您自訂叢集和節點的設定，以符合您的特定需求。如需所有設定和選項的清單，請輸入 `eksctl create cluster -h` 命令。如需詳細資訊，請參閱 `eksctl` 文件中的建立和管理叢集。

1. 創建一個 Amazon EC2 key pair。

   如果沒有現成的金鑰對，則可以執行以下命令來建立新金鑰對。使用您建立叢集所在的區域取代 `us-west-2`。

   ```
   aws ec2 create-key-pair --region us-west-2 --key-name myKeyPair
   ```

   將傳回的輸出儲存在本機電腦的檔案中。如需詳細資訊，請參閱 Amazon 執行個體EC2使用者指南中的建立或匯入 key pair。

   > **Note**
   >
   > 建立EKS叢集不需要 key pair。但是指定 key pair 允許您在創SSH建節點後訪問節點。只有在建立節點群組時，才能指定金鑰對。

2. 建立 EKS 叢集

   執行下列命令以建立EKS叢集和節點。Replace (取代) *`my-cluster`* 以及 *`myKeyPair`* 使用您自己的群集名稱和 key pair 名稱。Replace (取代) *`us-west-2`* 與您要在其中創建集群的區域。如需有關 Amazon EKS 支援區域的詳細資訊，請參閱 Amazon Elastic Kubernetes Service 端點和配額。

   ```
   eksctl create cluster \
   --name my-cluster \
   ```

```
--region us-west-2 \
--with-oidc \
--ssh-access \
--ssh-public-key myKeyPair \
--instance-types=m5.xlarge \
--managed
```

> **Important**
>
> 建立EKS叢集時，請使用 m5.xlarge 做為執行個體類型，或使用任何其他具有更高CPU記憶體和記憶體的執行個體類型。使用較低CPU或記憶體較 m5.xlarge 的執行個體類型，可能會因為叢集中的資源不足而導致工作失敗。若要查看建立的所有資源，請檢視 AWS Cloud Formation 主控台中名為 eksctl-*my-cluster*-cluster 的堆疊。

建立叢集和節點需要幾分鐘的時間。建立叢集和節點時，您會看到數行輸出。下列範例示範輸出的最後一行。

```
...
[#]  EKS cluster "my-cluster" in "us-west-2" region is ready
```

eksctl 已在 ~/.kube 中建立了 kubectl 組態檔案，或將新叢集的組態新增至 ~/.kube 中的現有組態檔案。

3. 檢視和驗證資源

執行下列命令，以檢視您的叢集節點。

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

下面顯示了範例輸出。

```
Amazon EC2 node output

NAME                                          STATUS   ROLES    AGE    VERSION
          INTERNAL-IP      EXTERNAL-IP     OS-IMAGE          KERNEL-VERSION
         CONTAINER-RUNTIME
ip-192-168-12-49.us-west-2.compute.internal    Ready    none     6m7s
 v1.18.9-eks-d1db3c   192.168.12.49    52.35.116.65    Amazon Linux 2
 4.14.209-160.335.amzn2.x86_64   docker://19.3.6
```

```
ip-192-168-72-129.us-west-2.compute.internal   Ready    none     6m4s
 v1.18.9-eks-d1db3c   192.168.72.129   44.242.140.21   Amazon Linux 2
 4.14.209-160.335.amzn2.x86_64   docker://19.3.6
```

如需詳細資訊，請參閱檢視節點。

使用下列命令，以檢視在叢集上執行的工作負載。

```
kubectl get pods --all-namespaces -o wide
```

下面顯示了範例輸出。

```
Amazon EC2 output

NAMESPACE     NAME                          READY   STATUS    RESTARTS   AGE     IP
               NODE                                                NOMINATED NODE
 READINESS GATES
kube-system   aws-node-6ctpm                1/1     Running   0          7m43s
 192.168.72.129   ip-192-168-72-129.us-west-2.compute.internal   none
 none
kube-system   aws-node-cbntg                1/1     Running   0          7m46s
 192.168.12.49    ip-192-168-12-49.us-west-2.compute.internal    none
 none
kube-system   coredns-559b5db75d-26t47      1/1     Running   0          14m
 192.168.78.81    ip-192-168-72-129.us-west-2.compute.internal   none
 none
kube-system   coredns-559b5db75d-9rvnk      1/1     Running   0          14m
 192.168.29.248   ip-192-168-12-49.us-west-2.compute.internal    none
 none
kube-system   kube-proxy-l8pbd              1/1     Running   0          7m46s
 192.168.12.49    ip-192-168-12-49.us-west-2.compute.internal    none
 none
kube-system   kube-proxy-zh85h              1/1     Running   0          7m43s
 192.168.72.129   ip-192-168-72-129.us-west-2.compute.internal   none
 none
```

如需有關此處所展示內容的詳細資訊，請參閱檢視工作負載。

使用 AWS Management Console 和建立EKS叢集 AWS CLI

您也可以使用 AWS Management Console 和 AWS CLI 來建立EKS叢集。請按照開始使用 Amazon 的步驟進行操作 EKS- AWS Management Console 和 AWS CLI。這種方式可讓您了解如何為EKS叢集建立每個資源，以及資源之間的互動方式。

> **Important**
>
> 為EKS叢集建立節點時，請使用 m5.xlarge 做為執行個體類型，或使用任何其他具有較高CPU記憶體和記憶體的執行個體類型。

建立EKS叢集 AWS Fargate

您也可以建立具有執行網繭的EKS叢集 AWS Fargate。

1. 若要使用 Fargate 上執行的網繭建立EKS叢集，請按照入門 AWS Fargate 使用 Amazon EKS 中概述的步驟進行操作。

   > **Note**
   >
   > Amazon EMR on EKS 需要在EKS叢集上執行任務DNS的核心。如果您只想在 Fargate 上執行網繭，則必須遵循更新核心DNS中的步驟。

2. 執行下列命令，以檢視您的叢集節點。

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

   下面顯示了範例 Fargate 輸出。

   ```
   Fargate node output

   NAME                                                  STATUS   ROLES    AGE
      VERSION              INTERNAL-IP       EXTERNAL-IP   OS-IMAGE         KERNEL-
   VERSION                   CONTAINER-RUNTIME
   fargate-ip-192-168-141-147.us-west-2.compute.internal   Ready    none
    8m3s    v1.18.8-eks-7c9bda   192.168.141.147   none          Amazon Linux 2
    4.14.209-160.335.amzn2.x86_64   containerd://1.3.2
   ```

```
fargate-ip-192-168-164-53.us-west-2.compute.internal    Ready    none
 7m30s    v1.18.8-eks-7c9bda    192.168.164.53    none          Amazon Linux 2
 4.14.209-160.335.amzn2.x86_64    containerd://1.3.2
```

如需詳細資訊，請參閱檢視節點。

3. 執行下列命令，以檢視在叢集上執行的工作負載。

```
kubectl get pods --all-namespaces -o wide
```

下面顯示了範例 Fargate 輸出。

```
Fargate output

NAMESPACE     NAME                          READY   STATUS    RESTARTS   AGE   IP
           NODE                                                    NOMINATED NODE
 READINESS GATES
kube-system   coredns-69dfb8f894-9z95l   1/1     Running   0          18m
 192.168.164.53    fargate-ip-192-168-164-53.us-west-2.compute.internal    none
         none
kube-system   coredns-69dfb8f894-c8v66   1/1     Running   0          18m
 192.168.141.147   fargate-ip-192-168-141-147.us-west-2.compute.internal   none
         none
```

如需詳細資訊，請參閱檢視工作負載。

EMR在上啟用 Amazon 的叢集存取 EKS

使用存取項目啟用叢集EKS存取 (建議)

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

Amazon EMR 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，以便在 Amazon 叢集的命名空間中執行 Amazon EMR Spark 任務。EKS當您從 Amazon 叢集命名空間建立虛擬EKS叢集時，Amazon EMR 會自動設定所有必要的許可，因此您無需在目前的工作流程中新增任何額外步驟。

> **Note**
>
> Amazon 與 Amazon 的集EMR成EKSCAM僅支持EKS虛擬集群EMR上的新 Amazon。您無法移轉現有的虛擬叢集以使用此整合。

必要條件

- 請確定您執行的是 2.15.3 或更高版本 AWS CLI
- 您的 Amazon EKS 群集必須是 1.23 或更高版本。

設定

要設置 Amazon EMR 和來自 Amazon 的 AccessEntry API操作之間的集成EKS，請確保您已完成以下項目：

- 請確定您authenticationMode的 Amazon EKS 叢集已設定為API_AND_CONFIG_MAP。

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

  如果尚未設定，請authenticationMode將設定為API_AND_CONFIG_MAP。

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

  如需有關驗證模式的詳細資訊，請參閱叢集驗證模式。
- 請確定您用來執行CreateVirtualCluster和DeleteVirtualClusterAPI作業的IAM角色也具有下列權限：

  ```
  {
    "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 相關的術語和概念的列表。EKS CAM

- 虛擬群集（VC）— 在 Amazon 中創建的命名空間的邏輯表示EKS。這是一個 1:1 鏈接到一個 Amazon EKS 集群命名空間。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。
- 存取原則 — 授與EKS叢集中IAM角色的存取權和動作的權限。
- 存取項目 — 以角色 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存取入口整合式虛擬叢集 — 使用 Amazon 的存取輸入API作業建立的虛擬叢集EKS。

啟用叢集存取使用 **aws-auth**

您必須執行下列動作，允許 Amazon EMR 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 使用者對應至服務連結角色。AWSServiceRoleForAmazonEMRContainers`eksctl`當IAM身份對應命令與服務名稱一起`emr-containers`使用時，這些動作會在中自動執行。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"
```

Replace (取代) *my_eks_cluster* 使用您的 Amazon EKS 群集的名稱並替換 *kubernetes_namespace* 使用為執行 Amazon 工作負載而建立的 Kubernetes 命名空間。EMR

> **Important**
>
> 必須使用上一個步驟 安裝 eksctl 下載最新的 eksctl 才能使用此功能。

EMR在 Amazon 上啟用叢集存取的手動步驟 EKS

您也可以使用下列手動步驟啟用 Amazon EMR on 的叢集存取EKS。

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 時，執行下列命令以在特定命名空間中建立 Kubernetes 角色。此角色授予 Amazon EMR 的必要RBAC許可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 及更低版本時，執行下列命令以在特定命名空間中建立 Kubernetes 角色。此角色授予 Amazon EMR 的必要RBAC許可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 的服務連結角色 EKS，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

   您可以使用下列其中一個選項，將 Amazon 對應EKS服務連結角色與EMR上一個步驟中與 Kubernetes 角色繫結的使用emr-containers者對應。

   選項 1：使用 **eksctl**

   執行下列eksctl命令，將 Amazon EMR 上的EKS服務連結角色對應至emr-containers使用者。

```
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**
      >
      > 如果您收到說明的錯誤訊息Error from server (NotFound): configmaps "aws-auth" not found，請參閱 Amazon 使用者指南中新增使EKS用者角色中的步驟，以套用庫存 ConfigMap。

   2. 將 Amazon EMR 上的EKS服務連結角色詳細資訊新增至「」下ConfigMapdata的一mapRoles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。

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 1.14 及更新EKS版本，以及在 2019 年 9 月 3 日或之後更新至 1.13 版或更新版本的EKS叢集使用。若要使用此功能，您可以將現有EKS叢集更新至版本 1.14 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，就會有與之URL相關聯的 OpenID Connect 發行者。您可以在 Amazon EKS 主控台URL中檢視此內容，也可以使用下列 AWS CLI 命令擷取它。

> **Important**
>
> 您必須使用的最新版本才 AWS CLI 能從此命令接收正確的輸出。

```
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
```

若要針對叢集中的服務帳戶使用IAM角色，您必須使用 eksctl 或. OIDC AWS Management Console

若要使用以建立叢集的IAMOIDC身分識別提供者 **eksctl**

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。

使用下列命令為叢集建立OIDC身分識別提供者。Replace (取代) *cluster_name* 用你自己的價值。

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

若IAMOIDC要使用 AWS Management Console

URL從叢集的 Amazon 主EKS控台說明擷取OIDC發行者，或使用下列 AWS CLI 命令。

使用下列命令從擷取OIDC簽URL發者 AWS CLI。

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

使用下列步驟URL從 Amazon EKS 主控台擷取OIDC發卡機構。

1. 在開啟IAM主控台https://console.aws.amazon.com/iam/。
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. 針對提供者 URL，貼上叢集URL的OIDC發行者。
    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 上執行工作負載EKS，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〉。

您也必須建立指定工作執行角色權限的原IAM則，然後將IAM原則附加至工作執行角色。

任務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。若要使用執行相同的程序 AWS CLI，您也可以使用 Amazon EMR on EKS Workshop 上的建立任務執行IAM角色一節中的步驟設定角色。

> **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。

```
{
    "Version": "2012-10-17",
```

```
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:PutObject",
                "s3:GetObject",
                "s3:ListBucket"
            ],
            "Resource": "arn:aws:s3:::example-bucket"
        },
        {
            "Effect": "Allow",
            "Action": [
                "logs:PutLogEvents",
                "logs:CreateLogStream",
                "logs:DescribeLogGroups",
                "logs:DescribeLogStreams"
            ],
            "Resource": [
                "arn:aws:logs:*:*:*"
            ]
        }
    ]
}
```

如需詳細資訊，請參閱使用工作執行角色、設定工作執行以使用 S3 日誌和設定任務執行以使用 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 命名空間上執行工作時，系統管理員必須在工作執行角色與EMR受管理服務帳戶的識別之間建立信任關係。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，受EMR管理的服務帳戶會在工作提交時自動建立，範圍限定在提交工作的命名空間。

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EMR對 Amazon 的存取權 EKS

對於您在 Amazon EMR 上執行的任何動作EKS，您都需要該動作的對應IAM許可。您必須建立一個IAM政策，讓您執行 Amazon EMR 的EKS動作，並將該政策附加到您使用的使用IAM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EMR在EKS環境上設置 Amazon 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。

建立新IAM原則並將其附加至IAM主控台中的使用者

建立新IAM策略

1. 登入 AWS Management Console 並開啟IAM主控台，位於https://console.aws.amazon.com/iam/。
2. 在IAM主控台的左側導覽窗格中，選擇 [原則]。
3. 在 Policies (政策) 頁面上，選擇 Create a policy (建立政策)。
4. 在「建立原則」視窗中，瀏覽至 JSON「編輯」索引標籤。建立具有一或多個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IAM主控台，網址為：https://console.aws.amazon.com/iam/
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，請建立具有下列權限的IAM原則。這些權限可讓您建立、列出、描述和刪除 AWS 帳戶中的虛擬叢集。

```
{
    "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 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，以便在 Amazon 叢集的命名空間中執行 Amazon EMR Spark 任務。EKS若要這麼做，您必須具備下列權限：

```
{
  "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 的叢集存取EKS。

第一次從 AWS 帳戶叫用CreateVirtualCluster操作時，您還需要CreateServiceLinkedRole許可以為 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工作，請建立具有下列權限的IAM原則。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。

```
{
    "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

若要存取推送至 Amazon S3 的日誌 CloudWatch，或在 Amazon EMR 主控台中檢視應用程式事件日誌，請建立具有下列權限的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。

```
{
    "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": "*"
        }
    ]
}
```

有關如何設定任務執行以將日誌推送到 Amazon S3 的詳細資訊 CloudWatch，請參閱設定任務執行以使用 S3 日誌和設定任務執行以使用 CloudWatch 日誌。

註冊 Amazon EKS 群集與 Amazon EMR

註冊叢集是設定 Amazon 執行工作負載的最後EMR一EKS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選擇的名稱的虛擬叢集。

> **Note**
>
> 每個虛擬叢集在所有EKS叢集中都必須具有唯一的名稱。如果兩個虛擬叢集的名稱相同，即使兩個虛擬叢集屬於不EKS同的叢集，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，您也可以建立包含虛擬叢集所需參數的JSON檔案，然後使用該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 Spark Operator 執行 Spark 作業

Amazon EMR 發布 6.10.0 及更高版本支持阿帕奇星火 Kubernetes 運營商，或星火運營商，作為 Amazon 的任務提交模型。EMR EKS使用 Spark 操作員，您可以在自己的 Amazon EKS 叢集上使用 Amazon EMR 發行執行階段部署和管理 Spark 應用程式。在 Amazon EKS 叢集中部署 Spark 操作員之後，您可以直接向操作員提交 Spark 應用程式。Operator 可管理 Spark 應用程式的生命週期。

> **Note**
>
> Amazon EMR EKS 根據 v CPU 和內存消耗在 Amazon 上計算定價。此計算適用於驅動程式和執行程式 Pod。此計算從您下載 Amazon EMR 應用程式映像時開始，直到 Amazon EKS 網繭終止並四捨五入到最接近的秒數為止。

主題

- 建立 Amazon EMR 的星火運營商 EKS
- 開始使用星火運營商 Amazon EMR 上 EKS
- 在 Amazon EMR 上與 Spark 運營商一起使用垂直自動調度 EKS
- 卸載 Amazon EMR 的星火運營商 EKS
- 安全和星火運營商與 Amazon EMR EKS

建立 Amazon EMR 的星火運營商 EKS

在 Amazon 上安裝 Spark 操作員之前，請完成以下任務以進行設置EKS。如果您已經註冊了 Amazon Web Services（AWS）並使用 AmazonEKS，那麼您幾乎可以EMR在上使用 Amazon EKS。完成以下任務即可為 Amazon 上的 Spark 運營商進行設置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 AWS CLI — 如果已經安裝 AWS CLI，請確認擁有最新版本。
- 安裝 eksctl — eksctl 是您用來與 Amazon 通信的命令行工具。EKS
- 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。
- 設定 Amazon EKS 叢集 — 按照以下步驟在 Amazon 中建立具有節點的新 Kubernetes 叢集。EKS
- 選擇 Amazon EMR 基礎映像 URI（版本 6.10.0 或更高版本）— Amazon 6.10.0 及更高EMR版本支持 Spark 運營商。

開始使用星火運營商 Amazon EMR 上 EKS

本主題可協助您EKS透過部署 Spark 應用程式和排程 Spark 應用程式，開始使用 Amazon 上的 Spark 運算子。

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 建立 Amazon EMR 的星火運營商 EKS 中的步驟。
2. 向 Amazon ECR 註冊表驗證您的 Helm 客戶端。在下列命令中，取代 *region-id* 值與您的首選 AWS 區域，以及相應的 *ECR-registry-account* 頁面中「區域Amazon ECR 註冊表帳戶按區域」的值。

   ```
   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 圖表`--version`參數，請使用您的 Amazon EMR 版本標籤，並刪除`emr-`前綴和日期後綴。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-7.2.0-latest` 版本，因此它會為 Helm Chart `--version` 指定 `7.2.0`。

   ```
   helm install spark-operator-demo \
     oci://895885662937.dkr.ecr.region-id.amazonaws.com/spark-operator \
     --set emrContainers.awsRegion=region-id \
     --version 7.2.0 \
   ```

```
  --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-7.2.0
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。有關更多信息，請參閱上的`spark-on-k8s-operator`文檔。 GitHub

執行 Spark 應用程式

星火運營商與 Amazon EMR 6.10.0 或更高版本的支持。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。使用以下步驟與星火運營商在 Amazon EMR EKS 6.10.0 或更高版本上運行 Spark 應用程序。

1. 在使用 Spark Operator 執行 Spark 應用程式之前，請先完成 建立 Amazon EMR 的星火運營商 EKS 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 運算子提交應用程式至 Spark 的詳細資訊，請參閱上的spark-on-k8s-operator文件SparkApplication中的使用 GitHub.

使用 Amazon S3 進行存儲

若要使用 Amazon S3 做為檔案儲存選項，請將下列組態新增至您的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 7.2.0 及更高EMR版本，則預設會包含組態。在這種情況下，您可以將文件路徑設置為，`s3://<bucket_name>/<file_path>`而不是`local://<file_path>`在 Spark 應用程序YAML文件中。

然後正常提交 Spark 應用程序。

在 Amazon EMR 上與 Spark 運營商一起使用垂直自動調度 EKS

從 Amazon EMR 7.0 開始，您可以EMR在EKS垂直自動調度資源上使用 Amazon 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。

必要條件

繼續之前，請先完成下列設定：

- 完成「建立 Amazon EMR 的星火運營商 EKS」中的步驟。
- (選擇性) 如果您先前安裝了較舊版本的 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。
- 授予您 Amazon S3 位置中檔案的存取權限：

  1. 使用具有 S3 許`JobExecutionRole`可的駕駛員和操作員服務帳戶註釋。

     ```
     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. 更新該命名空間中工作執行角色的信任原則。

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

3. 編輯工作執行IAM角色的角色信任原則，並emr-containers-sa-spark-*-*-xxxx將serviceaccount從更新為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，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 必要條件 中的步驟。

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

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

此組態可啟用垂直自動調度資源，而且是必要的簽名組態，可讓您為工作選擇簽名。

如需有關組態和參數值的詳細資訊，請參閱在EMR上EKS設定 Amazon 的垂直自動調度資源。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，請參閱垂直自動調度資源模式。

以下是使用垂直自動調度資源`spark-pi.yaml`的必要組態命名的範例`SparkApplication`定義檔案。

```
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-7.2.0: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 運算子提交應用程式至 Spark 的詳細資訊，請參閱上的spark-on-k8s-operator文件SparkApplication中的使用 GitHub.

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。

若要移除網繭和應用程式，請執行下列命令：

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

卸載 Amazon EMR 的星火運營商 EKS

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. 刪除星火運算符CustomResourceDefinitions (CRDs) ：

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

安全和星火運營商與 Amazon EMR EKS

主題

- 使用以角色為基礎的存取控制設定叢集存取權限 () RBAC
- 使用服務帳戶的IAM角色設定叢集存取權限 (IRSA)

使用以角色為基礎的存取控制設定叢集存取權限 () RBAC

為了部署 Spark 運營商，Amazon EMR 上EKS創建了 Spark 運營商和 Spark 應用程序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 會建立操作員服務帳戶和角色，以管SparkApplications理 Spark 任務和其他資源 (例如服務)。

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角色設定叢集存取權限 (IRSA)

本節使用範例來示範如何將 Kubernetes 服務帳戶設定為擔任角色。 AWS Identity and Access Management 接著，使用該服務帳戶的網繭就可以存取該角色具有存取權限的任何 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。若要這麼做，您可以設定服務帳戶 (IRSA) 的IAM角色，以驗證及授權 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。

必要條件

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。

將 Kubernetes 服務帳戶設定為擔任角色 IAM

使用下列步驟設定 Kubernetes 服務帳戶，以假定網繭可用來存取該IAM角色具有存取權限之 AWS 服務的角色。

1. 完成之後必要條件，請使 AWS Command Line Interface 用建立一個`example-policy.json`檔案，允許對您上傳到 Amazon S3 的檔案進行唯讀存取：

```
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策略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 服務帳戶產生關聯：

```
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 必要條件 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 運算子將應用程式提交至 Spark 的詳細資訊，請參閱 SparkApplication在 Apache Spark 的 Kubernetes 運算子中使用 (spark-on-k8s 運算子) 說明文件。 GitHub

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 根據 vCPU 和內存消耗計算 Amazon EKS 上的定價。此計算適用於驅動程式和執行程式 Pod。此計算從您下載 Amazon EMR 應用程式映像時開始，直到 Amazon EKS 網繭終止並四捨五入到最接近的秒數為止。

主題

- 為 Amazon EMR on EKS 設定 spark-submit
- 開始針對 Amazon EMR on EKS 使用 spark-submit
- spark-submit 的 Spark 驅動程式服務帳戶安全要求

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 AWS CLI — 如果已經安裝 AWS CLI，請確認擁有最新版本。
- 安裝 eksctl — eksctl 是一個命令列工具，可以用來與 Amazon EKS 通訊。

- 設定 Amazon EKS 叢集 — 按照以下步驟在 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 spark-submit 的 Spark 驅動程式服務帳戶安全要求。
- 設定本機 AWS 憑證設定檔。
- 從 Amazon EKS 主控台選擇您的 EKS 叢集，然後在「概觀」、「詳細資料」和「API 伺服器端點」下找到 EKS 叢集端點。

開始針對 Amazon EMR on EKS 使用 spark-submit

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. 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. 在 EKS 基礎映像上運行帶有 Amazon EMR 的容器。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。

spark-submit 的 Spark 驅動程式服務帳戶安全要求

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
```

執行每個命令，確認您擁有必要的權限。

```
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:
  - "*"
```

為服務帳戶 (IRSA) 設定 IAM 角色以進行火花提交

以下各節說明如何為服務帳戶 (IRSA) 設定 IAM 角色，以驗證和授權 Kubernetes 服務帳戶，以便您可以執行存放在 Amazon S3 中的 Spark 應用程式。

必要條件

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：

- 完成設定火花提交
- 創建了一個 S3 存儲桶並上傳了火花應用程序 jar

設定 Kubernetes 服務帳戶以扮演身分與存取權管理角色

下列步驟說明如何將 Kubernetes 服務帳戶設定為擔任 AWS Identity and Access Management (IAM) 角色。將網繭設定為使用服務帳戶後，他們就可以存取 AWS 服務 該角色具有存取權限的任何網繭。

1. 建立政策檔案以允許對您上傳的 Amazon S3 物件進行唯讀存取：

```
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. 建立身分與存取權管理角色，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權限的 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. 該kubectl命令應返回創建帳戶的確認。

```
serviceaccount/emr-containers-sa-spark created
clusterrolebinding.rbac.authorization.k8s.io/emr-containers-role-spark configured
```

運行星火應用

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. 請確定您已完成在 EKS 上為 Amazon EMR 設定火花提交中的步驟。
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. 火花驅動程式完成 Spark 工作後，您應該會在提交結束時看到記錄行，指出 Spark 工作已完成。

```
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

運行完應用程序後，可以使用以下命令執行清理。

```
kubectl delete -f spark-rbac.yaml
```

在 Amazon EMR 上使用阿帕奇利維 EKS

在 Amazon 7.1.0 及更高EMR版本中，您可以使用 Apache Livy 在 Amazon EMR 上提交任務。EKS使用 Apache Livy，您可以設定自己的 Apache Livy REST 端點，並使用它在您的 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 應用程式提交至您的 Livy 伺服器。服務器管理 Spark 應用程序的生命週期。

> **Note**
>
> Amazon EMR EKS 根據 v CPU 和內存消耗在 Amazon 上計算定價。此計算適用於驅動程式和執行程式 Pod。此計算從您下載 Amazon EMR 應用程式映像時開始，直到 Amazon EKS 網繭終止並四捨五入到最接近的秒數為止。

主題

- 在 Amazon EMR 上設置阿帕奇利維 EKS

- 在 Amazon EMR 上開始使用阿帕奇利維 EKS
- 運行與阿帕奇利維的星火應用程序 Amazon EMR EKS
- 在 Amazon EMR 上卸載阿帕奇利維 EKS
- 阿帕奇利維與 Amazon EMR 的安全 EKS
- 阿帕奇利維在 Amazon EMR 版本上的安裝屬性 EKS
- 故障診斷

在 Amazon EMR 上設置阿帕奇利維 EKS

您必須先完成下列任務，才能在 Amazon EKS 叢集上安裝 Apache Livy。

- 安裝 AWS CLI— 如果您已經安裝了 AWS CLI，請確認您擁有最新版本。
- 安裝 eksctl — eksctl 是您用來與 Amazon 通信的命令行工具。EKS
- 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。
- 設定 Amazon EKS 叢集 — 按照以下步驟在 Amazon 中建立具有節點的新 Kubernetes 叢集。EKS
- 選擇 Amazon EMR 版本標籤 — Amazon 7.1.0 及更高EMR版本支援 Apache Livy。
- 安裝ALB控制器 — 控制器會ALB管理 Kubernetes 叢集的 E AWS lastic Load Balancing。當您在設定 Apache Livy 時建立 Kubernetes 輸入時，它會建立 Net AWS work Load Balancer (NLB)。

在 Amazon EMR 上開始使用阿帕奇利維 EKS

請完成以下步驟來安裝阿帕奇利維。

1. 如果你還沒有，設置阿帕奇利維 Amazon EMR 上EKS。
2. 向 Amazon ECR 註冊表驗證您的 Helm 客戶端。您可以 AWS 區域 從 Amazon ECR 註冊表帳戶按區域找到相應的`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IRSA定服務帳戶，請參閱使用服務帳戶的IAM角色設定存取權限 (IRSA)。

4. 建立命名空間以執行 Spark 工作負載。

```
kubectl create ns <spark-ns>
```

5. 使用下面的命令來安裝 Livy。

此 Livy 端點僅供EKS叢集VPC中的內部使用。若要啟用以外的存取VPC，請`--set loadbalancer.internal=false`在 Helm 安裝指令中設定。

> **Note**
>
> 依預設，SSL不會在此 Livy 端點內啟用，且端點僅在EKS叢集VPC內可見。如果你設置`loadbalancer.internal=false`和`ssl.enabled=false`，你正在暴露一個不安全的端點到你的. VPC 若要設定安全的 Livy 端點，請參閱使TLS用/設定安全的 Apache Livy 端點。SSL

```
helm install livy-demo \
  oci://895885662937.dkr.ecr.region-id.amazonaws.com/livy \
  --version 7.2.0 \
  --namespace livy-ns \
  --set image=ECR-registry-account.dkr.ecr.region-id.amazonaws.com/livy/
emr-7.2.0: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「頭盔」圖表。

```
helm list -n livy-ns -o yaml
```

該helm list命令應該返回有關您的新 Helm 圖表的信息。

```
app_version: 0.7.1-incubating
chart: livy-emr-7.2.0
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，請確認 Network Load Balancer 中的目標群組健康狀況良好。

```
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"
            }
        }
    ]
}
```

一旦您的狀態NLB變成active並且您的目標群組是healthy，您就可以繼續。可能需要幾分鐘的時間。

9. 從 Helm 安裝中擷取 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. 從頭盔安裝中檢索星火服務帳戶

```
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. 如果您設定internalALB=true為從外部啟用存取VPC，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。如需有關安全地公開端點以外的詳細資訊VPC，請參閱使用/使用安全的 Apache Livy 端點設TLS定。SSL

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 S3 或呼叫 AWS APIor CLI操作等任何 AWS 資源，您必須將具有必要權限的 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的IAM角色設定存取權限 (IRSA)。

阿帕奇利維支持其他配置，您可以在安裝 Livy 時使用。如需詳細資訊，請參閱 Amazon EMR EKS 版本上 Apache Livy 的安裝屬性。

運行與阿帕奇利維的星火應用程序 Amazon EMR EKS

在您可以使用 Apache Livy 運行 Spark 應用程序之前，請確保您已經完成了在 Amazon EMR 上設置 Apache Livy EKS 和開始使用 Apache Livy 的步驟。EMR EKS

您可以使用 Apache 利維來運行兩種類型的應用程序：

- Batch 工作階段 — 用於提交 Spark 批次工作的一種 Livy 工作負載。
- 互動式工作階段 — 一種 Livy 工作負載，提供程式化和視覺化介面來執行 Spark 查詢。

> **Note**
>
> 來自不同會話的驅動程序和執行程序 Pod 可以相互通信。命名空間不保證網繭之間的任何安全性。Kubernetes 不允許對指定命名空間內的網繭子集進行選擇性權限。

執行批次工作階

若要提交批次處理工作，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-7.2.0:latest",
```

```
                "spark.kubernetes.authenticate.driver.serviceAccountName": "<spark-
service-account>"
            }
          }' <livy-endpoint>/batches
```

若要監視批次處理工作，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. 請確定您可以存取自我託管或受管理的 Jupyter 筆記本，例如 Jupyter 筆記本。 SageMaker 您的筆記本必須安裝火花。
2. 為星火配置創建一個桶spark.kubernetes.file.upload.path。確定 Spark 服務帳戶具有值區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角色設定存取權限 (IRSA)
3. 使用命令將閃光加載到 Jupyter 筆記本中。%load_ext sparkmagic.magics
4. 執行命令%manage_spark以使用 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再次運行以創建 Spark 上下文，然後轉到創建會話。選擇 Livy 端點，指定唯一的工作階段名稱選擇語言，然後新增下列內容。

   ```
   {
     "conf": {
       "spark.kubernetes.namespace": "livy-namespace",
       "spark.kubernetes.container.image": "public.ecr.aws/emr-on-eks/spark/
   emr-7.2.0:latest",
       "spark.kubernetes.authenticate.driver.serviceAccountName": "<spark-service-
   account>",
       "spark.kubernetes.file.upload.path": "<URI_TO_S3_LOCATION_>"
     }
   }
   ```

6. 提交應用程序並等待它創建 Spark 上下文。
7. 若要監視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
```

監控星火應用

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`。

在 Amazon EMR 上卸載阿帕奇利維 EKS

請按照下列步驟解除安裝阿帕奇利維。

1. 使用命名空間和應用程式名稱的名稱刪除 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。在再次在命名空間上安裝 Livy 之前，請確保已刪除資源。
3. 刪除星火命名空間。

   ```
   kubectl delete namespace spark-ns
   ```

阿帕奇利維與 Amazon EMR 的安全 EKS

請參閱以下頁面，進一步了解如何在 Amazon EMR 上為 Apache Livy 設定安全性 EKS

主題

- 使TLS用/設置一個安全的 Apache 利維端點 SSL
- 使用以角色為基礎的存取控制設定 Apache Livy 和 Spark 應用程式權限 () RBAC
- 使用服務帳戶的IAM角色設定存取權限 (IRSA)

使TLS用/設置一個安全的 Apache 利維端點 SSL

請參閱以下各節，進一步了解如何EKS使用 end-to-end TLS和SSL加密設定適用EMR於 Amazon 的 Apache Livy。

設定TLS與SSL加密

若要在 Apache Livy 端點上設定SSL加密，請依照下列步驟執行。

- 安裝 Secret 存放區CSI驅動程式和 AWS 密碼以及組態提供者 (ASCP) — Secret 存放區CSI驅動程式，並ASCP安全地儲存 Livy 伺服器網繭需要啟用的 Livy JKS 憑證和密碼。SSL您也可以只安裝 Secret 存放區CSI驅動程式，並使用任何其他受支援的密碼提供者。
- 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、金鑰密碼和金鑰庫密碼，以確保ALB端點與 Livy 伺服器之間的連線安全。
- 向 Livy 服務帳戶添加權限以從中檢索密碼 AWS Secrets Manager — Livy 服務器需要這些權限才能從中檢索密碼ASCP並添加 Livy 配置以保護 Livy 服務器。若要將IAM權限新增至服務帳戶，請參閱使用服務帳戶的IAM角色設定存取權限 (IRSA)。

使用金鑰和金鑰庫密碼設定JKS憑證 AWS Secrets Manager

請依照下列步驟來設定具有金鑰和金鑰庫密碼的JKS憑證。

1. 為 Livy 伺服器產生金鑰儲存檔案。

```
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. 建立信任庫檔案。

```
keytool -import -noprompt -alias <host>-file <cert_file> -
keystore <truststore_file> -storepass <truststorePassword>
```

4. 將JKS憑證儲存於中 AWS Secrets Manager。`livy-jks-secret`用您的密鑰和`fileb://mykeystore.jks`密鑰庫JKS證書的路徑替換。

```
aws secretsmanager create-secret \
```

```
--name livy-jks-secret \
--description "My Livy keystore JKS secret" \
--secret-binary fileb://mykeystore.jks
```

5. 在密鑰管理器中保存密鑰庫和密鑰密碼。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. 使用以下命令創建一個 Livy 服務器命名空間。

```
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>
```

開始使SSL用已啟用的阿帕奇利維

在 Livy 伺服器SSL上啟用之後，您必須設定以存取keyStore和keyPasswords密碼。serviceAccount AWS Secrets Manager

1. 建立 Livy 伺服器命名空間。

```
kubectl create namespace <livy-ns>
```

2. 設置 Livy 服務帳戶以訪問秘密管理器中的密碼。如需有關設定的詳細資訊IRSA，請參閱安裝 Apache Livy IRSA 時進行設定。

```
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. 安裝生活。對於頭盔圖-版本參數，請使用您的 Amazon EMR 發布標籤，例如。7.1.0您還必須將 Amazon ECR 註冊表帳戶 ID 和區域 ID 替換為您自己的 ID IDs。您可以 AWS 區域 從 Amazon ECR 註冊表帳戶按區域找到相應的ECR-registry-account值。

```
helm install <livy-app-name> \
  oci://895885662937.dkr.ecr.region-id.amazonaws.com/livy \
  --version 7.2.0 \
  --namespace livy-namespace-name \
  --set image=<ECR-registry-account.dkr.ecr>.<region>.amazonaws.com/livy/
emr-7.2.0: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 上安裝阿帕奇利維的步驟 5 繼續。EKS

使用以角色為基礎的存取控制設定 Apache Livy 和 Spark 應用程式權限 () RBAC

為了部署 Livy，Amazon EMR 上EKS創建了一個服務器服務帳戶和角色以及 Spark 服務帳戶和角色。這些角色必須具備完成安裝和執行 Spark 應用程式的必要RBAC權限。

RBAC伺服器服務帳戶和角色的權限

Amazon EMR on 會EKS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，並將流量路由到輸入和其他資源。

此服務帳戶的預設名稱為 emr-containers-sa-livy。它必須具有以下權限。

```
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"
```

RBACspark 服務帳戶和角色的權限

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。此服務帳戶需要權限才能管理執行程式網繭以及驅動程式網繭所需的任何資源。除非命名空間中的預設服務帳戶具有必要的權限，否則驅動程式會失敗並結束。需要下列RBAC權限。

```
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角色設定存取權限 (IRSA)

默認情況下，Livy 服務器和 Spark 應用程序的驅動程序和執行程序無法訪問資源。 AWS 伺服器服務帳戶和 spark 服務帳戶可控制 Livy 伺服器 AWS 資源的存取權，以及火花應用程式的網繭。若要授與存取權，您需要將服務帳戶對應至具有必要 AWS 權限的IAM角色。

您可以在安裝 Apache Livy 之前、安裝期間或完成安裝後設定IRSA對應。

IRSA在安裝 Apache Livy 時進行設置（用於服務器服務帳戶）

> **Note**
>
> 只有伺服器服務帳戶才支援此對應。

1. 確保您已經完成了 Amazon 的 Apache Livy 的設置，EKS並且正在與 Amazon EMR 上安裝 Apache Livy 的中間。EMR EKS
2. 為 Livy 伺服器建立一個 Kubernetes 命名空間。在此範例中，命名空間的名稱為`livy-ns`。
3. 建立包含您要網繭存取 AWS 服務 之權限的IAM原則。下列範例會針對 Spark 進入點建立取得 Amazon S3 資源的IAM政策。

```
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. 使用以下命令將您的 AWS 帳戶 ID 設置為變量。

```
account_id=$(aws sts get-caller-identity --query "Account" --output text)
```

5. 將叢集的 OpenID Connect (OIDC) 身分識別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。一定要使用你自己的價值觀。

```
export namespace=default
export service_account=my-service-account
```

7. 使用下列命令建立信任原則檔案。如果您想要將角色的存取權授StringEquals與命名空間內的所有服務帳戶，請複製下列命令，然後取代為StringLike並取代$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. 使用以下頭盔安裝命令將設置serviceAccount.executionRoleArn為映射IRSA。下面是頭盔安裝命令的一個例子。您可以 AWS 區域 從 Amazon ECR 註冊表帳戶按區域找到相應的ECR-registry-account值。

```
helm install livy-demo \
  oci://895885662937.dkr.ecr.us-west-2.amazonaws.com/livy \
  --version 7.2.0 \
  --namespace livy-ns \
  --set image=ECR-registry-account.dkr.ecr.region-id.amazonaws.com/livy/
emr-7.2.0:latest \
  --set sparkNamespace=spark-ns \
  --set serviceAccount.executionRoleArn=arn:aws:iam::123456789012:role/my-role
```

對應IRSA至星火服務帳戶

在對應IRSA至 Spark 服務帳戶之前，請確定您已完成下列項目：

- 確保您已經完成了 Amazon 的 Apache Livy 的設置，EKS並且正在與 Amazon EMR 上安裝 Apache Livy 的中間。EMR EKS
- 您的群集必須有一個現有的 IAM OpenID Connect（OIDC）程序。若要查看您是否已有或如何建立叢集，請參閱為叢集建立IAMOIDC提供者。
- 請確定您已安裝 0.171.0 或更新版本的eksctlCLI安裝或。 AWS CloudShell若要安裝或更新eksctl，請參閱eksctl文件的安裝。

請依照下列步驟對IRSA應至您的 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置變量。

```
export namespace=default
export service_account=my-service-account
```

3. 使用下列命令建立IAM角色的信任原則檔案。下列範例會授與命名空間內所有服務帳戶使用角色的權限。若要這麼做，請StringEquals使用 * 取代StringLike$service_account並取代。

```
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。確保使用自己的價值觀。

```
 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
```

阿帕奇利維在 Amazon EMR 版本上的安裝屬性 EKS

阿帕奇利維安裝允許您選擇一個版本的利維頭盔圖表。Helm 圖表提供各種屬性，可自訂您的安裝和設定體驗。Amazon EMR 在 7.1.0 及更高EKS版本上支援這些屬性。

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 利維時，您可以選擇利維頭盔圖表版本。若要在安裝期間設定屬性，請使用指令`--set <property>=<value>`。

屬性	描述	預設
image	Amazon EMR 版本URI的 Livy 服務器。這是必要的組態。	""
sparkNamespace	命名空間來運行利維星火會話。例如，指定「livy」。這是必要的組態。	""
nameOverride	請提供名稱，而不是`livy`。該名稱被設置為所有 Livy 資源的標籤	「生活」
fullnameOverride	提供要使用的名稱，而不是資源的完整名稱。	""
啟用 SSL	end-to-end SSL從 Livy 端點到 Livy 伺服器啟用。	FALSE
SSL。 certificateArn	如果SSL已啟用，則此ARN為服務所NLB建立之ACM憑證。	""
SSL。 secretProviderClass姓名	如果SSL啟用，這是要保NLB護 Livy 伺服器連線的秘密提供者類別名稱。SSL	""
SSL。 keyStoreObject姓名	如果SSL啟用，則為秘密提供者類別中金鑰儲存庫憑證的物件名稱。	""

屬性	描述	預設
SSL。 keyPasswordsObject姓名	如果SSL啟用，則為具有金鑰儲存庫和金鑰密碼之密碼的物件名稱。	""
瑞巴克創建	如果為 true，則會建立RBAC資源。	FALSE
serviceAccount. 創建	如果為真，則創建一個利維服務帳戶。	TRUE
serviceAccount. 名稱	要用於 Livy 的服務帳戶名稱。如果您沒有設置此屬性並創建服務帳戶，Amazon EMR on 會使用`fullname`覆蓋屬性EKS自動生成一個名稱。	"emr-containers-sa-livy"
serviceAccount.executionRoleArn	Livy 服務帳戶ARN的執行角色。	""
sparkServiceAccount. 創建	如果為真，則會在中建立星火服務帳戶 `.Release.Namespace`	TRUE
sparkServiceAccount. 名稱	要用於 Spark 的服務帳戶名稱。如果您不設置此屬性並創建一個 Spark 服務帳戶，Amazon EMR on 會EKS自動生成帶有`-spark-livy` 後綴`fullname0verride` 屬性的名稱。	「emr-containers-sa-spark生活」
服務. 名稱	利維服務的名稱	`"emr-containers-livy"`
服務. 註釋	活動服務註釋	`{}`

屬性	描述	預設
負載器. 啟用	是否為 Livy 服務建立負載平衡器，用來公開 Amazon EKS 叢集外部的 Livy 端點。	FALSE
負載器. 內部	是否將 Livy 端點設定為內部VPC或外部端點。 將此內容設定為將端點FALSE公開給。VPC我們建議您使用TLS/來保護您的端點SSL。如需詳細資訊，請參閱設定TLS和SSL加密。	FALSE
imagePullSecrets	用於從私有存儲庫中提取 Livy 映像的`imagePullSecret` 名稱列表。	[]
resources	Livy 容器的資源請求和限制。	{}
nodeSelector	要為其排程 Livy 網繭的節點。	{}
容差	包含要定義之 Livy 網繭公差的清單。	[]
親和力	Livy 網繭相似性規則。	{}
持續性. 已啟用	如果為 true，則啟用會話目錄的持續性。	FALSE
持久性。 subPath	要掛載至工作階段目錄的PVC子路徑。	""
持久性。 existingClaim	使PVC用而不是創建一個新的。	{}

屬性	描述	預設
持久性。 storageClass	要使用的儲存空間類別。若要定義此參數，請使用格式`storageClassName: <storageClass>` 。將此參數設定為"-"停用動態佈建。如果將此參數設定為 null 或未指定任何內容，Amazon EMR on EKS 不會設定 storageClassName 並使用預設佈建程式。	""
持久性。 accessMode	PVC訪問模式。	`ReadWriteOnce`
持久性. 大小	大PVC小。	二十基
持久性. 註釋	的其他註釋PVC。	{}
環境。 *	額外的環境設置為 Livy 容器。有關更多信息，請參閱在安裝 Livy 時輸入自己的 Livy 和 Spark 配置。	{}
envFrom.*	從 Kubernetes 配置圖或秘密設置為利維的其他 envs。	[]
livyConf.*	其他 livy.conf 項目可從已安裝的 Kubernetes 設定地圖或機密設定。	{}
sparkDefaultsConf.*	要從已安裝的 Kubernetes 組態對應或密碼設定的其他`spark-defaults.conf` 項目。	{}

故障診斷

在安裝 Livy 時輸入您自己的活力和火花配置

您可以配置任何 Apache 的利維或 Apache 星火環境變量與env.*頭盔屬性。請遵循下列步驟，將範例組態轉換example.config.with-dash.withUppercase為支援的環境變數格式。

1. 將大寫字母替換為 1 和字母的小寫字母。例如，example.config.with-dash.withUppercase 會變成 example.config.with-dash.with1uppercase。
2. 用 0 取代破折號 (-)。例如，example.config.with-dash.with1uppercase變成 example.config.with0dash.with1uppercase
3. 將點 (.) 取代為底線 (_)。例如，example.config.with0dash.with1uppercase 會變成 example_config_with0dash_with1uppercase。
4. 用大寫字母替換所有小寫字母。
5. 將前綴添加LIVY_到變量名稱中。
6. 在使用格式--set env 通過舵圖安裝 Livy 時使用該變量。*YOUR_VARIABLE_NAME*. 價值 =*yourvalue*

例如，要設置 Livy 和 Spark 配置spark.kubernetes.executor.podNamePrefix = my-prefix，livy.server.recovery.state-store = filesystem並使用以下頭盔屬性：

```
–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。

主題

- 使用 AWS CLI 管理作業執行
- 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 AWS CLI 管理作業執行

本頁介紹如何使用 AWS Command Line Interface (AWS CLI) 來管理作業執行。

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。如需更多詳細資訊，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。如需更多詳細資訊，請參閱 Amazon EMR 上EKS發布。
- `--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 上EKS發布。

  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。如需更多詳細資訊，請參閱 為 Amazon EMR 定制碼頭圖像 EKS。

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 *DOC-EXAMPLE-BUCKET-LOGGING* 取代為日誌儲存貯體的名稱。

```
{
    "Version": "2012-10-17",
```

```
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:PutObject",
                "s3:GetObject",
                "s3:ListBucket"
            ],
            "Resource": [
                "arn:aws:s3:::DOC-EXAMPLE-BUCKET-LOGGING",
                "arn:aws:s3:::DOC-EXAMPLE-BUCKET-LOGGING/*",
            ]
        }
    ]
}
```

> **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 AWS CLI 管理作業執行 中所示。

- 控制器日誌 - /*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。

```
{
     "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 AWS CLI 管理作業執行 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。

- 控制器日誌 - *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。

選項	描述
--total-executor-cores NUM	所有執行程式的核心總數。
--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

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。

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：^arn:(aws[a-zA-Z0-9-]*):iam:	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 搭配使用作業執行角色與 Amazon EMR on EKS。

錯誤訊息	錯誤情況	建議的後續步驟
:(\d{12})?:(role((\u002F)|(\u002F[\u0021-\u007F]+\u002F))[\w+=,.@-]+)		
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 上EKS 發布。
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。 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。 不支援 *`classification`* 分類。 失敗原因：VALIDATION_ERROR 狀態：FAILED。	StartJobRun 中的參數無效。	請參閱 Amazon EMR 上EKS 發布。
狀態詳細資訊：叢集 *`EKS Cluster ID`* 不存在。 失敗原因：CLUSTER_UNAVAILABLE 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。如需更多詳細資訊，請參閱 設置 Amazon EMR EKS。
狀態詳細資訊：叢集 *`EKS Cluster ID`* 沒有足夠的許可。 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。如需更多詳細資訊，請參閱 設置 Amazon EMR 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。 失敗原因：CLUSTER_UNAVAILABLE 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。如需更多詳細資訊，請參閱 設置 Amazon EMR EKS。
狀態詳細資訊：由於內部錯誤，JobRun 提交失敗。 失敗原因：INTERNAL_ERROR 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。 失敗原因：USER_ERROR 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。 叢集 *EKS Cluster ID* 不存在。 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。如需更多詳細資訊，請參閱 設置 Amazon EMR 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。 叢集 *EKS Cluster ID* 沒有足夠的許可。 失敗原因：CLUSTER_UNAVAILABLE 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。如需更多詳細資訊，請參閱 設置 Amazon EMR EKS。
狀態詳細資訊：監控 JobRun 時出現問題。 叢集 *EKS Cluster ID* 目前無法連線。 失敗原因：CLUSTER_UNAVAILABLE 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。如需更多詳細資訊，請參閱 設置 Amazon EMR EKS。
狀態詳細資訊：由於內部錯誤，無法監控 JobRun 失敗原因：INTERNAL_ERROR 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"
      }
  }]
}
```

使用作業提交器分類

概要

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。可以使用 `emr-job-submitter` 分類為作業提交器 Pod 設定節點選取器。

`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。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。

作為最佳實務，建議作業提交器 Pod 將節點放置在隨需執行個體上，而非 Spot 執行個體上。這是因為如果作業提交器 Pod 遭受 Spot 執行個體中斷，則作業將會失敗。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

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"
      }
    }
  }
}
```

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

建立並使用作業範本來啟動作業執行

本章節描述了如何建立作業範本並使用該範本，以透過 AWS Command Line Interface (AWS CLI) 啟動作業執行。

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。

```
{
    "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。

```
{
    "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。無法使用 Pod 範本來設定作業控制器 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。如需更多詳細資訊，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。如需更多詳細資訊，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://DOC-EXAMPLE-BUCKET-LOGGING"
    }
  }
}'
```

> **Note**
>
> AWS CLI 1.27.68 以後的版本中才提供 `retryPolicyConfiguration`。若要將 AWS CLI 更新至最新版本，請參閱安裝或更新 AWS CLI 的最新版本

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 活動監控任務。

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

- 必要條件
- 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

必要條件

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 AWS CLI — 如果已經安裝 AWS CLI，請確認擁有最新版本。
- 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 度量伺服器 — 您必須先安裝指標伺服器，以便垂直網繭自動配置器可以從 Kubernetes API 伺服器擷取指標。

- 設置 Amazon EKS 群集 (1.24 版或更高版本) – Amazon EKS 1.24 及更高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 Amazon ECR 註冊表帳戶按區域。
   - **$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

使用垂直自動擴展功能提交 Spark 作業

當您透過 StartJobRunAPI 提交工作時，請將下列兩個設定新增至驅動程式，讓 Spark 作業開啟垂直自動調度資源：

```
"spark.kubernetes.driver.label.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.label.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 API 提交 Amazon EMR Spark 任務時，您可以設定垂直自動調度資源。StartJobRun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.`

金鑰	描述	接受的值	預設值	Type	Spark 參數[1]
					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.label.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.label.emr-containers.amazonaws.com/dynamic.sizing.scale.memory
dynamic.sizing.scale.cpu	開啟或關閉 CPU 擴展	*true*, *false*	false	註釋	spark.kubernetes.driver.label.emr-containers.amazonaws.com/dynamic.sizing.scale.cpu

金鑰	描述	接受的值	預設值	Type	Spark 參數[1]
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.label.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.label.emr-containers.amazonaws.com/dynamic.sizing.scale.memory.max
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.label.emr-containers.amazonaws.com/dynamic.sizing.scale.cpu.min

金鑰	描述	接受的值	預設值	Type	Spark 參數[1]
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.label.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

可以使用 kubectl Kubernetes 命令列工具，列出叢集中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 上執行互動式工作負載 EKS

互動式端點是將 Amazon EMR Studio 連接到 Amazon 的閘道，以EKS便您可以執行互動式工作負載。EMR您可以將互動式端點與 EMR Studio 搭配使用，透過 Amazon S3 和 Amazon DynamoDB 等資料存放區中的資料集執行互動式分析。

使用案例

- 建立具有 EMR Studio IDE 體驗的ETL指令碼。轉換後會IDE擷取現場部署資料，並將其存放在 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 上建立互動式端點的先決條件 EKS
- 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
- 互動端點上的其他操作

互動端點概觀

互動式端點為 Amazon EMR Studio 等互動式用戶端提供連線到EKS叢集EMR上的 Amazon 以執行互動式工作負載的功能。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，可與以 Jupyter 為基礎的 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 在庫伯尼特
- 帶 Scala 的 Apache Spark

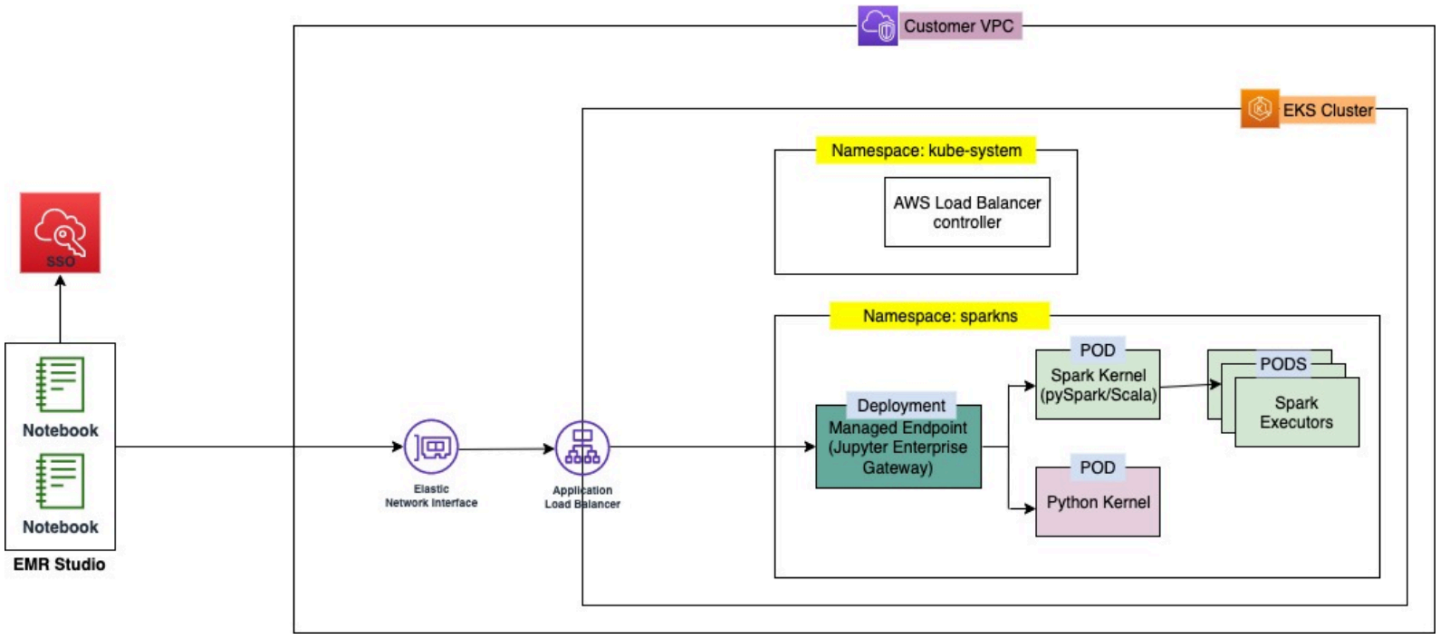
> **Note**
>
> Amazon 的EMR定EKS價適用於互動式端點和核心。如需詳細資訊，請參閱 Amazon EMR 的EKS定價頁面。

EMR工作室需要以下實體與 Amazon 連EMR接EKS。

- EKS虛擬叢集EMR上的 Amazon — 虛擬叢集是您向 Amazon 註冊的 Kubernetes 命名空間。EMRAmazon EMR 使用虛擬叢集執行任務和託管端點。可使用相同實體叢集來支援多個虛擬叢集。不過，每個虛擬叢集都會對應至 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。
- EKS互動式端點EMR上的 Amazon — 互動式端點是 EMR Studio 使用者可以連接工作區的HTTPS端點。您只能從EMR工作室存取HTTPS端點，並在 Amazon EKS 叢集的 Amazon 虛擬私有雲端 (AmazonVPC) 的私有子網路中建立端點。

  Python PySpark，和星火斯卡拉內核使用EKS任務執行角色 Amazon EMR 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。建議您為不同的使用者建立個別的端點，並讓使用者擁有不同的 AWS Identity and Access Management (IAM) 角色。
- AWS Ap@@ plication Load Balancer 器控制器 — AWS 應用程式負載平衡器控制器可管理 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 器 (ALB)。在 Amazon EKS 群集之外，但在同一個 Amazon 內部 ALB公開 Kubernetes 服務，例如交互式端點。VPC當您建立互動式端點時，也會部署 Ingress 資源，透過以ALB供互動式用戶端連線至的方式公開互動式端點。您只需為每個 Amazon EKS 叢集安裝一個 Ap AWS plication Load Balancer 器控制器。

下圖描述了在 Amazon EMR 上EKS的交互式端點架構。Amazon EKS 叢集包含用於執行分析工作負載的運算和互動式端點。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。建立互動式端點時，Amazon EMR EKS 控制平面會在 Amazon EKS 叢集中建立互動式端點部署。此外，負載平衡器控制器會建立應用程式負載平衡器輸 AWS 入的執行個體。應用程式負載平衡器為像 EMR Studio 這樣的用戶端提供外部介面，以連接到 Amazon EMR 叢集並執行互動式工作負載。



在 Amazon EMR 上建立互動式端點的先決條件 EKS

本節說明設定互動式端點的先決條件，讓 EMR Studio 可用來連接EKS叢集EMR上的 Amazon 和執行互動式工作負載。

AWS CLI

請按照中安裝 AWS CLI的步驟安裝最新版本的 AWS Command Line Interface (AWS CLI)。

安裝 eksctl

遵循 安裝 eksctl 中的步驟，安裝最新版本的 eksctl。如果您的 Amazon EKS 叢集使用 Kubernetes 1.22 版或更新版本，請使用大於 0.117.0 的 eksctl 版本。

Amazon EKS 集群

創建一個 Amazon EKS 群集。EMR在 Amazon 上將叢集註冊為虛擬叢集EKS。以下是此叢集的要求和考量事項。

- 叢集必須與您的工作EMR室位於相同的 Amazon Virtual Private Cloud (VPC) 中。
- 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。

- EMRStudio 和用於註冊虛擬叢集的 Amazon EKS 叢集之間至少必須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您的工作室和 Amazon EKS 叢集：

  - 建立 Amazon EKS 叢集，並將其與屬於您 EMR Studio 的子網路產生關聯。
  - 或者，建立 EMR Studio 並為您的 Amazon EKS 叢集指定私有子網路。
- EMR在EKS互動式端點AMIs上，Amazon 不支援 Amazon EKS 優化ARM的 Amazon Linux。
- 互動式端點可與使用 Kubernetes 版本最高 1.29 的 Amazon EKS 叢集搭配使用。
- 僅支援 Amazon EKS 受管節點群組。

授予 Amazon EMR 的叢集存取權 EKS

使用授與 Amazon EMR 的叢集存取權中的步驟，授與 Amazon EKS 存EMR取叢集中特定命名空間的存取權限。EKS

在 Amazon EKS 群集IRSA上激活

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IAM角色 (IRSA) 中的步驟進行操作。

建立IAM工作執行角色

您必須建立IAM角色才能在 Amazon 上在EKS互動式端點EMR上執行工作負載。在本文件中，我們將此IAM角色稱為工作執行角色。這個IAM角色會指派給互動式端點容器和當您使用 EMR Studio 提交工作時所建立的實際執行容器。您需要 Amazon 上的任務執行角色的 Amazon EMR 資源名稱（ARN）EKS。這需要兩個步驟：

- 建立工作執行的IAM角色。
- 更新作業執行角色的信任政策 。

授予使用者EMR對 Amazon 的存取權 EKS

提出建立互動式端點請求的IAM實體 (使用者或角色) 也必須具有下列 Amazon EC2 和`emr-containers`許可。請按照中描述的步驟授授予使用者EMR對 Amazon 的存取權 EKS予這些許可，以允許 Amazon EMR on 建立、管理和刪除EKS將入站流量限制到互動式端點的負載平衡器的安全群組。

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"
```

註冊 Amazon EKS 群集與 Amazon EMR

設定虛擬叢集並將其對應至 Amazon EKS 叢集中要執行任務的命名空間。對於 AWS Fargate僅限叢集，請為EKS虛擬叢集EMR上的 Amazon 和 Fargate 設定檔使用相同的命名空間。

如需在EKS虛擬叢集上設定 Amazon EMR 的相關資訊，請參閱註冊 Amazon EKS 群集與 Amazon EMR。

將 AWS Load Balancer 控制器部署到 Amazon EKS 叢集

您的 Amazon EKS 叢集需要 Ap AWS plication Load Balancer。您只需為每個 Amazon EKS 叢集設定一個 Application Load Balancer 器控制器。如需設定 Ap AWS plication Load Balancer 器控制器的相關資訊，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

本頁說明如何使用 AWS 指令行介面 (AWS CLI) 建立互動式端點。

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 在EKS支持使用 Amazon 6.7.0 及更高EMR版本創建交互式端點。

```
aws emr-containers create-managed-endpoint \
--type JUPYTER_ENTERPRISE_GATEWAY \
```

```
--virtual-cluster-id 1234567890abcdef0xxxxxxxx \
--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文件中創建具有指定參數的交互式端點

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`aws emr-containers create-managed-endpoint`會建立自我簽署的憑證，允許 EMR Studio 和互動式端點伺服器之間的HTTPS通訊。

如果您執行`create-managed-endpoint`但尚未完成先決條件，Amazon 會EMR傳回錯誤訊息，其中包含您必須採取的動作才能繼續。

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EMR在 Amazon 上註冊的虛擬叢集識別碼EKS。

`--name`

互動式端點的描述性名稱，可協助 EMR Studio 使用者從下拉式清單中選取它。

`--execution-role-arn`

Amazon 上的任務執行角色的 IAM Amazon 資源名稱 (ARN) EKS 是EMR在先決條件中建立的。

`--release-label`

用於端點的 Amazon EMR 版本發行版本的版本標籤。例如：`emr-6.9.0-latest`。Amazon EMR 上的EKS支持與 Amazon 6.7.0 及更高EMR版本的交互式端點。

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。您可以使用簡寫語法來提供配置，也可以參考檔案中的配置物件。JSON

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。您可以在單一物件中為多個應用程式指定多個分JSON類。可用的組態分類因 Amazon EMR EKS 發行版本而有所不同。如需每個 Amazon EMR 版本可用的組態分類清單EKS，請參閱Amazon EMR 上EKS發布。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，請參閱重複企業閘道 (JEG) 組態選項。

配置互動端點的設定

監控 Spark 任務

為了監控和疑難排解故障，請設定互動式端點，以便使用端點啟動的任務可以將日誌資訊傳送到 Amazon S3、Amazon CloudWatch Logs 或兩者。以下各節說明如何將 Spark 應用程式日誌傳送至 Amazon S3，以取得您EMR在EKS互動式端點上使用 Amazon 啟動的 Spark 任務。

設IAM定 Amazon S3 日誌的政策

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。Replace (取代) *DOC-EXAMPLE-BUCKET-LOGGING* 使用您的日誌存儲桶的名稱。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:PutObject",
                "s3:GetObject",
                "s3:ListBucket"
            ],
            "Resource": [
                "arn:aws:s3:::DOC-EXAMPLE-BUCKET-LOGGING",
                "arn:aws:s3:::DOC-EXAMPLE-BUCKET-LOGGING/*",
            ]
        }
    ]
}
```

> **Note**
>
> Amazon EMR 上也EKS可以創建一個 S3 存儲桶。如果 S3 儲存貯體不可用，請在IAM政策中包含`s3:CreateBucket`權限。

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您的 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 6.3.0 及更高EMR版本目前支援網繭範本。

如需有關網繭範本的詳細資訊，請參閱《Amazon EMR EKS 開發指南》中的使用網繭範本。

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網繭部署到節點群組

JEG(Jupyter 企業閘道) 網繭放置是一項功能，可讓您在特定節點群組上部署互動式端點。使用此功能，可以設定互動端點的設定，例如 `instance type`。

將JEG網繭與受管理的節點群組產生關聯

下列組態內容可讓您指定要部署JEG網繭之 Amazon EKS 叢集上受管節點群組的名稱。

```
//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，通常情況下，如果您使用建立叢集和受管節點群組`eksctl`。您可以使用以下步驟在 AWS 控制台中驗證這一點。

1. 前往 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加至 Amazon EKS 叢集安全群組，則需要將`for-use-with-emr-containers-managed-endpoint-sg=`*`ClusterName`*`/`*`NodeGroupName`*標籤附加到節點群組安全群組。使用下列步驟來附接此標籤。

1. 前往 Amazon 主EC2控台，然後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網繭與自我管理的節點群組建立關聯

下列組態內容可讓您在將部署JEG網繭的 Amazon EKS 叢集上指定自我管理或非受管節點群組的名稱。

```
//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. 導航到 Amazon EC2 控制台。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網繭與受管節點群組與隨需執行個體建立關聯

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。下列範例顯示如何針對JEG網繭使用隨需 Amazon EC2 執行個體。

```
--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。

重複企業閘道 (JEG) 組態選項

Amazon EMR 上EKS使用 Jupyter 企業閘道（JEG）打開交互式端點。您可以在建立端點時，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 會話參數

從 Amazon EMR EKS 版本 6.9.0 開始，在 Amazon EMR Studio 中，您可以通過在EMR筆記本電腦單元中執行`%%configure`魔術命令來調整與 PySpark 會話關聯的 Spark 配置。

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，您可以用來將檔案和 jar 相依性新增至 Spark 執行階段。pyFiles

```
%%configure -f
{
  "files": "s3://test-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 執行互動式工作負載時具有應用程式的正確相依性，您可以為互動式端點自訂 Docker 映像，並執行自訂的基本核心映像。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URI中的基本映像檔必須使用notebook-spark來取代. 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，請參閱如何選擇基本映像 URI。

2. 建立可與自訂映像搭配使用的互動端點。

   a. 創建custom-image-managed-endpoint.json具有以下內容的JSON文件。這個例子使用 Amazon EMR 版本 6.9.0。

      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式端點，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 Connect 到互動式端點。有關更多信息和要完成的步驟，請參閱工作室文檔中 EMR的 Amazon EKS 部分中的從 AWS 工作室連接。

監控互動端點

EMR在 6.10 版及更新EKS版本的 Amazon 中，互動式端點會發出 Amazon CloudWatch 指標，用於監控核心生命週期操作和疑難排解。指標是由互動式用戶端 (例如 EMR Studio 或自行託管 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。互動式端點發出的指標會顯示在您帳戶中的自訂命名空間下方。EMRContainers

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
4 XXError	當操作的請求導致在處理過程中出現 4xx 錯誤時發出。	計數
5 XXError	當操作的請求導致 5Xxx 伺服器端錯誤時發出。	計數

指標	描述	單位
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。
`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。

operationName	操作說明
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. KernelLaunchSuccess 具有統計值SUM和期間的度量1 day將提供過去 24 小時內發出的所有成功核心啟動要求。

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

您可以在 Amazon EC2 執行個體或您自己的 Amazon EKS 叢集上，以自託管 Jupyter JupyterLab 筆記本的形式託管和管理 Jupyter 或筆記本。然後，使用自助託管的 Jupyter 筆記本執行互動式工作負載。以下各節將逐步介紹在 Amazon 叢集上設定和部署自我託管 Jupyter 筆記本的程序。EKS

在叢集上建立自我代管的 Jupyter 筆記本 EKS

- 建立安全群組
- EMR在EKS互動式端點上建立 Amazon
- 擷取互動式端點URL的閘道伺服器

- 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

您必須先建立安全性群組來控制 JupyterLab 筆記本與互動式端點之間的流量，才能建立互動式端點並執行自我裝載的 Jupyter 或筆記本。若要使用 Amazon EC2 主控台或 Amazon EC2 SDK 建立安全群組，請參閱 Amazon 使用EC2者指南中建立安全群組中的步驟。您應該在您要部署筆記型電腦伺服器的VPC位置建立安全性群組。

若要遵循本指南中的範例，請使用與 Amazon EKS 叢集VPC相同的方法。如果您想要將筆記本託管在與 Amazon EKS 叢集不同的VPC筆記本中，則可能需要在這兩VPCs者之間建立對等連接。VPC如需在兩者之間建立對等連線的步驟VPCs，請參閱 Amazon VPC 入門指南中的建立對VPC等連線。

您需要安全群組的 ID，才能在下一個步驟中EMR在EKS互動式端點上建立 Amazon。

EMR在EKS互動式端點上建立 Amazon

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 。

插入安全 ID 代替 *your-notebook-security-group-id* 在以下配置覆蓋設置中：

```
--configuration-overrides '{
    "applicationConfiguration": [
        {
            "classification": "endpoint-configuration",
            "properties": {
                "notebook-security-group-id": "your-notebook-security-group-id"
            }
        }
    ],
    "monitoringConfiguration": {
    ...'
```

擷取互動式端點URL的閘道伺服器

建立互動式端點後，請URL使用中的`describe-managed-endpoint`指令擷取閘道伺服器 AWS CLI。您需要此功URL能才能將筆記型電腦連接到端點。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。當您部署自託管URL的 Jupyter 或筆記本時，您需要此功能才能將筆記本連接到端點。 JupyterLab

擷取驗證字符以連接到互動端點

若要從 Jupyter 或 JupyterLab 筆記本連線到互動式端點，您必須使用. `GetManagedEndpointSessionCredentials` API 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

您ARN的端點。您可以ARN在`describe-managed-endpoint`呼叫的結果中找到。

virtualClusterArn

虛擬叢集ARN的。

executionRoleArn

執ARN行角色的。

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。請記下您在部署自我託管 Jupyter 或筆記本時將使用的*session-token*值。JupyterLab

```
{
    "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，您可以嘗試此範例程序，透過互動式端點將 JupyterLab 筆記本部署到 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 擷取互動式端點URL的閘道伺服器 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. 將網繭規格部署到您的 Amazon EKS 叢集：

```
kubectl apply -f notebook.yaml -n namespace
```

這將啟動EMR在EKS交互式端點上連接到 Amazon 的最小 JupyterLab 筆記本電腦。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https://console.aws.amazon.com/eks/家裡 #/集群打開 Amazon EKS 控制台。

   c. 導覽至 Amazon EKS 叢集的運算索引標籤，然後選取`describe pod`命令所識別的節點。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。

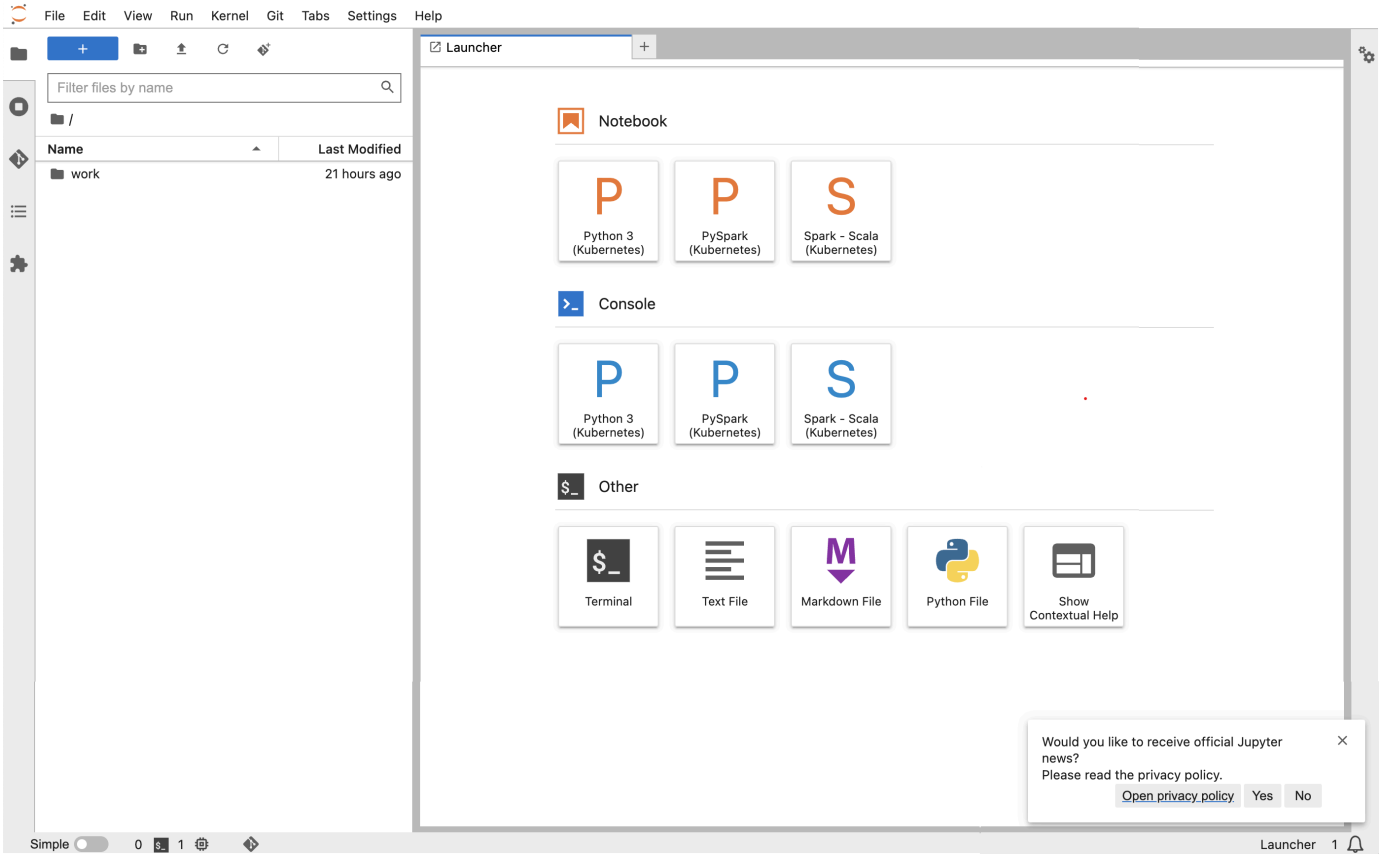
e. 如果您要在上部署 Jupyter 筆記本網繭 AWS Fargate，請建立套用[SecurityGroupPolicy]至具有角色標籤的 Jupyter 筆記本網繭：

```
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，端口轉發，以便您可以在本地訪問接 JupyterLab 口：

```
kubectl port-forward jupyter-notebook 8888:8888 -n namespace
```

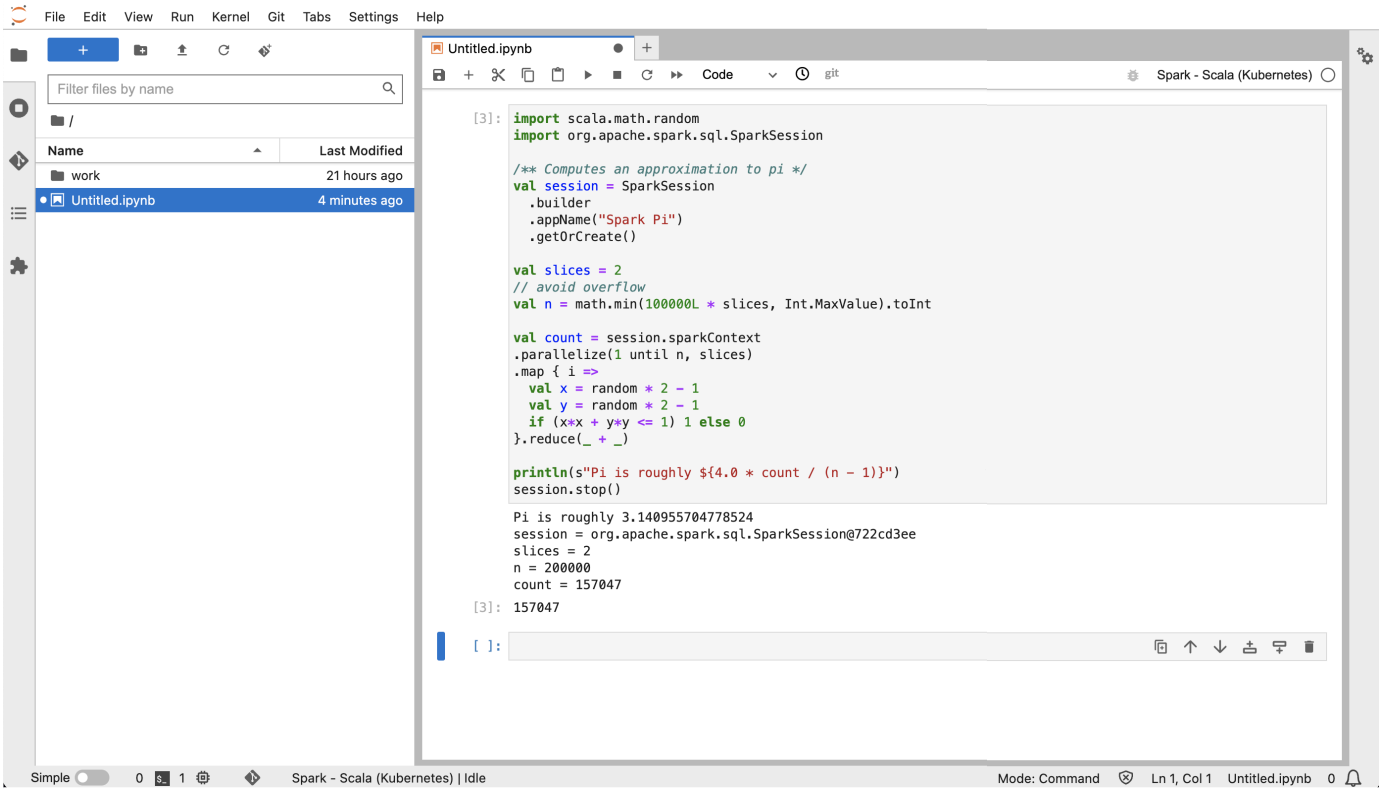
運行後，導航到本地瀏覽器並訪問`localhost:8888`以查看 JupyterLab 界面：



6. 從中 JupyterLab創建一個新的 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。

互動端點上的其他操作

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

建立互動式端點後，您可以使用describe-managed-endpoint AWS CLI 指令擷取其詳細資料。插入您自己的值 *managed-endpoint-id*, *virtual-cluster-id* 和 *region*:

```
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，具有指定的端點ARN，例如 ID 和名稱。

```
{
   "id": "as3ys2xxxxxxx",
   "name": "endpoint-name",
    "arn": "arn:aws:emr-containers:us-east-1:1828xxxxxxxx:/virtualclusters/
lbhl6kwwyoxxxxxxxxxxxxxx/endpoints/as3ysxxxxxxxx",
    "virtualClusterId": "lbhl6kwwyoxxxxxxxxxxxxxx",
    "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

使用此命`list-managed-endpoints` AWS CLI 令可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EKS虛擬叢集EMR上與 Amazon 相關聯的互動式端點，請使用delete-managed-endpoint AWS CLI 指令。刪除互動式端點時，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 — 與您要刪除之互動式端點相關聯的虛擬叢集識別碼。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"
}
```

EMR在 Amazon 上將數據上傳到 Amazon S3 快遞一個區域 EKS

使用 Amazon 7.2.0 及更高EMR版本，您可以將 Amazon EKS 與 Amazon EMR S3 快速單區域儲存類別搭配使用，以提高執行任務和工作負載時的效能。S3 Express One Zone 是一種高效能的單區域 Amazon S3 儲存類別，可為大多數對延遲敏感的應用程式提供一致、10 毫秒的資料存取。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。

必要條件

您必須具備下列先決條件EKS，才能搭配 Amazon EMR 使用 S3 快速單一區域：

- 已完成 Amazon EMR 的設置EKS。
- 在上設定 Amazon 之後 EMREKS，請建立虛擬叢集。

開始使用 S3 Express One Zone

請按照下列步驟開始使用 S3 快速單區

1. 將CreateSession權限新增至您的工作執行角色。當 S3 Express 單一區域最初在 S3 物件PUT上執行或類似GET的動作時，儲存類別會代表您呼叫CreateSession。LIST以下是如何授與CreateSession權限的範例。

```
{
  "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 來訪問 S3 快速存儲桶，因此請更改 Amazon S3 URIs 以使用該s3a方案以使用連接器。如果他們不使用該方案，則可以更改用於s3和s3n方案的文件系統實現。

   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，請指定下列叢集配置：

   ```
   [
     {
       "Classification": "core-site",
       "Properties": {
         "fs.s3n.impl": "org.apache.hadoop.fs.s3a.S3AFileSystem",
         "fs.AbstractFileSystem.s3n.impl": "org.apache.hadoop.fs.s3a.S3A"
       }
     }
   ]
   ```

3. 在您的火花提交配置中，使用 Web 身份憑證提供者。

   ```
   "spark.hadoop.fs.s3a.aws.credentials.provider=com.amazonaws.auth.WebIdentityTokenCredential
   ```

監控任務

主題

- 使用 Amazon CloudWatch 活動監控任務
- 透過活動在 EKS 上自動執行 Amazon EMR CloudWatch
- 範例：設定調用 Lambda 的規則
- 使用 Amazon CloudWatch 事件以重試政策監控任務的驅動程式網繭

使用 Amazon CloudWatch 活動監控任務

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。您也可以使用 Amazon E CloudWatch vents 來定義任務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。如需有關 CloudWatch 事件的詳細資訊，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 E CloudWatch vents 設定事件處理的規則時，JSON 物件特別重要，因為規則會尋求符合 JSON 物件中的模式。有關更多信息，請參閱 Amazon EventBridge 用戶指南中的 Amazon EventBridge 事件模式和 EKS 事件上的 Amazon EMR。

作業執行狀態變更事件

State	嚴重性	訊息
SUBMITTED	INFO	Job 執行 *JobRunId*(*JobRunName*) 已*VirtualClusterId* 在 UTC *##*順利提交至虛擬叢集。
RUNNING (執行中)	INFO	虛擬叢集中的 Job 執行 *JobRunId*(*JobRunName*) 會在時*#VirtualClusterId* 開始執行。

State	嚴重性	訊息
COMPLETED (已完成)	INFO	虛擬叢集中的 Job 執行 *jobRunId*(*JobRunName*) *VirtualClusterId* 依時*#*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	*####VirtualClusterId ###Job # ##JobRunId(JobRunName) ######### #Job ########*
失敗	ERROR	虛擬叢集中的 Job 執行 *JobRunId*(*JobRunName*) 在時*#VirtualClusterId* 失敗。

透過活動在 EKS 上自動執行 Amazon EMR CloudWatch

您可以使用 Amazon E CloudWatch vents 將 AWS 服務自動化，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傳送至「 CloudWatch 活動」。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：

- 調用一 AWS Lambda 個函數
- 呼叫 Amazon EC2 執行命令
- 將事件轉傳至 Amazon Kinesis Data Streams
- 啟動 AWS Step Functions 狀態機
- 通知 Amazon Simple Notification Service (SNS) 主題或 Amazon Simple Queue Service (SQS) 佇列

在 EKS 上搭配 Amazon EMR 使用 CloudWatch 事件的一些範例包括：

- 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

CloudWatch 針對 "" "detail-type:EMR Job Run State Change" 的事件是由 Amazon EMR 在 EKS 上針對SUBMITTED、RUNNINGCANCELLED、FAILED和COMPLETED狀態變更產生。

範例：設定調用 Lambda 的規則

使用下列步驟設定在發生「EMR Job 執行狀態變更」 CloudWatch 事件時叫用 Lambda 的事件規則。

```
aws events put-rule \
--name cwe-test \
--event-pattern '{"detail-type": ["EMR Job Run State Change"]}'
```

將您擁有的 Lambda 函數新增為新目標，並授予「 CloudWatch 事件」權限以叫用 Lambda 函數，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 事件以重試政策監控任務的驅動程式網繭

您可以使用 CloudWatch 事件監視已在具有重試原則的作業中建立的驅動程式網繭。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 設置 Amazon EMR 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 的教程 EKS

本節說明EMR在EKS應用程式上使用 Amazon 時的常見使用案例。

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
- 使用 AWS Glue 與熔接
- 使用阿帕奇胡迪與阿帕奇 Flink
- 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

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 發布 7.0.0 及更高版本使用三角洲湖 3.0，重命名`delta-core.jar`為。`delta-spark.jar`如果您使用 Amazon EMR 7.0.0 或更高版本，請務必使用正確的檔案名稱，如下列範例所示：
>
> ```
> --jars local:///usr/share/aws/delta/lib/delta-spark.jar
> ```

2. 包括三角洲湖的其他配置，並使用 AWS Glue 數據目錄作為您的中繼存儲。

```
--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

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://DOC-EXAMPLE-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。

使用 PyFlink

EKS 上的 Amazon EMR 發布 6.15.0 及更高版本的支持。 PyFlink如果您已經有 PyFlink 指令碼，您可以執行下列其中一項作業：

- 創建一個包含 PyFlink 腳本的自定義映像。
- 將您的指令碼上傳到 Amazon S3 位置

如果您還沒有指令碼，可以使用下列範例來啟動 PyFlink 工作。此範例會從 S3 擷取指令碼。如果您使用的自訂映像檔已包含在影像中的指令碼，則必須將指令碼路徑更新為儲存指令碼的位置。如果指令碼位於 S3 位置，EKS 上的 Amazon EMR 將擷取該指令碼，並將其放置在 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
```

使用 AWS Glue 與熔接

使用 Apache Flink 版本 6.15.0 及更高版本的 EKS 支援使用 AWS Glue 資料型錄做為串流和批次 SQL 工作流程的中繼資料存放區的 Amazon EMR。

您必須先建立一個名`default`為的 AWS Glue 資料庫，做為 Flink SQL 目錄。此 Flink 目錄存儲元數據，例如數據庫，表格，分區，視圖，函數以及其他訪問其他外部系統中的數據所需的信息。

```
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 指令碼快速發出一些 Flink SQL 陳述式，以便與 AWS Glue 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
```

以下是您的 PyFlink 指令碼可能看起來像的範例。

```
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()
```

使用阿帕奇胡迪與阿帕奇 Flink

Apache Hudi 是一個開放原始碼的資料管理架構，其中包含插入、更新、更新和刪除等記錄層級作業，您可以使用這些作業簡化資料管理和資料管線開發。Hudi 與 Amazon S3 中的有效資料管理結合使用

時，可讓您即時擷取和更新資料。Hudi 會維護您在資料集上執行的所有作業的中繼資料，因此所有動作都會保持原子和一致性。

阿帕奇胡迪是可EMR在 Amazon EKS 與阿帕奇 Flink 與 Amazon EMR 版本 7.2.0 及更高版本。請參閱下列步驟，瞭解如何開始使用和提交 Apache Hudi 工作。

提交阿帕奇胡迪工作

請參閱下列步驟，瞭解如何提交 Apache Hudi 工作。

1. 建立名為的 AWS Glue 資料庫`default`。

```
aws glue create-database --database-input "{\"Name\":\"default\"}"
```

2. 請遵循 Flink 庫伯內特斯運算SQL子範例來建立檔案。`flink-sql-runner.jar`
3. 創建一個 Hudi SQL 腳本，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. 將您的 Hudi SQL 指令碼和`flink-sql-runner.jar`檔案上傳到 S3 位置。
5. 在您的`FlinkDeployments`YAML檔案中，`hudi.enabled`將設定為`true`。

```
spec:
  flinkConfiguration:
    hudi.enabled: "true"
```

6. 創建一個YAML文件來運行您的配置。此範例檔案的名稱為`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-7.2.0-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 胡迪工作提交給 Flink 庫伯內特斯操作員。

```
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 設置 Amazon EMR 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

使用 AWS Secrets Manager 來擷取憑證並連線到 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

下列程式碼範例用 PySpark 於透過資料來源 API 和 SparkSQL，從 Amazon Redshift 資料庫讀取和寫入範例資料。

Data source API

用 PySpark 於使用資料來源 API 從 Amazon Redshift 資料庫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

用於 PySpark 使用 SparkSQL 從 Amazon Redshift 資料庫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

- 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。如需了解相關範例，請參閱使用 AWS Secrets Manager 擷取連線至 Amazon Redshift 的登入資料。
- 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。

概要

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. 建立 Amazon EMR 的星火運營商 EKS

   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。

概要

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. 建立 Amazon EMR 的星火運營商 EKS

   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

雲端安全是 AWS 最重視的一環。身為 AWS 的客戶，您將能從資料中心和網路架構中獲益，這些都是專為最重視安全的組織而設計的。

安全性是 AWS 與您共同肩負的責任。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：

- 雲端本身的安全 – AWS 負責保護在 AWS Cloud 中執行 AWS 服務的基礎設施。AWS 也提供您可安全使用的服務。第三方稽核人員會定期測試和驗證我們安全性的有效性，作為 AWS 合規計畫的一部分。若要了解適用於 Amazon EMR 的合規計畫，請參閱 AWS 合規計畫的服務範圍 。
- 雲端內部的安全 – 您的責任取決於所使用的 AWS 服務。您也必須對其他因素負責，包括資料的機密性、您公司的請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。您也將了解如何使用其他 AWS 服務，幫助您監控並保護 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
- 記錄和監控
- EMR在 Amazon 上使用 Amazon S3 訪問贈款 EKS
- Amazon EMR on EKS 的合規驗證
- Amazon EMR on EKS 的恢復能力
- Amazon EMR 的基礎設施安全 EKS
- 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。執行應用程式碼的作業 Pod 會在連線至其他 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 上EKS發布及如何選擇基本映像 URI。

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。
- Kubernetes RBAC 許可能夠建立 Pod，以防止使用者使用 Kubernetes ServiceAccount (它可能映射至比使用者具有更多 AWS 權限的 IAM 角色) 建立 Pod。
- 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

- 建議將最低 AWS 許可指派給節點群組的 IAM 角色。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

了解如何將 AWS 共同責任模型套用於 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 AWS 雲端的全球基礎設施。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。如需有關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，建議您保護 AWS 帳戶憑證，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 AWS CloudTrail 設定 API 和使用者活動記錄。
- 使用 AWS 加密解決方案，以及 AWS 服務內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。
- 如果您在透過命令列介面或 API 存取 AWS 時，需要 FIPS 140-2 驗證的加密模組，請使用 FIPS 端點。如需 FIPS 和 FIPS 端點的詳細資訊，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 AWS 服務 (這些服務使用主控台、API、AWS CLI 或 AWS SDK) 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。您可從數個選項中選擇，包括 AWS Key Management Service 所管理的金鑰、Amazon S3 所管理的金鑰以及來自您所提供自訂提供者的金鑰和憑證。使用 AWS KMS 做為您的金鑰提供者時，將適用儲存體及使用加密金鑰的費用。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 AWS KMS 時，將適用儲存體及使用加密金鑰的費用。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 來設定適用於 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。物件在上傳到 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。有三種不同的 AWS 本機儲存選項可與 Kubernetes 搭配使用：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。如需其他資訊，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) 可協助系統管理員安全地控制 AWS 資源存取權。AWS 服務 IAM管理員控制哪些人可以通過身份驗證 (登入) 和授權 (具有權限) 在EKS資源EMR上使用 Amazon。IAM是您 AWS 服務 可以免費使用的。

主題

- 物件
- 使用身分驗證
- 使用政策管理存取權
- Amazon EMR 如EKS何與 IAM
- 使用 Amazon EMR on EKS 的服務連結角色
- Amazon EMR 的受管政策 EKS
- 搭配使用作業執行角色與 Amazon EMR on EKS
- Amazon 網站上的基於身份的政策示例 EMR EKS
- 標籤型存取控制政策
- Amazon EMR 在EKS身分識別和存取上的疑難

物件

你如何使用 AWS Identity and Access Management （ IAM ）不同，這取決於你在 Amazon EMR 上做的工作EKS。

服務使用者 — 如果您使用 Amazon EMR on EKS 服務執行工作，則管理員會為您提供所需的登入資料和許可。當您EMR在EKS功能上使用更多 Amazon 來完成工作時，您可能需要額外的許可。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。如果您無法在EMR上存取 Amazon 中的某項功能EKS，請參閱Amazon EMR 在EKS身分識別和存取上的疑難。

服務管理員 — 如果您負責 Amazon 公司EMR的EKS資源，則可能擁有對 Amazon EMR 的完全訪問權限EKS。您的任務是確定服務使用者應存取哪些 Amazon EMR 的EKS功能和資源。然後，您必須向IAM管理員提交請求，才能變更服務使用者的權限。檢閱此頁面上的資訊，以瞭解的基本概念IAM。若要進一步了解貴公司如何EMR在 Amazon 上使IAM用EKS，請參閱Amazon EMR 如EKS何與 IAM。

IAM管理員 — 如果您是管理IAM員，您可能想要了解如何編寫政策以管理 Amazon EMR 的存取權限的詳細資訊EKS。若要檢視您可以在中使用的EKS基於身分的政策的 Amazon EMR 範例IAM，請參閱。Amazon 網站上的基於身份的政策示例 EMR EKS

使用身分驗證

驗證是您 AWS 使用身分認證登入的方式。您必須以IAM使用者身分或假設IAM角色來驗證 (登入 AWS)。 AWS 帳戶根使用者

您可以使用透過 AWS 身分識別來源提供的認證，以聯合身分識別身分登入。 AWS IAM Identity Center (IAM身分識別中心) 使用者、貴公司的單一登入驗證，以及您的 Google 或 Facebook 認證都是聯合身分識別的範例。當您以同盟身分登入時，您的管理員先前會使用IAM角色設定聯合身分識別。當您使 AWS 用同盟存取時，您會間接擔任角色。

根據您的使用者類型，您可以登入 AWS Management Console 或 AWS 存取入口網站。如需有關登入的詳細資訊 AWS，請參閱《AWS 登入 使用指南》 AWS 帳戶中的如何登入您的。

如果您 AWS 以程式設計方式存取，請 AWS 提供軟體開發套件 (SDK) 和命令列介面 (CLI)，以使用您的認證以密碼編譯方式簽署您的要求。如果您不使用 AWS 工具，則必須自行簽署要求。如需使用建議的方法自行簽署要求的詳細資訊，請參閱使用IAM者指南中的簽署 AWS API要求。

無論您使用何種身分驗證方法，您可能都需要提供額外的安全性資訊。例如， AWS 建議您使用多重要素驗證 (MFA) 來增加帳戶的安全性。若要深入瞭解，請參閱使用AWS IAM Identity Center 者指南中的多重要素驗證和使用多重要素驗證 (MFA) AWS的使用IAM者指南。

AWS 帳戶 根使用者

當您建立時 AWS 帳戶，您會從一個登入身分開始，該身分可完整存取該帳戶中的所有資源 AWS 服務和資源。此身分稱為 AWS 帳戶 root 使用者，可透過使用您用來建立帳戶的電子郵件地址和密碼登入來存取。強烈建議您不要以根使用者處理日常任務。保護您的根使用者憑證，並將其用來執行只能由根使用者執行的任務。如需需要您以 root 使用者身分登入的完整工作清單，請參閱《使用指南》中的〈需要 root 使用者認證的IAM工作〉。

聯合身分

最佳作法是要求人類使用者 (包括需要系統管理員存取權的使用者) 使用與身分識別提供者的同盟，才能使用臨時登入資料進行存取 AWS 服務 。

聯合身分識別是來自企業使用者目錄的使用者、Web 身分識別提供者、Identi ty Center 目錄，或使用透過身分識別來源提供的認證進行存取 AWS 服務 的任何使用者。 AWS Directory Service同盟身分存取時 AWS 帳戶，他們會假設角色，而角色則提供臨時認證。

對於集中式存取權管理，我們建議您使用 AWS IAM Identity Center。您可以在 IAM Identity Center 中建立使用者和群組，也可以連線並同步處理至您自己身分識別來源中的一組使用者和群組，以便在所有應用程式 AWS 帳戶 和應用程式中使用。如需IAM身分識別中心的相關資訊，請參閱IAM識別中心是什麼？ 在《AWS IAM Identity Center 使用者指南》中。

IAM 使用者和群組

IAM使用者是您內部的身分，具 AWS 帳戶 有單一人員或應用程式的特定權限。在可能的情況下，我們建議您仰賴臨時登入資料，而不要建立具有長期認證 (例如密碼和存取金鑰) 的IAM使用者。不過，如果您的特定使用案例需要使用IAM者的長期認證，建議您輪換存取金鑰。如需詳細資訊，請參閱《使用指南》中的「IAM定期輪換存取金鑰」以瞭解需要長期認證的使用案例。

IAM群組是指定IAM使用者集合的身分識別。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的群組，IAMAdmins並授與該群組管理IAM資源的權限。

使用者與角色不同。使用者只會與單一人員或應用程式建立關聯，但角色的目的是在由任何需要它的人員取得。使用者擁有永久的長期憑證，但角色僅提供暫時憑證。要了解更多信息，請參閱《IAM用戶指南》中的創建用戶（而不是角色）的IAM時間。

IAM角色

IAM角色是您 AWS 帳戶 中具有特定權限的身份。它類似於用IAM戶，但不與特定人員相關聯。您可以切換角色來暫時擔任中 AWS Management Console 的角色。IAM您可以呼叫 AWS CLI 或 AWS API作業或使用自訂來擔任角色URL。如需有關使用角色方法的詳細資訊，請參閱《使用指南》中的IAM〈使用IAM角色〉。

IAM具有臨時認證的角色在下列情況下很有用：

- 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。如需聯合角色的相關資訊，

請參閱《使用指南》中的〈建立第三方身分識別提供IAM者的角色〉。如果您使用IAM身分識別中心，則需要設定權限集。為了控制身分驗證後可以存取的內IAM容，IAMIdentity Center 會將權限集與中的角色相關聯。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使用者權限 — IAM 使用者或角色可以假定某個IAM角色，暫時取得特定工作的不同權限。
- 跨帳戶存取 — 您可以使用IAM角色允許不同帳戶中的某個人 (受信任的主體) 存取您帳戶中的資源。角色是授予跨帳戶存取權的主要方式。但是，對於某些策略 AWS 服務，您可以將策略直接附加到資源（而不是使用角色作為代理）。若要瞭解跨帳戶存取角色與以資源為基礎的政策之間的差異，請參閱《IAM使用指南》IAM中的〈跨帳號資源存取〉。
- 跨服務訪問 — 有些 AWS 服務 使用其他 AWS 服務功能。例如，當您在服務中撥打電話時，該服務通常會在 Amazon 中執行應用程式EC2或將物件存放在 Amazon S3 中。服務可能會使用呼叫主體的許可、使用服務角色或使用服務連結角色來執行此作業。
  - 轉寄存取工作階段 (FAS) — 當您使用使用IAM者或角色執行中的動作時 AWS，您會被視為主參與者。使用某些服務時，您可能會執行某個動作，進而在不同服務中啟動另一個動作。FAS會使用主參與者呼叫的權限 AWS 服務，並結合要求 AWS 服務 向下游服務發出要求。FAS只有當服務收到需要與其他 AWS 服務 資源互動才能完成的請求時，才會發出請求。在此情況下，您必須具有執行這兩個動作的許可。有關提出FAS請求時的策略詳細信息，請參閱轉發訪問會話。
  - 服務角色 — 服務角色是指服務代表您執行動作所代表的IAM角色。IAM管理員可以從中建立、修改和刪除服務角色IAM。如需詳細資訊，請參閱《IAM使用指南》 AWS 服務中的建立角色以將權限委派給
  - 服務連結角色 — 服務連結角色是連結至. AWS 服務服務可以擔任代表您執行動作的角色。服務連結角色會顯示在您的中， AWS 帳戶 且屬於服務所有。IAM管理員可以檢視 (但無法編輯服務連結角色) 的權限。
- 在 Amazon 上執行的應用程式 EC2 — 您可以使用IAM角色來管理在執行個體上EC2執行的應用程式以及發出 AWS CLI 或 AWS API請求的臨時登入資料。這比在EC2實例中存儲訪問密鑰更好。若要將 AWS 角色指派給EC2執行個體並讓其所有應用程式都能使用，請建立附加至執行個體的執行個體設定檔。執行個體設定檔包含角色，可讓執行個體上EC2執行的程式取得臨時登入資料。如需詳細資訊，請參閱使用者指南中的使用IAM角色將許可授與在 Amazon EC2 執行個體上執行的應IAM用程式。

要了解是否使用IAM角色還是用IAM戶，請參閱《用戶指南》中的「IAM創建IAM角色的時機（而不是用戶）」。

使用政策管理存取權

您可以透 AWS 過建立原則並將其附加至 AWS 身分識別或資源來控制中的存取。原則是一個物件 AWS ，當與身分識別或資源相關聯時，會定義其權限。 AWS 當主參與者 (使用者、root 使用者或角色工作階段) 提出要求時，評估這些原則。政策中的許可決定是否允許或拒絕請求。大多數原則會 AWS 以JSON文件的形式儲存在中。如需有關JSON原則文件結構和內容的詳細資訊，請參閱《IAM使用指南》中的策略概觀。JSON

管理員可以使用 AWS JSON策略來指定誰可以存取什麼內容。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。若要授與使用者對所需資源執行動作的權限，IAM管理員可以建立IAM策略。然後，系統管理員可以將IAM原則新增至角色，使用者可以擔任這些角色。

IAM原則會定義動作的權限，不論您用來執行作業的方法為何。例如，假設您有一個允許 `iam:GetRole` 動作的政策。具有該原則的使用者可以從 AWS Management Console AWS CLI、或取得角色資訊 AWS API。

身分型政策

以身分識別為基礎的原則是您可以附加至身分識別 (例如使用者、使用IAM者群組或角色) 的JSON權限原則文件。這些政策可控制身分在何種條件下能對哪些資源執行哪些動作。若要瞭解如何建立以身分識別為基礎的策略，請參閱《IAM使用指南》中的〈建立IAM策略〉。

身分型政策可進一步分類成內嵌政策或受管政策。內嵌政策會直接內嵌到單一使用者、群組或角色。受管理的策略是獨立策略，您可以將其附加到您的 AWS 帳戶. 受管政策包括 AWS 受管政策和客戶管理的策略。若要了解如何在受管策略或內嵌策略之間進行選擇，請參閱《IAM使用手冊》中的「在受管策略和內嵌策略之間進行選擇」。

資源型政策

以資源為基礎的JSON策略是您附加至資源的政策文件。以資源為基礎的政策範例包括IAM角色信任政策和 Amazon S3 儲存貯體政策。在支援資源型政策的服務中，服務管理員可以使用它們來控制對特定資源的存取權限。對於附加政策的資源，政策會定義指定的主體可以對該資源執行的動作以及在何種條件下執行的動作。您必須在資源型政策中指定主體。主參與者可以包括帳戶、使用者、角色、同盟使用者或。 AWS 服務

資源型政策是位於該服務中的內嵌政策。您無法在以資源為基礎的策略IAM中使用 AWS 受管政策。

存取控制清單 (ACLs)

存取控制清單 (ACLs) 控制哪些主參與者 (帳戶成員、使用者或角色) 具有存取資源的權限。ACLs類似於以資源為基礎的策略，雖然它們不使用JSON政策文件格式。

Amazon S3 和 Amazon VPC 是支持服務的示例ACLs。 AWS WAF若要進一步了解ACLs，請參閱 Amazon 簡單儲存服務開發人員指南中的存取控制清單 (ACL) 概觀。

其他政策類型

AWS 支援其他較不常見的原則類型。這些政策類型可設定較常見政策類型授予您的最大許可。

- 權限界限 — 權限界限是一項進階功能，您可以在其中設定以身分識別為基礎的原則可授與給IAM實體 (IAM使用者或角色) 的最大權限。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。如需有關權限界限的詳細資訊，請參閱《IAM使用指南》中的IAM實體的權限界限。
- 服務控制策略 (SCPs) — SCPs 是指定中組織或組織單位 (OU) 最大權限的JSON策略 AWS Organizations。 AWS Organizations 是一種用於分組和集中管理您企業擁 AWS 帳戶 有的多個服務。如果您啟用組織中的所有功能，則可以將服務控制策略 (SCPs) 套用至您的任何或所有帳戶。SCP限制成員帳戶中實體的權限，包括每個帳戶 AWS 帳戶根使用者。如需有關 Organizations 的詳細資訊SCPs，請參閱AWS Organizations 使用指南中的服務控制原則。
- 工作階段政策 – 工作階段政策是一種進階政策，您可以在透過編寫程式的方式建立角色或聯合使用者的暫時工作階段時，作為參數傳遞。所產生工作階段的許可會是使用者或角色的身分型政策和工作階段政策的交集。許可也可以來自資源型政策。所有這類政策中的明確拒絕都會覆寫該允許。如需詳細資訊，請參閱IAM使用指南中的工作階段原則。

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。若要瞭解如何在涉及多個原則類型時 AWS 決定是否允許要求，請參閱IAM使用指南中的原則評估邏輯。

Amazon EMR 如EKS何與 IAM

在您用IAM來管理EMR對 Amazon 的存取權限之前EKS，請先了解哪些IAM功能可與 Amazon EMR 搭配使用EKS。

IAM您可以EMR在 Amazon 上使用的功能 EKS

IAM特徵	Amazon EMR 的EKS支持
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
ACLs	否
ABAC(策略中的標籤)	是
暫時性憑證	是
主體許可	是
服務角色	否
服務連結角色	是

若要深入瞭解 Amazon EMR on EKS 和其他 AWS 服務如何使用大多數IAM功能，請參閱IAM使用者指南IAM中的可使用AWS 服務。

Amazon 網站上的基於身份的政策 EMR EKS

支援身分型政策：是

以身分識別為基礎的原則是您可以附加至身分識別 (例如使用者、使用IAM者群組或角色) 的JSON權限原則文件。這些政策可控制身分在何種條件下能對哪些資源執行哪些動作。若要瞭解如何建立以身分識別為基礎的策略，請參閱《IAM使用指南》中的〈建立IAM策略〉。

使用以IAM身分識別為基礎的策略，您可以指定允許或拒絕的動作和資源，以及允許或拒絕動作的條件。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。若要瞭解可在JSON策略中使用的所有元素，請參閱《使用IAM者指南》中的IAMJSON策略元素參考資料。

Amazon 網站上的基於身份的政策示例 EMR EKS

若要檢視 Amazon EMR 以EKS身分識別為基礎的政策範例，請參閱。Amazon 網站上的基於身份的政策示例 EMR EKS

Amazon EMR 內部基於資源的政策 EKS

支援資源型政策：否

以資源為基礎的JSON策略是您附加至資源的政策文件。以資源為基礎的政策範例包括IAM角色信任政策和 Amazon S3 儲存貯體政策。在支援資源型政策的服務中，服務管理員可以使用它們來控制對特定資源的存取權限。對於附加政策的資源，政策會定義指定的主體可以對該資源執行的動作以及在何種條件下執行的動作。您必須在資源型政策中指定主體。主參與者可以包括帳戶、使用者、角色、同盟使用者或。 AWS 服務

若要啟用跨帳戶存取，您可以在以資源為基礎的策略中指定一個或多個帳戶中的一個或多個帳戶中的IAM實體作為主體。新增跨帳戶主體至資源型政策，只是建立信任關係的一半。當主參與者和資源不同時 AWS 帳戶，受信任帳戶中的IAM管理員也必須授與主參與者實體 (使用者或角色) 權限，才能存取資源。其透過將身分型政策連接到實體來授與許可。不過，如果資源型政策會為相同帳戶中的主體授予存取，這時就不需要額外的身分型政策。如需詳細資訊，請參閱《IAM使用指南》IAM中的〈跨帳號資源存取〉。

Amazon EMR 的政策行動 EKS

支援政策動作：是

管理員可以使用 AWS JSON策略來指定誰可以存取什麼內容。也就是說，哪個主體在什麼條件下可以對什麼資源執行哪些動作。

JSON策略`Action`元素描述了您可以用來允許或拒絕策略中存取的動作。策略動作通常與關聯 AWS API操作具有相同的名稱。有一些例外情況，例如沒有匹配API操作的僅限權限的操作。也有一些作業需要政策中的多個動作。這些額外的動作稱為相依動作。

政策會使用動作來授予執行相關聯動作的許可。

若要查看 Amazon EMR 的EKS動作清單，請參閱服務授權參考EKS中的 Amazon EMR 動作、資源和條件金鑰。

Amazon EMR 中的政策動作在動作之前EKS使用以下前綴：

```
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 以EKS身分識別為基礎的政策範例，請參閱。Amazon 網站上的基於身份的政策示例 EMR EKS

Amazon EMR 的政策資源 EKS

支援政策資源：是

管理員可以使用 AWS JSON策略來指定誰可以存取什麼內容。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。

ResourceJSON原則元素會指定要套用動作的一或多個物件。陳述式必須包含 Resource 或 NotResource 元素。最佳做法是使用其 Amazon 資源名稱 (ARN) 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": "*"
```

要查看有EMR關EKS資源類型及其類型的 Amazon 列表ARNs，請參閱服務授權參考EKS中由 Amazon EMR 定義的資源。若要了解您可以為每個資源指定哪些動作，請參閱 Amazon EMR on ARN 的動作、資源和條件金鑰EKS。

若要檢視 Amazon EMR 以EKS身分識別為基礎的政策範例，請參閱。Amazon 網站上的基於身份的政策示例 EMR EKS

Amazon EMR 的政策條件密鑰 EKS

支援服務特定政策條件金鑰：是

管理員可以使用 AWS JSON策略來指定誰可以存取什麼內容。也就是說，哪個主體在什麼條件下可以對什麼資源執行哪些動作。

Condition 元素 (或 Condition 區塊) 可讓您指定使陳述式生效的條件。Condition 元素是選用項目。您可以建立使用條件運算子的條件運算式 (例如等於或小於)，來比對政策中的條件和請求中的值。

若您在陳述式中指定多個 Condition 元素，或是在單一 Condition 元素中指定多個索引鍵， AWS 會使用邏輯 AND 操作評估他們。如果您為單一條件索引鍵指定多個值，請使用邏輯OR運算來 AWS 評估條件。必須符合所有條件，才會授與陳述式的許可。

您也可以在指定條件時使用預留位置變數。例如，只有在IAM使用者名稱標記資源時，您才可以授與IAM使用者存取資源的權限。如需詳細資訊，請參閱《IAM使用指南》中的IAM政策元素：變數和標籤。

AWS 支援全域條件金鑰和服務特定條件金鑰。若要查看所有 AWS 全域條件索引鍵，請參閱《使用指南》中的AWS 全域條件內IAM容索引鍵。

若要查看 Amazon EMR 上的EKS條件金鑰清單，並瞭解您可以使用條件金鑰的動作和資源，請參閱服務授權參考EKS中適用EMR於 Amazon 的動作、資源和條件金鑰。

若要檢視 Amazon EMR 以EKS身分識別為基礎的政策範例，請參閱。Amazon 網站上的基於身份的政策示例 EMR EKS

Amazon EMR 上的訪問控制列表（ACLs）EKS

支持ACLs：無

存取控制清單 (ACLs) 控制哪些主參與者 (帳戶成員、使用者或角色) 具有存取資源的權限。ACLs類似於以資源為基礎的策略，雖然它們不使用JSON政策文件格式。

基於屬性的訪問控制（ABAC）與 Amazon 上 EMR EKS

支援 ABAC (策略中的標記)	是

以屬性為基礎的存取控制 (ABAC) 是一種授權策略，可根據屬性定義權限。在中 AWS，這些屬性稱為標籤。您可以將標籤附加至IAM實體 (使用者或角色) 和許多 AWS 資源。標記實體和資源是的第一步ABAC。然後，您可以設計ABAC策略，以便在主參與者的標籤與他們嘗試存取的資源上的標籤相符時允許作業。

ABAC在快速成長的環境中很有幫助，並且有助於原則管理變得繁瑣的情況。

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。

如需有關的詳細資訊ABAC，請參閱什麼是ABAC？ 在《IAM使用者指南》中。若要檢視包含設定步驟的自學課程ABAC，請參閱《使用指南》中的〈使用以屬性為基礎的存取控制 (ABAC) IAM 〉。

EMR在 Amazon 上使用臨時登入資料 EKS

支援臨時憑證：是

當您使用臨時憑據登錄時，某些 AWS 服務 不起作用。如需其他資訊，包括哪些 AWS 服務 與臨時登入資料搭配使用 AWS 服務 ，請參閱《IAM使用指南》IAM中的使用方式。

如果您使用除了使用者名稱和密碼以外的任何方法登入，則您正在 AWS Management Console 使用臨時認證。例如，當您 AWS 使用公司的單一登入 (SSO) 連結存取時，該程序會自動建立臨時認證。當您以使用者身分登入主控台，然後切換角色時，也會自動建立臨時憑證。如需有關切換角色的詳細資訊，請參閱《IAM使用者指南》中的〈切換到角色 (主控台)〉。

您可以使用 AWS CLI 或手動建立臨時認證 AWS API。然後，您可以使用這些臨時登入資料來存取 AWS。 AWS 建議您動態產生臨時登入資料，而非使用長期存取金鑰。如需詳細資訊，請參閱IAM。

Amazon EMR 上的跨服務主體許可 EKS

支援轉寄存取工作階段 (FAS)：是

當您使用使用IAM者或角色在中執行動作時 AWS，您會被視為主參與者。使用某些服務時，您可能會執行某個動作，進而在不同服務中啟動另一個動作。FAS會使用主參與者呼叫的權限 AWS 服務，並結合要求 AWS 服務 向下游服務發出要求。FAS只有當服務收到需要與其他 AWS 服務 資源互動才能完成的請求時，才會發出請求。在此情況下，您必須具有執行這兩個動作的許可。有關提出FAS請求時的策略詳細信息，請參閱轉發訪問會話。

Amazon EMR 的服務角色 EKS

支援服務角色	否

Amazon EMR 上的服務連結角色 EKS

支援服務連結角色	是

如需有關建立或管理服務連結角色的詳細資訊，請參閱使用IAM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，且內含該服務代您呼叫其他 AWS 服務所需的所有許可。

服務連結角色可讓設定 Amazon 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，並且該許可政策不能連接到任何其他 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 Yes (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。檢視 AWS Management Console 中的最新政策 AmazonEMRContainersServiceRolePolicy。

```
{
    "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"
            ],
            "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 CLI 或 AWS API 中，建立一個服務名稱為 `emr-containers.amazonaws.com` 的服務連結角色。如需詳細資訊，請參閱 IAM 使用者指南中的建立服務連結角色。如果您刪除此服務連結角色，您可以使用此相同的程序以再次建立該角色。

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。如需更多詳細資訊，請參閱 Amazon EMR on EKS 服務端點和配額。

Amazon EMR 的受管政策 EKS

檢視自 2021 年 3 月 1 日EKS起 Amazon AWS EMR 受管政策更新的詳細資訊。

變更	描述	日期
`AmazonEMRContainersServiceRolePolicy` -新增了描述和列出 Amazon EKS 節點群組、描述負載平衡器目標群組以及說明負載平衡器目標運作狀態的權限。	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
AmazonEMRContainersServiceRolePolicy -在中添加了導入和刪除證書的權限 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 開EKS始跟踪變化	Amazon EMR 開EKS始跟踪其 AWS 受管政策的更改。	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 設置 Amazon EMR 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。

```
  {
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Federated": "arn:aws:iam::AWS_ACCOUNT_ID: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。

```
  {
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "emr-containers:StartJobRun",
      "Resource": "arn:aws:emr-containers:REGION: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 網站上的基於身份的政策示例 EMR EKS

依預設，使用者和角色沒有EMR在EKS資源上建立或修改 Amazon 的權限。他們也無法使用 AWS Management Console、 AWS Command Line Interface (AWS CLI) 或執行工作 AWS API。若要授與使用者對所需資源執行動作的權限，IAM管理員可以建立IAM策略。然後，系統管理員可以將IAM原則新增至角色，使用者可以擔任這些角色。

若要瞭解如何使用這些範例原則文件來建立以IAM身分識別為基礎的JSON策略，請參閱使用指南中的IAM建立IAM策略。

有關 Amazon EMR on 定義的動作和資源類型的詳細資訊EKS，包括每種資源類型的格式，請參閱服務授權參考EKS中適用EMR於 Amazon 的動作、資源和條件金鑰。ARNs

主題

- 政策最佳實務
- EMR在EKS控制台上使用 Amazon
- 允許使用者檢視他們自己的許可

政策最佳實務

以身分識別為基礎的政策決定某人是否可以在您帳戶中的EKS資源EMR上建立、存取或刪除Amazon。這些動作可能會讓您的 AWS 帳戶產生費用。當您建立或編輯身分型政策時，請遵循下列準則及建議事項：

- 開始使用 AWS 受管原則並邁向最低權限權限 — 若要開始授與使用者和工作負載的權限，請使用可授與許多常見使用案例權限的AWS 受管理原則。它們在您的 AWS 帳戶. 建議您透過定義特定於您使用案例的 AWS 客戶管理政策，進一步降低使用權限。如需詳細資訊，請參閱AWS 《IAM使用指南》中針對工作職能的AWS 受管理策略或受管理的策略。
- 套用最低權限權限 — 當您使用原則設定權限時，IAM只授與執行工作所需的權限。為實現此目的，您可以定義在特定條件下可以對特定資源採取的動作，這也稱為最低權限許可。如需有關使用套用權限IAM的詳細資訊，請參閱《使用指南》IAM中的IAM《策略與權限》。
- 使用IAM策略中的條件進一步限制存取 — 您可以在策略中新增條件，以限制對動作和資源的存取。例如，您可以撰寫政策條件，以指定必須使用傳送所有要求SSL。您也可以使用條件來授與對服務動作的存取權 (如透過特定) 使用這些動作 AWS 服務，例如 AWS CloudFormation。如需詳細資訊，請參閱《IAM使用指南》中的IAMJSON策略元素：條件。
- 使用 IAM Access Analyzer 驗證您的原IAM則，以確保安全和功能性的權限 — IAM Access Analyzer會驗證新的和現有的原則，以便原則遵循IAM原則語言 (JSON) 和IAM最佳做法。IAMAccess Analyzer 提供超過 100 項原則檢查和可行的建議，協助您撰寫安全且功能正常的原則。如需詳細資訊，請參閱IAM使IAM用指南中的存取分析器原則驗證。
- 需要多因素驗證 (MFA) — 如果您的案例需要使IAM用者或 root 使用者 AWS 帳戶，請開啟以取得額外MFA的安全性。若要在呼叫API作業MFA時需要，請在原則中新增MFA條件。如需詳細資訊，請參閱《IAM使用指南》中的 < 設定MFA受保護的API存取 > 。

如需有關中最佳作法的詳細資訊IAM，請參閱《IAM使用指南》IAM中的「安全性最佳作法」。

EMR在EKS控制台上使用 Amazon

若要EMR在EKS主控台上存取 Amazon，您必須擁有最少一組許可。這些許可必須允許EKS您EMR在AWS 帳戶. 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。

您不需要為只對 AWS CLI 或撥打電話的使用者允許最低主控台權限 AWS API。而是只允許存取符合他們嘗試執行之API作業的動作。

為了確保使用者和角色仍然可以EMR在EKS主控台上使用 Amazon，請將 Amazon EMR on EKS `ConsoleAccess` 或`ReadOnly` AWS 受管政策附加到實體。如需詳細資訊，請參閱《使用指南》中的〈將權限新增至IAM使用者〉。

允許使用者檢視他們自己的許可

此範例顯示如何建立原則，讓使IAM用者檢視附加至其使用者身分識別的內嵌和受管理原則。此原則包含在主控台上或以程式設計方式使用或完成此動作的 AWS CLI 權限 AWS API。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
                "iam:GetUser"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
        {
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
            "Resource": "*"
        }
    ]
}
```

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。

下列範例示範將條件運算子與 Amazon EMR on EKS 條件金鑰搭配使用的不同案例和方法。這些IAM原則陳述式僅用於示範目的，不應在生產環境中使用。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。如需有關規劃和測試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。

以下範例示範以身分為基礎的許可政策，這些政策用於控制 Amazon EMR 在EKS虛擬叢集上允許的動作。

僅在具有特定標籤值的資源上允許動作

在下列原則範例中， StringEquals 條件運算子會嘗試比對 dev 與標籤部門的值。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。

```
{
  "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。

```
{
  "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。

```
{
  "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。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": [
        "emr-containers:TagResource",
        "emr-containers:UntagResource"
      ],
      "Resource": "*",
      "Condition": {
        "StringNotEquals": {
          "aws:ResourceTag/department": "dev"
        }
      }
    }
  ]
}
```

Amazon EMR 在EKS身分識別和存取上的疑難

使用下列資訊協助您診斷和修正使用 Amazon EMR 和時可能會遇到的常見問題IAM。EKS

主題

- 我沒有授權在 Amazon EMR 上執行操作 EKS
- 我沒有授權執行 iam：PassRole
- 我想允許我 AWS 帳戶以外的人EMR在EKS資源上訪問我的 Amazon

我沒有授權在 Amazon EMR 上執行操作 EKS

如果 AWS Management Console 告訴您您沒有執行動作的授權，則您必須聯絡您的管理員以尋求協助。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。

我沒有授權執行 iam：PassRole

如果您收到未獲授權執行iam:PassRole動作的錯誤訊息，則必須更新您的政策以允許您將角色傳遞給 Amazon EMR on EKS。

有些 AWS 服務 允許您將現有角色傳遞給該服務，而不是建立新的服務角色或服務連結角色。如需執行此作業，您必須擁有將角色傳遞至該服務的許可。

當名為的使用IAM者marymajor嘗試使用主控台在 Amazon EMR on 中執行動作時，會發生下列範例錯誤EKS。但是，動作請求服務具備服務角色授予的許可。Mary 沒有將角色傳遞至該服務的許可。

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

在這種情況下，Mary 的政策必須更新，允許她執行 iam:PassRole 動作。

如果您需要協助，請聯絡您的 AWS 管理員。您的管理員提供您的簽署憑證。

我想允許我 AWS 帳戶以外的人EMR在EKS資源上訪問我的 Amazon

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。對於支援以資源為基礎的政策或存取控制清單 (ACLs) 的服務，您可以使用這些政策授與人員存取您資源的權限。

如需進一步了解，請參閱以下內容：

- 若要了解 Amazon EMR on 是否EKS支援這些功能，請參閱Amazon EMR 如EKS何與 IAM。

- 若要瞭解如何提供您所擁有資 AWS 帳戶 源的存取權，請參閱《IAM使用指南》中的〈提供存取權給您 AWS 帳戶 所擁有的其他IAM使用者〉。
- 若要瞭解如何將您的資源存取權提供給第三方 AWS 帳戶，請參閱《IAM使用指南》中的提供第三方 AWS 帳戶 擁有的存取權。
- 若要瞭解如何透過聯合身分識別提供存取權，請參閱使用指南中的提供對外部驗證使用IAM者的存取權 (身分聯合)。
- 若要瞭解針對跨帳號存取使用角色與以資源為基礎的政策之間的差異，請參閱《使用IAM者指南》IAM中的〈跨帳號資源存取〉。

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 提供由使用者、角色或 AWS 服務在 Amazon EMR on EKS 中所採取之動作的記錄。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。如需更多詳細資訊，請參閱 使用 AWS CloudTrail 記錄 Amazon EMR on EKS API 呼叫。
- 搭配 Amazon EMR on EKS 使用 CloudWatch Events - CloudWatch Events 會提供近乎即時的系統事件串流，其會描述 AWS 資源的變化情形。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。如需更多詳細資訊，請參閱 使用 Amazon CloudWatch 活動監控任務。

使用 AWS CloudTrail 記錄 Amazon EMR on EKS API 呼叫

Amazon EMR on EKS 已與 AWS CloudTrail 整合，這項服務可提供由使用者、角色或 Amazon EKS 中 AWS 服務所採取之動作的記錄。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 Event history (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

當您建立帳戶時，系統即會在 AWS 帳戶中啟用 CloudTrail。在 Amazon EMR on EKS 中發生活動時，該活動將與事件歷史紀錄中的其他 AWS 服務事件一起記錄在 CloudTrail 事件中。您可以檢視、搜尋和下載 AWS 帳戶的最新事件。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 帳戶中正在進行的事件 (包括 Amazon EMR on EKS 的事件)，請建立追蹤。追蹤能讓 CloudTrail 將日誌檔交付至 Amazon S3 儲存貯體。根據預設，當您在主控台建立線索時，線索會套用到所有 AWS 區域。該追蹤會記錄來自 AWS 分割區中所有區域的事件，並將日誌檔案交付到您指定的 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 記錄檔案中產生項目。

每一筆事件或日誌項目都會包含產生請求者的資訊。身分資訊可協助您判斷下列事項：

- 該請求是否透過根或 AWS Identity and Access Management (IAM) 使用者憑證來提出。
- 提出該請求時，是否使用了特定角色或聯合身分使用者的暫時安全憑證。
- 該請求是否由另一項 AWS 服務提出。

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項目。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"
}
```

EMR在 Amazon 上使用 Amazon S3 訪問贈款 EKS

Amazon EMR 上的 S3 訪問授權概述 EKS

Amazon S3 存取授權使用 Amazon 6.15.0 及更高EMR版本時，提供可擴展的存取控制解決方案，您可以使用這些解決方案擴大對 Amazon S3 資料的存取。EMR EKS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 存取權授與擴大對 Amazon S3 資料的存取權限，超越執行時期IAM角色授予的許可或附加至可存取 Amazon EMR on EKS 叢集的身分的角色。

如需詳細資訊，請參閱 Amazon 管理指南中的使用適用於 Amazon EMR 的 S3 存取授權EMR管理存取和 Amazon 簡單儲存服務使用者指南中的使用 S3 存取授權管理存取。

本頁說明使EKS用 S3 存取授權整合在 Amazon EMR 上執行 Spark 任務的需求。開EMR啟 Amazon 時EKS，S3 存取授權需要任務的執行角色中的額外IAM政策聲明，以及 `StartJobRun`API. 如需使用其他 Amazon EMR 部署設定 S3 存取授權的步驟，請參閱下列文件：

- 透過 Amazon 使用 S3 存取授權 EMR
- 搭配EMR無伺服器使用 S3 存取授權

使用 S3 存取授權啟動EKS叢集EMR上的 Amazon，以進行資料管理

您可以在 Amazon EMR 上啟用 S3 訪問授權EKS並啟動 Spark 任務。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 EKS 叢集設定任務執行角色。包含執行 Spark 作業所需的必要IAM權限，以`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**
>
> 如果您指定具有任何其他權限直接存取 S3 的任務執行IAM角色，則無論您在 S3 Access Grants 中定義的許可為何，使用者都可以存取資料

2. 將 Amazon EMR 版本標籤為 6.15 或更高版本且`emrfs-site`分類的任務提交給EKS叢集EMR上的 Amazon，如下列範例所示。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-7.2.0-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"
         }
      }
    ],
  }
}
```

Amazon EMR 上的 S3 存取授予考量事項 EKS

如需在 Amazon EMR 上使用 Amazon S3 時的重要支援、相容性和行為資訊EKS，請參閱 Amazon EMR 管理指南EMR中的 S3 存取授與考量事項。

Amazon EMR on EKS 的合規驗證

在多個 AWS 合規計畫中，第三方稽核人員會評估 Amazon EMR on EKS 的安全與合規。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 區域與可用區域為中心建置的。AWS區域提供多個分開且隔離的實際可用區域，並以低延遲、高輸送量和高度備援聯網功能相互連結。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、容錯能力和擴充能力，均較單一或多個資料中心的傳統基礎設施還高。

如需有關 AWS 區域與可用區域的詳細資訊，請參閱 AWS 全球基礎設施。

除了 AWS 全球基礎設施外，Amazon EMR on EKS 還透過 EMRFS 提供與 Amazon S3 的整合，以支援資料恢復能力和備份需求。

Amazon EMR 的基礎設施安全 EKS

作為一項受管服務，Amazon EMR 受到 AWS 全球網路安全的保護。有關 AWS 安全服務以及如何 AWS 保護基礎結構的詳細資訊，請參閱AWS 雲端安全 若要使用基礎架構安全性的最佳做法來設計您的 AWS 環境，請參閱安全性支柱架構良 AWS 好的架構中的基礎結構保護。

您可以使用 AWS 已發佈的API呼叫EMR透過網路存取 Amazon。使用者端必須支援下列專案：

- 傳輸層安全性 (TLS)。我們需要 TLS 1.2 並推薦 TLS 1.3。
- 具有完美前向保密（ ）的密碼套件，例如（短暫的迪菲-赫爾曼PFS）或DHE（橢圓曲線短暫迪菲-赫爾曼）。ECDHE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主體相關聯的秘密存取金鑰來簽署。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 會處理訪客作業系統 (OS) 和資料庫修補、防火牆組態和災難復原等基本安全任務。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。使用介面 VPC 端點時，VPC 和 Amazon EMR on EKS 之間的通訊完全在 AWS 網路中進行。每個 VPC 端點皆會由一個或多個具私有 IP 地址彈性網絡介面 (ENI) 來表示，而該介面位於 VPC 子網路中。

介面 VPC 端點可以直接將 VPC 連接至 Amazon EMR on EKS，無需透過網際網路閘道、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。

可以使用 AWS Management Console 或 AWS Command Line Interface (AWS CLI) 命令，建立介面 VPC 端點來連接至 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》中的介面 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 可用來拒絕所有來自指定 AWS 帳戶之存取的 VPC 端點政策

下列 VPC 端點政策拒絕 AWS 帳戶 *123456789012* 對使用該端點之資源的任何存取。

```
{
    "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 端點政策只允許 AWS 帳戶 *123456789012* 中的 IAM 使用者 *lijuan* 的完整存取權。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權，但拒絕 AWS 帳戶 *123456789012* 對於在叢集 ID 為 *A1B2CD34EF5G* 的虛擬叢集上執行的動作的任何存取權。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。跨帳戶存取權可讓某個 AWS 帳戶中的使用者執行 Amazon EMR on EKS 作業，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，請在登入下列 AWS 帳戶時完成任務：

- `AccountA`-一個 AWS 帳戶，透過在 EKS 叢集上使用命名空間註冊 Amazon EMR 而在其中建立 Amazon EMR on EKS 虛擬叢集。
- `AccountB` – 一個 AWS 帳戶，它包含您希望 Amazon EMR on EKS 作業存取的 Amazon S3 儲存貯體或 DynamoDB 資料表。

在設定跨帳戶存取權之前，必須在 AWS 帳戶中準備好下列項目：

- 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。

```
{
    "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。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "dynamodb:*",
            "Resource": "arn:aws:dynamodb:MyRegion: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。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::AccountA: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 AccountA 的 IAM 主控台中，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。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "sts:AssumeRole",
            "Resource": "arn:aws:iam::AccountB: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 來處理標籤

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。

標籤允許您按照屬性將 AWS 資源分類，例如目的、擁有者或環境。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。

如果使用 AWS Identity and Access Management (IAM)，您可以控制您的 AWS 帳戶中的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及虛擬叢集狀態。

當虛擬叢集終止時，會清除標籤且無法再存取。

如果使用 Amazon EMR on EKS API、AWS CLI 或 AWS SDK，則可以使用相關 API 動作中的標籤參數，將標籤套用到新資源。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	支援在建立時加上標簽 (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，請記得，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。因為僅預留給 AWS 使用。

使用 AWS CLI 和 Amazon EMR on EKS API 來處理標籤

使用下列 AWS CLI 命令或 Amazon EMR on EKS API 操作來新增、更新、列出及刪除資源的標籤。

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"]
         }

    ]

    driverRoleRules = [
        {
            "apiGroups": [""],
            "resources": ["persistentvolumeclaims"],
            "verbs": ["list", "create", "delete"]
        },
        {
            "apiGroups": [""],
            "resources": ["services"],
            "verbs": ["get", "list", "describe", "create", "delete", "watch"]
        }
    ]

    clientRoleRules = [
        {
            "apiGroups": [""],
            "resources": ["persistentvolumeclaims"],
            "verbs": ["list", "create", "delete"]
        }
    ]

    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
```

   - driver-role-patch.yaml

```
- apiGroups:
  - ""
  resources:
  - persistentvolumeclaims
  verbs:
  - list
  - create
  - delete
- apiGroups:
  - ""
  resources:
  - services
  verbs:
  - get
  - list
  - describe
  - create
  - delete
  - watch
```

   - client-role-patch.yaml

```
- apiGroups:
  - ""
  resources:
  - persistentvolumeclaims
```

```
  verbs:
  - list
  - create
  - delete
```

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到 AWS 服務，請使用端點。除了標準 AWS 端點之外，某些 AWS 服務還在選取的區域提供 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 (孟買)	ap-south-1	emr-containers.ap-south-1.amazonaws.com	HTTPS
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

AWS 區域 名稱	代碼	端點	通訊協定
亞太區域 (香港)	ap-east-1	emr-containers.ap-east-1.amazonaws.com	HTTPS
加拿大 (中部)	ca-central-1	emr-containers.ca-central-1.amazonaws.com	HTTPS
歐洲 (法蘭克福)	eu-central-1	emr-containers.eu-central-1.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
歐洲 (斯德哥爾摩)	eu-north-1	emr-containers.eu-north-1.amazonaws.com	HTTPS
南美洲 (聖保羅)	sa-east-1	emr-containers.sa-east-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

EKS 上的 Amazon EMR 會針對每個區域針對每個 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。您可以要求提高帳戶的 AWS API 節流配額。

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
CreateManagedEndpoint	25	1
CreateVirtualCluster	25	1
DeleteManagedEndpoint	25	1
DeleteVirtualCluster	25	1
DescribeJobRun	100	20
DescribeManagedEndpoint	100	5
DescribeVirtualCluster	100	5
ListJobRun	100	5
ListManagedEndpoint	25	1
ListVirtualCluster	100	5
StartJobRun	25	1
At the AWS account level, the bucket maximum capacity and refill rate for the sum of all API actions listed in this table	200	20

Amazon EMR 上EKS發布

Amazon EMR 版本是來自大數據生態系統的一組開放原始碼應用程式。每個版本都包含不同的大數據應用程式、元件和功能，讓 Amazon EMR 在執行任務時EKS部署和設定這些功能。

從 Amazon EMR 版本 5.32.0 和 6.2.0 開始，您可以在. EMR EKS 舊EMR版 Amazon 不提供此部署選項。提交作業時，必須指定支援的發行版本。

Amazon EMR on EKS 使用以下形式的發布標籤：`emr-x.x.x-latest`或`emr-x.x.x-yyyymmdd`具體的發布日期。例如 `emr-7.2.0-latest` 或 `emr-7.2.0-20210129`。使用`-latest`尾碼時，請確保您的 Amazon EMR 版本始終包含最新的安全更新。

> **Note**
>
> 有EMR關 Amazon 上EMR運行的 Amazon EKS 和 Amazon 之間的比較EC2，請參閱 AWS 網站EMRFAQs上的亞馬遜。

主題

- Amazon EMR EKS 7.2.0 版本
- Amazon EMR EKS 7.1.0 版本
- Amazon EMR 在 EKS 7.0.0 版本上
- Amazon EMR 在 EKS 6.15.0 版本上發布
- Amazon EMR 在 EKS 6.14.0 版本上發布
- Amazon EMR 在 EKS 6.13.0 版本上發布
- Amazon EMR 在 EKS 6.12.0 版本上發布
- Amazon EMR 在 EKS 6.11.0 版本上發布
- Amazon EMR 在 EKS 6.10.0 版本上發布
- Amazon EMR EKS 6.9.0 版本
- Amazon EMR EKS 6.8.0 版本
- Amazon EMR EKS 6.7.0 版本
- Amazon EMR EKS 6.6.0 版本
- Amazon EMR EKS 6.5.0 版本

- Amazon EMR EKS 6.4.0 版本
- Amazon EMR EKS 6.3.0 版本
- Amazon EMR EKS 6.2.0 版本
- Amazon EMR 在 EKS 5.36.0 版本上發布
- Amazon EMR 在 EKS 5.35.0 版本上發布
- Amazon EMR 在 EKS 5.34.0 版本上發布
- Amazon EMR 在 EKS 5.33.0 版本上發布
- Amazon EMR 在 EKS 5.32.0 版本上發布

Amazon EMR EKS 7.2.0 版本

本頁面說明 Amazon EKS 部署時專用EMR的新功能和更新功能。EMR有關 Amazon 在 Amazon 上EMR運行以EC2及一般 Amazon EMR 7.2.0 版本的詳細信息，請參閱 Amazon 發EMR布指南中的Amazon EMR 7.2.0。

EKS7.2 版本EMR上的 Amazon

以下 Amazon EMR 7.2.0 版本可用於 Amazon EMR 上EKS。選取特定的 emr-7.2.0 XXXX 版本以檢視更多詳細資料，例如相關的容器映像檔標籤。

Flink releases

EKS當您運行 Flink 應用程序時，Amazon EMR EMR 7.2.0 版本可用於 Amazon。

- emr-7.2.0-打火石最新
- 火石火石

Spark releases

EKS當您運行 Spark 應用程序時，Amazon EMR EMR 7.2.0 版本可用於 Amazon。

- 電腦 -7.2.0-最新
- 電子郵件 -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

7.2.0 EMR 上 Amazon 的發行EKS公告

- 支援的應用程式：AWS SDK for Java 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

  與和一起使StartJobRun用 CreateManagedEndpointAPIs：

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。

專門用於 CreateManagedEndpointAPIs：

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。這些通常對應於應用程序的配置XML文件，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 的 7.2.0 版本包含以下功能。EKS

- 應用升級 — Amazon EMR EKS 7.2.0 應用程序升級包括星火 3.5.1，Flink 1.18.1 和 Flink 運營商 1.8.0。
- Flink 更新的自動配置器 — 7.2.0 版本使用開放原始碼組態**job.autoscaler.restart.time-tracking.enabled**來啟用重新調整時間估算，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 自動調度資源。
- Amazon 上的阿帕奇胡迪整合 Apache Flink EKS — 此版本增加了 Apache 胡迪和 Apache Flink 之間的整合，因此您可以使用 Flink Kubernetes 運EMR算子來執行胡迪任務。Hudi 允許您使用記錄級操作，可用於簡化數據管理和數據管道開發。
- Amazon S3 快遞單區與 Amazon EMR 整合 EKS — 使用 7.2.0 及更高版本，您可以使用 Amazon EMR 將數據上傳到 S3 快遞一個區域。EKSS3 Express One Zone 是一種高效能的單區域 Amazon S3 儲存類別，可為大多數對延遲敏感的應用程式提供一致、10 毫秒的資料存取。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 運營商中的默認配置 Support — Amazon 上的 Spark 運營商EKS現在支持與 7.2.0 及更高版本上 Amazon EMR 上的開始任務運行模型相同EKS的默認配置。這表示 Amazon S3 等功能EMRFS不再需要在 yaml 檔案中進行手動設定。

電腦 -7.2.0-最新

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EMR於 Amazon 所支援的所有區域EKS。如需詳細資訊，請參閱EKS服務端點EMR上的 Amazon。

容器映像標籤：emr-7.2.0:latest

電子郵件 -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（星火）的初始版本。

區域：emr-7.2.0-20240610適用EMR於 Amazon 所支援的所有區域EKS。如需詳細資訊，請參閱EKS服務端點EMR上的 Amazon。

容器映像標籤：emr-7.2.0:20240610

emr-7.2.0-打火石最新

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EMR於 Amazon 所支援的所有區域EKS。如需詳細資訊，請參閱EKS服務端點EMR上的 Amazon。

容器映像標籤：emr-7.2.0-flink:latest

火石火石

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EMR於 Amazon 所支援的所有區域EKS。如需詳細資訊，請參閱EKS服務端點EMR上的 Amazon。

容器映像標籤：emr-7.2.0-flink:20240610

Amazon EMR EKS 7.1.0 版本

本頁面說明 Amazon EKS 部署時專用EMR的新功能和更新功能。EMR有關 Amazon 在 Amazon 上EMR運行以EC2及一般 Amazon EMR 7.1.0 版本的詳細信息，請參閱 Amazon 發EMR布指南中的 Amazon EMR 7.1.0。

EKS7.1 版本EMR上的 Amazon

以下 Amazon EMR 7.1.0 版本可在 Amazon EMR 上EKS使用。選取特定的 emr-7.1.0 XXXX 版本，以檢視更多詳細資料，例如相關的容器映像檔標籤。

Flink releases

EKS當您運行 Flink 應用程序時，Amazon EMR EMR 7.1.0 版本可用於 Amazon。

- emr-7.1.0-打火石最新
- 火石火石

Spark releases

EKS當您運行 Spark 應用程序時，Amazon EMR EMR 7.1.0 版本可用於 Amazon。

- 電子報 -7.1.0-最新
- 電腦 -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 在 EKS 7.1.0 上的發行公告

- 支援的應用程式：AWS SDK for Java 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

  與和一起使StartJobRun用 CreateManagedEndpointAPIs：

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。

  專門用於 CreateManagedEndpointAPIs：

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。這些通常對應於應用程序的配置XML文件，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 的 7.1.0 版本包含以下功能。EKS

- 對 Amazon 的 Apache Livy 支持 EKS — EMR 在 7.1.0 及更高EKS版本EMR上使用 Amazon，您可以在 Amazon EKS 集群上使用 Apache Livy 創建一個 Apache Livy REST 界面，以提交 Spark 任務或 Spark 代碼片段。這樣做可讓您以同步和非同步方式擷取結果，同時仍可利用 Amazon EMR EMR 優化的 Spark 執行階段、SSL啟用的 Livy 端點和程式設定體驗等EKS權益。

電子報 -7.1.0-最新

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EMR於 Amazon 所支援的所有區域EKS。如需詳細資訊，請參閱EKS服務端點EMR上的 Amazon。

容器映像標籤：emr-7.1.0:latest

電腦 -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（星火）的初始版本。

區域：emr-7.1.0-20240321適用EMR於 Amazon 所支援的所有區域EKS。如需詳細資訊，請參閱EKS服務端點EMR上的 Amazon。

容器映像標籤：emr-7.1.0:20240321

emr-7.1.0-打火石最新

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EMR於 Amazon 所支援的所有區域EKS。如需詳細資訊，請參閱EKS服務端點EMR上的 Amazon。

容器映像標籤：emr-7.1.0-flink:latest

火石火石

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EMR於 Amazon 所支援的所有區域EKS。如需詳細資訊，請參閱EKS服務端點EMR上的 Amazon。

容器映像標籤：`emr-7.1.0-flink:20240321`

Amazon EMR 在 EKS 7.0.0 版本上

本頁面說明 Amazon EKS 部署時專用EMR的新功能和更新功能。EMR有關在 Amazon 上EMR運行的 Amazon 以EC2及一般 Amazon EMR 7.0.0 版本的詳細信息，請參閱 Amazon 版EMR本指南中的 Amazon EMR 7.0.0。

EKS7.0 版本EMR上的 Amazon

以下 Amazon EMR 7.0.0 版本可在 Amazon EMR 上EKS使用。選取特定的 emr-7.0.0 XXXX 版本，以檢視更多詳細資料，例如相關的容器映像檔標籤。

Flink releases

EKS當您運行 Flink 應用程序時，Amazon EMR EMR 7.0.0 版本可用於 Amazon。

- emr-7.0.0-flink-latest
- 火石火石
- emr-7.0.0-flink-20231211

Spark releases

EKS當您運行 Spark 應用程序時，Amazon EMR EMR 7.0.0 版本可用於 Amazon。

- 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

7.0.0 EMR 上 Amazon 的發行EKS公告

- 支援的應用程式：AWS SDK for Java 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

  與和一起使StartJobRun用 CreateManagedEndpointAPIs：

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。

  專門用於 CreateManagedEndpointAPIs：

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。這些通常對應於應用程序的配置XML文件，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 在 7.0 版本中包含以下功能EKS。

- 應用程序升級 — Amazon EMR EKS 7.0.0 應用程序升級包括星火 3.5，Flink 1.18 和 F link 運營商 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針對任務量身打造的最佳參數。

變更

Amazon EMR 在 7.0 版本中包含以下變更EKS。

- Amazon 2023 — 使用 EKS 7.0.0 及更高版本的 AmazonEMR，所有容器映像都基於 Amazon 2023。
- 星火使用 Java 17 作為默認運行時-Amazon EMR EKS 7.0.0 星火使用 Java 17 作為默認運行時。如有需要，您可以透過 EKS7.0 版本EMR上的 Amazon 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EMR於 Amazon 所支援的所有區域EKS。如需詳細資訊，請參閱EKS服務端點EMR上的 Amazon。

容器映像標籤：emr-7.0.0:latest

電腦 -7.0.0-2024321

發行公告：7.0.0-2024321已於 2024 年 3 月 11 日發行。與舊版相比，此版本已透過最近更新的 Amazon Linux 套件和重大修正程式進行重新整理。

區域：emr-7.0.0-2024321適用EMR於 Amazon 所支援的所有區域EKS。如需詳細資訊，請參閱EKS服務端點EMR上的 Amazon。

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（星火）的初始版本。

區域：emr-7.0.0-20231211適用EMR於 Amazon 所支援的所有區域EKS。如需詳細資訊，請參閱EKS服務端點EMR上的 Amazon。

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EMR於 Amazon 所支援的所有區域EKS。如需詳細資訊，請參閱EKS服務端點EMR上的 Amazon。

容器映像標籤：emr-7.0.0-flink:latest

火石火石

發行公告：7.0.0-flink-2024321已於 2024 年 3 月 11 日發行。與舊版相比，此版本已透過最近更新的 Amazon Linux 套件和重大修正程式進行重新整理。

區域：emr-7.0.0-flink-2024321適用EMR於 Amazon 所支援的所有區域EKS。如需詳細資訊，請參閱EKS服務端點EMR上的 Amazon。

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EMR於 Amazon 所支援的所有區域EKS。如需詳細資訊，請參閱EKS服務端點EMR上的 Amazon。

容器映像標籤：emr-7.0.0-flink:20231211

Amazon EMR 在 EKS 6.15.0 版本上發布

本頁面說明 Amazon EKS 部署時專用EMR的新功能和更新功能。EMR有關 Amazon 在 Amazon 上EMR運行以EC2及有關 Amazon EMR 6.15.0 版本的詳細信息，請參閱 Amazon 發布指南中的 Amazon EMR 6.15.0。EMR

Amazon EMR EKS 6.15 版本

以下 Amazon EMR 6.15.0 版本可用於 Amazon EMR 上。EKS選取特定的 emr-6.15.0 XXXX 版本，以檢視更多詳細資料，例如相關的容器映像標記。

Flink releases

當您運行 Flink 應用程序EKS時，Amazon EMR EMR 6.15.0 版本可用於 Amazon。

- emr-6.15.0-flink-latest
- 火石火石
- emr-6.15.0-flink-20231109

Spark releases

EKS當您運行 Spark 應用程序時，Amazon EMR EMR 6.15.0 版本可用於 Amazon。

- 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 在 EKS 6.15.0 上的發行公告

- 支援的應用程式：AWS SDK for Java 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

  與和一起使StartJobRun用 CreateManagedEndpointAPIs：

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。

  專門用於 CreateManagedEndpointAPIs：

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。這些通常對應於應用程序的配置XML文件，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 6.15 版本包含以下功能。EKS

- Amazon EKS 與 Apache Flink-EMR 在 EKS 6.15.0 EMR 上使用 Amazon，您可以在同一 Amazon 集群上運行基於 Apache 燧石的應用程序以及其他類型的應用程序。EKS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 應用程式中充分利用 Spot 執行個體，並透過 Amazon 進行精細的復原和任務本機復原，實現更快的重新啟動時間。EBS可存取性和監控功能包括使用存放在 Amazon S3 的 jar 啟動 Flink 應用程式、存取 AWS Glue 資料型錄、監控與 Amazon S3 和 Amazon 的整合 CloudWatch，以及容器日誌輪替。

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EMR於 Amazon 所支援的所有區域EKS。如需詳細資訊，請參閱EKS服務端點EMR上的 Amazon。

容器映像標籤：emr-6.15.0:latest

電子商品 -6.15.0-20240105

發行公告：6.15.0-20240105已於 2024 年 1 月 17 日發行。與舊版相比，此版本已透過最近更新的 Amazon Linux 套件和重大修正程式進行重新整理。

區域：emr-6.15.0-20240105適用EMR於 Amazon 所支援的所有區域EKS。如需詳細資訊，請參閱EKS服務端點EMR上的 Amazon。

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EMR於 Amazon 所支援的所有區域EKS。如需詳細資訊，請參閱EKS服務端點EMR上的 Amazon。

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EMR於 Amazon 所支援的所有區域EKS。如需詳細資訊，請參閱EKS服務端點EMR上的 Amazon。

容器映像標籤：emr-6.15.0-flink:latest

火石火石

發行公告：6.15.0-flink-20240105已於 2024 年 1 月 17 日發行。與舊版相比，此版本已透過最近更新的 Amazon Linux 套件和重大修正程式進行重新整理。

區域：emr-6.15.0-flink-20240105適用EMR於 Amazon 所支援的所有區域EKS。如需詳細資訊，請參閱EKS服務端點EMR上的 Amazon。

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EMR於 Amazon 所支援的所有區域EKS。如需詳細資訊，請參閱EKS服務端點EMR上的 Amazon。

容器映像標籤：emr-6.15.0-flink:20231109

Amazon EMR 在 EKS 6.14.0 版本上發布

本頁面說明 Amazon EKS 部署時專用EMR的新功能和更新功能。EMR有關 Amazon 在 Amazon 上EMR運行以EC2及有關 Amazon EMR 6.14.0 版本的詳細信息，請參閱 Amazon 發布指南中的 Amazon EMR 6.14.0。EMR

Amazon EMR EKS 6.14 版本

以下 Amazon EMR 6.14.0 版本可用於 Amazon EMR 上。EKS選取特定的 emr-6.14.0 XXXX 版本，以檢視更多詳細資料，例如相關的容器映像標記。

- 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

6.14.0 EMR 上 Amazon 的發行EKS公告

- 支持的應用程序- AWS SDK for Java 1.12.543, 阿帕奇星火 3.4.1-上午 1, 阿帕奇胡迪 0.13.1-安贊 -2, 阿帕奇冰山 1.3.0-安贊 0, 三角洲 2.4.0, 阿帕奇星火 23.06.0-安贊 -2, 木星企業網關 2.7.0 RAPIDS
- 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

  與和一起使StartJobRun用 CreateManagedEndpointAPIs：

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-log4j	變更 log4j2.properties Spark 檔案中的值。
emr-job-submitter	作業提交者 Pod 的組態。

專門用於 CreateManagedEndpointAPIs：

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。這些通常對應於應用程序的配置XML文件，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 6.14 版本包含以下功能。EKS

- 阿帕奇利維支持-Amazon EKS 現EMR在支持阿帕奇利維與。spark-submit

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EMR於 Amazon 所支援的所有區域EKS。如需詳細資訊，請參閱EKS服務端點EMR上的 Amazon。

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 的初始版本。

區域：`emr-6.14.0-20231005`適用EMR於 Amazon 所支援的所有區域EKS。如需詳細資訊，請參閱EKS服務端點EMR上的 Amazon。

容器映像標籤：`emr-6.14.0:20231005`

Amazon EMR 在 EKS 6.13.0 版本上發布

本頁面說明 Amazon EKS 部署時專用EMR的新功能和更新功能。EMR有關 Amazon 在 Amazon 上EMR運行以EC2及有關 Amazon EMR 6.13.0 版本的詳細信息，請參閱 Amazon 發布指南中的 Amazon EMR 6.13.0。EMR

Amazon EMR EKS 6.13 版本

以下 Amazon EMR 6.13.0 版本可用於 Amazon EMR 上。EKS選取特定的 emr-6.13.0 XXXX 版本，以檢視更多詳細資料，例如相關的容器映像標記。

- 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 在 EKS 6.13.0 上的發行公告

- 支持的應用程序- AWS SDK for Java 1.12.513, 阿帕奇星火 3.4.1-安贊 0, 阿帕奇胡迪 0.13.1-安贊 0, 阿帕奇冰山 1.3.0-安贊 0, 三角洲 2.4.0, 阿帕奇星火 23.06.0 和 1, 木星企業網關 2.6.0. RAPIDS

- 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

  與和一起使StartJobRun用 CreateManagedEndpointAPIs：

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。

  專門用於 CreateManagedEndpointAPIs：

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。這些通常對應於應用程序的配置XML文件，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 6.13 版本包含以下功能。EKS

- Amazon Linux 2023-隨著 Amazon EMR EKS 6.13 及更高版本，你可以推出星火與 AL2 023 作為操作系統與 Java 17 運行時一起。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 AL2 023 和 Java 17。
- Amazon EMR 上EKS與阿帕奇 Flink（公開預覽）-Amazon EMR 在 6.13 版和更高EKS版本支持阿帕奇 Flink，可在公開預覽。此次啟動後，您可以在同一個 Amazon EKS 叢集上執行 Apache Flink 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。如果您已經在 Amazon 上執行大數據架構EKS，您現在可以讓 Amazon EMR 自動化佈建和管理。

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EMR於 Amazon 所支援的所有區域EKS。如需詳細資訊，請參閱EKS服務端點EMR上的 Amazon。

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 的初始版本。

區域：emr-6.13.0-20230814適用EMR於 Amazon 所支援的所有區域EKS。如需詳細資訊，請參閱EKS服務端點EMR上的 Amazon。

容器映像標籤：emr-6.13.0:20230814

Amazon EMR 在 EKS 6.12.0 版本上發布

本頁面說明 Amazon EKS 部署時專用EMR的新功能和更新功能。EMR有關 Amazon 在 Amazon 上EMR運行以EC2及一般 Amazon EMR 6.12.0 版本的詳細信息，請參閱 Amazon 發布指南中的 Amazon EMR 6.12.0。EMR

Amazon EMR EKS 6.12 版本

以下 Amazon EMR 6.12.0 版本可用於 Amazon EMR 上。EKS選取特定的 emr-6.12.0 XXXX 版本，以檢視更多詳細資料，例如相關的容器映像標記。

- emr-6.12.0-latest
- 電腦 -6.12.0-20240321
- emr-6.12.0-20230701
- 埃姆尔 -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
- 埃姆尔 -6.12.0-17-最新 spark-rapids-java
- 電腦 -6.12.0-17-20230701 spark-rapids-java
- notebook-spark/emr-6.12.0-latest
- notebook-spark/emr-6.12.0-20230701
- 筆記型電腦火花/電腦 -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
- 筆記本-蟒蛇/電腦 -6.12.0-spark-rapids-latest
- notebook-python/emr-6.12.0-spark-rapids-20230701

版本備註

Amazon EMR 在 EKS 6.12.0 版發行公告

- 支持的應用程序- AWS SDK for Java 1.12.490, 阿帕奇星火 3.4.0-安贊 0, 阿帕奇胡迪 0.13.1-安贊 0, 阿帕奇冰山 1.3.0-安贊 0, 三角洲 2.4.0, 阿帕奇星火 23.06.0-上午 0, 木星企業網關 2.6.0 RAPIDS
- 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

  與和一起使StartJobRun用 CreateManagedEndpointAPIs：

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。

  專門用於 CreateManagedEndpointAPIs：

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。這些通常對應於應用程序的配置XML文件，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 6.12 版本包含以下功能。EKS

- Java 17-隨著 Amazon EMR EKS 6.12 及更高版本，您可以使用 Java 17 運行時啟動星火。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EMR於 Amazon 所支援的所有區域EKS。如需詳細資訊，請參閱EKS服務端點EMR上的 Amazon。

容器映像標籤：emr-6.12.0:latest

電腦 -6.12.0-20240321

發行公告：6.12.0-20240321已於 2024 年 3 月 11 日發行。與舊版相比，此版本已透過最近更新的 Amazon Linux 套件和重大修正程式進行重新整理。

區域：emr-6.12.0-20240321適用EMR於 Amazon 所支援的所有區域EKS。如需詳細資訊，請參閱EKS服務端點EMR上的 Amazon。

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 的初始版本。

區域：emr-6.12.0-20230701適用EMR於 Amazon 所支援的所有區域EKS。如需詳細資訊，請參閱EKS服務端點EMR上的 Amazon。

容器映像標籤：emr-6.12.0:20230701

Amazon EMR 在 EKS 6.11.0 版本上發布

本頁面說明 Amazon EKS 部署時專用EMR的新功能和更新功能。EMR有關 Amazon 在 Amazon 上EMR運行以EC2及一般 Amazon EMR 6.11.0 版本的詳細信息，請參閱 Amazon 發布指南中的 Amazon EMR 6.11.0。EMR

Amazon EMR EKS 6.11 版本

以下 Amazon EMR 6.11.0 版本可用於 Amazon EMR 上。EKS選取特定的 emr-6.11.0 XXXX 版本，以檢視更多詳細資料，例如相關的容器映像標記。

- emr-6.11.0-latest
- 電腦 -6.11.0-20230905
- emr-6.11.0-20230509

- 電腦 -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 在 EKS 6.11.0 上的發行公告

- 支持的應用程序- AWS SDK for Java 1.12.446, 阿帕奇星火 3.3.2-安贊 0, 阿帕奇胡迪 0.13.0-安贊 0, 阿帕奇冰山 1.2.0-安贊 0, 三角洲 2.2.0, 阿帕奇星火 23.02.0-安贊 0, 木星企業網關 2.6.0 RAPIDS

- 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

  與和一起使StartJobRun用 CreateManagedEndpointAPIs：

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。

  專門用於 CreateManagedEndpointAPIs：

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。這些通常對應於應用程序的配置XML文件，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 6.11 版本包含以下功能。EKS

- Amazon ECR 公共EMR圖庫中的EKS基本映像 — 如果您使用自定義映像功能，我們的基本映像提供了必要的罐子，配置和庫與 Amazon EMR 進行交互EKS。現在，您可以在 Amazon ECR 公共畫廊中找到基本圖像。
- 火花容器日誌輪替 — EKS 6.11 EMR 上的 Amazon 支持 Spark 容器日誌輪替。您可以在的MonitoringConfiguration作業containerLogRotationConfiguration中啟用功能StartJobRunAPI。您可以設定rotationSize和，maxFilestoKeep以指定您希望 Amazon EMR 在 Spark 驅動程式和執行程式網繭中EKS保留的日誌檔的數量和大小。如需詳細資訊，請參閱使用 Spark 容器日誌輪換。
- 火山支持 Spark 運營商和火花提交 — Amazon EKS 6.11 EMR 上支持使用火山作為 Kubernetes 自定義調度程序在 Spark 操作員中運行 Spark 任務並火花提交。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EMR於 Amazon 所支援的所有區域EKS。如需詳細資訊，請參閱EKS服務端點EMR上的 Amazon。

容器映像標籤：emr-6.11.0:latest

電腦 -6.11.0-20230905

發行公告：6.11.0-20230905已於 2023 年 9 月 29 日發行。與舊版相比，此版本已透過最近更新的 Amazon Linux 套件和重大修正程式進行重新整理。

區域：emr-6.11.0-20230509適用EMR於 Amazon 所支援的所有區域EKS。如需詳細資訊，請參閱EKS服務端點EMR上的 Amazon。

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 的初始版本。

區域：emr-6.11.0-20230509適用EMR於 Amazon 所支援的所有區域EKS。如需詳細資訊，請參閱EKS服務端點EMR上的 Amazon。

容器映像標籤：emr-6.11.0:20230509

Amazon EMR 在 EKS 6.10.0 版本上發布

以下 Amazon EMR 6.10.0 版本可用於 Amazon EMR 上。EKS選取特定的 emr-6.10.0 XXXX 版本，以檢視更多詳細資料，例如相關的容器映像標記。

- emr-6.10.0-latest
- 電腦 -6.10.0-20230905
- emr-6.10.0-20230624
- emr-6.10.0-20230421
- emr-6.10.0-20230403
- emr-6.10.0-20230220
- 電腦 -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 6.10.0 版的發行公告

- 支持的應用程序- AWS SDK for Java 1.12.397，星火 3.3.1-安培 -0，胡迪 0.12.2-安鋅 -0，冰山 1.1.0-安贊 -0，三角洲 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：

  與和一起使StartJobRun用 CreateManagedEndpointAPIs：

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。

  專門用於 CreateManagedEndpointAPIs：

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。這些通常對應於應用程序的配置XML文件，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

- 星火運算子-使用 EKS 6.10.0 及更高EMR版本的 AmazonEMR，您可以使用 Apache Spark 的 Kubernetes 運算子或 S park 運算子，在您自己的 Amazon 叢集上部署和管理 Spark 應用程式。EKS如需詳細資訊，請參閱使用 Spark Operator 執行 Spark 作業。
- Java 11-隨著 EKS 6.10 及更高版本的 AmazonEMR，您可以使用 Java 11 運行時啟動星火。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。
- 對於阿帕奇星火 Amazon Redshift 集成，Amazon EKS 6.10.0 EMR 上刪除了依賴關係minimal-json.jar，並自動將所需的spark-redshift相關罐子添加到 Spark 的執行程序類路徑：spark-redshift.jar，和。spark-avro.jar RedshiftJDBC.jar

變更

- EMRFSS3 優化的提交者現在默認情況下為實木複合地板和基於文本的格式（包ORC括和）啟用。CSV JSON

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EMR於 Amazon 所支援的所有區域EKS。如需詳細資訊，請參閱EKS服務端點EMR上的 Amazon。

容器映像標籤：emr-6.10.0:latest

電腦 -6.10.0-20230905

發行公告：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EMR於 Amazon 所支援的所有區域EKS。如需詳細資訊，請參閱EKS服務端點EMR上的 Amazon。

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EMR於 Amazon 所支援的所有區域EKS。如需詳細資訊，請參閱EKS服務端點EMR上的 Amazon。

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EMR於 Amazon 所支援的所有區域EKS。如需詳細資訊，請參閱EKS服務端點EMR上的 Amazon。

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EMR於 Amazon 所支援的所有區域EKS。如需詳細資訊，請參閱EKS服務端點EMR上的 Amazon。

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 的初始版本。

區域：`emr-6.10.0-20230220`適用EMR於 Amazon 所支援的所有區域EKS。如需詳細資訊，請參閱EKS服務端點EMR上的 Amazon。

容器映像標籤：`emr-6.10.0:20230220`

Amazon EMR EKS 6.9.0 版本

以下 Amazon EMR 6.9.0 版本可用於 Amazon EMR 上EKS。選取特定的 emr-6.9.0 XXXX 版本以檢視更多詳細資料，例如相關的容器映像檔標籤。

- emr-6.9.0-latest
- 電腦 -6.9.0-20230905
- emr-6.9.0-20230624
- emr-6.9.0-20221108
- 電腦 -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 6.9.0 的發行公告

- 支持的應用程序- AWS SDK for Java 1.12.331，火花 3.3.0-安區 -1，胡迪 0.12.1-安鋅 -0，冰山 0.14.1 安贊 -0，三角洲 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：

  與和一起使StartJobRun用 CreateManagedEndpointAPIs：

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。

  專門用於 CreateManagedEndpointAPIs：

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。這些通常對應於應用程序的配置XML文件，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

- 適用於 Apache Spark-Amazon EMR 的 Nvidia 加速RAPIDS器，使用EC2圖形處理單元 (GPU) 執行個體類型加速 Spark。EKS若要搭配RAPIDS加速器使用星火影像，請將釋出標籤指定為 emr-6.9.0-。spark-rapids-latest如需進一步了解，請造訪文件頁面。
- 火花紅移連接器-Amazon Redshift 集成阿帕奇星火包含在亞馬遜 6.9.0 及更高EMR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-互動式端點現在支援修改與 EMR Studio Jupyter 筆記本中工 PySpark 作階段相關聯的 Spark 參數。請造訪修改 PySpark 工作階段參數以深入瞭解。

已解決的問題

- 當您在 Amazon 6.6.0、6.7.0 和 6.8.0 EMR 版上搭配 Spark 使用 DynamoDB 連接器時，即使輸入分割參照非空白資料，從表格中讀取的所有讀取都會傳回空白結果。Amazon EMR 版本 6.9.0 修復了此問題。
- Amazon EKS 6.8.0 EMR 上錯誤地填充使用 Apache 星火生成的實木複合地板文件元數據構建哈希。此問題可能會導致從 Amazon EKS 6.8.0 EMR 上產生的 Parquet 檔案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EMR於 Amazon 所支援的所有區域EKS。如需詳細資訊，請參閱EKS服務端點EMR上的 Amazon。

容器映像標籤：`emr-6.9.0:latest`

電腦 -6.9.0-20230905

版本備註：emr-6.9.0-20230905。與舊版相比，此版本已透過最近更新的 Amazon Linux 套件和重大修正程式進行重新整理。

區域：emr-6.9.0-20230905適用EMR於 Amazon 所支援的所有區域EKS。如需詳細資訊，請參閱EKS服務端點EMR上的 Amazon。

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EMR於 Amazon 所支援的所有區域EKS。如需詳細資訊，請參閱EKS服務端點EMR上的 Amazon。

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 的初始版本。

區域：emr-6.9.0-20221108適用EMR於 Amazon 所支援的所有區域EKS。如需詳細資訊，請參閱EKS服務端點EMR上的 Amazon。

容器映像標籤：emr-6.9.0:20221108

Amazon EMR EKS 6.8.0 版本

以下 Amazon EMR 6.8.0 版本可用於 Amazon EMR 上EKS。選取特定的 emr-6.8.0 XXXX 版本，以檢視更多詳細資料，例如相關的容器映像檔標記。

- emr-6.8.0-latest
- 電腦 -6.8.0-20230905
- emr-6.8.0-20230624
- emr-6.8.0-20221219
- emr-6.8.0-20220802

Amazon EMR 6.8.0 的發行公告

- 支持的應用程序- AWS SDK for Java 1.12.170，星火 3.3.0-安培 -0，胡迪 0.11.1-安鋅 -0，冰山 0.14.0-安贊 -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。這些通常對應於應用程序的配置XML文件，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

- EKS6.8 EMR 上的 S@@ park3.3.0-Amazon 包括 Spark 3.3.0，它支持為 Spark 驅動程序執行器網繭使用單獨的節點選擇器標籤。這些新標籤可讓您在中分別定義驅動程式和執行程式網繭的節點類型 StartJobRun API，而無需使用網繭範本。
  - 驅動程序節點選擇器屬性：閃光. [labelKey]
  - 執行程序節點選擇器屬性：閃耀 .kubernetes.Executor。 [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。 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。 此屬性只有在 Config 閃耀 .stage 時才起作用。 extraDetailsOnFetchFailures。啟用被設置為真。	emr-6.8

如需詳細資訊，請參閱 Apache Spark 組態文件。

已知問題

- Amazon EKS 6.8.0 EMR 上錯誤地填充使用 Apache 星火生成的實木複合地板文件元數據構建哈希。此問題可能會導致從 Amazon EKS 6.8.0 EMR 上產生的 Parquet 檔案剖析中繼資料版本字串的工具失敗。從 Parquet 中繼資料剖析版本字串並依賴組建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

- 核 pySpark 心的中斷核心功能-正在進行中的互動式工作負載 (藉由在筆記本中執行儲存格而觸發) 可使用此Interrupt Kernel功能來停止。引入了修復程序，以便此功能適用於 pySpark 內核。這也可以在開放原始碼中取得，位於處理 PySpark Kubernetes 核心 #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EMR於 Amazon 所支援的所有區域EKS。如需詳細資訊，請參閱EKS服務端點EMR上的 Amazon。

容器映像標籤：emr-6.8.0:latest

電腦 -6.8.0-20230905

發行公告：emr-6.8.0-20230905已於 2023 年 9 月 29 日發行。與舊版相比，此版本已透過最近更新的 Amazon Linux 套件和重大修正程式進行重新整理。

區域：emr-6.8.0-20230905適用EMR於 Amazon 所支援的所有區域EKS。如需詳細資訊，請參閱EKS服務端點EMR上的 Amazon。

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。與舊版相比，此版本已透過最近更新的 Amazon Linux 套件和重大修正程式進行重新整理。

區域：emr-6.8.0-20230624適用EMR於 Amazon 所支援的所有區域EKS。如需詳細資訊，請參閱EKS服務端點EMR上的 Amazon。

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。與舊版相比，此版本已透過最近更新的 Amazon Linux 套件和重大修正程式進行重新整理。

區域：emr-6.8.0-20221219適用EMR於 Amazon 所支援的所有區域EKS。如需詳細資訊，請參閱EKS服務端點EMR上的 Amazon。

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 的初始版本。

區域：emr-6.8.0-20220802適用EMR於 Amazon 所支援的所有區域EKS。如需詳細資訊，請參閱EKS服務端點EMR上的 Amazon。

容器映像標籤：emr-6.8.0:20220802

Amazon EMR EKS 6.7.0 版本

以下 Amazon EMR 6.7.0 版本可在 Amazon EMR 上EKS使用。選取特定的 emr-6.7.0 XXXX 版本，以檢視更多詳細資料，例如相關的容器映像檔標籤。

- emr-6.7.0-latest
- 電腦 -6.7.0-20240321
- emr-6.7.0-20230624
- emr-6.7.0-20221219
- emr-6.7.0-20220630

Amazon EMR 6.7.0 的發行公告

- 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，內核管理現在是異步的，這意味著當內核啟動正在進行時，JEG不會阻止交易。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。這些通常對應於應用程序的配置XML文件，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

- EKS6.7 版EMR的 Amazon 修正了在使用 Apache Spark 的網繭範本功能與互動式端點時，6.6 中的問題。這個問題出現EMR在 Amazon EKS 版本 6.4，6.5 和 6.6 上。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。
- 在先前的 Amazon EKS 發EMR行版本中，Jupyter 企業閘道會在核心啟動進行時封鎖交易，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EMR於 Amazon 所支援的所有區域EKS。如需詳細資訊，請參閱EKS服務端點EMR上的 Amazon。

容器映像標籤：emr-6.7.0:latest

電腦 -6.7.0-20240321

發行公告：emr-6.7.0-20240321已於 2024 年 3 月 11 日發行。與舊版相比，此版本已透過最近更新的 Amazon Linux 套件和重大修正程式進行重新整理。

區域：emr-6.7.0-20240321適用EMR於 Amazon 所支援的所有區域EKS。如需詳細資訊，請參閱EKS服務端點EMR上的 Amazon。

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。與舊版相比，此版本已透過最近更新的 Amazon Linux 套件和重要修正程式進行重新整理。

區域：emr-6.7.0-20230624適用EMR於 Amazon 所支援的所有區域EKS。如需詳細資訊，請參閱EKS服務端點EMR上的 Amazon。

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。與舊版相比，此版本已透過最近更新的 Amazon Linux 套件和重要修正程式進行重新整理。

區域：emr-6.7.0-20221219適用EMR於 Amazon 所支援的所有區域EKS。如需詳細資訊，請參閱EKS服務端點EMR上的 Amazon。

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 的初始版本。

區域：emr-6.7.0-20220630適用EMR於 Amazon 所支援的所有區域EKS。如需詳細資訊，請參閱EKS服務端點EMR上的 Amazon。

容器映像標籤：emr-6.7.0:20220630

Amazon EMR EKS 6.6.0 版本

以下 Amazon EMR 6.6.0 版本可用於 Amazon EMR 上EKS。選取特定的 emr-6.6.0 XXXX 版本以檢視更多詳細資料，例如相關的容器映像檔標籤。

- emr-6.6.0-latest
- 電腦 -6.0-20240321
- emr-6.6.0-20230624
- emr-6.6.0-20221219
- emr-6.6.0-20220411

Amazon EMR 6.6.0 的發行公告

- 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。這些通常對應於應用程序的配置XML文件，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

- 在 6.4、6.5 和 6.6 版上，具有互動式端點的 Spark Pod 範EKS本功能無法在 Amazon EMR 上運作。

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EMR於 Amazon 所支援的所有區域EKS。如需詳細資訊，請參閱EKS服務端點EMR上的 Amazon。

容器映像標籤：`emr-6.6.0:latest`

電腦 -6.0-20240321

發行公告：`emr-6.6.0-20240321`已於 2024 年 3 月 11 日發行。與舊版相比，此版本已透過最近更新的 Amazon Linux 套件和重大修正程式進行重新整理。

區域：`emr-6.6.0-20240321`適用EMR於 Amazon 所支援的所有區域EKS。如需詳細資訊，請參閱EKS服務端點EMR上的 Amazon。

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`。與舊版相比，此版本已透過最近更新的 Amazon Linux 套件和重要修正程式進行重新整理。

區域：`emr-6.6.0-20230624`適用EMR於 Amazon 所支援的所有區域EKS。如需詳細資訊，請參閱EKS服務端點EMR上的 Amazon。

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。與舊版相比，此版本已透過最近更新的 Amazon Linux 套件和重要修正程式進行重新整理。

區域：emr-6.6.0-20221219適用EMR於 Amazon 所支援的所有區域EKS。如需詳細資訊，請參閱EKS服務端點EMR上的 Amazon。

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 的初始版本。

區域：emr-6.6.0-20220411適用EMR於 Amazon 所支援的所有區域EKS。如需詳細資訊，請參閱EKS服務端點EMR上的 Amazon。

容器映像標籤：emr-6.6.0:20220411

Amazon EMR EKS 6.5.0 版本

以下 Amazon EMR 6.5.0 版本可用於 Amazon EMR 上EKS。選取特定的 emr-6.5.0 XXXX 版本以檢視更多詳細資料，例如相關的容器映像檔標記。

- emr-6.5.0-latest
- 電腦 -6.5.0-20240321
- emr-6.5.0-20221219
- emr-6.5.0-20220802
- emr-6.5.0-20211119

Amazon EMR 6.5.0 的發行公告

- 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。這些通常對應於應用程序的配置XML文件，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

- 在 6.4 EKS 版和 6.5 版上，具有互動式端點的 Spark Pod 範本功能無法EMR在 Amazon 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EMR於 Amazon 所支援的所有區域EKS。如需詳細資訊，請參閱EKS服務端點EMR上的 Amazon。

容器映像標籤：emr-6.5.0:latest

電腦 -6.5.0-20240321

發行公告：emr-6.5.0-20240321已於 2024 年 3 月 11 日發行。與舊版相比，此版本已透過最近更新的 Amazon Linux 套件和重要修正程式進行重新整理。

區域：emr-6.5.0-20240321適用EMR於 Amazon 所支援的所有區域EKS。如需詳細資訊，請參閱EKS服務端點EMR上的 Amazon。

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。與舊版相比，此版本已透過最近更新的 Amazon Linux 套件和重要修正程式進行重新整理。

區域：emr-6.5.0-20221219適用EMR於 Amazon 所支援的所有區域EKS。如需詳細資訊，請參閱EKS服務端點EMR上的 Amazon。

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。與之前的版本相比，此版本已使用最近更新的 Amazon Linux 套件進行了重新整理。

區域：emr-6.5.0-20220802適用EMR於 Amazon 所支援的所有區域EKS。如需詳細資訊，請參閱EKS服務端點EMR上的 Amazon。

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 的初始版本。

區域：emr-6.5.0-20211119適用EMR於 Amazon 所支援的所有區域EKS。如需詳細資訊，請參閱EKS服務端點EMR上的 Amazon。

容器映像標籤：emr-6.5.0:20211119

Amazon EMR EKS 6.4.0 版本

以下 Amazon EMR 6.4.0 版本可用於 Amazon EMR 上EKS。選取特定的 emr-6.4.0 XXXX 版本以檢視更多詳細資料，例如相關的容器映像檔標籤。

- emr-6.4.0-latest
- 電腦 -6.4.0-20240321

- emr-6.4.0-20221219
- emr-6.4.0-20210830

Amazon EMR 6.4.0 的發行公告

- 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。這些通常對應於應用程序的配置XML文件，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

- 在 6.4 EKS 版本EMR上，Amazon 無法使用具有互動式端點的 Spark Pod 範本功能。

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EMR於 Amazon 所支援的所有區域EKS。如需詳細資訊，請參閱EKS服務端點EMR上的 Amazon。

容器映像標籤：emr-6.4.0:latest

電腦 -6.4.0-20240321

發行公告：emr-6.4.0-20240321已於 2024 年 3 月 11 日發行。與舊版相比，此版本已透過最近更新的 Amazon Linux 套件和重大修正程式進行重新整理。

區域：emr-6.4.0-20240321適用EMR於 Amazon 所支援的所有區域EKS。如需詳細資訊，請參閱EKS服務端點EMR上的 Amazon。

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。與之前的版本相比，此版本已使用最近添加的 Amazon Linux 軟件包進行了刷新。

區域：emr-6.4.0-20221219適用EMR於 Amazon 所支援的所有區域EKS。如需詳細資訊，請參閱EKS服務端點EMR上的 Amazon。

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 的初始版本。

區域：emr-6.4.0-20210830適用EMR於 Amazon 所支援的所有區域EKS。如需詳細資訊，請參閱EKS服務端點EMR上的 Amazon。

容器映像標籤：emr-6.4.0:20210830

Amazon EMR EKS 6.3.0 版本

以下 Amazon EMR 6.3.0 版本可用於 Amazon EMR 上EKS。選取特定的 emr-6.3.0 XXXX 版本以檢視更多詳細資料，例如相關的容器映像檔標籤。

- emr-6.3.0-latest
- 電腦 -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 6.3.0 的發行公告

- 新功能-從 6.x 版本系列的 Amazon EMR 6.3.0 開始，Amazon EMR 上EKS支持 Spark 的網繭模板功能。您也可以開啟 Amazon EMR 的 Spark 事件日誌輪替功能EKS。如需詳細資訊，請參閱 使用 Pod 範本 和 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。這些通常對應於應用程序的配置XML文件，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EMR於 Amazon 所支援的所有區域EKS。如需詳細資訊，請參閱EKS服務端點EMR上的 Amazon。

容器映像標籤：emr-6.3.0:latest

電腦 -6.3.0-20240321

發行公告：emr-6.3.0-20240321已於 2024 年 3 月 11 日發行。與舊版相比，此版本已透過最近更新的 Amazon Linux 套件和重大修正程式進行重新整理。

區域：emr-6.3.0-20240321適用EMR於 Amazon 所支援的所有區域EKS。如需詳細資訊，請參閱EKS服務端點EMR上的 Amazon。

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。與之前的版本相比，此版本已使用最近更新的 Amazon Linux 套件進行了重新整理。

區域：emr-6.3.0-20220802適用EMR於 Amazon 所支援的所有區域EKS。如需詳細資訊，請參閱EKS服務端點EMR上的 Amazon。

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EMR於 Amazon 所支援的所有區域EKS。如需詳細資訊，請參閱EKS服務端點EMR上的 Amazon。

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EMR於 Amazon 所支援的所有區域EKS。如需詳細資訊，請參閱EKS服務端點EMR上的 Amazon。

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 的初始版本。

區域：emr-6.3.0-20210429適用EMR於 Amazon 所支援的所有區域EKS。如需詳細資訊，請參閱EKS服務端點EMR上的 Amazon。

容器映像標籤：emr-6.3.0:20210429

Amazon EMR EKS 6.2.0 版本

以下 Amazon EMR 6.2.0 版本可用於 Amazon EMR 上EKS。選取特定的 emr-6.2.0 XXXX 版本以檢視更多詳細資料，例如相關的容器映像檔標籤。

- emr-6.2.0-latest
- 電腦 -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 6.2.0 的發行公告

- 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。這些通常對應於應用程序的配置XML文件，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EMR於 Amazon 所支援的所有區域EKS。如需詳細資訊，請參閱EKS服務端點EMR上的 Amazon。

容器映像標籤：emr-6.2.0:20240321

電腦 -6.2.0-20240321

發行公告：emr-6.2.0-20240321已於 2024 年 3 月 11 日發行。與舊版相比，此版本已透過最近更新的 Amazon Linux 套件和重大修正程式進行重新整理。

區域：emr-6.2.0-20240321適用EMR於 Amazon 所支援的所有區域EKS。如需詳細資訊，請參閱EKS服務端點EMR上的 Amazon。

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。與之前的版本相比，此版本已使用最近更新的 Amazon Linux 套件進行了重新整理。

區域：emr-6.2.0-20220802適用EMR於 Amazon 所支援的所有區域EKS。如需詳細資訊，請參閱EKS服務端點EMR上的 Amazon。

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 的初始版本。

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 在 EKS 5.36.0 版本上發布

以下 Amazon EMR 5.36.0 版本可用於 Amazon EMR 上。EKS選取特定的 emr-5.36.0 XXXX 版本以檢視更多詳細資料，例如相關的容器映像檔標籤。

- emr-5.36.0-latest
- 埃姆尔 -5.36.0-20240321
- emr-5.36.0-20221219
- emr-5.36.0-20220620

- emr-5.36.0-20220525

Amazon EMR 5.36.0 的發行公告

- 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。
- 支持的組件- aws-hm-client,, emr-ddb aws-sagemaker-spark-sdk, EMR-好吃的東西, EMR-運動, 核心服務器.
- 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。這些通常對應於應用程序的配置XML文件，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EMR於 Amazon 所支援的所有區域EKS。如需詳細資訊，請參閱EKS服務端點EMR上的 Amazon。

容器映像標籤：emr-5.36.0:latest

埃姆尔 -5.36.0-20240321

發行公告：emr-5.36.0-20240321已於 2024 年 3 月 11 日發行。與舊版相比，此版本已透過最近更新的 Amazon Linux 套件和重大修正程式進行重新整理。

區域：emr-5.36.0-20240321適用EMR於 Amazon 所支援的所有區域EKS。如需詳細資訊，請參閱EKS服務端點EMR上的 Amazon。

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。與之前的版本相比，此版本已使用最近更新的 Amazon Linux 套件進行了重新整理。

區域：emr-5.36.0-20221219適用EMR於 Amazon 所支援的所有區域EKS。如需詳細資訊，請參閱EKS服務端點EMR上的 Amazon。

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。與之前的版本相比，此版本已使用最近更新的 Amazon Linux 套件進行了重新整理。

區域：emr-5.36.0-20220620適用EMR於 Amazon 所支援的所有區域EKS。如需詳細資訊，請參閱EKS服務端點EMR上的 Amazon。

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 的初始版本。

區域：emr-5.36.0-20220525適用EMR於 Amazon 所支援的所有區域EKS。如需詳細資訊，請參閱EKS服務端點EMR上的 Amazon。

容器映像標籤：emr-5.36.0:20220525

Amazon EMR 在 EKS 5.35.0 版本上發布

以下 Amazon EMR 5.35.0 版本可在 Amazon EMR 上使用。EKS選取特定的 emr-5.35.0 XXXX 版本，以檢視更多詳細資料，例如相關的容器映像標記。

- emr-5.35.0-latest
- 埃姆尔 -5.35.0-20240321
- emr-5.35.0-20221219
- emr-5.35.0-20220802
- emr-5.35.0-20220307

Amazon EMR 5.35.0 的發行公告

- 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)。
- 支持的組件- aws-hm-client （Glue 連接器）， aws-sagemaker-spark-sdk，EMR-S3-選擇，EMRFS，EMR-DDB，火花。
- 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。這些通常對應於應用程序的配置XML文件，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EMR於 Amazon 所支援的所有區域EKS。如需詳細資訊，請參閱EKS服務端點EMR上的 Amazon。

容器映像標籤：emr-5.35.0:latest

埃姆尔 -5.35.0-20240321

發行公告：emr-5.35.0-20240321已於 2024 年 3 月 11 日發行。與舊版相比，此版本已透過最近更新的 Amazon Linux 套件和重大修正程式進行重新整理。

區域：emr-5.35.0-20240321適用EMR於 Amazon 所支援的所有區域EKS。如需詳細資訊，請參閱EKS服務端點EMR上的 Amazon。

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。與之前的版本相比，此版本已使用最近更新的 Amazon Linux 套件進行了重新整理。

區域：emr-5.35.0-20221219適用EMR於 Amazon 所支援的所有區域EKS。如需詳細資訊，請參閱EKS服務端點EMR上的 Amazon。

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。與之前的版本相比，此版本已使用最近更新的 Amazon Linux 套件進行了重新整理。

區域：emr-5.35.0-20220802適用EMR於 Amazon 所支援的所有區域EKS。如需詳細資訊，請參閱EKS服務端點EMR上的 Amazon。

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。與之前的版本相比，此版本已使用最近更新的 Amazon Linux 套件進行了重新整理。

區域：emr-5.35.0-20220307適用EMR於 Amazon 所支援的所有區域EKS。如需詳細資訊，請參閱EKS服務端點EMR上的 Amazon。

容器映像標籤：emr-5.35.0:20220307

Amazon EMR 在 EKS 5.34.0 版本上發布

以下 Amazon EMR 5.34.0 版本可在 Amazon EMR 上使用。EKS選取特定的 emr-5.34.0 XXXX 版本以檢視更多詳細資料，例如相關的容器映像檔標籤。

- emr-5.34.0-latest
- 埃姆尔 -5.34.0-20240321
- emr-5.34.0-20220802

Amazon EMR 5.34.0 的發行公告

- 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。這些通常對應於應用程序的配置XML文件，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EMR於 Amazon 所支援的所有區域EKS。如需詳細資訊，請參閱EKS服務端點EMR上的 Amazon。

容器映像標籤：emr-5.34.0:latest

埃姆尔 -5.34.0-20240321

發行公告：emr-5.34.0-20240321已於 2024 年 3 月 11 日發行。與舊版相比，此版本已透過最近更新的 Amazon Linux 套件和重大修正程式進行重新整理。

區域：emr-5.34.0-20240321適用EMR於 Amazon 所支援的所有區域EKS。如需詳細資訊，請參閱EKS服務端點EMR上的 Amazon。

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。與之前的版本相比，此版本已使用最近更新的 Amazon Linux 套件進行了重新整理。

區域：emr-5.34.0-20220802適用EMR於 Amazon 所支援的所有區域EKS。如需詳細資訊，請參閱EKS服務端點EMR上的 Amazon。

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。與之前的版本相比，此版本已使用最近更新的 Amazon Linux 套件進行了重新整理。

區域：emr-5.34.0-20211208適用EMR於 Amazon 所支援的所有區域EKS。如需詳細資訊，請參閱EKS服務端點EMR上的 Amazon。

容器映像標籤：emr-5.34.0:20211208

Amazon EMR 在 EKS 5.33.0 版本上發布

以下 Amazon EMR 5.33.0 版本可用於 Amazon EMR 上。EKS選取特定的 emr-5.33.0 XXXX 版本以檢視更多詳細資料，例如相關的容器映像檔標籤。

- emr-5.33.0-latest
- 埃姆尔 -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 5.33.0 的發行公告

- 新功能-從 5.x 版本系列中的 Amazon EMR 5.33.0 開始，Amazon EMR 上EKS支持 Spark 的網繭模板功能。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。這些通常對應於應用程序的配置XML文件，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EMR於 Amazon 所支援的所有區域EKS。如需詳細資訊，請參閱EKS服務端點EMR上的 Amazon。

容器映像標籤：emr-5.33.0:latest

埃姆尔 -5.33.0-20240321

發行公告：emr-5.33.0-20240321已於 2024 年 3 月 11 日發行。與舊版相比，此版本已透過最近更新的 Amazon Linux 套件和重大修正程式進行重新整理。

區域：emr-5.33.0-20240321適用EMR於 Amazon 所支援的所有區域EKS。如需詳細資訊，請參閱EKS服務端點EMR上的 Amazon。

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。與舊版相比，此版本已透過最近更新的 Amazon Linux 套件和重要修正程式進行重新整理。

區域：emr-5.33.0-20221219適用EMR於 Amazon 所支援的所有區域EKS。如需詳細資訊，請參閱EKS服務端點EMR上的 Amazon。

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。與之前的版本相比，此版本已使用最近更新的 Amazon Linux 套件進行了重新整理。

區域：emr-5.33.0-20220802適用EMR於 Amazon 所支援的所有區域EKS。如需詳細資訊，請參閱EKS服務端點EMR上的 Amazon。

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EMR於 Amazon 所支援的所有區域EKS。如需詳細資訊，請參閱EKS服務端點EMR上的 Amazon。

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EMR於 Amazon 所支援的所有區域EKS。如需詳細資訊，請參閱EKS服務端點EMR上的 Amazon。

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EMR於 Amazon 所支援的所有區域EKS。如需詳細資訊，請參閱EKS服務端點EMR上的 Amazon。

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 的初始版本。

區域：emr-5.33.0-20210323適用EMR於 Amazon 所支援的所有區域EKS。如需詳細資訊，請參閱EKS服務端點EMR上的 Amazon。

容器映像標籤：emr-5.33.0-20210323

Amazon EMR 在 EKS 5.32.0 版本上發布

以下 Amazon EMR 5.32.0 版本可用於 Amazon EMR 上。EKS選取特定的 emr-5.32.0 XXXX 版本以檢視更多詳細資料，例如相關的容器映像檔標籤。

- emr-5.32.0-latest
- 埃姆尔 -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 5.32.0 的發行公告

- 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。這些通常對應於應用程序的配置XML文件，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EMR於 Amazon 所支援的所有區域EKS。如需詳細資訊，請參閱EKS服務端點EMR上的 Amazon。

容器映像標籤：emr-5.32.0:latest

埃姆尔 -5.32.0-20240321

發行公告：emr-5.32.0-20240321已於 2024 年 3 月 11 日發行。與舊版相比，此版本已透過最近更新的 Amazon Linux 套件和重大修正程式進行重新整理。

區域：emr-5.32.0-20240321適用EMR於 Amazon 所支援的所有區域EKS。如需詳細資訊，請參閱EKS服務端點EMR上的 Amazon。

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。與之前的版本相比，此版本已使用最近更新的 Amazon Linux 套件進行了重新整理。

區域：emr-5.32.0-20220802適用EMR於 Amazon 所支援的所有區域EKS。如需詳細資訊，請參閱EKS服務端點EMR上的 Amazon。

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 的初始版本。

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

下表說明自 Amazon EMR 上次發行以來文件的重要變更EKS。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
新版本	Amazon EMR EKS 7.2.0 版本	2024年7月25日
新版本	Amazon EMR EKS 7.1.0 版本	2024年4月17日
新版本	Amazon EMR 在 EKS 7.0.0 版本上	2023 年 12 月 22 日
新版本	Amazon EMR 在 EKS 6.15.0 版本上發布	2023 年 11 月 17 日
新版本	Amazon EMR 在 EKS 6.14.0 版本上發布	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 在 EKS 6.13.0 版本上發布，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 在 EKS 6.12.0 版本上發布	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
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	更新了在 Amazon ECR 公共圖庫中查找基本映像信息的自定義映像文檔。	2023 年 6 月 8 日
新版本	Amazon EMR 在 EKS 6.11.0 版本上發布	2023 年 6 月 8 日

變更	描述	日期
新內容	在 使用 Amazon EMR on EKS 執行作業 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	EMR在 Amazon 上創建了一個文檔歷史記錄頁面EKS。	2023 年 3 月 13 日
受管政策頁面	EMR在 Amazon 上創建了一個受管政策頁面EKS。	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。